Vision par ordinateur: outils fondamentaux

Radu Horaud, Olivier Monga

Vision par ordinateur : outils fondamentaux

Radu HORAUD et Olivier MONGA

Deuxième édition

Editions Hermès

Vision par ordinateur : outils fondamentaux

Radu HORAUD, CNRS
et
Olivier MONGA, INRIA

Deuxième édition

"Ouvrage rédigé avec le concours du Ministère de l'enseignement supérieur et de la recherche (DIST)"

Editions Hermès

Table des matières

Avant-propos

La vision par ordinateur est une discipline dont les premières bases théoriques ont été jétées dans les années '60. Depuis, étant donné le spectre très large d'applications industrielles, militaires, aérospatiales et médicales qui peut être envisagé, la vision par ordinateur a vite fait de dépasser le cadre relativement restreint des laboratoires de recherche. Aujourd'hui, rares sont les écoles d'ingénieurs, les instituts de technologie ou les 2^e et 3^e cycles universitaires scientifiques ne proposant pas un ou plusieurs cours de vision par ordinateur.

Cet ouvrage s'adresse à l'étudiant, à l'ingénieur, à l'enseignant, au chercheur et à tous ceux qui désirent :

- s'initier au domaine de la vision par ordinateur et à ces travaux de recherche les plus récents,
- se spécialiser dans ce domaine,
- acquérir les connaissances permettant par la suite d'aborder des aspects plus techniques,
- suivre des cours et préparer des examens,
- enseigner cette discipline.

Les outils fondamentaux de la vision par ordinateur – détection et segmentation, extraction d'indices visuels, géométrie et calibration des capteurs, stéréoscopie, localisation et reconnaissance d'objets, reconstruction, traitement d'images volumiques – sont présentés dans un langage mathématique simple, le souci de base étant la clarté. L'ouvrage peut ainsi être abordé par le néophyte ayant des connaissances de base en mathématiques et en informatique (niveau DEUG ou équivalent). Il sera également utile au spécialiste comme manuel de référence. L'ouvrage contient par ailleurs de nombreux exemples d'utilisation de la vision par ordinateur dans deux domaines de technologie de pointe : la

robotique et l'imagerie médicale. Enfin, le texte est complété par 185 références bibliographiques commentées tout le long de l'exposé.

Remerciements

Nous tenons tout d'abord à remercier nos organismes de tutelle, le CNRS et l'INRIA. Sans la liberté dont nous jouissons en tant que chercheurs de ces organismes, cet ouvrage n'aurait pu voir le jour.

Beaucoup de personnes ont contribué à ameliorer le contenu technique et la présentation de cet ouvrage. Sans pouvoir les citer individuellement, nous remercions les étudiants qui ont suivi nos cours et qui ont bien voulu nous faire part de leurs remarques et critiques sur des versions préliminaires. Ils nous ont appris à être clairs, nous leur en sommes reconnaissants.

Nous tenons ensuite à exprimer notre gratitude à toutes celles et tous ceux de nos collègues qui ont, d'une manière ou d'une autre, participé à l'élaboration de ce texte : Nicholas Ayache, Serge Benayoun, Horst Beyer, Jean-Daniel Boissonnat, Patrick Bouthémy, Serge Castan, Philippe Cinquin, François Chaumette, Guillaume Chambleboux, Alain Chéhikian, James Crowley, Marc Berthod, Boubakeur Boufama, Philippe Bobet, Pascal Brand, Jacques Demongeot, Rachid Deriche, Fadi Dornaika, Olivier Faugeras, Jean-Pierre Gambotto, Gérard Giraudon, Patrick Gros, Marsha Jo Hannah, Laurent Hérault, Claude Inglebert, Naamen Keskes, Stéphane Lavallée, Bernard Lacolle, Augustin Lux, Grégoire Malandain, Bruno Mazier, Luce Morin, Roger Mohr, Long Quan, Thai Quynh Phong, Stephen Pollard, François Robert, Peter Sander, Thomas Skordas, Humberto Sossa, Françoise Veillon et Brigitte Wrobel-Dautcourt.

Finalement, nous remercions Nathalie Gaudechoux pour sa contribution à la réalisation matérielle d'une partie du texte et des figures.

Avant-propos de la deuxième édition

Cette deuxième édition de l'ouvrage a été augmentée d'environ cinquante pages. Nous avons également fait un effort considérable d'harmonisation des notations, tâche qui n'est pas du tout facile, compte tenu de la diversité des sujets abordés. Les chapitres 2, 3, 5, 6, 8, 11 et 12 ont été remaniés de la façon suivante :

Le chapitre 2 "Détection de contours" a été complété par une discussion quant à l'intérêt de l'isotropie des filtres de lissage. De plus, on décrit en détail une méthode permettant la synthèse par des filtres récursifs du filtre gaussien ainsi que de ses dérivées.

Les méthodes d'approximation polygonale décrites au chapitre 3 ont été illustrées par un exemple.

Le chapitre 5 "Géométrie et calibration des caméras" a subi plusieurs modifications : la section 5.2.6. a été corrigée et rendue plus claire, la section 5.6. a été rajoutée ce qui permettra au lecteur d'aborder facilement le cas d'une "caméra affine".

Trois sections du chapitre 6 "Vision stéréoscopique" ont subi des modifications importantes : les sections 6.3.3. (Estimation de la matrice fondamentale), 6.3.4. (Reconstruction) et 6.5.3. (Appariement par relaxation).

Une méthode "quasi" linéaire de calcul de la transformation objet-caméra a été rajoutée au chapitre 8. Cette méthode permet une utilisation judicieuse d'un modèle simplifié de caméra (nouvellement introduit au chapitre 5) dans le cadre d'un algorithme itératif très simple et très performant – section 8.6.

Le chapitre 11 "Des images volumiques 3D à la géométrie des surfaces" aborde, d'une manière générale, le problème de la modélisation locale d'une surface. Nous y avons rajouté un exemple d'utilisation de l'approche multi-échelle. Cet exemple illustre à la fois l'intérêt ainsi que les difficultés associés avec cette approche.

Le chapitre 12 “Des cartes de profondeur à la géométrie des surfaces” a été sensiblement étoffé. Ce chapitre donne maintenant une ouverture à la fois sur la caractérisation de surfaces à partir de cartes de profondeur et sur les algorithmes les plus récents d'extraction d'indices visuels.

Chapitre 1

Introduction

1.1. Qu'est-ce que la vision ?

L'espace qui nous entoure a une structure tri-dimensionnelle (3D). Lorsque l'on demande à une personne de décrire ce qu'elle *voit*, elle n'éprouve aucune difficulté à nommer les objets qui l'entourent : téléphone, table, livre... Et pourtant l'information qui est réellement disponible sur la rétine des ses yeux n'est, ni plus ni moins, une collection de points (environ un million !). En chaque point ou *pixel* (picture element) il y a tout simplement une information qui donne une indication quant à la quantité de lumière et la couleur qui proviennent de l'espace environnant et qui ont été projetées à cet endroit de la rétine. Le téléphone, la table ou le livre *n'existent pas* sur la rétine. Guidé à la fois par l'information codée dans l'image (ou la rétine) et par ses propres connaissances, le processus visuel *construit* des percepts. Le téléphone ou le livre sont le réponses finales, résultant d'un processus *d'interprétation* qui fait partie intégrante du système de vision. De plus, il n'y a pas de correspondance terme à terme entre l'information sensorielle (la lumière et la couleur) et la réponse finale (des objets 3D). Le système de vision doit *fournir* les connaissances nécessaires afin de permettre une interprétation non ambiguë.

1.2. Comprendre la vision

Il n'est pas suffisant de constater qu'un problème est complexe. Encore faut-il essayer de le comprendre dans ces moindres détails et de proposer une solution.

La vision a suscité l'intérêt de nombreux scientifiques et philosophes depuis déjà très longtemps. Parmi ceux-ci, les neurobiologistes mènent des recherches théoriques et expérimentales afin d'essayer de comprendre l'anatomie et le fonctionnement du cerveau dans son ensemble. Ils ont découvert une structure très complexe qui est loin de leur avoir révélé tous ses secrets. La tâche des neurobiologistes semble être à la fois grandiose et illusoire. Grandiose, parce que le cerveau est une des plus complexes *inventions* de la nature. Il reste et restera pour longtemps le bastion encore inconnu que les sciences humaines se proposent de conquérir. Illusoire, car on ne connaît pas ses limites. Ces limites ne sont-elles par repoussées à chaque découverte ? David Hubel [99] a merveilleusement bien exprimé ce paradoxe : *Le cerveau peut-il comprendre le cerveau ?*

Avec la naissance de machines de calcul de plus en plus sophistiquées, un certain nombre de scientifiques se sont attaqués au problème de la vision d'un point de vue quantitatif : est-il possible de construire un *modèle computationnel* pour la perception visuelle ? Attention : il ne s'agit pas de fournir une explication de comment marche la vision biologique mais de **créer un modèle** qui, vu de l'extérieur, possède des propriétés semblables.

Ce modèle *artificiel* peut-il être d'une utilité quelconque quant à la vision biologique ? Peut-il constituer la base d'une nouvelle technologie – *des machines qui voient* ?

Il est certainement trop tôt pour répondre à ces questions et pour tirer des conclusions. Malgré les efforts non négligeables, il y a très peu de résultats convaincants. Nous pensons que deux démarches doivent être suivies simultanèment :

- essayer d'élaborer une théorie de la **vision par ordinateur** qui doit nous guider à long terme ;
- tenter de résoudre des problèmes spécifiques dans le cadre de cette théorie : de tels résultats partiels permettraient de confirmer ou au contraire de mettre en cause certains aspects de la théorie.

1.3. Une théorie de la vision

L'élaboration d'une théorie scientifique demande trois étapes :

1. énoncer la théorie, spécifier et élaborer les concepts de base : ces concepts doivent exprimer le cadre formel qui est à la base de la théorie,

2. exprimer ces concepts sous forme mathématique,

3. réaliser un ensemble expérimental qui permette de vérifier la théorie.

Voici comment la vision par ordinateur peut s'énoncer briévement dans les termes de ce paradigme. La vision est un processus de traitement de l'information. Elle utilise des stratégies bien définies afin d'atteindre ses buts. L'entrée d'un système de vision est constituée par une séquence d'images. Le système lui-même apporte un certain nombre de connaissances qui interviennent à tous les niveaux. La sortie est une description de l'entrée en termes d'objets et de relations entre ces objets.

Deux types de stratégies sont mises en jeu : ascendantes et descendantes. Les stratégies ascendantes tentent de construire à partir de l'information sensorielle une représentation la plus abstraite possible (par exemple, un ensemble de primitives géométriques 3D). Les stratégies descendantes déduisent à partir de l'ensemble d'objets connus par le système une description compatible avec les primitives extraites de l'image. Il est alors possible de mettre en correspondance la représentation extraite de l'image avec les descriptions des objets afin de décrire les données sensorielles en termes de ces objets.

Les connaissances mises en jeu peuvent être de trois types : physiques, géométriques et sémantiques. Les lois physiques imposent des contraintes aux signaux lumineux qui partant d'une source, traversent la scène et se projettent sur l'image. La gravitation impose à la scène (et donc à l'image) une structure hétérogène : prépondérance de lignes verticales et horizontales pour ne citer qu'un exemple. La forme des objets (l'ensemble de ses surfaces) et la géométrie de la formation de l'image imposent des contraintes très strictes quant aux structures susceptibles d'être présentes dans l'image. A un niveau plus élevé, un objet peut être décrit par sa fonction dans le contexte d'un raisonnement symbolique. Cette fonction n'est pas directement mesurable dans l'image. On devrait pouvoir dériver des contraintes sur la forme et l'emplacement d'un objet à partir de sa fonction. Par exemple, le mot *chaise* désigne une classe d'objets réels (un objet réel est un objet qui occupe une place dans l'espace physique). Cependant il y a une grande variété de chaises quant à la forme et à la couleur pour ne citer que deux propriétés. Quelles sont les propriétés communes à toutes les chaises, **mesurables** dans l'image ?

L'étape suivante consiste à exprimer ces stratégies et connaissances dans le cadre d'un formalisme mathématique et à construire les algorithmes correspon-

dants. Les performances de ces algorithmes doivent correspondre aux qualités exigées d'un tel système : *la reconnaissance visuelle doit être fiable et rapide.*

1.4. Le paradigme de David Marr

Vers la fin des années 70, David Marr [121] a proposé un modèle calculatoire pour le traitement et la représentation de l'information visuelle. Voici quels sont les principaux traits de ce paradigme :

- à partir d'une ou de plusieurs images un processus d'extraction de caractéristiques produit un description en termes d'attributs bi-dimensionnels ; ce niveau de représentation est appelé *première ébauche* (primal sketch) ;

- la première ébauche constitue l'entrée d'un certain nombre de processus plus ou moins indépendants qui calculent des propriétés tri-dimensionnelles locales relatives à la scène ; il s'agit d'une représentation centrée sur l'observateur, appelée *ébauche 2.5D* ; ces processus opèrent sur une séquence d'images (analyse du mouvement) sur une paire d'images (stéréoscopie) ou sur une seule image. Dans ce dernier cas il s'agit de processus d'inférence qui utilisent des connaissances géométriques (analyse des contours), géométriques et statistiques (analyse des textures), photométriques (analyse des ombrages) ou colorimétriques (analyse des reflets) ;

- l'ébauche 2.5D est mise en correspondance avec des connaissances 3D afin de construire une description de la scène en termes d'objets et de relations entre les objets ; il s'agit maintenant d'une représentation *centrée sur la scène* (la description ne dépend plus de la position de l'observateur).

1.5. Segmentation, reconstruction, reconnaissance

En pratique, le paradigme de David Marr se traduit par trois étapes de traitement : segmentation, reconstruction et reconnaissance.

La segmentation d'images étant la pierre de base de tout système de vision, de nombreux travaux lui ont été consacrés. La diversité des images, la difficulté du problème, les origines variées des chercheurs, l'évolution de la puissance de calcul des ordinateurs, et un certain empirisme dans l'évaluation des résultats ont conduit à l'introduction d'une multitude d'algorithmes.

Quelle que soit son origine, une image constitue une représentation d'un univers composé d'entités : objets dans une scène d'intérieur, cellules, surfaces sismiques, organes du corps humain ...Le but de toute méthode de segmentation est l'extraction d'attributs caractérisant ces entités. Les attributs étudiés correspondent à des points d'intérêt ou à des zones caractéristiques de l'image : contours et régions. La détection des contours (chapitre 2) implique la recherche des discontinuités locales de la fonction des niveaux de gris de l'image. La segmentation des contours (chapitre 3) consiste à approximer les contours par des représentations analytiques, telles que des droites ou des coniques. L'extraction de régions (chapitre 4) revient à déterminer des zones homogènes en niveaux de gris de l'image. Par exemple, dans le cas d'images réelles, les contours correspondent aux frontières des objets et les régions à leurs surfaces. Ces deux approches "contour" et "région" sont duales en ce sens qu'une région définit une ligne par son contour, et qu'une ligne fermée définit une région. Elles amènent cependant à des algorithmes complètement différents et ne fournissant pas les mêmes résultats. Cette dualité est cependant peu exploitée dans la plupart des méthodes existantes.

Un autre aspect de la segmentation est celui qui consiste à retrouver la géométrie des objets à partir des images. On obtient ainsi des représentations intrinsèques, aisément manipulables et utilisables, à partir de la réalité physique induite par l'image. De manière à obtenir ces caractéristiques géométriques, on est souvent conduit à définir une suite hiérarchique de représentations de l'information image permettant finalement d'obtenir des indices visuels servant à résoudre une tâche donnée. Dans les chapitres 10 et 11 nous illustrons ces principes dans le cas des images bi-dimensionnelles et des images volumiques.

La calibration est la première étape indispensable pour toute méthode de reconstruction (à moins que la calibration ait lieu en même temps que la reconstruction). Le chapitre 5 décrit en détail les modèles géométriques de plusieurs capteurs basés sur une caméra ainsi que plusieurs techniques de détermination des paramètres de ces capteurs (calibration). On étudiera ainsi la caméra matricielle, la caméra linéaire ainsi que les capteurs stéréoscopiques passifs et actifs. Le chapitre 12 aborde le problème de la caractérisation des cartes de profondeur obtenues avec un capteur stéréoscopique actif (caméra et faisceau laser).

Le chapitre 6 décrit en détail les principes de reconstruction tri-dimensionnelle à partir d'un système stéréoscopique. Plus particulièrement, le problème de

mise en correspondance stéréo est abordé d'un point de vue géométrique et algorithmique. La reconstruction de surfaces polyédriques par vision stéréoscopique à partir d'images 2D peut être réalisée par une approche régions ou par une approche contours. Le chapitre 10 présente une approche de type géométrie algorithmique pour résoudre le problème de reconstruction polyédrique.

La reconnaissance consiste essentiellement à comparer des indices visuels bi- ou tri-dimensionnels avec les indices des objets à reconnaître. Les méthodes de reconnaissance sont souvent couplées avec des méthodes de localisation. Le chapitre 7 décrit quelques méthodes de localisation (position et orientation avec six degrés de liberté) à partir d'indices visuels 3D mis en correspondance avec des indices d'objets 3D. Le chapitre 8 décrit une méthode de localisation à partir d'indices visuels 2D mis en correspondance avec des indices d'objets 3D. Enfin le chapitre 9 montre comment on peut combiner les méthodes de localisaton avec des méthodes de recherche arborescente pour pouvoir reconnaître des objets rigides.

1.6. Quelques références bibliographiques

Le premier ouvrage consacré partiellement à la vision par ordinateur est celui de Duda et Hart, datant de 1973 [50]. A une première partie consacrée à la reconnaissance des formes, fait suite une deuxième partie qui introduit les bases théoriques d'une approche géométrique de l'interprétation d'une image.

Les ouvrages de Gonzales et Wintz (1977) et de Rosenfeld et Kak (1982, seconde édition) passent en revue l'état de l'art de ce qu'on appelle aujourd'hui la "vision bas niveau" [73], [160].

Pendant longtemps, l'ouvrage de référence en vision par ordinateur a été celui de Ballard et Brown (1982), [15]. Sans rentrer dans les détails mathématiques, ce texte fournit une vue synthétique des travaux de recherche dans les années 80.

L'ouvrage de Horn, publié en 1986, aborde quelques aspects de la vision d'un point de vue plus fondamental [97]. Les bases mathématiques de la formation d'une image, de la détection de contours et de régions, des propriétés photométriques ainsi que de la perception du mouvement sont clairement présentées.

L'utilisation de la vision par ordinateur pour la navigation des robots est le thème de l'ouvrage d'Ayache, publié en 1989 [8] et en 1991 [9] (version anglaise). On y trouve notamment les détails de l'utilisation du filtre de Kalman étendu

pour intégrer l'information provenant de plusieurs cartes stéréoscopiques.

Enfin, l'ouvrage de Faugeras est, avec le nôtre, le plus récent [55]. Il propose une approche géométrique (géometrie projective et euclidienne) pour résoudre notamment le problème de reconstruction. Très clair et très détaillé, contenant de nombreux exemples ainsi que des exercices, ce texte rend compte de 10 années de travaux de recherche effectués par l'auteur et par son équipe de l'INRIA.

Chapitre 2

Détection de contours

Les contours des objets dans des images de dimensions quelconques (images naturelles 2D, images médicales 3D ...) correspondent le plus souvent aux extréma locaux du gradient ou aux zéros du laplacien de la fonction des niveaux de gris. Les difficultés de la détection des contours proviennent du bruit important présent dans les images (bruit du capteur, bruit d'échantillonnage, irrégularités de la surface des objets ...). On est donc confronté au problème de la différenciation d'un signal bruité. Pour le résoudre, il est d'abord nécessaire de définir des critères de performance d'un opérateur incluant une modélisation des contours recherchés et la notion de bruit. Dans une deuxième phase, on déduit de ces critères une famille de filtres optimaux. Dans une troisième phase, on résout le problème posé par l'implantation de ces filtres.

Nous mettrons l'accent dans cette partie sur l'utilisation de filtres linéaires. En effet, dans le cas où on ne dispose pas de connaissances a priori sur l'image, ce type de méthode fournit des algorithmes de faible complexité dont les résultats sont satisfaisants sur la plupart des types d'images. Nous nous attacherons plus particulièrement aux filtres séparables récursifs car ils peuvent se généraliser à une dimension quelconque (séparabilité) , et permettent une implantation de faible coût algorithmique pour des opérateurs de réponse impulsionnelle infinie (récursivité).

2.1. Généralités

2.1.1. Filtrage linéaire d'une image

Rappelons rapidement que filtrer signifie convoluer une image $I(x,y)$ avec une fonction $f(x,y)$ qui s'appelle réponse impulsionnelle du filtre. Dans le cas continu l'image filtrée est donnée par :

$$\begin{aligned} I_f(x,y) &= (f * I)(x,y) & [2.1]\\ &= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x',y')I(x-x',y-y')dx'dy' & [2.2]\\ &= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x-x',y-y')I(x',y')dx'dy' & [2.3] \end{aligned}$$

Dans le cas discret les domaines de I et de f sont bornés. Le domaine de I est $[-N/2,+N/2]$ et le domaine de f est $[-K/2,+K/2]$. On a nécessairement $K \leq N$, N étant la taille de l'image. Dans le cas discret la convolution s'écrit :

$$\begin{aligned} I_f(x,y) &= (f * I)(x,y) & [2.4]\\ &= \sum_{i'=-K/2}^{i'=+K/2}\sum_{j'=-K/2}^{j'=+K/2} f(i-i',j-j')I(i',j') & [2.5] \end{aligned}$$

On notera que le filtrage linéaire consiste simplement à remplacer chaque niveau de gris par une combinaison linéaire des niveaux de gris des points voisins ; les cœfficients de cette combinaison linéaire sont définis par la réponse impulsionnelle du filtre. Cette réponse impulsionnelle est la réponse du filtre à la fonction impulsion (d'où son nom!).

2.1.2. Le gradient d'une image

Le gradient d'une image se calcule comme suit :

$$\begin{aligned} I_x(x,y) &= \frac{\partial I_f(x,y)}{\partial x} & [2.6]\\ I_y(x,y) &= \frac{\partial I_f(x,y)}{\partial y} & [2.7] \end{aligned}$$

En chaque point (x, y) de l'image nous pouvons donc calculer le vecteur *gradient*. Le module et la direction de ce vecteur sont donnés par :

$$G = (2_x + 2_y)^{\frac{1}{2}} \quad [2.8]$$

$$\simeq \max(I_x, I_y) \quad [2.9]$$

$$\simeq |\, I_x \,| + |\, I_y \,| \quad [2.10]$$

$$\phi = \arctan(\frac{I_y}{I_x}) \quad [2.11]$$

La direction du gradient maximise la dérivée directionnellle, et la norme du gradient est la valeur de cette dérivée. On obtient la dérivée de I dans une direction quelconque $\vec{d}$ à partir des deux dérivées directionnelles définissant le gradient I_x et I_y de la manière suivante : $L_{\vec{d}}(I) = \nabla I \cdot \vec{d}$.

2.1.3. Interprétation géométrique du gradient

Afin de mieux comprendre la signification du gradient, considérons une image constituée de deux régions "plates" séparées par une marche rectiligne (voir figure 2.3.). Essayons d'exprimer analytiquement cette image. Nous avons besoin pour cela de la fonction marche :

$$u(t) = \begin{cases} 1 & \text{si} \quad t \geq 0 \\ 0 & \text{si} \quad t < 0 \end{cases} \quad [2.12]$$

Nous avons également besoin de la fonction impulsion :

$$\delta(t) = \begin{cases} 1 & \text{si} \quad t = 0 \\ 0 & \text{sinon} \end{cases} \quad [2.13]$$

La fonction marche peut être écrite comme l'intégrale de la fonction de Dirac :

$$u(z) = \int_{-\infty}^{z} \delta(t)dt \quad [2.14]$$

La dérivée de $u(t)$ est :

$$u'(t) = \delta(t) \quad [2.15]$$

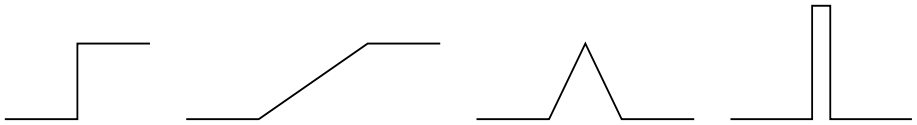


Figure 2.1. *Quelques profils de contours : marche, rampe, toit, pic.*

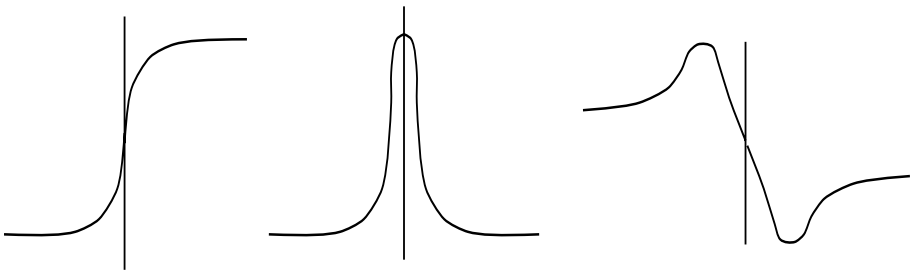


Figure 2.2. *Les deux principes de la détection de contours : dérivée première et dérivée seconde (discontinuité du signal, dérivée première, dérivée seconde).*

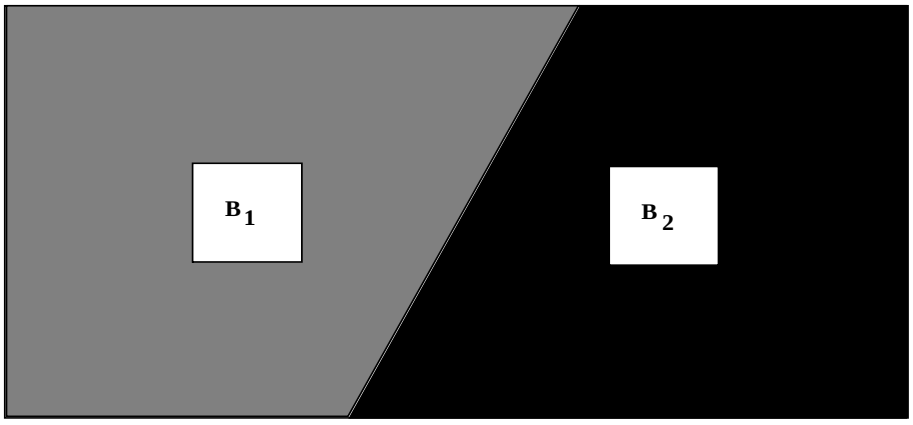


Figure 2.3. *Deux régions plates séparées par une marche d'escalier.*

Ecrivons maintenant l'équation de la droite décrivant la marche de l'image. Cette droite fait un angle θ avec l'axe des x, son équation s'écrit donc :

$$-x\sin\theta + y\cos\theta + \rho = 0 \qquad [2.16]$$

Nous pouvons maintenant écrire la fonction image :

$$I(x,y) = B_1 + (B_2 - B_1)u(-x\sin\theta + y\cos\theta + \rho) \qquad [2.17]$$

Les composantes du vecteur gradient au point image (x,y) sont donc :

$$I_x = \cos(\theta + \frac{\pi}{2})(B_2 - B_1)\delta(-x\sin\theta + y\cos\theta + \rho) \qquad [2.18]$$

$$I_y = \sin(\theta + \frac{\pi}{2})(B_2 - B_1)\delta(-x\sin\theta + y\cos\theta + \rho) \qquad [2.19]$$

Le gradient est nul partout dans l'image sauf le long du contour en marche. Le vecteur gradient se trouve dans le plan image et il est *perpendiculaire* à la direction du contour :

$$\phi = \theta + \frac{\pi}{2} \qquad [2.20]$$

La norme du gradient est égale à la dérivée de l'image le long de la normale au contour soit : $B_2 - B_1$.

2.2. Dérivation et séparabilité des filtres

L'objectif de cette partie est de montrer que sous réserve d'hypothèses raisonnables, on peut ramener la détection de contours au lissage et à la dérivation par filtrage linéaire d'un signal monodimensionnel. On notera que les méthodes décrites dans les développements suivants s'appliquent pour une image de dimension quelconque.

Soit $I(x,y)$ une image de dimension 2 (le même développement peut être réalisé pour un signal de dimension quelconque).

On admet communément que les contours correspondent aux discontinuités d'ordre 0 de I . On peut noter que cette hypothèse peut être mise à mal par certains contours associés aux discontinuités d'ordre 1 (contours en toit). Quoi qu'il en soit peu de travaux ont été consacrés à la détection de ce deuxième type de contours.

La détection de contours s'effectue classiquement de deux manières :

1. calcul du gradient et extraction des extréma locaux de la norme du gradient dans la direction du gradient : **approche gradient**.

2. calcul du laplacien et détermination des passages par zéro : **approche laplacien**.

On remarque que la première approche se ramène à extraire les zéros de la dérivée seconde dans la direction du gradient, ce qui est différent des passages par zéro du laplacien. Pratiquement ces deux types de méthodes fournissent cependant des résultats très proches.

Soit G le gradient de I :

$$G(x,y) = (\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})^t$$

Soit L le laplacien de I :

$$L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

Il s'agit donc de trouver des méthodes permettant de calculer $G(I)$ et $L(I)$. Une manière commode de poser ce problème est de rechercher des filtres linéaires permettant l'approximation du gradient ou du laplacien. Si on suppose le bruit de moyenne nulle, le filtrage linéaire est susceptible d'amener des solutions satisfaisantes. Pour la plupart des images traitées l'hypothèse d'un bruit blanc gaussien est raisonnable. Cependant dans certains cas un bruit impulsionnel nécessite au préalable un filtrage de lissage non linéaire par exemple un filtrage médian [122]. Le principe du filtre médian est de calculer en un point non pas une combinaison linéaire des niveaux de gris de ses voisins, mais une valeur médiane dépendant d'un tri des niveaux de gris des points voisins. Les figures 2.4. à 2.7. illustrent cette remarque en dimension 1 :

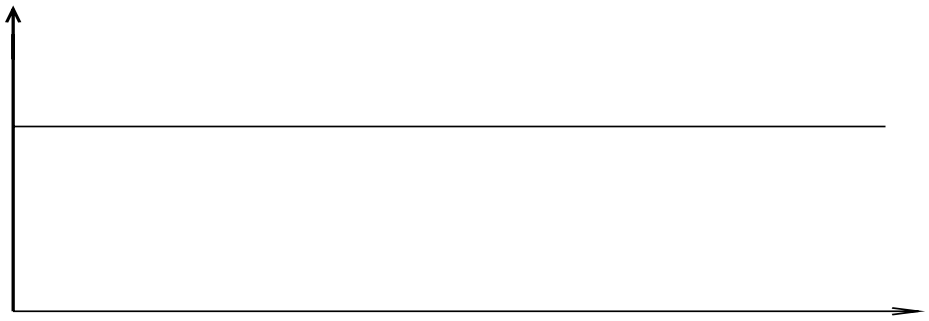


Figure 2.4. *Signal idéal.*

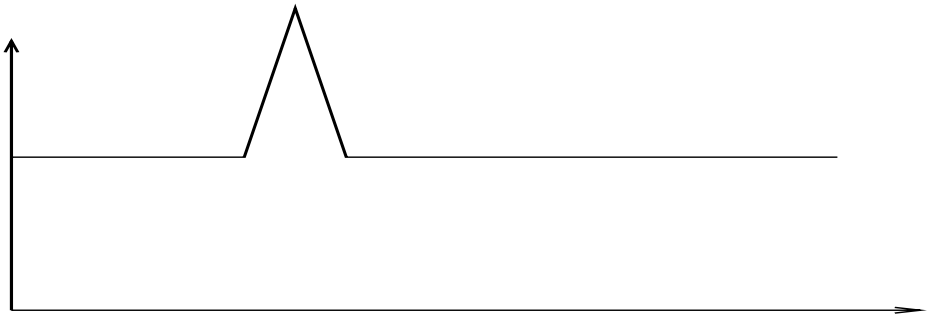


Figure 2.5. *Signal bruité par un bruit impulsionnel.*

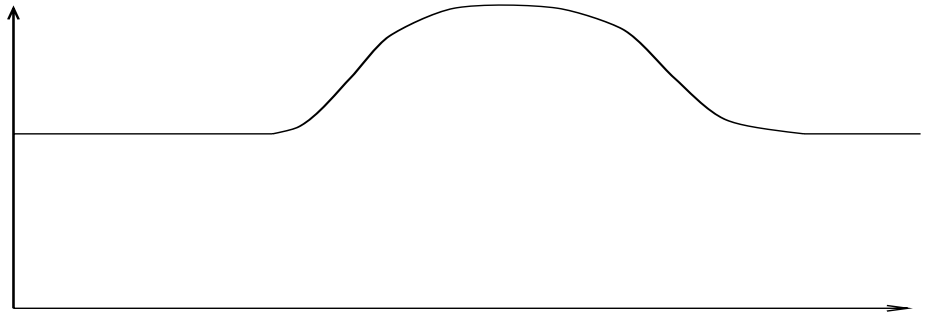


Figure 2.6. *Signal lissé par un filtre linéaire : pas d'issue !*

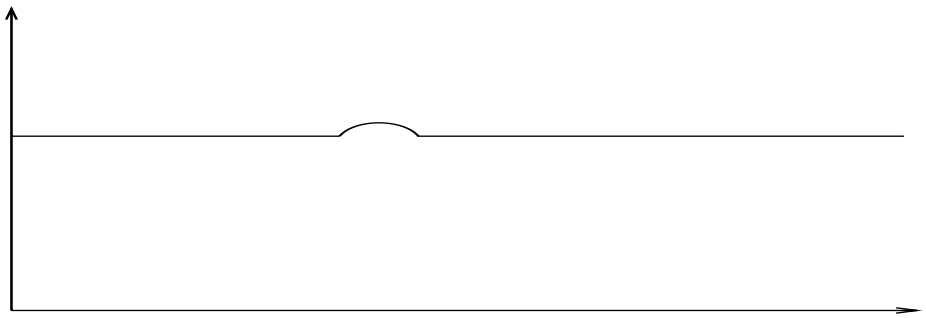


Figure 2.7. *Signal après un filtrage médian de dimension 3*
Soit $(i_1, i_2, \ldots i_n)$ *la suite des valeurs de* I
Soit $(j_1, j_2, \ldots j_n)$ *la suite des valeurs de* I *après filtrage*
j_p *: valeur médiane de* (i_{p-1}, i_p, i_{p+1})
$j_p = i_p$ *si* $i_p \leq i_{p+1}$ *et* $i_{p-1} \leq i_p$
$j_p = i_{p-1}$ *si* $i_{p+1} \leq i_{p-1}$ *et* $i_{p-1} \leq i_p \ldots$

On a donc ramené la détection de contours à la détermination de filtres linéaires permettant d'approximer le gradient ou le laplacien de l'image (le gradient ou le laplacien permettent de définir les discontinuités d'ordre 0 de la fonction des niveaux de gris). Soit un filtre linéaire de réponse impulsionnelle f et I un signal, la dérivée du signal convolué avec f est égale à l'image convoluée avec la dérivée de f :

$$(I * f)' = I * f'$$

$$(I * f)'' = I * f''$$

A cause de cette propriété les approches "dérivée première" et "dérivée seconde" se ramènent à déterminer un filtre de lissage de réponse $f(x, y)$. On calcule les dérivées premières et secondes par rapport aux variables x, y en convoluant avec $\frac{\partial f(x,y)}{\partial x}$, $\frac{\partial^2 f(x,y)}{\partial x^2} \cdots$ On remarque que le calcul du laplacien peut s'effectuer directement par convolution avec le filtre de réponse :

$$\frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

On obtient :

$$\begin{aligned} I_x(x,y) &= \frac{\partial(I*f)}{\partial x} = I * \frac{\partial f}{\partial x} = I * f_x \\ I_y(x,y) &= \frac{\partial(I*f)}{\partial y} = I * \frac{\partial f}{\partial y} = I * f_y \\ \Delta I(x,y) &= I * \Delta(f) \end{aligned} \qquad [2.21]$$

On remarquera que parfois l'expression $\Delta(f)$ se simplifie bien. Une autre solution pour le calcul du laplacien consiste à l'approximer par différence entre l'image lissée par deux filtres de lissage de largeurs différentes [165]. Cependant les résultats obtenus avec cette deuxième solution sont nettement moins satisfaisants que ceux fournis par un calcul direct du laplacien. Un des premiers algorithmes de détection de contours dans des images bidimensionnelles introduits par Prewitt [129] revient à approximer le gradient par convolution de l'image avec des masques de convolution (les masques de convolution représentent la fonction discrète f définie dans la section 2.2.1 : "filtrage d'une image") :

$$L = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \qquad f_x = \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix}$$

$$f_y = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

Figure 2.8. *Masques de Prewitt.*

Dans le cas de l'opérateur de Prewitt, l'expression analytique de la réponse impulsionnelle de L n'étant pas dérivable, on dérive alors par différences finies.

Les premiers algorithmes de détection de contours par approximation du laplacien consistent à calculer le laplacien par convolution de l'image avec un masque (par exemple voir figure 2.9.). Il est important de remarquer que dans le cas des approches reposant sur le calcul de la dérivée première, le but du calcul du gradient est de déterminer la direction selon laquelle la variation locale des niveaux de gris est la plus forte (direction du gradient) ainsi que l'intensité de cette variation (norme du gradient). Dans le cas d'un signal continu (déri-

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} \qquad \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \begin{pmatrix} -1 & 0 & 0 & -1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ -1 & 0 & 0 & -1 \end{pmatrix}$$

Figure 2.9. *Masques d'approximation du laplacien.*

vable) , il suffit de dériver selon deux directions non colinéaires pour obtenir la norme et la direction du gradient invariants selon les directions de dérivation choisies. Ainsi, dans des images comprenant des points où se rejoignent plusieurs contours, et où le bruit est important, il est intéressant de calculer le gradient par une suite de dérivations directionnelles. Le principe de ces méthodes consiste à discrétiser l'espace des orientations possibles pour un contour, et à définir un masque de dérivation pour chaque direction. On sélectionne ensuite en chaque point l'orientation selon laquelle le gradient est maximum. La figure 2.10. présente les masques proposés par Kirsch [129]. Ces masques sont obtenus par rotation d'un masque de dérivation selon 8 orientations (la figure 2.10. présente les masques obtenus pour 4 orientations). Cependant, le coût élevé de l'implantation de ce type de méthodes et l'amélioration souvent assez ténue des résultats qu'elles permettent font qu'elles sont assez peu utilisées. Quoi qu'il en soit, ces algorithmes se révélent très efficaces dans les zones de l'image où plusieurs contours se croisent. Cependant, même lorsque l'on

$$N \begin{pmatrix} 5 & 5 & -3 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{pmatrix} \qquad N-W \begin{pmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{pmatrix}$$

$$W \begin{pmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{pmatrix} \qquad S-W \begin{pmatrix} 5 & 5 & -3 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{pmatrix}$$

Figure 2.10. *Masques de Kirsch correspondant aux directions* $N, N-W, W, S-W$.

choisit d'approximer le gradient par dérivation selon n directions (n : dimension de l'image, $n = 2$ dans le cas traité), des problèmes de coût algorithmique

se posent. En effet, par exemple en dimension 2 pour une image de dimensions $d_x \times d_y$, le coût d'une convolution $p \times p$ est $p^2 d_x d_y$. Si on approxime les dérivées partielles à l'aide d'un filtre s'implantant avec un masque de convolution $p \times p$ il en coûte : $2p^2 d_x d_y$. Ces remarques ont amené l'introduction de filtres à réponse impulsionnelle séparable selon les directions x, y :

$f(x, y) = f_1(x) f_2(y)$

Le filtrage séparable comporte plusieurs avantages :

- réduction du temps de calcul d'une convolution $p \times p$ de p^2 à $2p$ (pour le cas d'une image de dimension 2),
- possibilité de prise en compte d'un bruit de caractéristiques différentes selon chaque direction (par exemple dans certaines images le bruit est différent selon les deux axes X et Y),
- généralisation directe d'un filtre à une dimension quelconque,
- utilisation du filtrage récursif.

L'inconvénient majeur du filtrage séparable est que l'on peut déboucher sur des filtres anisotropiques selon les directions différentes de X, Y (l'anisotropie étant définie par rapport à la distance euclidienne). On provoque alors une légère délocalisation des contours fournis. On peut noter que les filtres gaussiens sont parmi les rares filtres séparables et isotropes (voir paragraphe 2.6.).

Dans le cas où f est un filtre séparable on obtient :

$$I * f(x, y) = I * (f_1(x) f_2(y)) \qquad [2.22]$$

Or la convolution d'un signal par le produit de deux réponses impulsionnelles séparables peut s'écrire comme la convolution par la deuxième réponse du signal convolué par la première réponse :

$$I * (f_1(x) f_2(y)) = (I * f_1(x)) * f_2(y)$$

f peut donc s'implanter par la cascade des filtres f_1 et f_2.

Si on suppose le bruit homogène selon toutes les directions on peut poser :

$$f_1 = f_2 = S$$

On supposera dans la suite que le bruit est isotrope, mais l'approche que nous décrirons s'adapte directement au cas d'un bruit anisotropique mais homogène

selon X, Y. On verra en effet que le filtrage récursif ne peut être utilisé que si le bruit est homogène selon chaque variété de dimension 1 : $x = cste$ ou $y = cste$.

On obtient :

$$I * f(x,y) = I * (S(x)S(y)) = I * S(x) * S(y)$$

On pose :

$$\begin{aligned} D &= S' \\ P &= S'' \end{aligned}$$

d'où, par exemple, si on note I_x et I_{xx} les dérivées première et seconde de l'image par rapport à x on obtient :

$$\begin{aligned} I_x &= \frac{\partial I(x,y)}{\partial x} = I * (D(x)S(y)) \\ I_{xx} &= \frac{\partial^2 I(x,y)}{\partial x^2} = I * (P(x)S(y)) \\ \Delta I &= I * (S(x)P(y) + S(y)P(x)) \end{aligned}$$

Exemple :
Le premier opérateur utilisant le filtrage séparable a été introduit par Sobel (voir figure 2.11.) [160] :

$$H1 = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \qquad H2 = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

Figure 2.11. *Masques de Sobel (à gauche gradient en X, à droite gradient en Y).*

Si on pose :

$$S = (1,2,1) \quad ; \quad D = (1,0,-1)$$

On obtient :

$$S(x) = (1,2,1) \quad ; \quad S(y) = (1,2,1)^t$$

$$D(x) = (1, 0, -1) \quad ; \quad D(y) = (1, 0, -1)^t$$

Soit :

$$H1 = D(x) * S(y) \quad ; \quad H2 = S(x) * D(y)$$

En effet :

$$\begin{aligned}
I_{D(x)*S(y)}(i,j) &= \sum_{i'=-1}^{i'=1} \sum_{j'=-1}^{j'=1} D(i-i')S(j-j')I(i',j') \\
&= I(i-1,j-1) + 2I(i-1,j) + I(i-1,j+1) \\
&- I(i+1,j-1) - 2I(i+1,j) - I(i+1,j+1) \\
&= I_{H_1}(i,j) \\
I_{S(x)*D(y)}(i,j) &= \sum_{i'=-1}^{i'=1} \sum_{j'=-1}^{j'=1} S(i-i')D(j-j')I(i',j') \\
&= I(i-1,j-1) + 2I(i,j-1) + I(i+1,j-1) \\
&- I(i-1,j+1) - 2I(i,j+1) - I(i+1,j+1) \\
&= I_{H_2}(i,j)
\end{aligned}$$

On a donc réduit le problème de la dérivation et du lissage d'un signal d'une dimension quelconque au cas monodimensionnel. On est donc ramené à la recherche de filtres de lissage 1D.

2.3. Du gradient ou du laplacien vers les points de contours

On remarque que le calcul du gradient ou du laplacien bien que constituant la partie essentielle de la détection des contours ne fournit pas directement les points de contour. En effet dans les deux cas deux étapes supplémentaires sont nécessaires. Nous décrivons brièvement ces traitements **communs** à tous les opérateurs avant d'introduire des opérateurs de filtrage 1D.

Soit $I(x, y)$ une image

Soit $G(x, y)$ le gradient de I au point (x, y) :

$$G(x, y) = (I_x(x, y), I_y(x, y))^t$$

Soit $\Delta(I)$ le laplacien de I au point (x, y)

2.3.1. Approche gradient

Dans le cas d'une approche dérivée première, on dispose donc de la valeur du gradient en tout point de l'image soit de la fonction G. Dans les premières approches [160] l'extraction des points de contour s'effectuait par sélection des points de norme de gradient élevée grâce aux deux étapes suivantes :

1. **calcul de la norme du gradient**

$$N(x,y) = (I_x(x,y)^2 + I_y(x,y)^2)^{1/2}$$

2. **sélection des points de fort gradient**

On détermine les points tels que :

$$N(x,y) > s$$

s : seuil fixé a priori

Dans le cas d'images où la norme du gradient aux points de contour varie fortement selon les parties de l'image cette méthode se révèle inefficace. En effet il n'existe alors pas de seuil s permettant d'obtenir les vrais points de contours sans sélectionner aussi ceux dus au bruit.

Un moyen de tourner cette difficulté est d'extraire non pas les points de norme de gradient élevée mais les extréma locaux de la norme du gradient. Une méthode efficace consiste à déterminer les maxima de la norme du gradient dans la direction du gradient [32]. Ces extréma correspondent aux passages par zéro de la dérivée de la norme du gradient dans la direction du gradient. Dans une deuxième étape on élimine les points de norme de gradient faible avec un seuillage par hystérésis [32]. Ce type de seuillage permet l'obtention de points de contour bien connectés entre eux. Il faut cependant noter qu'il utilise une propriété topologique : la connexité. On obtient donc les deux traitements suivants :

1. **extraction des extréma locaux du gradient**

Soit un point M de gradient $G(\vec{M})$ et d une distance seuil (par exemple $d = 1$). Soient M_1 et M_2 deux points de la droite passant par M et de vecteur directeur $G(\vec{M})$ situés à une distance d de M ; M_1 est pris dans le sens du gradient et M_2 dans le sens inverse.

On détermine une approximation du gradient aux points M_1 et M_2 par exemple par interpolation linéaire avec les points voisins.

Le point M est sélectionné si $N(M) > N(M_2)$ et $N(M) \geq N(M_1)$; le fait d'imposer que le maxima soit strict dans un sens revient à choisir si on localise le point de contour dans la zone de plus faible ou de plus forte valeur de la fonction des niveaux de gris. Cela revient à se déplacer le long de la normale au contour (approximée par le gradient), et à détecter le point de plus fort gradient.

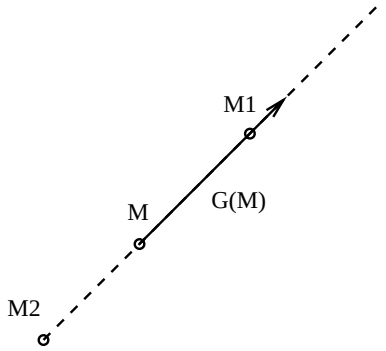


Figure 2.12. *Extraction des extréma locaux : cas 2D.*

On remarquera que les points obtenus vérifient l'équation $\nabla(\|\nabla I\|).\nabla I = 0$.

2. **seuillage par hystérésis des extréma**

Le principe du seuillage par hystérésis est de sélectionner parmi tous les extréma dont la norme du gradient est supérieure à un seuil bas s_b, ceux tels qu'il existe un chemin composé de points dont la norme du gradient est plus élevée que le seuil bas entre l'extremum considéré et un extremum de norme de gradient plus élevé qu'un seuil haut s_h. Le seuil haut s_h et le seuil bas s_b, peuvent par exemple être déterminés à partir de l'histogramme cumulé des valeurs de la norme du gradient aux extréma [129]. L'algorithme se décompose donc en deux étapes :

- détermination de deux images I_h et I_b telles que :

$$\begin{aligned} I_h(M) &= 0 \text{ si } N(M) < s_h \\ I_h(M) &= 1 \text{ si } N(M) \geq s_h \end{aligned}$$

$$I_b(M) = 0 \text{ si } N(M) < s_b$$
$$I_b(M) = 1 \text{ si } N(M) \geq s_b$$

- expansion en composantes connexes à partir de tous les points tels que $I_h(M) = 1$ sur tous les points tels que $I_b(M) = 1$. Cette étape revient à déterminer le graphe d'adjacence des points de I tels que $I_b(M) = 1$, puis à sélectionner les composantes connexes des nœuds tels que $I_h(M) = 1$.

Le seuillage par hystérésis peut être amélioré en effectuant une expansion en composantes connexes non pas dans toutes les directions, mais simplement dans la direction perpendiculaire au gradient (direction du contour). Ceci permet de diminuer la valeur du seuil bas sans introduire de contours parasites. Cette idée se révèle particulièrement intéressante pour la détection de contours 3D où une expansion dans toutes les directions marque parfois de faux points de contours [129]. Il est important de noter que ce type de traitement introduit de manière implicite des contraintes sur la morphologie des contours, et ne repose pas uniquement sur l'information du "signal image" comme les étapes précédentes.

2.3.2. Approche laplacien

Dans le cas d'une approche dérivée seconde, on dispose donc de la valeur du laplacien en chaque point de l'image soit de la fonction Δ. On considère que les points de contours sont localisés aux passages par zéro du laplacien. Si le calcul du laplacien était exact il suffirait de sélectionner les points M tels que $\Delta(M) = 0$. Mais comme généralement l'approximation du laplacien est assez bruitée, on détecte les points où il change de signe. Une dernière étape de seuillage est là encore nécessaire afin d'éliminer les points de trop faible gradient. L'extraction de ces passages par zéro s'effectue classiquement en trois étapes :

1. **détermination d'une image de polarité**

On calcule une image I_p telle que :

$$I_p(M) = 0 \text{ si } \Delta(M) > 0$$

$$I_p(M) \quad = \quad 1 \text{ si } \Delta(M) \leq 0$$

2. **détection des passages par zéro**

On calcule une image I_z telle que :

$I_z(M) = 1$ si M correspond à une transition 0-1 ou 1-0 dans I_p,

$I_z(M) = 0$ sinon.

On remarque que le choix de la localisation du passage par zéro au point de laplacien positif ou négatif revient, comme pour l'extraction des extréma locaux, à définir les points de contour dans la région la plus claire ou la plus foncée.

3. **seuillage des passages par zéro**

L'élimination des passages par zéro de faible norme de gradient peut s'effectuer par un algorithme de seuillage quelconque. L'algorithme de seuillage par hystérésis décrit pour l'approche dérivée première peut par exemple être utilisé. On peut aussi se servir du fait que les passages par zéro extraits définissent des lignes fermées délimitant les régions de points connexes où le laplacien est positif ou négatif. Des méthodes reposant sur le suivi de ces frontières et sur un calcul local du gradient peuvent aussi être utilisées [165].

Il faut noter que la méthodologie que nous avons développée dans cette partie, tant pour le filtrage que les traitements complémentaires, s'applique à l'extraction de contours dans des images de dimensions quelconques. Cette approche nous a donc permis de ramener ce problème à celui du lissage d'un signal 1D bruité. La partie suivante est donc consacrée au filtrage 1D . Ensuite nous décrirons les algorithmes que nous obtenons dans le cas 2D.

2.4. Mise en oeuvre du filtrage 1D

Nous présentons dans cette partie des critères de performance pour un filtre de lissage 1D, et le filtre de dérivation correspondant. Nous déduisons de ces critères des filtres optimaux. Il faut cependant noter que des liens de causalité existant entre les filtres et les critères d'optimalité qu'ils vérifient peuvent être vus dans les deux sens. Nous montrons enfin que le filtrage récursif amène, pour certains de ces filtres, des implantations de faible complexité algorithmique.

2.4.1. Modèles de contours

Une première étape est de spécifier le type de contours auxquels on s'intéresse ainsi que les caractéristiques du bruit.

- *type de contours*

Plusieurs types de contour existent dans une image. Horn en a définit une classification : marche, arête, toit ...

- *bruit*

Des bruits de caractéristiques diverses sont présents dans les images : bruit convolutif, bruit additif, bruit blanc, bruit impulsionnel ...

- *modèle classique de contours*

Les contours de type "marche" étant très fréquents dans les images et le bruit blanc facile à modéliser, la plupart des méthodes se focalisent sur la détection de contours marche avec un bruit blanc additif. Néanmoins, cette approche s'applique pour d'autres types de contours.

Soit A l'amplitude de la marche et η_0^2 la variance du bruit blanc. Le signal d'entrée $I(x)$ peut être représenté par :

$$I(x) = Au_{-1}(x) + \eta(x)$$

avec

$$\eta_0^2 = E[\eta^2(x)]$$

$$u_{-1}(x) = \begin{cases} 0 & \text{pour x dans}[-\infty, 0] \\ 1 & \text{pour x dans}]0, +\infty] \end{cases}$$

2.4.2. Critères de performances

Il est généralement difficile d'apprécier le résultat d'une détection de contours et ceci pour plusieurs raisons :

- les résultats sont assez souvent estimés à "l'œil", ce qui ne fournit pas de jugement objectif.

- les images traitées sont en général en nombre et en type assez limités.

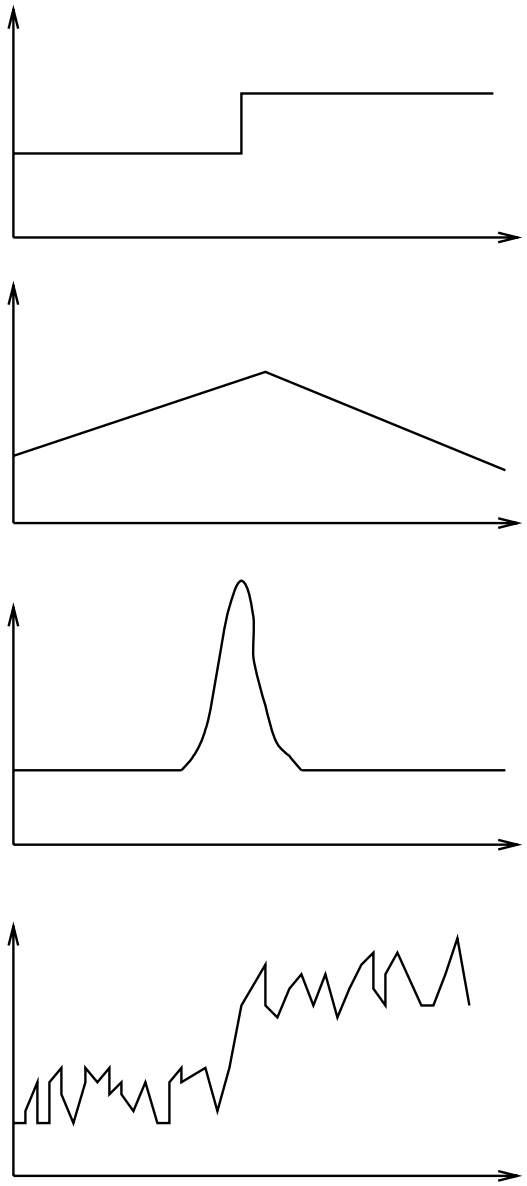


Figure 2.13. *Différents types de contours : contour idéal, contour en toit, contour en pointe, contour idéal bruité.*

Aussi, il est nécessaire de définir au préalable des critères de performance d'un détecteur utilisant évidemment le modèle de contour et le modèle de bruit choisis. Les performances d'un détecteur se caractérisent essentiellement par :

- *détection* : l'opérateur doit donner une réponse au voisinage d'un contour,
- *localisation* : le contour doit être localisé avec précision,
- *réponse unique* : un contour doit provoquer une seule réponse de l'opérateur d'extraction.

On suppose classiquement que la détection est effectuée en convoluant le contour bruité avec une fonction spatiale antisymétrique $f(x)$ et que l'on marque les contours avec des maxima de la sortie $\theta(x_0)$ de cette convolution :

$$\theta(x_0) = \int_{-\infty}^{+\infty} I(x) f(x_0 - x) dx$$

A partir de cette modélisation Canny [32] définit des critères évaluant :

- *la détection des points de contours* : faible probabilité de ne pas détecter un vrai point de contour, et faible probabilité de marquer de faux points de contours ; ce critère correspond à maximiser le rapport signal sur bruit RSB (sortie du filtre au point de discontinuité pour le contour marche / écart type de la réponse du filtre du bruit)

$$RSB = \frac{A}{\eta_0} \frac{\int_{-\infty}^{0} f(x) dx}{(\int_{-\infty}^{+\infty} f^2(x) dx)^{\frac{1}{2}}} \qquad [2.23]$$

Si on note

$$\sigma = \frac{\int_{-\infty}^{0} f(x) dx}{(\int_{-\infty}^{+\infty} f^2(x) dx)^{\frac{1}{2}}} \qquad [2.24]$$

on obtient :

$$RSB = \frac{A}{\eta_0} \sigma \qquad [2.25]$$

- *la localisation des points de contours* : les points marqués comme contours par le détecteur doivent être aussi près que possible du centre du contour véritable. Ce critère correspond à maximiser l'écart type de la position des

passages par zéro (voir 2.6. pour le calcul détaillé), et correspond à l'inverse de l'espérance de la distance entre le vrai point de contour, et le point de contour détecté (maximum de la sortie de l'opérateur).

$$L = \frac{A}{\eta_0} \frac{\mid f'(0) \mid}{(\int_{-\infty}^{+\infty} f'^2(x)dx)^{\frac{1}{2}}} \qquad [2.26]$$

Si on note

$$\lambda = \frac{\mid f'(0) \mid}{(\int_{-\infty}^{+\infty} f'^2(x)dx)^{\frac{1}{2}}} \qquad [2.27]$$

on obtient :

$$L = \frac{A}{\eta_0}\lambda \qquad [2.28]$$

• une réponse à un contour : le détecteur ne doit pas fournir de multiples réponses à un seul contour ; la distance moyenne entre les pics dans la réponse au bruit de f, noté $xmax$, est posée comme une fraction k de la largeur de l'opérateur W (la largeur d'un opérateur est la taille du masque de convolution nécessaire pour l'implanter). Cette expression ($xmax$) se calcule aisément car on connait la distance moyenne entre les passages par zéro de la dérivée seconde de la réponse d'un filtre à un bruit blanc gaussien (Rice 1944).

$$x_{max} = kW = 2\Pi \left(\frac{\int_{-\infty}^{+\infty} f'^2(x)dx}{\int_{-\infty}^{+\infty} f''^2(x)dx} \right)^{\frac{1}{2}} \qquad [2.29]$$

On notera que l'introduction de ce troisième critère est dû au fait que le critère de détection ne prend en compte que la réponse du filtre au point de contour et non dans son voisinage. Ce critère est le seul utilisant l'hypothèse d'un bruit gaussien.

Les critères de détection et de localisation étant antinomiques, on les combine d'une manière significative en maximisant le produit $\sigma\lambda$ sous la contrainte du 3ème critère. (d'autres solutions sont envisageables, par exemple maximiser le produit des trois termes...). On obtient ainsi une équation différentielle dont la solution est :

$$\begin{aligned} f(x) &= a_1 e^{\alpha x} \sin \omega x + a_2 e^{\alpha x} \cos \omega x \\ &+ a_3 e^{-\alpha x} \sin \omega x + a_4 e^{-\alpha x} \cos \omega x + c \end{aligned} \qquad [2.30]$$

avec les conditions initiales :

$$f(0) = 0 \quad f(W) = 0 \quad f'(0) = S \quad f'(-W) = 0$$

W étant la taille du filtre défini dans $[0, W]$.
Si on fait tendre W vers $+\infty$ on obtient

$$a_1 = a_2 = a_3 = c = 0$$

Soit le filtre optimal devient :

$$f(x) = a_3 e^{\alpha x} \sin wx$$

Le filtre étant impair on obtient pour le filtre de dérivation optimal :

$$f(x) = a_3 e^{\alpha |x|} \sin wx \qquad [2.31]$$

α définit la largeur du filtre soit le compromis détection-localisation désiré (la localisation est une fonction croissante de α et la détection une fonction décroissante de α).

Shen et Castan [165] proposent de déterminer les opérateurs de lissage optimisant un critère incluant la détection et la localisation. Les critères qu'ils obtiennent correspondent aux critères de détection et de localisation de Canny et les filtres obtenus assez similaires dans la pratique.

2.4.3. Filtres optimaux de lissage/dérivation

Les filtres de dérivation correspondant aux filtres de lissage sont obtenus simplement par dérivation des réponses impulsionnelles et renormalisation. Il est important de remarquer que l'utilisation de filtres de dérivation pour la détection des contours revient à supposer que les contours correspondent aux discontinuités d'ordre 0. Si on s'attachait à la recherche des contours en toit, il faudrait mettre en œuvre des opérateurs de détection des discontinuités d'ordre 1 (celà revient à déterminer la courbure du signal).

1. **filtre Shen-Castan**

Le filtre de lissage Shen-Castan (voir figure 2.14) s'écrit :

$$s_1(x) = ce^{-\alpha . |x|} \qquad [2.32]$$

c est choisi de manière à obtenir un filtre normalisé (dans l'espace discret) soit :

$$\sum_{-\infty}^{+\infty} s_1(n) = 1$$

d'où :

$$c = \frac{1 - e^{-\alpha}}{1 + e^{-\alpha}}$$

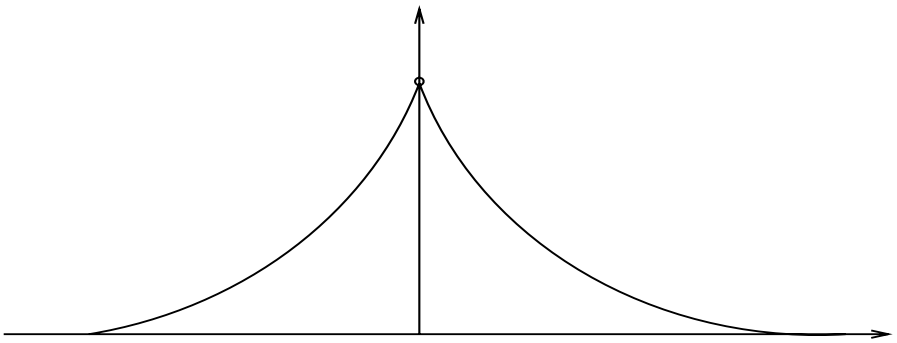


Figure 2.14. *Réponse impulsionnelle de s_1.*

le filtre de dérivation correspondant (d_1) est obtenu par normalisation de la dérivée du filtre de lissage.

$$d_1(x) = \begin{cases} d.e^{-\alpha|x|} \text{ si } x \geq 0 \\ -d.e^{-\alpha|x|} \text{ si } x \leq 0 \end{cases}$$

d est choisi de manière à obtenir un filtre normalisé dans l'espace discret soit :

$$\sum_{0}^{+\infty} d_1(n) = 1$$

d'où

$$d = 1 - e^{-\alpha}$$

Le paramètre α définit la "largeur" du filtre ; plus α est petit, plus le lissage effectué par le filtre est important.

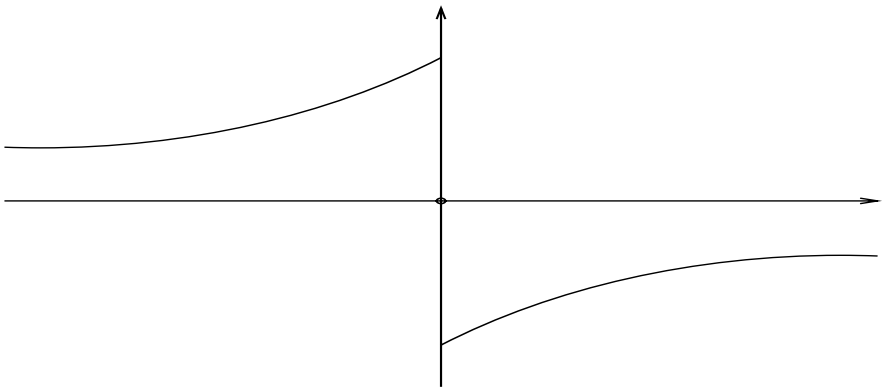


Figure 2.15. *Réponse impulsionnelle de d_1.*

Ce filtre a été introduit par Shen et Castan pour l'approximation du laplacien par différence entre l'image originale et l'image lissée [165]. Le filtre obtenu par dérivation constitue une solution optimale pour la première partie des critères de Canny (voir équations [2.23] et [2.26]). Il correspond à une valeur optimale pour le produit $\sigma\lambda$, soit au meilleur compromis détection-localisation. La discontinuité d'ordre 1 au point 0 permet d'éviter une délocalisation importante des contours dans l'image lissée même avec des valeurs faibles de α. Cependant cette discontinuité peut entrainer la détection de contours multiples [32].

2. **filtre moyenne** Le filtre moyenne s'écrit :

$$s_2(x) = \begin{cases} \frac{1}{(2n+1)} \text{ si } x \text{ dans } [-n, n] \\ 0 \text{ sinon} \end{cases}$$

Dans ce cas, le filtre dérivé d_2 est approximé par :

$$d_2(x) = \begin{cases} \frac{-1}{(2n+1)} \text{pour } x \text{ } dans \text{ } [n, 0] \\ \frac{1}{(2n+1)} \text{ pour } x \text{ dans }]0, n] \\ 0 \text{ sinon} \end{cases} \qquad [2.33]$$

Ce filtre est le plus simple à mettre en œuvre et est utilisé depuis longtemps. Il peut par exemple être mis en œuvre pour approximer le laplacien par différence entre les images obtenues par lissage avec deux filtres de largeurs différentes [104]. Il est équivalent au filtre s_1 lorsque α tend vers zéro. Il est

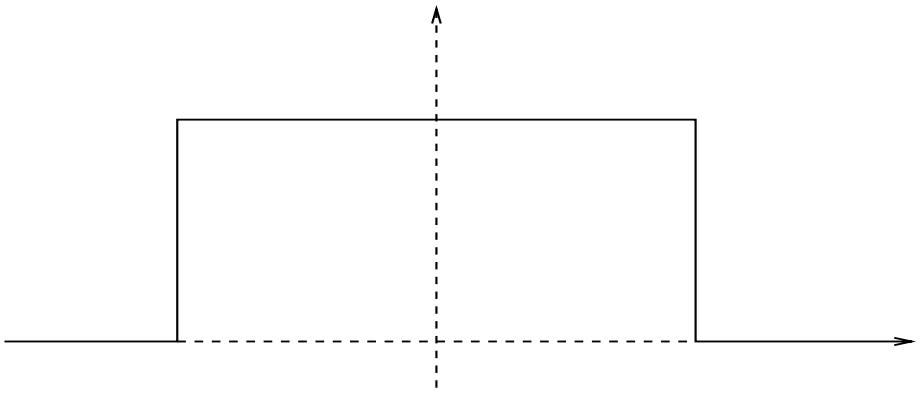


Figure 2.16. *Réponse impulsionnelle de s_2.*

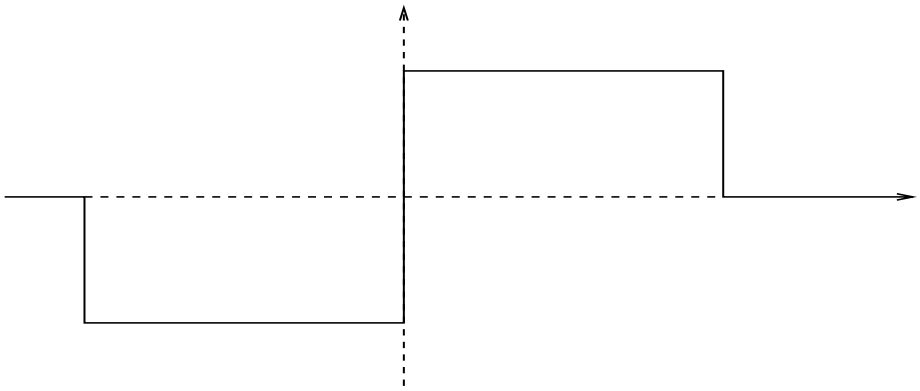


Figure 2.17. *Réponse impulsionnelle de d_2.*

recommandé de l'utiliser avec des valeurs du paramètre α faibles par rapport à la distance minimale entre deux contours, car sinon la dégradation des contours dans l'image lissée est importante.

3. **filtre gaussien** Le filtre gaussien s'écrit :

$$s_3(x) = ce^{-\frac{x^2}{2\sigma^2}}$$

c est choisi de manière à obtenir un filtre normalisé.
Le filtre de dérivation correspondant s'écrit :

$$d_3(x) = c\frac{-x}{\sigma^2}e^{-\frac{x^2}{2\sigma^2}} \qquad [2.34]$$

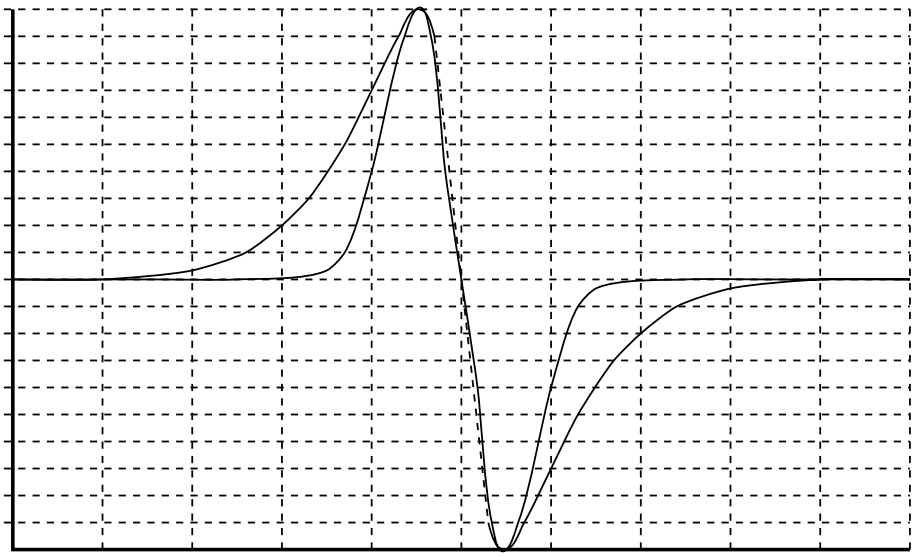


Figure 2.18. *Réponses impulsionnelles de d_3 et de d_4 ($d_4 \leq d_3$).*

Ce filtre a été introduit pour la détection de contours par Marr et Hildreth pour le calcul du laplacien [122]. Il correspond à la valeur optimale pour le produit de la variance temporelle et de la variance spectrale. Le filtre de dérivation correspondant $d_3(x)$ est une solution approchée à l'équation de Canny. En effet par optimisation numérique Canny trouve que la plus grande valeur de k qui peut être obtenue est 0.58 et que sa performance est donnée par $\sigma\lambda = 1.12$ (voir équations [2.23], [2.26], [2.29]). Il propose l'utilisation de la première dérivée d'une gaussienne pour laquelle : $\sigma\lambda = 0.92$ et $k = 0.51$ (voir équation [2.34]).

4. **filtre Deriche** Le filtre Deriche s'écrit :

$$s_4(x) = k(\alpha \mid x \mid +1)e^{-\alpha|x|}$$

k est choisi de manière à obtenir un filtre discret normalisé soit :

$$\sum_{-\infty}^{+\infty} s_4(x) = 1 \Longleftrightarrow k = \frac{(1-e^{-\alpha})^2}{1+2\alpha e^{-\alpha} - e^{-2\alpha}}$$

Ce filtre a été proposé récemment par R. Deriche [43] et sa dérivée est une solution exacte à l'équation de Canny étendue à des filtres infinis. Il est obtenu en étendant l'équation différentielle [2.30] aux opérateurs infinis et antisymétriques et constitue ainsi une solution exacte à l'équation de Canny.

En effet, soit $f(x)$ le filtre solution de l'équation de Canny étendue aux filtres à réponses infinies :

$$f(x) = -de^{-\alpha|x|}\sin wx$$

α, ω, c réels positifs.

Si on pose :

$$\alpha = m\omega$$

on obtient :

(a)	$m >> 1$	$\lambda = (2\alpha)^{\frac{1}{2}}$	$\sigma \approx (\frac{2}{\alpha})^{\frac{1}{2}}$	$\sigma\lambda \approx 2$	$k \approx .44$
(b)	$m << 1$	$\lambda = (2\alpha)^{\frac{1}{2}}$	$\sigma \approx \frac{\lambda}{m}$	$\sigma\lambda \approx 2m$	$k \approx 1$
(c)	$m = 1$	$\lambda = (2\alpha)^{\frac{1}{2}}$	$\sigma = (\frac{1}{\alpha})^{\frac{1}{2}}$	$\sigma\lambda = 1.414$	$k = .58$
(d)	$m = (3)^{\frac{1}{2}}$	$\lambda = (2\alpha)^{\frac{1}{2}}$	$\sigma = (\frac{3}{2\alpha})^{\frac{1}{2}}$	$\sigma\lambda = 1.732$	$k = .5$

Le cas (d) montre que pour une même valeur de k la performance de la première dérivée d'une gaussienne est moins bonne que celle de l'opérateur f de plus de 90 %.

Le cas (c) montre que la forme finale utilisée par Canny pour son opérateur optimal (la dérivée d'une gaussienne, $\sigma\lambda = 1.112, k = .58$) est moins bonne que l'opérateur f de plus de 25 %.

Le cas (b) présente une réponse idéale pour k mais le produit $\sigma\lambda$ devrait être beaucoup plus petit que l'unité.
Le cas (a) présente le meilleur compromis. On peut alors écrire ce filtre optimal de dérivation comme :

$$\begin{aligned} d_4(x) &= -kxe^{-\alpha|x|} \\ k &= \frac{(1-e^{-\alpha})^2}{e^{-\alpha}} \end{aligned}$$

Le filtre de lissage s'obtient par intégration du filtre de dérivation.

Les filtres de dérivée seconde s'obtiennent en dérivant deux fois les filtres de lissage correspondants (ou une fois les filtres de dérivation). Ils permettent d'approximer la dérivée seconde du signal. Nous remarquerons que pour le filtre moyenne et le filtre Shen il est nécessaire d'effectuer une approximation à cause de la discontinuité au point 0. De toute manière, même s'il n'existe pas d'expression analytique des réponses impulsionnelles, il est toujours possible de les implanter par cascade de deux filtres de dérivation. Les filtres de dérivée seconde et troisième de Deriche sont décrits dans le chapitre "Reconstruction et modélisation de surfaces" (partie : "Des dérivées partielles des images 3D aux courbures des surfaces").

2.4.4. Récursivité et implantation des filtres

La détermination de filtres ayant des propriétés intéressantes pour la détection de contours répond à la question : Que calculer (voir introduction). Maintenant il nous faut trouver avec quels algorithmes. En d'autres termes, comment implanter efficacement la convolution d'un signal 1D par une réponse impulsionnelle infinie.

L'implantation classique consiste à tronquer les réponses de manière à ne conserver que les cœfficients significatifs (les filtres de lissage et de dérivation première ou seconde considérés décroissent vers 0 à l'infini). Ensuite on procède par implantation directe d'un produit de convolution. On obtient ainsi par exemple pour une gaussienne d'écart type σ un masque de convolution de taille 8σ, et pour un filtre Shen ou un filtre Deriche de paramètre α un masque de taille $\frac{8}{\alpha}$. Pour des étendues de filtres assez importantes on obtient donc des complexités prohibitives et ceci d'autant plus que la dimension de l'image est

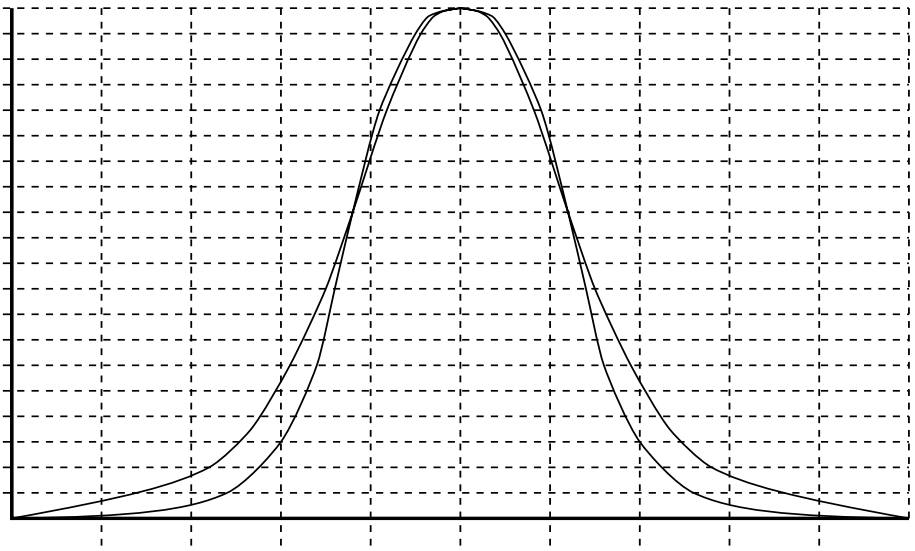


Figure 2.19. *Réponses impulsionnelles de s_3 et de s_4 ($s_4 \leq s_3$).*

importante. De plus la troncature des réponses introduit un effet "cut off" qui nuit à la qualité des contours détectés.

Une solution pour éviter ce compromis entre la taille des masques et la qualité des résultats consiste à utiliser le filtrage récursif. Le filtrage récursif est une technique bien connue en traitement du signal et a été introduite récemment en vision artificielle pour la détection des contours [43, 165]. elle permet d'implanter des filtres de réponse impulsionnelle infinie avec un coût faible. Il se trouve que le filtre Shen et le filtre Deriche admettent des réalisations récursives respectivement d'ordres 1 et 2. Le filtre gaussien peut être approximé par des filtres récursifs d'ordre 4 [42]. Dans cette partie nous montrerons comment obtenir la mise en œuvre récursive des filtres de Canny, Deriche et Shen.

Nous allons montrer, par exemple, que le filtre optimal de Canny ($f(x)$) se réalise par un filtrage récursif d'ordre 2 [43].

$$f(x) = -ce^{-\alpha|x|} \sin wx$$

Soit $f(n)$ la fonction d'échantillonnage de $f(x)$ et $F(Z)$ sa transformée en

Z :

$$F(Z) = \sum_{-\infty}^{+\infty} f(n) Z^{-n}$$

On réécrit $f(n)$ comme :

$$\begin{aligned}
f(n) &= f_-(n) + f_+(n) \quad \text{où} \\
f_-(n) &= \begin{cases} 0 & n \geq 0 \\ \alpha_1 (p_2)^n + \alpha_1^* (p_2^*)^n & n < 0 \end{cases} \\
f_+(n) &= \begin{cases} \alpha_1 (p_1)^n + \alpha_2^* (p_1^*)^n & n \geq 0 \\ 0 & n < 0 \end{cases}
\end{aligned}$$

$$\begin{aligned}
\alpha_1 &= \tfrac{-c}{2ip_1} = e^{-\alpha + i\omega} \qquad p_2 = e^{\alpha + i\omega} \\
\alpha^* &: \text{conjugué de } \alpha
\end{aligned}$$

Les équations précédentes reviennent à séparer la réponse en une partie positive (f_+) et une partie négative (f_-). En utilisant la transformée en Z, on reconnaît que chaque facteur $\alpha_i (p_i)^n$ a une transformée en Z égale à $\frac{\alpha_i}{(1 - p_i Z^{-1})}$ et on obtient :

$$\begin{aligned}
F(Z) &= F_-(Z) + F_+(Z^{-1}) \quad \text{où} \\
F_+(Z^{-1}) &= \frac{a Z^{-1}}{(1 + b_1 Z^{-1} + b_2 \times Z^{-2})} \\
F_-(Z) &= \frac{-aZ}{(1 + b_1 Z + b_2 Z^2)}
\end{aligned}$$

avec :

$$\begin{aligned}
a &= -c.e^{-\alpha} \sin\omega \\
b1 &= -2.e^{-\alpha} \cos\omega \\
b2 &= e^{-2\alpha}
\end{aligned}$$

Toutes les singularités de $F_+(Z^{-1})$ (resp. $F_-(Z)$) sont à l'intérieur (resp. à l'extérieur) du cercle unité pour une valeur de α positive, ces deux transformées en Z correspondent donc à deux fonctions de transfert rationnelles de filtres stables du second ordre récursifs de la gauche vers la droite (F_+) et de la droite vers la gauche (F_-).

En particulier la séquence de sortie $y(m)$ en réponse à l'entrée $x(m)$ pour un système de réponse impulsionnelle $f(n)$ peut être obtenue récursivement de la manière suivante :

$$\begin{aligned} y^+(m) &= ax(m-1) - b_1 y^+(m-1) - b_2 y^+(m-2) \qquad m = 1, \ldots, N \\ y^-(m) &= -ax(m+1) - b_1 y^-(m+1) - b_2 y^-(m+2) \qquad m = N, \ldots, 1 \\ y(m) &= y^+(m) + y^-(m) \qquad m = 1, \ldots, N \end{aligned}$$

Les cœfficients de la récursion se déduisent de la transformée en Z de la manière suivante :

$$F(Z) = \frac{a_0 + a_1 Z^{-1} + \ldots\ldots a_{n-1} Z^{-(n-1)}}{1 + b_1 Z^{-1} + \ldots.. b_n Z^{-n}} \text{ si } |z| > 1$$

caractérise un système récursif causal d'ordre n de réalisation :

$$y(i) = \sum_{l=0}^{l=n-1} a_l x(i-l) - \sum_{k=1}^{k=n} b_k y(i-k)$$

$$F(Z) = \frac{a_0 + a_1 Z^{1} + \ldots\ldots a_{n-1} Z^{(n-1)}}{1 + b_1 Z^{1} + \ldots.. b_n Z^{n}} \text{ si } |z| < 1$$

caractérise un système récursif anticausal d'ordre n de réalisation

$$y(i) = \sum_{l=0}^{l=n-1} a_l x(i+l) - \sum_{k=1}^{k=n} b_k y(i+k)$$

La récursivité du filtre de Canny permet une réalisation qui requiert seulement 6 multiplications et 5 additions par point, et cela avec un paramètre de filtre (α) quelconque.

En appliquant la même technique on peut montrer que l'échantillonnage h de l'intégrale $h(x)$ du filtre précédent $f(x)$ s'implante aussi récursivement .

En effectuant le même type de calcul on montre que les filtres de lissage et de dérivation de Shen s'implantent par filtrage récursif d'ordre 1. s_1 (filtre de lissage Shen) est un filtre récursif du premier ordre de réalisation [165] :

$$s_1(x) = ce^{-\alpha.|x|}$$

$$\begin{aligned} y^+(m) &= ax(m) + by^+(m-1) \\ &\quad \text{pour } m = 1, \ldots, N \\ y^-(m) &= abx(m+1) + by^-(m+1) \\ &\quad \text{pour } m = N, \ldots, 1 \\ y(m) &= y^-(m) + y^+(m) \\ &\quad \text{pour } m = 1, \ldots, N \end{aligned}$$

avec

$$a = c = \frac{1 - e^{-\alpha}}{1 + e^{-\alpha}} \; ; \; b = e^{-\alpha}$$

d_1 est un filtre récursif du premier ordre de réalisation :

$$d_1(x) = \begin{cases} d.e^{-\alpha|x|} & \text{si } x \geq 0 \\ -d.e^{-\alpha|x|} & \text{si } x \leq 0 \end{cases}$$

$$\begin{aligned} y^+(m) = &\quad dx(m) + ey^+(m-1) &&\quad \text{pour } m = 1, \ldots, N \\ y^-(m) = &\quad -dx(m) + ey^-(m+1) &&\quad \text{pour } m = N, \ldots, 1 \\ y(m) = &\quad y^-(m) + y^+(m) &&\quad \text{pour } m = 1, \ldots, N \end{aligned}$$

avec

$$d = e = 1 - e^{-\alpha} \; ; \; f = e^{-\alpha}$$

s_4 (filtre de lissage Deriche) est un filtre récursif du second ordre de réalisation :

$$s_4(x) = k(\alpha \mid x \mid +1)e^{-\alpha|x|}$$

$$\begin{aligned} y^+(m) &= a_0 x(m) + a_1 x(m-1) - b_1 y^+(m-1) - b_2 y^+(m-2) \\ &\quad \text{pour } m = 1, \ldots, N \\ y^-(m) &= a_2 x(m+1) + a_3 x(m+2) - b_1 y^-(m+1) - b_2 y^-(m+2) \\ &\quad \text{pour } m = N, \ldots, 1 \\ y(m) &= y^-(m) + y^+(m) \\ &\quad \text{pour } m = 1, \ldots, N \end{aligned}$$

avec

$$a_0 = k \ ; \ a_1 = k(\alpha - 1)e^{-\alpha} \ ; \ a_2 = k(\alpha + 1)e^{-\alpha} \ ; \ a_3 = -ke^{-2\alpha} \ ;$$

$$b_1 = -2e^{-\alpha} \ ; \ b_2 = e^{-2\alpha} \ ; \ k = \frac{(1 - e^{-\alpha})^2}{1 + 2\alpha e^{-\alpha} - e^{-2\alpha}}$$

d_4 (filtre de dérivation Deriche) est un filtre récursif du second ordre de réalisation :

$$d_4(x) = -cxe^{-\alpha|x|}$$

$$\begin{array}{lcll} y^+(m) = & ax(m-1) - b_1 y^+(m-1) - b_2 y^+(m-2) & \text{pour } m = 1, \ldots, N \\ y^-(m) = & -ax(m+1) - b_1 y^-(m+1) - b_2 y^-(m+2) & \text{pour } m = N, \ldots, 1 \\ y(m) = & y^-(m) + y^+(m) & \text{pour } m = 1, \ldots, N \end{array}$$

avec

$$a = ce^{-\alpha} \ ; \ b_1 = -2e^{-\alpha} \ ; \ b_2 = e^{-2\alpha}$$

$$c = \frac{(1 - e^{-\alpha})^2}{e^{-\alpha}}$$

l_4 (filtre dérivée seconde Deriche) est un filtre récursif du second ordre de réalisation :

$$l_4(x) = c_2(1 - c_3\alpha \mid x \mid)e^{-\alpha|x|}$$

$$\begin{array}{rcl} y^+(m) & = & a_0 x(m) + a_1 x(m-1) - b_1 y^+(m-1) - b_2 y^+(m-2) \\ & & \text{pour } m = 1, \ldots, N \\ y^-(m) & = & a_2 x(m+1) + a_3 x(m+2) - b_1 y^-(m+1) - b_2 y^-(m+2) \\ & & \text{pour } m = N, \ldots, 1 \\ y(m) & = & y^-(m) + y^+(m) \\ & & \text{pour } m = 1, \ldots, N \end{array}$$

avec

$$a_0 = -1 \ ; \ a_1 = (1 + k\alpha)e^{-\alpha} \ ; \ a_2 = -(1 - k\alpha)e^{-\alpha} \ ; \ a_3 = -e^{-2\alpha} \ ;$$

$$b_1 = -2e^{-\alpha} \ ; \ b_2 = e^{-2\alpha} \ ; \ k = \frac{1 - e^{-2\alpha}}{2\alpha e^{-\alpha}}$$

2.5. Résumé : algorithmes de calcul du gradient et du laplacien

Nous présentons dans cette section les différentes étapes des algorithmes de détection de contours issus de la méthodologie précédemment décrite.

1. **choix d'un filtre de lissage 1D :** $s(x)$

On recommande de choisir un filtre qui s'implante récursivement soit par exemple :

$$s(x) = s_4(x) = k(\alpha \mid x \mid +1)e^{-\alpha|x|} \quad \text{filtre de lissage Deriche}$$

où

$$s(x) = s_1(x) = ce^{-\alpha|x|} \quad \text{filtre de lissage Shen}$$

Théoriquement le filtre de dérivation $s_4'(x)$ est meilleur par rapport au critère de réponse multiple, mais $s_1'(x)$ présente le meilleur compromis détection-localisation. Pour des α petits la forme de $s_4'(x)$ induit des problèmes de délocalisation des contours comme pour la plupart des filtres continus au point 0 [160]. Néanmoins, sur une grande variété d'images les résultats obtenus sont comparables. On notera que les deux filtres ne sont pas de la même espèce (discontinu ou continu au point 0) donc ne sont pas théoriquement comparables.

2. **choix du type d'approche : gradient, laplacien** En général, on constate que le calcul de la dérivée seconde est plus sensible au bruit. D'un autre côté, la complexité du calcul du gradient est généralement plus élevée que pour le laplacien. Ceci est dû aux simplifications qui apparaissent lors du calcul de la réponse du filtre d'approximation du laplacien par multiplication et addition de filtres de lissage et de dérivée seconde. Il faut cependant noter que le plus souvent une étape de seuillage des passages par zéro du laplacien est utile, et qu'elle nécessite le calcul de la norme du gradient (aux points de passage par zéro). La localisation des contours obtenus est pratiquement la même. On remarque que l'approche laplacien a tendance à arrondir les angles tout en maintenant une localisation exacte pour les coins. On choisira donc plutôt une approche gradient si des angles aigus sont présents dans les images (scènes d'intérieur par exemple), et plutôt une approche laplacien dans le cas de contours à faible variation de courbure.

3. **détermination d'une cascade de filtres pour approximer le gradient ou le laplacien**

Une fois que l'on a choisi le type d'approche : gradient ou laplacien, il nous faut déterminer un moyen de les calculer.

- **calcul du gradient**

  Détermination d'une implantation des filtres 1D de réponses : $s(x)$ (lissage) et $d(x)$ (dérivation). On choisira de préférence des implantations récursives. Des exemples sont donnés dans la section précédente.

  Soit $I(x, y)$ une image,

  Soit $G(x, y)$ le gradient de I

  $$G(x,y) = (\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})^t = (I_x, I_y)^t$$

  Le calcul des composantes du gradient $\frac{dI}{dx_i}$ s'effectue en calculant les images correspondant aux dérivées partielles des points par rapport à x et à y par filtrage séparable récursif :

  $$I_x(x,y) = (I * s(y)) * d(x)$$

  $$I_y(x,y) = (I * s(x)) * d(y)$$

  Pour une image de dimension d^2 le calcul de chaque composante du gradient nécessite donc le calcul de 2 convolutions par point ; on obtient pour le calcul total du gradient $2d^2$ convolutions par point. Si on utilise l'implantation directe d'un masque de convolution (1D) de taille k on obtient une complexité de $2d^2k$. Le filtrage récursif d'ordre r permet d'obtenir une complexité de l'ordre de $2d^2r$. Par exemple pour le filtre Deriche on obtient 13 multiplications et 12 additions par points et ceci quelle que soit la valeur de α. Si on utilise un filtre récursif le calcul de $I * s(x)$ s'effectue en filtrant successivement les droites d'équation $y = cste$.

- **calcul du laplacien**

  On peut le calculer en déterminant d'abord les dérivées secondes puis en les additionnant. Le calcul des dérivées secondes s'effectue avec la structure algorithmique précédente dans laquelle on remplace l'opérateur de dérivée première $d(x)$ par l'opérateur de dérivée seconde $l(x)$. Cependant, dans certains cas, la réponse impulsionnelle du filtre permettant d'approximer le laplacien L se simplifie et se réécrit sous une forme séparable. On obtient alors :

$$L(x,y) \quad = \quad s(x)s^{''}(y) + s^{''}(x)s(y)$$

4. **extraction des extréma locaux du gradient ou des passages par zéro du laplacien**

Bien que ne constituant pas la partie essentielle de la détection des contours cette étape doit être effectuée avec attention car elle conditionne fortement le résultat final. La méthode proposée dans la section précédente permet d'obtenir des résultats satisfaisants pour des images 2D [43] ou 3D [128].

5. **seuillage des extréma locaux du gradient ou des passages par zéro du laplacien**

Le plus souvent les filtres utilisés ne suffisent pas à éliminer complètement le bruit présent dans les images. On obtient alors des extréma locaux ou des passages par zéro du laplacien qui ne correspondent pas à de vrais points de contour. Fréquemment ces points parasites ont une norme de gradient faible. Ainsi un seuillage selon une estimation de la norme du gradient permet le plus souvent d'éliminer la plupart de ces faux points de contour. L'algorithme de seuillage par hystérésis décrit précédemment semble être une bonne réponse à ce problème de seuillage.

2.6. Analyse des performances des filtres

On traitera dans cette section le cas 3D afin de montrer l'applicabilité à une dimension quelconque des développements précédents et d'expliciter dans un cas particulier le calcul des critères de Canny. On remarquera que ces calculs sont réalisés avec des filtres normalisés dans le domaine continu.

2.6.1. Introduction

Le filtrage séparable permet de dériver des opérateurs d'approximation du gradient ou du laplacien 2D ou 3D (mêmes développements que précédemment en rajoutant une dimension) à partir d'opérateurs 1D optimaux au sens des critères de Canny (voir partie 1.2.5). Dans cette partie nous nous intéressons au calcul des performances des filtres 2D ou 3D correspondant pour le filtre

Deriche (voir partie 1.2.5). Nous verrons que l'anisotropie des réponses impulsionnelles obtenues nécessite la prise en compte de l'orientation du contour pour le calcul des performances. Nous remarquons que dans le cas du filtre gaussien, l'isotropie de la réponse impulsionnelle fait que les performances du filtre 1D sont les performances des filtres 2D, 3D à une constante multiplicative près.

2.6.2. Modèle de contour 2D

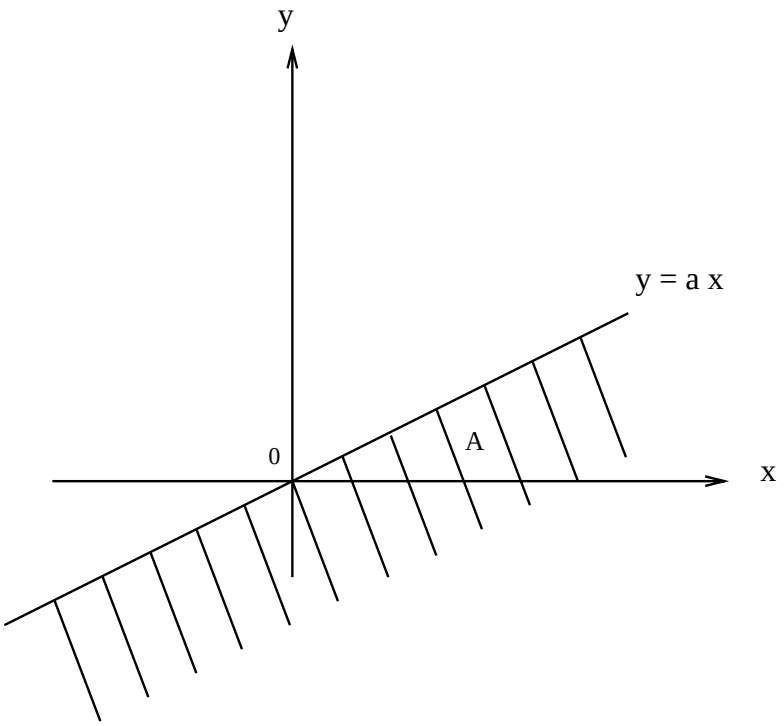


Figure 2.20. *Contour idéal 2D.*

Un contour parfait est défini par une droite séparant l'espace en 2 régions homogènes (voir figure 2.20.) :

$$H(x, y) = Au_{-1}(x, y)$$

avec

$$u_{-1}(x,y) = \begin{cases} 0 & \text{si} \quad y > ax \\ 1 & \text{si} \quad y \leq ax \end{cases}$$

avec $a \geq 0$
Un contour bruité est défini par :

$$C(x,y) = H(x,y) + \eta(x,y)$$

où $\eta(x,y)$ représente le bruit.

On suppose que le bruit est décorrélé en x, y soit :

$$\eta(x,y) = \eta_1(x) + \eta_2(y)$$

où η_1 et η_2 sont des bruits blancs gaussiens indépendants de moyenne nulle et de variance η_0^2. On supposera dans la suite par commodité que les variances de η_1 et η_2 sont égales.

2.6.3. Modèle de contour 3D

Dans le cas 3D le contour parfait est défini par un plan divisant l'espace en deux régions de niveaux de gris uniformes (voir figure 2.6.3.) :

$$H(x,y,z) = Au_{-1}(x,y,z)$$

avec

$$u_{-1}(x,y,z) = \begin{cases} 0 & \text{si} \quad z > ax + by, \\ 1 & \text{si} \quad z \leq ax + by \end{cases}$$

avec $a, b \geq 0$ Un contour bruité est défini par :

$$C(x,y,z) = H(x,y,z) + \eta(x,y,z)$$

avec :

$$\eta(x,y,z) = \eta_1(x) + \eta_2(y) + \eta_3(z)$$

où les η_i sont des bruits blancs gaussiens indépendants de variances η_0^2.

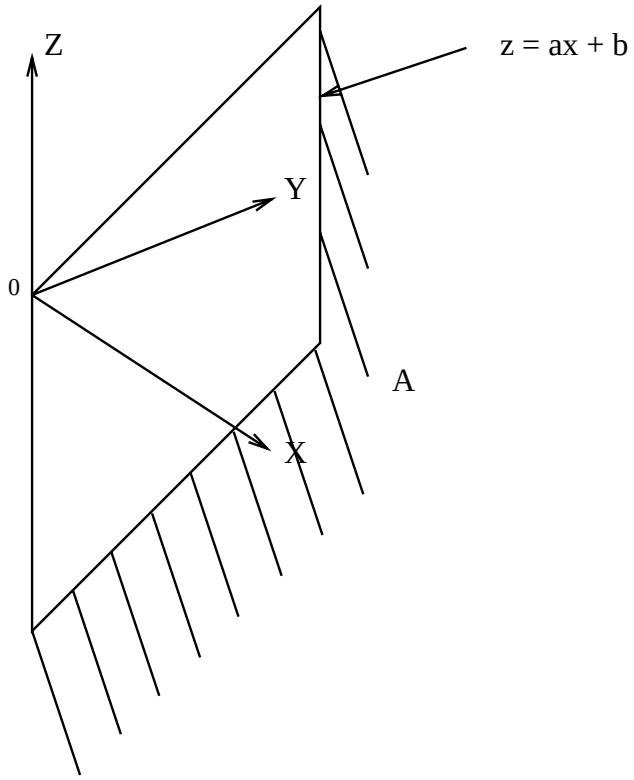


Figure 2.21. *Contour idéal 3D.*

2.6.4. Performances du filtre Deriche 2D, 3D

Dans cette partie on calcule les performances des filtres 2D et 3D de Deriche en fonction de l'orientation du contour. On commencera par le calcul dans le cas 3D, le cas 2D étant déduit directement.

2.6.5. Opérateurs

$$\begin{aligned} s_4(x) &= k(\alpha \mid x \mid +1)e^{-\alpha|x|} : \text{composante de lissage} \\ d_4(x) &= -cxe^{-\alpha|x|} : \text{composante de dérivée première} \\ l_4(x) &= f(1-\alpha \mid x \mid)e^{-\alpha|x|} : \text{composante de dérivée seconde} \\ D(x,y,z) &= l_4(x)s_4(y)s_4(z)s_4(x)l_4(y)s_4(z) + \\ & \quad s_4(x)s_4(y)l_4(z) : \text{filtre de calcul du laplacien} \end{aligned} \qquad [2.35]$$

et:

$$\begin{aligned} g_1(x,y,z) &= d_4(x)s_4(y)s_4(z) \\ g_2(x,y,z) &= s_4(x)d_4(y)s_4(z) \\ g_3(x,y,z) &= s_4(x)s_4(y)d_4(z) \end{aligned}$$

sont les filtres de calcul des dérivées par rapport à x, y, z. Pour les calculs suivants, les constantes de normalisation k, c, f sont déterminées dans l'espace continu.

2.6.6. Performances

Détection pour l'approche gradient

De manière à déterminer RSB, on calcule la réponse du filtre en un point du contour pour le contour idéal ($H(x,y,z)$) soit :

$$\begin{aligned} G_{x_0} &= I_x(x_0,y_0,z_0) = [H * g_1]_0 \\ &= \int\int\int H(x,y,z)g_1(x_0-x, y_0-y, z_0-z)dxdydz, \\ G_{y_0} &= [H * g_2]_0, \\ G_{z_0} &= [H * g_3]_0, \end{aligned}$$

Soit le vecteur gradient $G_0 = (G_{x_0}, G_{y_0}, G_{z_0})^t$, dont la direction est orthogonale au contour défini par $z = ax + by$. Après calcul, on obtient la norme du gradient qui dépend de l'orientation du contour et est donnée par :

$$\|G_0\| = Ag(a,b)$$

où $g(a,b)$ est le rapport de deux polynômes en a et b.

$$\begin{aligned} g(a,b) &= \frac{(ab+b+a)\sqrt{a^2+b^2+1}(b^4(a+1)^2 + b^3(3a^3+8a^2+8a+3))}{(b+1)^3(a+b)^3(a+1)^3} \\ &+ \frac{(ab+b+a)\sqrt{a^2+b^2+1}(b^2(a^4+8a^3+15a^2+8a+1))}{(b+1)^3(a+b)^3(a+1)^3} \\ &+ \frac{(ab+b+a)\sqrt{a^2+b^2+1}(b(2a^4+8a^3+8a^2+2a)+a^4+3a^3+a^2)}{(b+1)^3(a+b)^3(a+1)^3} \end{aligned}$$

Il est intéressant d'étudier comment varie $g(a,b)$ avec la direction (a,b). On atteint la valeur maximale de 1 pour une normale au contour alignée avec un des axes du repère, soit si a et b tendent tous les deux vers 0 ou si un des deux tend vers l'infini. On atteint la valeur minimale 0.85 pour $(a,b) = (1,1)$ soit pour un plan orienté à 45 degrés.

De manière à obtenir le cas 2D on pose $b = 0$ et dans ce cas :

$$g(a,0) = \frac{(a^2+3a+1)\sqrt{a^2+1}}{(a+1)^3}$$

De nouveau la valeur maximale 1 est atteinte pour $a = 0$ ou $d \to \infty$. La valeur minimum $g(1,0) \simeq 0.88$ est obtenue pour $a = 1$ ce qui correspond à un contour orienté à 45 degrés (voir figure 2.22.).

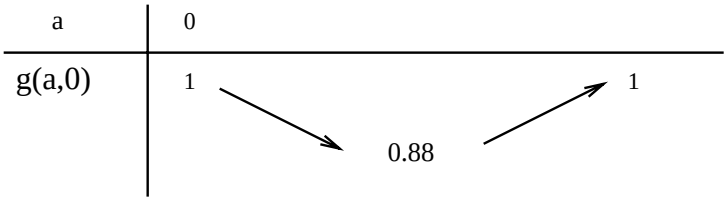


Figure 2.22. *Tableau des variations de* $g(a,0)$.

Calcul de l'écart type de la réponse au bruit :

En utilisant le fait que $\int_{-\infty}^{+\infty} l_4(x)dx = 0$ (la réponse du filtre dérivée seconde à un signal constant est nulle), et $\int_{-\infty}^{+\infty} s_4(x)dx = 1$ (la réponse du filtre de lissage à un signal constant est ce signal) on obtient :

$$\begin{aligned} N_0 &= \int\int\int (\eta_1(x) + \eta_2(y) + \eta_3(z)) D(x_0 - x, y_0 - y, z_0 - z) dxdydz \\ &= \int \eta_1(x) l_4(x) dx + \int \eta_2(y) l_4(y) dy + \int \eta_3(z) l_4(z) dz \end{aligned}$$

La valeur du critère de détection de Canny (voir équation 2.23) s'obtient en faisant le rapport de la norme du gradient en un point du contour idéal et de l'écart type de la réponse au bruit :

$$\text{RSB} = \frac{\|G_0\|}{\sqrt{E(N_0^2)}} = \frac{Ag(a,b)}{\eta_0 \sqrt{\alpha} \sqrt{\frac{3}{2}}}$$

Localisation pour l'approche laplacien

Un point (x_0, y_0, z_0) est considéré comme point de contour si et seulement si :

$$\theta_0 = \theta(x_0, y_0, z_0) = \int\int\int C(x,y,z) D(x - x_0, y - y_0, z - z_0) dxdydz = 0$$

Ce produit de convolution peut être réécrit comme :

$$\theta_0 = P_0 + N_0$$

où $P_0 = [H * D]_0$ et $N_0 = [\eta * D]_0$.

On calcule d'abord la réponse au contour idéal ($H(x,y,z)$) soit :

$$P_0 = A \int\int\int_{z \leq ax+by} D(x_0 - x, y_0 - y, z_0 - z) dxdydz$$

Et ensuite on développe P_0 au premier ordre autour de l'origine (ou de n'importe quel point du plan d'équation $z = ax + by$). Ce développement limité se justifie car on suppose que le point de contour détecté est au voisinage du vrai point de contour.

$$P_0 \simeq A(ax_0 + by_0 - z_0)\alpha^2 m(a,b)$$

où

$$\begin{aligned} m(a,b) &= \frac{(a^3b^3+3a^2b^3+3ab^3+b^3+7a^2b^2)(a^2+b^2+1)}{(a+b)^3(a+1)^3(b+1)^3} \\ &+ \frac{(3a^3b^2+3ab^2+3a^3b+3a^2b+a^3)(a^2+b^2+1)}{(a+b)^3(a+1)^3(b+1)^3} \end{aligned}$$

Etant donné que $\theta_0 = 0$ aux points de contours, le point (x_0, y_0, z_0) est un point de contour si $P_0 + N_0 = 0$. Ainsi pour chaque point de contour détecté on a $E(P_0^2) = E(N_0^2)$.

On obtient :

$$\begin{aligned} E\left[N_0^2\right] &= \frac{3}{2}\alpha^3\eta_0^2, \\ E\left[P_0^2\right] &= E\left[(ax_0+by_0-z_0)^2\right]2\alpha^4 m^2(a,b) \end{aligned}$$

L'écart type de la distance du point de contour détecté au plan contenant les vrais points de contour (inverse du critère de localisation de Canny) est donc :

$$\sigma[\frac{ax_0+by_0-z_0}{(a^2+b^2+1)^{\frac{1}{2}}}] = \sqrt{\frac{3}{2\alpha}}\frac{\eta_0}{A}h(a,b)$$

où :

$$h(a,b) = \frac{1}{m(a,b)\sqrt{a^2+b^2+1}}$$

Par définition le critère de localisation de Canny (voir équation [2.26]) correspond à l'inverse de cet écart type soit :

$$L = \sqrt{\frac{2\alpha}{3}}\frac{A}{\eta_0 h(a,b)}$$

On obtient ainsi une estimation quantitative de la qualité de la localisation des contours. Comme dans le cas 1D, la localisation est une fonction croissante de α. La fonction $h(a,b)$ apparaît comme un facteur de correction dépendant de l'orientation du contour, et qui est minimum quand la normale au contour est parallèle à un des trois axes, c.a.d. quand a et b tendent vers 0 ou quand l'un des deux tend vers l'infini. Dans ces trois cas, $h(a,b) \to 1$. La valeur maximale de $h(a,b)$ est $h(1,1) = 128/21\sqrt{3} \simeq 3.52$, obtenue pour un plan de contour dont la normale est parallèle au vecteur $(1,1,-1)$. Cela montre clairement l'importance de la prise en compte de l'orientation du contour pour la localisation.

Le cas 2D est obtenu en posant $b = 0$ ce qui entraine :

$$h(a,0) = \frac{1}{m(a,b)\sqrt{a^2+1}} = \frac{(a+1)^3}{(a^2+1)^{\frac{3}{2}}}$$

Dans ce cas, la valeur minimale de $h(a,0)$ est 1 pour $a = 0$ ou $a \to \infty$ (contours parallèles aux axes de coordonnées). La valeur maximale (produisant la pire localisation) est $2\sqrt{2} \simeq 2.83$ et est de nouveau obtenue pour $a = 1$ (voir figure 2.23.)

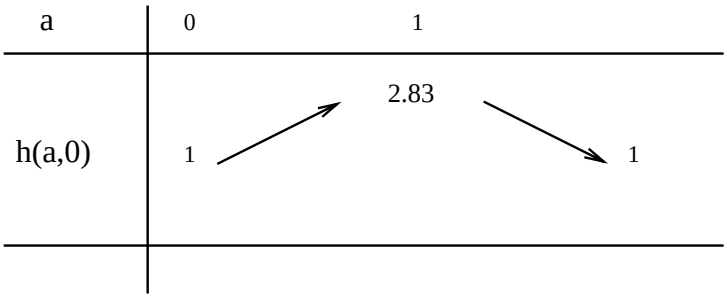


Figure 2.23. *Tableau des variations de $h(a,0)$.*

2.6.7. Isotropie et invariants euclidiens pour les filtres linéaires

Le développement précédent illustre dans le cas du filtre de Deriche et de la norme du gradient les problèmes d'invariance (ou plutot de non-invariance) engendrés par des filtres de lissage non-isotropes. Dans cette partie, nous allons aborder ces problèmes de façon plus générale en montrant que l'invariance euclidienne est conservée si les dérivées partielles sont calculées à partir de filtres issues d'un filtre de lissage isotrope. On notera que l'isotropie est en théorie une condition suffisante mais non nécessaire et en pratique une condition aussi nécessaire.

Nous discuterons les propriétés des filtres dans l'espace continu et nous traiterons comme précédemment le cas 3D qui est plus général (une image se note $I(x,y,z)$ dans l'espace continu et $I(i,j,k)$ dans l'espace discret).

Soit $I(x,y,z)$ une image $3D$.

Supposons que nous cherchions les dérivées partielles de $I(x,y,z)$: $\dfrac{\partial^n (I(x,y,z))}{\partial x^m \partial y^p \partial z^q}$, $n = m+p+q$ que nous noterons avec la notation abrégée : $I_{x^m y^p z^q}$ (nous écri-

rons seulement les variables dont l'exposant est non nul, par exemple $I_{x^1y^0z^0}$ devient I_x).

Comme on l'a vu précédemment, si $f(x,y,z)$ est la réponse impulsionnelle du filtre de lissage, l'image restorée I_r est égale à $I * f$, où $*$ est le produit de convolution. On obtient donc :

$$\frac{\partial^n I_r}{\partial x^m \partial y^p \partial z^q} = \frac{\partial^n (I * f)}{\partial x^m \partial y^p \partial z^q} = I * (\frac{\partial^n f}{\partial x^m \partial y^p \partial z^q})$$

Ainsi, la réponse impulsionnelle du filtre qui calcule $I_{x^m y^p z^q}$ est $\frac{\partial^n f}{\partial x^m \partial y^p \partial z^q}$

Supposons que le filtre de lissage soit l'opérateur de Deriche :

$$f(x,y,z) = f_0(x) f_0(y) f_0(z)$$

avec

$$f_0(x) = c_0(1 + \alpha \mid x \mid)e^{-\alpha|x|}$$

où c_0 est une constante de normalisation. L'inconvénient de l'opérateur $f(x,y,z)$ est qu'il n'est pas isotrope (invariant par rotation). Par exemple, celà implique que la norme du gradient calculé avec les filtres de dérivation correspondants dépend de l'orientation du contour (voir paragraphe précédent). On pourrait aussi montrer la même chose pour le Laplacien ou d'autres invariants différentiels euclidiens.

On va maintenant montrer que si les dérivées partielles sont calculées avec des filtres dérivant d'un opérateur de lissage isotrope alors les invariants différentiels euclidiens "demeurent" invariants. On donnera la démonstration pour une image de dimension 3 mais elle s'applique pour une dimension quelconque.

Soit $I(x,y,z)$ une image $3D$.

Soit $J(x,y,z)$ une image I transformée par un déplacement rigide :

$$J(X_0, Y_0, Z_0) = I(x_0, y_0, z_0)$$

avec

$$(X_0, Y_0, Z_0)^t = R(x_0, y_0, z_0)^t + T$$

où R et T sont respectivement des matrices définissant une rotation autour de l'origine et une translation.

Soit f la réponse impulsionnelle d'un filtre de lissage isotrope, nous avons :

$$f(x_1, y_1, z_1) = f(x_2, y_2, z_2)$$

où

$$(x_2, y_2, z_2)^t = R(x_1, y_1, z_1)^t$$

ainsi

$$(I * f)(x_0, y_0, z_0) = (J * f)(X_0, Y_0, Z_0)$$

Soit E une fonction des dérivées partielles de I. Si E est un invariant différentiel euclidien de I nous avons :

$$E((I_x)(x_0, y_0, z_0), (I_y)(x_0, y_0, z_0), (I_z)(x_0, y_0, z_0), (I_{xx})(x_0, y_0, z_0)), (I_{xy})(x_0, y_0, z_0) \ldots) =$$
$$E((J_x)(X_0, Y_0, Z_0), (J_y)(X_0, Y_0, Z_0), (J_z)(X_0, Y_0, Z_0), (J_{xx})(X_0, Y_0, Z_0)), (J_{xy})(X_0, Y_0, Z_0) \ldots)$$

Si nous calculons les dérivées partielles en utilisant nos filtres, on obtient :

$$E((I * f_x)(x_0, y_0, z_0), (I * f_y)(x_0, y_0, z_0), (I * f_z)(x_0, y_0, z_0) \ldots.) =$$
$$E((I * f)_x(x_0, y_0, z_0), (I * f)_y(x_0, y_0, z_0), (I * f)_z(x_0, y_0, z_0) \ldots)$$

Etant donné que E est aussi un invariant différentiel euclidien de $I * f$, la dernière expression est égale à :

$$E((J * f)_x(X_0, Y_0, Z_0), (J * f)_y(X_0, Y_0, Z_0), (J * f)_z(X_0, Y_0, Z_0) \ldots) =$$
$$E((J * f_x)(X_0, Y_0, Z_0), (J * f_y)(X_0, Y_0, Z_0), (J * f_z)(X_0, Y_0, Z_0) \ldots)$$

Par conséquent, nous avons montré que :

$$E((I * f_x)(x_0, y_0, z_0), (I * f_y)(x_0, y_0, z_0), (I * f_z)(x_0, y_0, z_0) \ldots.) =$$
$$E((J * f_x)(X_0, Y_0, Z_0), (J * f_y)(X_0, Y_0, Z_0), (J * f_z)(X_0, Y_0, Z_0) \ldots)$$

L'équation précédente signifie que E reste invariant par changement de repère (invariant euclidien) si les dérivées partielles sont calculés à partir d'opérateurs dérivés de f.

On notera au passage qu'il existe deux types d'invariants différentiels euclidiens dans les images :

- Les invariants différentiels euclidiens pour chaque point de l'image : gradient, laplacien, courbures de la surface (hypersurface dans le cas $3D$) définie par l'image (voir chapitre : "Des images volumiques 3D à la géométrie des surfaces").

• Les invariants différentiels euclidiens définis seulement sur les points de contour ; si on suppose que le gradient est paralléle à la normale à la surface (à la courbe dans le cas $2D$) les courbures s'expriment en fonction des dérivées partielles de l'image; les expressions correspondantes sont donc invariantes seulement sur les points de contour (voir chapitre "Des images volumiques 3D à la géométrie des surfaces").

La preuve précédente est aussi valide dans le second cas si on suppose que le gradient est paralléle à la normale pour $I * f$ ce qui est une hypothèse raisonnable.

Nous avons donc montré que si les dérivées partielles d'une image sont calculées avec des filtres dérivés d'un opérateur de lissage isotrope, alors les invariants différentiels euclidiens calculés en utilisant ces dérivées partielles, sont invariants par transformation rigide (invariance euclidienne).

La figure 2.24. illustre l'intérêt d'utiliser des filtres issus d'un filtre de lissage isotropique pour calculer les dérivées partielles ; on compare les courbures sur les contours d'un disque plein (voir chapitre "Des images volumiques 3D à la géométrie des surfaces" pour le calcul des courbures en fonction des dérivées). Les dérivées sont successivement calculées avec le filtre de Deriche et ensuite avec un filtre gaussien. On choisit la largeur des filtres de manière à ce que la courbure (inverse du rayon du cercle) ait la bonne valeur (en pixels). On discerne clairement les fausses variations de courbure induites par les filtres dérivant d'un opérateur anisotrope [136].

2.7. Filtres récursifs pour approximer les filtres gaussiens

La partie précédentes montrent clairement l'intérêt d'utiliser des opérateurs dérivés d'un filtre de lissage isotrope afin de calculer les dérivées partielles d'une image. D'un autre côté, on a aussi intérêt a mettre en œuvre des filtres récursifs séparables de manière à obtenir un temps de calcul raisonnable. Un moyen de joindre ces deux points antagonistes est d'utiliser des approximations récursives des filtres gaussiens et de leurs dérivées [45].

2.7.1. Calcul des dérivées d'un signal 1D

L'opérateur de lissage gaussien 1D est donné par :

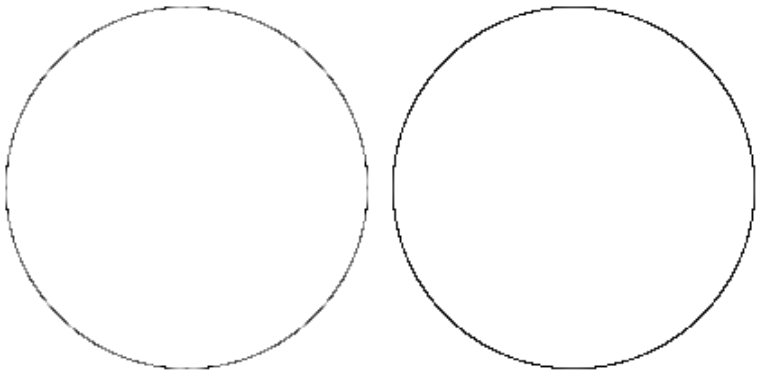


Figure 2.24. *Image gauche : Valeurs de la courbure obtenues avec les filtres dérivés d'un opérateur anisotrope (filtre de Deriche) ; Image droite : Valeurs de la courbure obtenues avec les filtres dérivés d'un opérateur isotrope (filtre gaussien)*

$$g(x) = e^{-\frac{x^2}{2\sigma^2}}$$

Les parties positives et négatives de g et de ses dérivées normalisées d'ordre $1, 2, 3$ peuvent être approximées (avec une erreur normalisée au sens des moindres carrées d'environ $\epsilon^2 = 10^{-6}$) par un filtre récursif (IIR) d'ordre 4 de la forme :

$$h(x) = (a_0 \cos(\frac{\omega_0}{\sigma}x) + a_1 \sin(\frac{\omega_0}{\sigma}x))e^{-\frac{b_0}{\sigma}x} + (c_0 \cos(\frac{\omega_1}{\sigma}x) + c_1 \sin(\frac{\omega_1}{\sigma}x))e^{-\frac{b_1}{\sigma}x} \quad [2.36]$$

On réalise ces approximations de la manière suivante.

$h(x)$ peut s'écrire :

$$h(x) = \sum_{i=1}^{m} \alpha_i e^{-\frac{\lambda_i x}{\sigma}}$$

où les cœfficients α_i et λ_i sont des nombres complexes conjuguées (dans le cas présent m est égal à 4). L'erreur au sens des moindres carrées est définie par :

$$\epsilon^2 = \frac{\sum_{k=0}^{10\sigma}(h(k) - u(k))^2}{\sum_{k=0}^{10\sigma} u(k)^2}$$

où u est la fonction que l'on veut approximer avec h.

Ce problème de minimisation peut être résolu en utilisant un algorithme classique de minimisation. Un bon point de départ pour une méthode itérative de minimisation peut être obtenu avec la méthode de Prony.

La méthode de Prony utilise un échantillonage équidistant de la fonction $((x_0, y_0 = f(x_0)), \ldots, (x_n, y_n = f(x_n)))$. On peut poser $x_r = x_0 + rh$, $c_r = \alpha_r e^{\lambda_r x_0}$ et $v_r = h\lambda_r$, pour $r = 0, \ldots m$, et écrire $f(x) = \sum_{i=1}^{m} \alpha_i \lambda_i x$. Ainsi, on obtient le système suivant :

$$\begin{cases} c_1 + c_2 + \ldots + c_m = y_0 \\ c_1 v_1 + c2 v_2 + \ldots + c_m v_m = y_1 \\ \vdots \\ c_1 v_1^m + c_2 v_2^m + \ldots + c_m v_m^m = y_m \end{cases} \qquad [2.37]$$

Si on construit les polynômes :

$$\prod_{i=1}^{m} (v - v_i) = \sum_{i=1}^{m} s_i v_{m-i} = \phi(v) \qquad [2.38]$$

avec $s_0 = 1$, en multipliant chaque équation de [2.37] par s_i, $i = 0 \ldots m$, et en additionant chacune de ces nouvelles équations, on obtient :
$\sum_{i=1}^{m} \phi(v_i) c_i = \sum_{i=0}^{m} s_i y_{m-i} = 0$, puisque $\phi(v_r) = 0$.
Si on choisit $n \geq m$, on peut aussi écrire :

$$\begin{cases} y_{m-1} s_1 + y_{m-2} s_2 + \ldots + y_0 s_m = -y_m \\ y_m s_1 + y_{m-1} s_2 + \ldots + y_1 s_m = -y_{m+1} \\ \vdots \\ y_{n-1} s_1 + y_{n-2} s_2 + \ldots + y_{n-m} s_m = -y_n \end{cases}$$

Ces $n - m + 1$ équations avec m inconnues s_i définissent un sytème au sens des moindres carrés. Ensuite, nous obtenons les inconnues v_i en résolvant [2.38]. Finalement, nous obtenons les cœfficients λ_i avec la formule $\lambda_i = \frac{\ln v_i}{h}$ les c_i à partir de [2.37], et les α_i avec $\alpha_i = c_i v_i^{-\frac{x_0}{h}}$.

L'avantage de cette méthode est qu'il n'est pas nécessaire d'affecter à σ une valeur particulière pour déterminer les cœfficients λ_i et α_i. Les cœfficients a_0, a_1, b_0, b_1, c_0, c_1, ω_0, ω_1 sont aisément obtenus à partir des α_i et λ_i.

Nous allons maintenant donner les résultats d'approximation obtenus pour g et ses dérivées normalisées d'ordre 1,2,3 que nous noterons : g_1, g_2, g_3. Ces opérateurs sont normalisés de maniére à ce que le résultat obtenu soit correct si on les applique à des polynômes. Soit :

$$\sum_{-\infty}^{+\infty} g(x)dx = 1;$$

$$\sum_{-\infty}^{+\infty} x g_1(x)dx = 1;$$

$$\sum_{-\infty}^{+\infty} g_2(x)dx = 0 \text{ et} \sum_{-\infty}^{+\infty} \frac{x^2}{2} g_2(x)dx = 1;$$

$$\sum_{-\infty}^{+\infty} \frac{x^3}{6} g_3(x)dx = 1 \text{ et } \sum_{-\infty}^{+\infty} x g_3(x)dx = 0.$$

On définit γ_1, γ_2, γ_3, γ_4 comme suit :

- $\gamma_1 = \sum_{k=0}^{\infty} -\frac{k^2}{2\sigma^2}$
- $\gamma_2 = \sum_{k=0}^{\infty} k^2 - \frac{k^2}{2\sigma^2}$
- $\gamma_3 = \sum_{k=0}^{\infty} k^4 - \frac{k^2}{2\sigma^2}$
- $\gamma_4 = \sum_{k=0}^{\infty} k^6 - \frac{k^2}{2\sigma^2}$.

On peut montrer qu'une fonction $F(\sigma) = \sum_{k=1}^{\infty} k^{\alpha} - \frac{k^2}{(2\sigma^2)}, \alpha > 0$ peut être approximée avec une très faible erreur par une fonction polynômiale $F_a(\sigma) = I(\alpha)\sigma^{\alpha+1}$ avec $I(\alpha) = \int_0^{\infty} t^{\alpha} - \frac{t^2}{2} dt$.
Nous avons $I(0) = \sqrt{\frac{\pi}{2}}, I(1) = 1$, et $I(\alpha+2) = (\alpha+1)I(\alpha)$.
Si $\alpha = 0$, on approxime $F(\sigma)$ avec $F_a(\sigma) = 1.25\sigma - 0.5$ pour $\sigma \geq 0.5$.
Ainsi nous obtenons les approximations suivantes :

- $\gamma_1 \approx 0.5 + 1.25\sigma$
- $\gamma_2 \approx \sigma^3 \sqrt{\frac{\pi}{2}}$
- $\gamma_3 \approx 3\sigma^5 \sqrt{\frac{\pi}{2}}$
- $\gamma_4 \approx 15\sigma^7 \sqrt{\frac{\pi}{2}}$

Les cœfficients de normalisation sont égaux à :

- $\beta_0 = \dfrac{1}{2\gamma_1 - 1} \approx \dfrac{0.4}{\sigma}$

- $\beta_1 = -\dfrac{1}{2\gamma_2} \approx -\dfrac{0.4}{\sigma^3}$
- $\beta_2 = -\dfrac{2\gamma_2}{-2\gamma_2^2 + 2\gamma_3\gamma_1 - \gamma_3} \approx -\dfrac{0.4}{\sigma^3}$
- $\beta_3 = \dfrac{2\gamma_1 - 1}{2\gamma_2} \approx \dfrac{1}{\sigma^2}$
- $\beta_4 = \dfrac{3}{\frac{\gamma_4\gamma_2}{\gamma_3\sigma^2} - \frac{\gamma_3}{\sigma^2}} \approx \dfrac{1.2}{\sigma^3}$
- $\beta_5 = \dfrac{3\gamma_2}{\gamma_3(\frac{\gamma_4\gamma_2}{\gamma_3\sigma^2} - \frac{\gamma_3}{\sigma^2})} \approx \dfrac{0.4}{\sigma^5}$

Nous obtenons les résultats suivants pour les cœfficients des filtres d'approximation :

	$g(x) = \beta_0 e^{-\frac{x^2}{2\sigma^2}}$	$g_1(x) = \beta_1 x e^{-\frac{-x^2}{2\sigma^2}}$
a_0	$0.657/\sigma$	$-0.173/\sigma^2$
a_1	$1.979/\sigma$	$-2.004/\sigma^2$
c_0	$-0.258/\sigma$	$0.173/\sigma^2$
c_1	$-0.239/\sigma$	$0.444/\sigma^2$
b_0	1.906	1.561
b_1	1.881	1.594
ω_0	0.651	0.700
ω_1	2.053	2.145

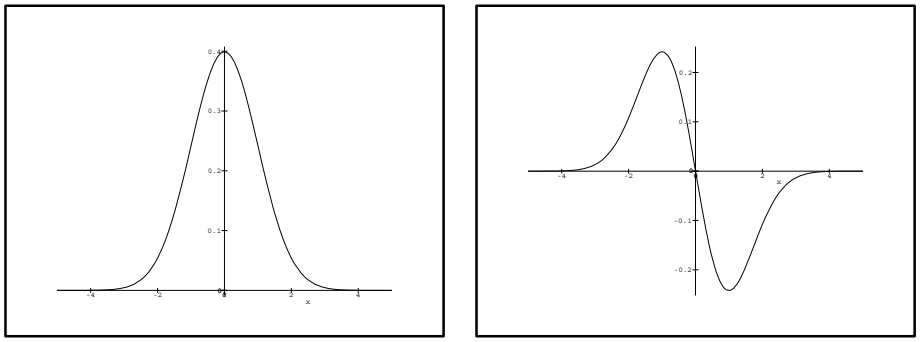


Figure 2.25. *Le filtre gaussien normalisé et sa première dérivée ($\sigma = 1$)*

Pour g et $\sigma = 1$, $\epsilon^2 = 9.7E-09$. Pour g_1, $\epsilon^2 = 1.2E-06$.

Pour $g_2(x) = \beta_2(1 - \beta_3 x^2)e^{-\frac{x^2}{2\sigma^2}}$ et $g_3(x) = \frac{1}{\sigma^2}(\beta_4 x - \beta_5 x^3)e^{-\frac{x^2}{2\sigma^2}}$, on obtient :

	$g_2(x)$	$g_3(x)$
a_0	-0.724/σ^3	1.286/σ^4
a_1	1.689/σ^3	-0.290/σ^4
c_0	0.325/σ^3	-1.286/σ^4
c_1	-0.721/σ^3	0.262/σ^4
b_0	1.295	1.012
b_1	1.427	1.273
ω_0	0.779	0.947
ω_1	2.234	2.338

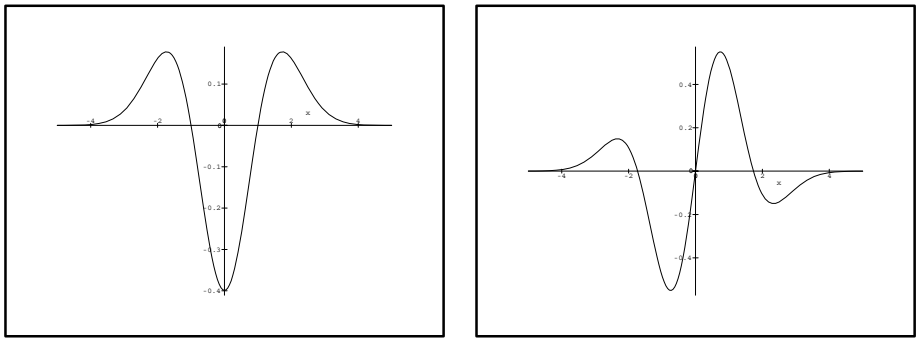


Figure 2.26. *Dérivées deuxième et troisième du filtre gaussien normalisées ($\sigma = 1$)*

Pour g_2 et $\sigma = 1$, $\epsilon^2 = 7.3E-06$. Pour g_3 et $\sigma = 1$, $\epsilon^2 = 1.9E-04$.

Une fois que les cœfficients $a_0, a_1, c_0, c_1, b_0, b_1, \omega_0, \omega_1$ sont calculés, on peut aisément déterminer une réalisation récursive de $h(n)$. On calcule la transformée en Z : $H_1(z) = \sum_{n=0}^{\infty} h(n)z^{-n}$ pour la partie causale du filtre, et $H_2(z) = \sum_{n=-\infty}^{-1} h(n)z^{-n}$ pour la partie anticausale. Les détails d'implémentation peuvent être trouvés par exemple dans [45].

2.7.2. Conclusion

La prise en compte de modèles de contour réellement 2D ou 3D même aussi simples qu'un contour orienté remet en question l'optimalité des opérateurs. L'utilisation de modèles plus complets incluant des propriétés d'angles ou de courbures est sans doute intéressante d'un point de vue théorique. Cependant il paraît difficile d'obtenir des opérateurs de mise en œuvre aisée avec des modèles plus complexes.

Les algorithmes basés sur le filtrage séparable récursif apportent néanmoins des solutions intéressantes pour la détection de contours. Ces méthodes décrites précedemment permettent en effet d'obtenir des résultats satisfaisants sur la plupart des types d'images.

2.8. Détection de contours par lissage : Haralick

L'idée de ce modèle est d'approximer localement le signal discret formant l'image par une fonction dérivable, puis de dériver cette fonction [81].

Soit $I(r,c)$ l'image, $(r,c,I(r,c))$ forme une surface.

On suppose que localement l'image peut être approximée par une fonction polynômiale

$$I(r,c) = k_1 + k_2 r + k_3 c + k_4 r^2 + k_5 rc + k_6 c^2 + k_7 r^3 + k_8 r^2 c + k_9 rc^2 + k_{10} c^3$$

On recherche les cœfficients de ce polynôme en construisant une base de l'ensemble des fonctions polynômiales dans $R \times C$ (R et C étant des ensembles d'indices ; $r \in R \Rightarrow -r \in R$; $c \in C \Rightarrow -c \in C$). On va montrer que dans certains cas particuliers il existe un polynôme discret (défini dans l'espace des entiers relatifs) interpolant unique passant par tous les points de l'image. On montrera aussi que les cœfficients de ce polynôme peuvent être déterminés par des produits de convolution.

Soit $\{P_0(r), \ldots P_{N-1}(r)\}$ un ensemble de polynômes discrets orthogonaux entre eux.

$\{Q_0(r,c), Q_1(r,c) \ldots Q_{p-1}(r,c)\}$ peut être obtenu par produit tensoriel.

On prend pour $P_o(r), \ldots P_{n-1}(r)$ les polynômes discrets de Tchebycheff :

On définit : $P_o(r) = 1$.

On suppose : $P_0(r) \ldots P_{n-1}(r)$ définis par récurrence comme suit :

$$\begin{cases} P_n(r) = r^n + a_{n-1}r^{n-1} + \ldots + a_1 r + a_0 \\ P_n(r) \perp P_i(r) \qquad \forall i \in \{0, \ldots, n-1\} \end{cases}$$

Soit

$$\begin{cases} \sum_{r \in R} P_k(r)(r^n + a_{n-1}r^{n-1} + \ldots + a_1 r + a_0) = 0 \\ k = 0, \ldots, n-1 \end{cases}$$

On obtient ainsi n équations linéaires d'inconnues $a_0, \ldots, a_{n-1}$. Pour $n = 5$ la résolution de ce système permet l'obtention d'une base orthogonale de l'ensemble des polynômes de degré 4 :

$$\begin{cases} P_0(r) = 1 \\ P_1(r) = r \\ P_2(r) = r^2 - \frac{u_2}{u_0} \\ P_3(r) = (r^3 - \frac{u_4}{u_2})r \\ P_4(r) = \frac{r^4 + (u_2u_4 - u_0u_6)r^2 + (u_2u_6 + u_2^4)}{u_0u_4 - u_2^2} \end{cases}$$

$\{P_0(r), \ldots P_N(r)\}$ est un ensemble de polynômes discrets et orthogonaux sur R.

$\{Q_0(c), \ldots Q_M(c)\}$ est un ensemble de polynômes discrets et orthogonaux sur C.

$$\begin{cases} r \in R \Rightarrow -r \in R \\ c \in C \Rightarrow -c \in C \end{cases} \Rightarrow \{P_0(r)Q_0(c), \ldots P_n(r)Q_n(c), \ldots P_N(r)Q_M(c)\}$$

est un ensemble de polynômes discrets et orthogonaux sur $R \times C$.

En effet

$$\sum_{r \in R} \sum_{c \in C} P_i(r)Q_j(c)P_n(r)Q_m(c) = (\sum_{r \in R} P_i(r)P_n(r))(\sum_{c \in C} Q_j(c)Q_n(c)) = 0$$

Si on pose : $R = C = \{-1, 0, 1\}$ on obtient une base orthogonale des polynômes discrets de degré 2 sur $R \times C$:

$$\{1, r, c, r^2 - \frac{2}{3}, rc, c^2 - \frac{2}{3}, r(c^2 - \frac{2}{3}), c(r^2 - \frac{2}{3}), (r^2 - \frac{2}{3})(c^2 - \frac{2}{3})\}$$

Au voisinage d'un point on a :

$$I(r,c) = \sum_{n=0}^{N-1} a_n P_n(r,c)$$

avec a_i tel que $e^2 = \sum_{(r,c)\in R\times C}[I(r,c) - \sum_{n=0}^{N-1} a_n P_n(r,c)]^2$ minimum

Soit

$$a_m = \frac{\sum_{(r,c)\in R\times C} P_m(r,c) I(r,c)}{\sum_{(r,c)\in R\times C} P_m^2(r,c)} \quad \text{car} \quad < P_m, P_n >= 0$$

On obtient dans ce *cas particulier* des cœfficients a_m tels que $e^2 = 0$ par produit scalaire (ce sont les composantes du polynôme d'interpolation dans la base (P_i)) D'où les cœfficients a_m du polynôme recherché s'expriment comme combinaison linéaire des valeurs de l'image dans un voisinage 3×3 autour du point. On peut donc générer des masques permettant le calcul des cœfficients du polynôme :

$$\tfrac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad : \quad \text{cœfficient de } 1$$

$$\tfrac{1}{6}\begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} \quad : \quad \text{cœfficient de } r$$

$$\tfrac{1}{6}\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \quad : \quad \text{cœfficient de } c$$

$$\tfrac{1}{6}\begin{pmatrix} 1 & 1 & 1 \\ -2 & -2 & -2 \\ 1 & 1 & 1 \end{pmatrix} \quad : \quad \text{cœfficient de } (r^2 - \tfrac{2}{3})$$

$$\tfrac{1}{4}\begin{pmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{pmatrix} \quad : \quad \text{cœfficient de } (rc)$$

$$\tfrac{1}{6}\begin{pmatrix} 1 & -2 & 1 \\ 1 & -2 & 1 \\ 1 & -2 & 1 \end{pmatrix} \quad : \quad \text{cœfficient de } (c^2 - \tfrac{2}{3})$$

$$\frac{1}{4}\begin{pmatrix} -1 & 0 & 1 \\ 2 & 0 & -2 \\ -1 & 0 & 1 \end{pmatrix} \quad : \quad \text{cœfficient de } (r^2 - \tfrac{2}{3})c$$

$$\frac{1}{4}\begin{pmatrix} -1 & 2 & -1 \\ 0 & 0 & 0 \\ 1 & -2 & 1 \end{pmatrix} \quad : \quad \text{cœfficient de } (c^2 - \tfrac{2}{3})r$$

$$\frac{1}{4}\begin{pmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{pmatrix} \quad : \quad \text{cœfficient de } (r^2 - \tfrac{2}{3})(c^2 - \tfrac{2}{3})$$

On obtient ainsi neuf masques correspondant à une fenêtre 3×3 permettant de calculer par des produits de convolution un polynôme approximant localement l'image en chaque point (ce polynôme interpole les 9 points correspondant au point considéré et à ces 8 voisins). La base de fonctions polynômes utilisée définit implicitement un modèle du bruit. Le problème de ce type de méthodes est la sensibilité des approximations polynômiales au bruit.

2.9. Détection de contours par optimisation

Le principe de cette approche est de rechercher directement dans l'image un modèle de contour basé sur la séparation locale de l'image en deux zones homogènes [100].

Le modèle utilisé est une droite subdivisant le mieux possible un disque centré en un point en deux régions homogènes :

$$T(x, y, c, s, l, d, b) = \begin{cases} b & \text{si} \quad cx + sy \leq l \\ b + d & \text{si} \quad cx + sy > l \end{cases}$$

On considère qu'il existe une bijection entre les contours idéaux et les fonctions $T(x, y, c, s, l, d, b)$.

Soit $I(x, y)$ l'image :

On cherche (c, s, l, d, b) tel que :

$$\epsilon^2 = \int\int_D (I(x, y) - T(x, y, s, l, b, d))^2 dxdy = \|I - T\| \text{ minimum.}$$

De manière à minimiser cette fonction, on décompose selon une base orthogonale de l'espace hilbertien des fonctions sur D.

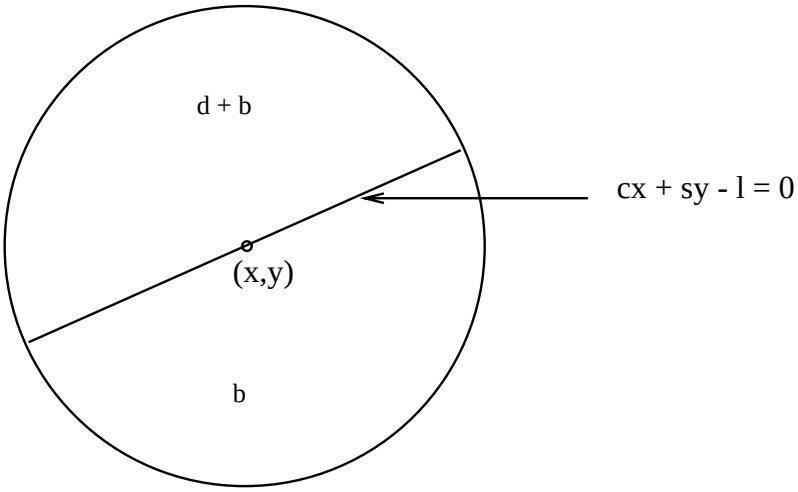


Figure 2.27. *Contour idéal au sens de Hueckel.*

$\{H_i\}_{i=0}^{\infty}$ base de l'espace hilbertien des fonctions sur D.
On obtient :

$$\begin{aligned} a_i &= \int H_i(x,y)I(x,y)dxdy = H_i \cdot I \\ s_i &= \int H_i(x,y)T(x,y,c,s,l,b,d)dxdy = H_i \cdot T \\ a_i &: \text{constantes} \\ s_i &: \text{variables} \end{aligned}$$

$$I(x,y) = \sum_i a_i H_i(x,y)$$

$$T(x,y,c,s,l,b,d) = \sum_i s_i H_i(x,y)$$

$$\begin{aligned} \int\int_D [I(x,y) - T(x,y,c,s,l,b,d)]^2 dxdy &= \\ \int\int_D [\sum_i (a_i - s_i) H_i(x,y)]^2 dxdy &= \sum_i (a_i - s_i)^2 \end{aligned}$$

On décompose les fonctions sur D en produits de composantes radiales et angulaires :

$$r^2 = x^2 + y^2, \quad Q(r) = (1 - r^2)^{\frac{1}{2}}$$

$$\begin{aligned}
H_0(x,y) &= (\frac{5}{6\pi})Q(r)(1+2r^2) \\
H_1(x,y) &= (\frac{8}{27\pi})^{\frac{1}{2}}Q(r)(5-2) \\
H_2(x,y) &= (\frac{3}{\pi})^{\frac{1}{2}}Q(r)x(4r^2-1) \\
H_3(x,y) &= (\frac{3}{\pi})^{\frac{1}{2}}Q(r)y(4r^2-1) \\
H_4(x,y) &= (\frac{3}{\pi})^{\frac{1}{2}}Q(r)x(3-4r^2) \\
H_5(x,y) &= (\frac{3}{\pi})^{\frac{1}{2}}Q(r)y(3-4r^2) \\
H_6(x,y) &= (\frac{32}{3\pi})^{\frac{1}{2}}Q(r)(x^2-y^2) \\
H_7(x,y) &= (\frac{32}{3\pi})^{\frac{1}{2}}Q(r)2xy
\end{aligned}$$

On obtient :

$$\begin{aligned}
H_1 &= (\frac{9}{2})^{\frac{1}{2}}H_1 \\
H_2 &= H_2 + H_4 \\
H_3 &= H_3 + H_5 \\
H_4 &= (\frac{3}{2})H_6 \\
H_5 &= (\frac{3}{2})H_7 \\
H_6 &= (\frac{5}{4})^{\frac{1}{2}}(H_2 - H_4) \\
H_7 &= (\frac{5}{4})^{\frac{1}{2}}(H_3 - H_5)
\end{aligned}$$

Ensuite on minimise par décomposition sur cette base de fonctions et on obtient (T, d, b) en fonction des luminances des points considérés.

Etudions maintenant le cas particulier de l'opérateur de Hueckel pour la détection de contour dans un voisinage 2×2 traité par Jean-François Abramatic.

$$s(x,y) = \begin{cases} a & \text{si} \quad x\sin\theta \geq y\cos\theta \\ b & \text{sinon} \end{cases}$$

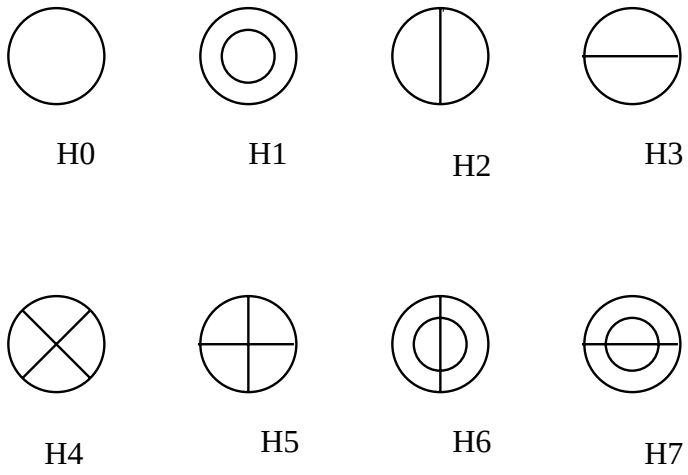


Figure 2.28. *Représentations graphiques des fonctions de la base de Hueckel.*

Soit (a^*, b^*) le couple de valeurs optimales, on va montrer que

$$\mid a^* - b^* \mid = \max(\mid B - C \mid, \mid A - D \mid)$$

Soit (H_1, H_2, H_3, H_4) la base canonique de l'espace des fonctions sur le voisinage 2×2 :

$$H_1 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \quad H_2 = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} H_3 = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \quad H_4 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

$$s(x, y, a, b, \theta) = \sum_{i=1}^{4} s_i H_i$$

$$\begin{aligned} f &= \sum_{i=1}^{4} f_i H_i \\ f &= AH_1 + BH_2 + CH_3 + DH_4 \end{aligned}$$

$$f_0 = A \ ; \ f_1 = b \ ; \ f_2 = C \ ; \ f_3 = D$$

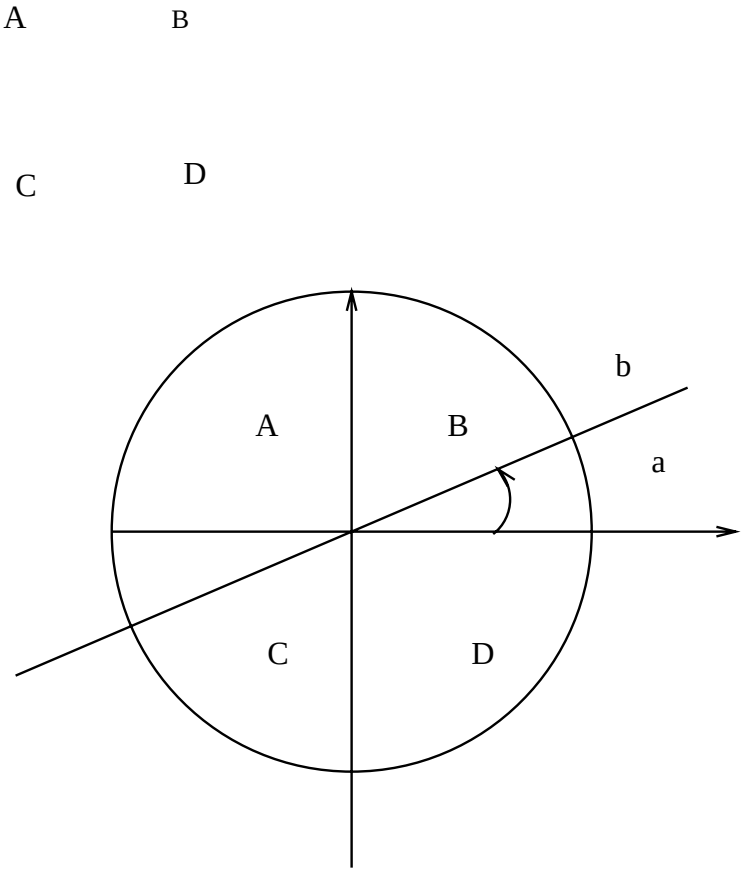


Figure 2.29. *Cas d'un voisinage* 2×2 *pour le modèle de Hueckel.*

$$s_0 = \begin{cases} b & \text{pour } 0 \le \theta \le \frac{\pi}{2} \\ a(\frac{\theta-\frac{\pi}{2}}{\frac{\pi}{2}}) + b(\frac{\pi-\theta}{\frac{\pi}{2}}) & \text{pour } \frac{\pi}{2} \le \theta \le \pi \\ a & \text{pour } \pi \le \theta \le \frac{3\pi}{2} \\ a(\frac{2\pi-\theta}{\frac{\pi}{2}}) + b(\frac{\theta-\frac{3\pi}{2}}{\frac{\pi}{2}}) & \text{pour } \frac{3\pi}{2} \le \theta \le 2\pi \end{cases}$$

$$s_1 = \begin{cases} a(\frac{\theta}{\frac{\pi}{2}}) + b(\frac{\frac{\pi}{2}-\theta}{\frac{\pi}{2}}) & \text{pour } 0 \le \theta \le \frac{\pi}{2} \\ a & \text{pour } \frac{\pi}{2} \le \theta \le \pi \\ a(\frac{\frac{3\pi}{2}-\theta}{\frac{\pi}{2}}) + b(\frac{\theta-\pi}{\frac{\pi}{2}}) & \text{pour } \pi \le \theta \le \frac{3\pi}{2} \\ b & \text{pour } \frac{3\pi}{2} \le \theta \le \pi \end{cases}$$

$$s_2 = \begin{cases} a(\frac{\frac{\pi}{2}-\theta}{\frac{\pi}{2}}) + b(\frac{\theta}{\frac{\pi}{2}}) & \text{pour } 0 \le \theta \le \frac{\pi}{2} \\ b & \text{pour } \frac{\pi}{2} \le \theta \le \pi \\ a(\frac{\theta-\frac{\pi}{2}}{\frac{\pi}{2}}) + b(\frac{\frac{3\pi}{2}-\theta}{\frac{\pi}{2}}) & \text{pour } \pi \le \theta \le \frac{3\pi}{2} \\ a & \text{pour } \frac{3\pi}{2} \le \theta \le \pi \end{cases}$$

$$s_3 = \begin{cases} a & \text{pour } 0 \le \theta \le \frac{\pi}{2} \\ a(\frac{\pi-\theta}{\frac{\pi}{2}}) + b(\frac{\theta-\frac{\pi}{2}}{\frac{\pi}{2}}) & \text{pour } \frac{\pi}{2} \le \theta \le \pi \\ b & \text{pour } \pi \le \theta \le \frac{3\pi}{2} \\ a(\frac{\theta-\frac{3\pi}{2}}{\frac{\pi}{2}}) + b(\frac{2\pi-\theta}{\frac{\pi}{2}}) & \text{pour } \frac{3\pi}{2} \le \theta \le 2\pi \end{cases}$$

Pour des raisons de symétrie on n'étudie que le premier et le second cadran.

Pour le premier cadran on obtient :

$$\epsilon^2 = (b-A)^2 + (a-D)^2 + (\frac{2\theta}{\pi}a + (1-\frac{2\theta}{\pi})b - B)^2 + (\frac{2\theta}{\pi}b + (2-\frac{2\theta}{\pi})a - c)^2$$

ϵ^2 minimum $\Longrightarrow \frac{\partial\epsilon^2}{\partial a} = \frac{\partial\epsilon^2}{\partial b} = \frac{\partial\epsilon^2}{\partial\theta} = 0$

Soit :

$$\begin{cases} a = \frac{3A+B+C-D}{4} \\ b = \frac{3D+B+C-A}{4} \\ \theta = \frac{\pi}{4} - \pi 4\frac{C-B}{A-D} \end{cases}$$

Les calculs sont similaires pour le 2ème cadran. On obtient :

$$\mid a^* - b^* \mid = \max(\mid B - C \mid, \mid A - D \mid)$$

Ce qui correspond à l'opérateur de Roberts.

Comme dans le cas du modèle d'Haralick, le problème du modèle de Hueckel est la définition non explicite du bruit. Par contre, ces deux modèles sont réellement 2D ce qui n'est pas le cas du modèle de Canny.

2.10. Perspectives

Les propriétés essentielles d'un opérateur de détection de contour dans des images bidimensionnelles sont :

1. définition explicite du bruit,
2. prise en compte des propriétés bidimensionnelles d'un contour : courbures, angles,
3. coût calculatoire.

Les opérateurs issus du modèle de Canny apportent une réponse satisfaisante pour les points 1 et 3 mais négligent le point 2. Les opérateurs associés à des modèles de type Haralick ou Hueckel prennent en compte, du moins implicitement, le point 2 et conduisent à des algorithmes de coût raisonnable mais négligent le point 1.

Une autre alternative fournissant des résultats intéressants consiste à approximer la fonction des niveaux de gris de l'image par une fonction spline [35].

On peut raisonnablement penser que l'utilisation de mathématiques plus sophistiquées pourra permettre de définir des modèles prenant en compte à la fois une définition explicite du bruit et des propriétés géométriques locales. Cependant il semble peu vraisemblable que de tels modèles débouchent sur des algorithmes de coût raisonnable. On peut alors espérer que l'émergence des machines parallèles ou d'ordinateurs séquentiels plus puissants permettra leur implantation.

2.11. Résultats expérimentaux

La méthode reposant sur le filtrage séparable récursif que nous avons décrite a été appliquée aux cas 2D [43], 3D [128], et 4D [159] en utilisant les filtres de Shen et de Deriche. On obtient ainsi de bons résultats, au moins meilleurs que ceux des méthodes plus anciennes, sur une grande variété d'images : images 2D de scènes d'intérieur et d'extérieur, images aériennes, images médicales 3D obtenues par résonance magnétique nucléaire, images 4D constituées par des séquences temporelles d'images médicales 3D ...

Nous présentons dans cette section, à titre indicatif, quelques résultats expérimentaux (figures 2.30. à 2.11.). Sur une station de travail SUN-4 et pour une image $512 \times 512 \times 512$ le temps total de calcul incluant le filtrage, l'extraction des extréma et le seuillage par hystéresis est de l'ordre de 30 secondes CPU.

Figure 2.30. *Image originale d'une scène d'intérieur.*

Figure 2.31. *Valeur absolue du gradient en X de l'image précédente calculé avec le filtre Deriche ($\alpha = 1$).*

Figure 2.32. *Valeur absolue du gradient en Y de l'image précédente calculé avec le filtre Deriche ($\alpha = 1$).*

Figure 2.33. *Extréma locaux de la norme du gradient dans la direction du gradient (la valeur affichée est la norme du gradient).*

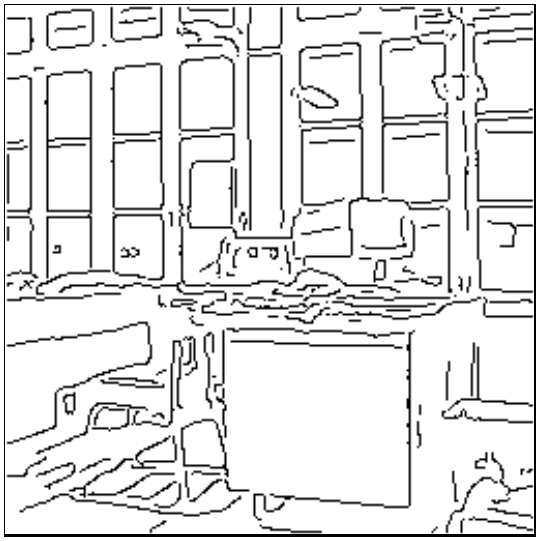


Figure 2.34. *Contours obtenus après seuillage des extréma locaux ($\alpha = 1$).*

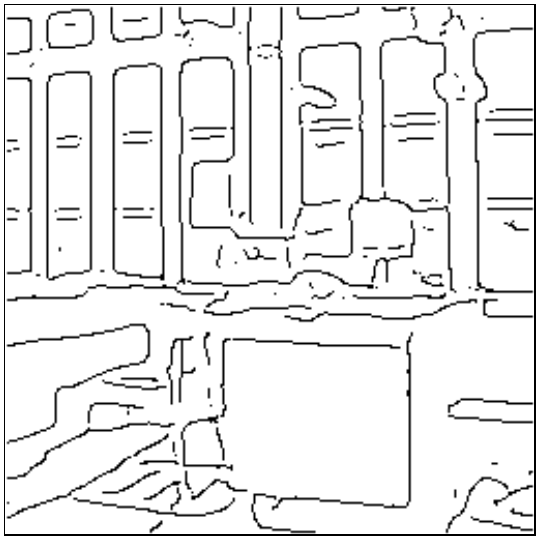


Figure 2.35. *Contours obtenus après seuillage des extréma locaux ($\alpha = 0.5$).*

Figure 2.36. *Contours obtenus après seuillage des extréma locaux ($\alpha = 1.5$).*

Figure 2.37. *Image originale (coupe sagittale du cerveau obtenue par IRM).*

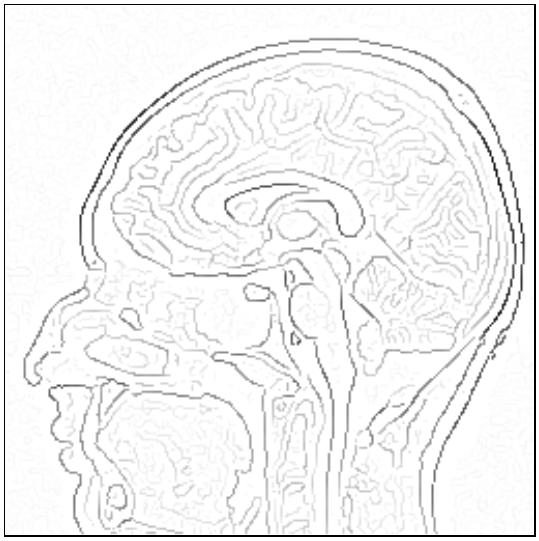


Figure 2.38. *Extréma locaux obtenus avec le filtre Deriche.*

Figure 2.39. *Contours obtenus par seuillage par hystéresis des extréma locaux.*

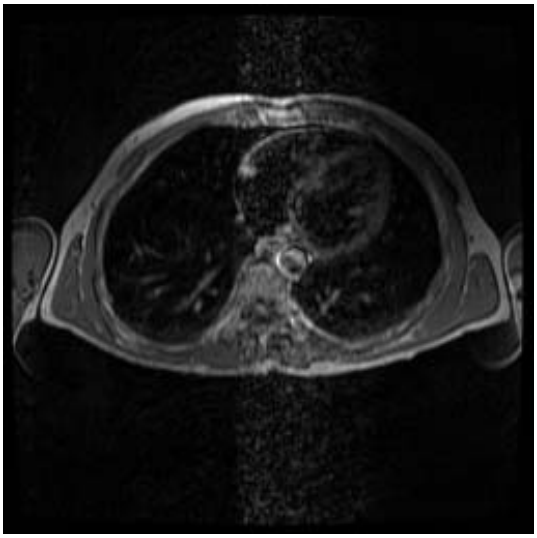


Figure 2.40. *Image originale (coupe d'une image IRM du thorax).*

Figure 2.41. *Extréma locaux obtenus avec le filtre de Deriche.*

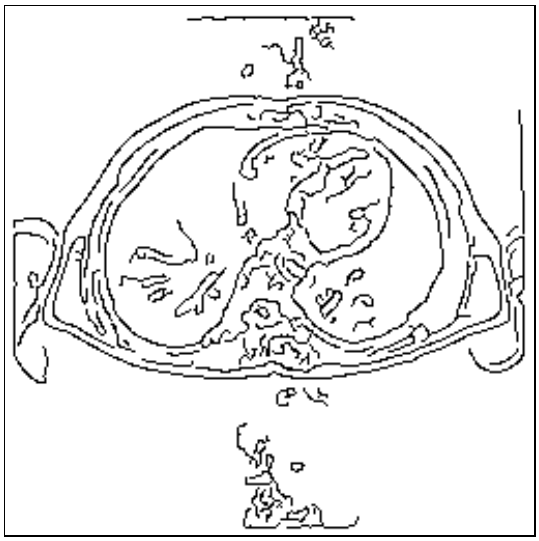


Figure 2.42. *Contours obtenus par seuillage par hystéresis des extréma locaux.*

Figure 2.43. *Image originale.*

Figure 2.44. *Extréma locaux obtenus avec le filtre de Deriche.*

Figure 2.45. *Contours obtenus par seuillage par hysteresis des extréma locaux.*

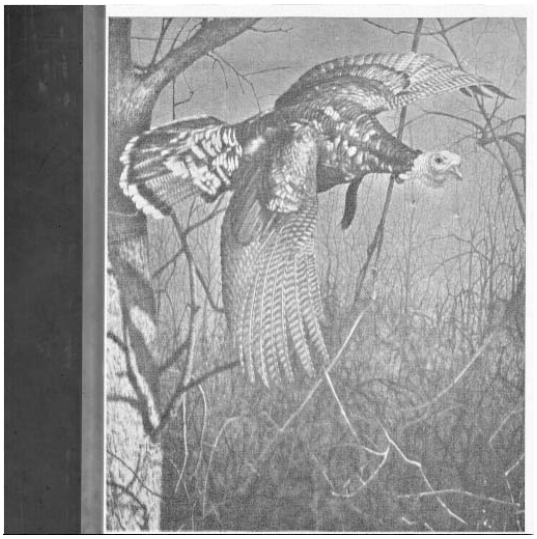


Figure 2.46. *Image originale.*

Figure 2.47. *Extréma locaux obtenus avec le filtre de Deriche.*

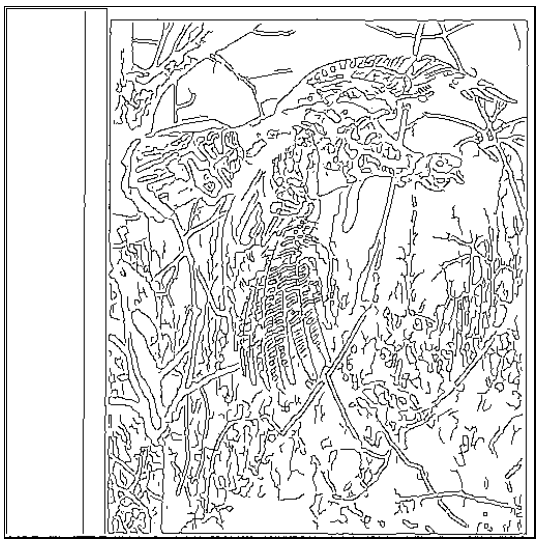


Figure 2.48. *Contours obtenus par seuillage par hysteresis des extréma locaux.*

2.12. Amélioration, suivi et chaînage des contours

Il est souvent difficile de sélectionner des seuils adéquats pour l'étape de seuillage. Des seuils élevés permettent de supprimer les points de contour dus au bruit mais aussi de vrais points de contour. Des seuils bas permettent d'obtenir tous les vrais points de contour mais aussi ceux dus au bruit. Le choix de seuils définit un compromis entre les vrais et les faux contours. La raison de cette limitation des algorithmes de détection de contours réside dans la nature locale des opérateurs. En effet lorsque le bruit de l'image est important il est nécessaire de définir aussi des critères globaux. Ces critères s'expriment généralement par une fonction à optimiser sur un ensemble d'attributs locaux. Le plus souvent on choisit de rechercher dans l'image des chemins optimisant la somme des normes du gradient calculé en chaque point à l'aide d'un opérateur local.

On peut aussi rajouter un critère de forme de contour minimisant les variations de courbure et la longueur des contours seconde (minimisation des courbures). Lorsqu'on prend simplement en compte un critère de normes de gradient, des algorithmes de recherche de chemins optimaux dans un graphe suffisent pour l'optimisation [123]. Si on considère aussi les courbures on utilise alors le calcul variationnel pour minimiser une fonctionnelle incluant la

somme des normes du gradient sur le contour, la longueur et la somme des courbures [102, 38]. On ajuste alors un modèle déformable sur les données, ces algorithmes sont connus sous le nom de "snake" (serpent) mais leur danger réside dans le choix de la fonctionnelle à utiliser (cette fonctionnelle résulte le plus souvent de l'addition de quantités non comparables). Généralement on applique les méthodes de suivi non pas sur toute l'image mais simplement dans les zones où les opérateurs locaux ne donnent pas des résultats satisfaisants. Par exemple, si on suppose que les contours réels sont fermés, on peut mettre en œuvre un algorithme de suivi de contours à partir des extrémités des lignes de contours ouverts.

2.12.0.1. Recherche de chemin optimal dans un graphe

L'idée de ce type d'algorithme est de poser le problème de la détection de contour comme la recherche d'un chemin de coût minimum dans un graphe. Les données sont une carte incomplète des contours et les résultats une meilleure carte des contours. Les composantes critiques de ces méthodes sont :

1. prendre un bon point de départ ;

2. comment les résultats déjà obtenus doivent-ils influencer le choix du point suivant : fonction à optimiser ?

3. critère d'arrêt.

L'approche classique consiste à rechercher un chemin de coût minimum dans le graphe d'adjacence des points de l'image.

On définit une fonction d'évaluation :

- $F(n) = G(n) + H(n)$
- n : état
- F : fonction de coût
- G : coût pour aller de l'état n vers un état de départ
- H : estimation du coût pour aller de l'état n vers un état but.

Un exemple de ce type d'algorithme a été proposé par Martelli [123] :

But : Une ligne de contraste qui commence sur la ligne en haut de l'image et se termine sur la ligne du bas de l'image.

Etats : Eléments de contours définis par deux points :
$A = (i, j)$, $B = (i, j + 1)$ définissant le contour orienté AB

Contour : Séquence d'éléments de contour commençant en haut de l'image et se terminant en bas, ne contenant pas de boucle, et n'ayant aucun élément de direction haut.

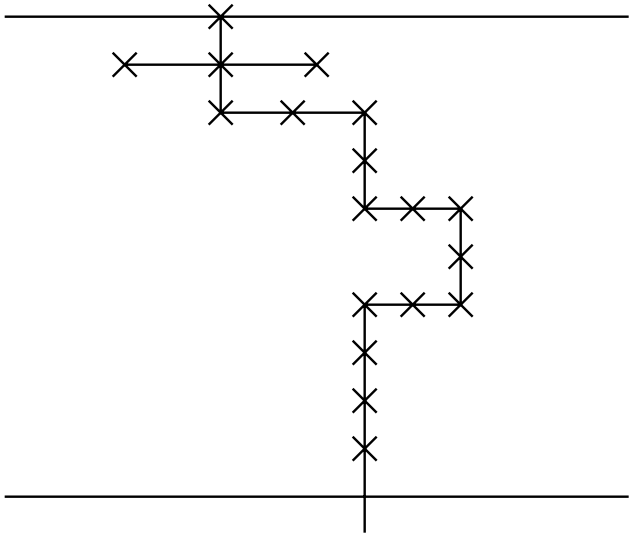


Figure 2.49. *Chemins recherchés.*

Un contour est donc un chemin dans le graphe représentant l'espace des états.

La détermination du meilleur contour se ramène à la recherche d'un chemin optimal dans le graphe.

Soit Max le niveau de gris maximum de l'image I

$$
\begin{aligned}
n &= A_r A_s \\
C(n) &= \text{Max} - I(A_r) + I(A_s) \\
G(n) &= \text{coût pour aller de l'état } n \text{ à un état initial} = \sum_s C(s), \\
&\quad s \text{ parcourant le chemin optimal reliant l'état de départ à l'état } n \\
H(n) &= \text{estimation du coût du chemin optimal reliant l'état } n \text{ à un état final} \\
F(n) &= G(n) + H(n)
\end{aligned}
$$

Pour optimiser F on peut utiliser des méthodes de programmation dynamique ou un algorithme de type A^*.

2.12.1. Chaînage de contours

Dans la plupart des applications la description matricielle des contours fournis par la détection de contours n'est pas suffisante. Il est alors nécessaire de passer à une description sous forme de listes chaînées de contours. Nous décrivons les principes d'un algorithme de chaînage opérant dans le cas 2D uniquement.

L'objectif du chaînage de contours est de passer d'une description matricielle des contours à une description sous forme de listes de contours chaînées. Cette étape qui semble aisée pose en fait des problèmes difficiles :

- le temps de calcul : cette étape n'étant que transitoire, il faut éviter des temps de calcul trop prohibitifs ;
- la représentation des relations entre les chaînes : les relations entre les chaînes doivent être gérées sans perdre d'informations par rapport à l'information matricielle ;
- la structuration de l'information : il faut bien structurer l'information pour que les manipulations de données effectuées par la suite soient aisées.

P. Garnesson et G. Giraudon [68] ont mis au point un algorithme de chaînage qui résout en partie ces problèmes en permettant :

- un chaînage rapide en une seule lecture de l'image ;
- une gestion efficace des liens de parenté ;
- la possibilité d'un stockage étendu pour divers types d'informations (niveau de gris, pixel, amplitude du gradient, ...) ;
- la possibilité d'éliminer des chaînes ;
- la minimisation du nombre de chaînes ;
- la possibilité de travailler avec des stockages d'images réduits ;
- une faisabilité temps réel envisageable pour un coût réduit.

Pour cela, ils ont repris l'idée de décomposition d'un pavé 3 x 3 centré autour d'un pixel, en passé, présent, futur, idée déjà utilisée pour le coloriage de taches ou sur des problèmes de restauration d'images. La méthode se divise en trois étapes principales :

- création des chaînes en parallèle en une passe par un balayage de l'image avec un pavé 3 x 3 ; les chaînes sont créées par mise à jour itérative par de simples jeux de pointeurs ;
- fusion des chaînes obtenues afin de minimiser leur nombre ;
- élimination de chaînes suivant un critère.

L'originalité de cette approche par rapport aux nombreux algorithmes de chaînage existant est de proposer une structure de données "propre" permettant d'effectuer le chaînage et de représenter son résultat. En effet il faut noter que dans les travaux antérieurs, l'aspect structure de données n'avait pas été abordé aussi clairement. Une méthode d'approximation polygonale peut permettre ensuite d'obtenir à partir des chaînes de contours une approximation polygonale des contours (voir le chapitre 3).

- création des chaînes en parallèle en une passe par un balayage de l'image avec un pavé 3 x 3 ; les chaînes sont créées par mise à jour itérative par de simples jeux de pointeurs ;
- fusion des chaînes obtenues afin de minimiser leur nombre ;
- élimination de chaînes suivant un critère.

L'originalité de cette approche par rapport aux nombreux algorithmes de chaînage existant est de proposer une structure de données "propre" permettant d'effectuer le chaînage et de représenter son résultat. En effet il faut noter que dans les travaux antérieurs, l'aspect structure de données n'avait pas été abordé aussi clairement. Une méthode d'approximation polygonale peut permettre ensuite d'obtenir à partir des chaînes de contours une approximation polygonale des contours (voir le chapitre 3).

Chapitre 3

Segmentation de contours

Les chaînes de contours peuvent être partitionnées dans des segments de courbes qui ont une description analytique connue telles des lignes droites et des coniques. La plus simple description d'une courbe est la ligne droite mais dans de nombreux cas ce n'est pas suffisant et l'utilisation de courbes du deuxième ordre s'avère alors nécessaire. Ce chapitre est en grande partie basé sur les travaux de Pavlidis & Horowitz [147].

3.1. Segmentation d'une chaîne

Soit $C = \{c_i, i = 0, 1, \cdots, n\}$ l'ensemble des points d'une chaîne où $c_i = [x_i, y_i]$ est le $i^{ème}$ point de la chaîne avec ses coordonnées dans l'image. Ce point n'a en principe que deux voisins (sauf pour le premier et dernier point) : c_{i-1} et c_{i+1}. Soit S une partition de C. Soit M un sous-ensemble de C contenant les points de cassure correspondant à la partition S. On a :

$$S = \{s_0, s_1, \cdots, s_{N-1}\} \; et \; M = \{m_0, m_1, \cdots, m_N\}$$

Le segment s_k commence au point m_k et se termine au point m_{k+1}. Les éléments m de M appartiennent également à C. Chaque m correspond donc à un c : $m_k = c_{i_k}$. Un segment s_k est donc un sous ensemble de points de C tel que (figure 3.1.) :

$$s_k = \{c_j \mid i_k \leq j \leq i_{k+1},\; c_{ik} = m_k,\; c_{ik+1} = m_{k+1}\}$$

La segmentation d'une chaîne de contours posera donc deux problèmes :

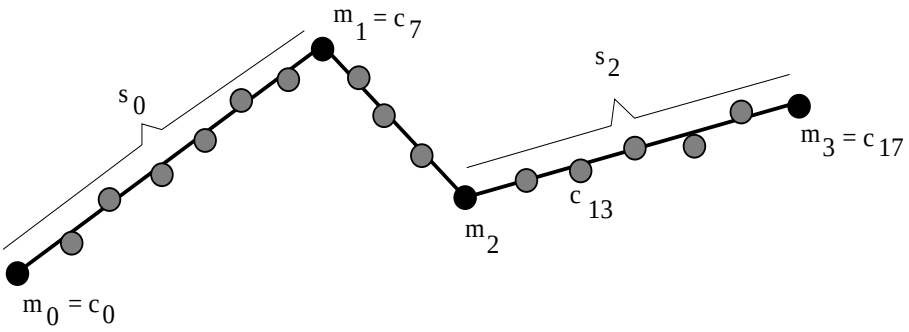


Figure 3.1. *Un exemple de segmentation.*

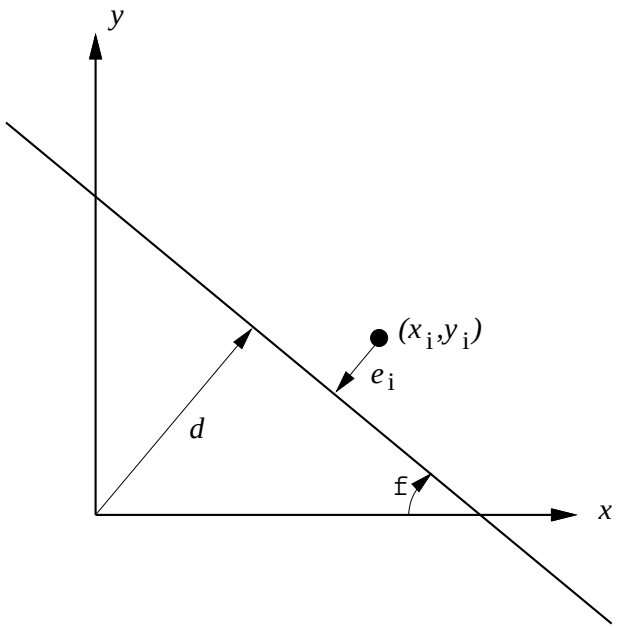


Figure 3.2. *La paramétrisation d'une droite utilise la distance de la droite à l'origine et l'inclinaison de la droite par rapport à l'axe horizontal.*

1. trouver un partionnement de la chaîne en segments $(s_0, \cdots, s_k, \cdots)$ et
2. trouver pour chaque segment s_k la meilleure approximation analytique.

Nous allons commencer par donner une solution simple d'approximation d'un ensemble de points par un segment de droite et par un arc de cercle. Ensuite nous allons décrire quelques méthodes de partionnement d'une chaîne de points en segments.

3.2. Approximer un segment par une droite

Soit l'équation d'une droite (figure 3.2.) :

$$x \ \sin\phi + y \ \cos\phi = d \qquad [3.1]$$

La distance d'un point $[x_i, y_i]$ à cette droite est :

$$e_i = \mid x_i \ \sin\phi + y_i \ \cos\phi - d \ \mid$$

Nous voulons trouver la droite (les paramètres ϕ et d) qui minimise la grandeur :

$$E = \sum_{i=1}^{n} e_i^2 \qquad [3.2]$$

L'ensemble des paramètres de la droite est la solution du système d'équations :

$$\left\{ \begin{array}{rcl} \frac{\partial E}{\partial \phi} & = & 0 \\ \frac{\partial E}{\partial d} & = & 0 \end{array} \right. \qquad [3.3]$$

Après calculs on obtient pour d et ϕ :

$$d = V_x \sin\phi + V_y \cos\phi \qquad [3.4]$$

$$\phi = \frac{1}{2} \arctan \frac{2V_{xy}}{V_{yy} - V_{xx}} \qquad [3.5]$$

avec :

$$\begin{array}{rcl} V_x & = & \frac{1}{n} \sum_{i=1}^{n} x_i \\ V_y & = & \frac{1}{n} \sum_{i=1}^{n} y_i \end{array}$$

$$\begin{aligned} V_{xx} &= \sum_{i=1}^{n}(x_i - V_x)^2 \\ V_{yy} &= \sum_{i=1}^{n}(y_i - V_y)^2 \\ V_{xy} &= \sum_{i=1}^{n}(x_i - V_x)(y_i - V_y) \end{aligned}$$

3.3. Approximer un segment par un cercle

Soit l'équation d'un cercle :

$$(x-a)^2 + (y-b)^2 = r^2 \qquad [3.6]$$

Avec les notations suivantes :

$$\begin{aligned} A &= -2a \\ B &= -2b \\ C &= a^2 + b^2 - r^2 \end{aligned}$$

L'équation du cercle peut s'écrire à nouveau :

$$x^2 + y^2 + Ax + By + C = 0 \qquad [3.7]$$

La distance d'un point à un cercle s'écrit :

$$d_i = \mid \sqrt{(x_i - a)^2 + (y_i - b)^2} - r \mid \qquad [3.8]$$

Cependant ceci conduit à un problème d'optimisation non linéaire. La distance euclidienne d'un point à un cercle peut être remplacée par la distance algébrique :

$$\begin{aligned} e_i &= \mid (x_i - a)^2 + (y_i - b)^2 - r^2 \mid \\ &= \mid x_i^2 + y_i^2 + Ax_i + By_i + C \mid \qquad [3.9] \end{aligned}$$

Le critère à minimiser est :

$$E = \sum_{i=1}^{n} e_i^2 \qquad [3.10]$$

La solution pour A, B et C est donnée par le système d'équations suivant :

$$\left\{ \begin{array}{rcl} \frac{\partial E}{\partial A} & = & 0 \\ \frac{\partial E}{\partial B} & = & 0 \\ \frac{\partial E}{\partial C} & = & 0 \end{array} \right. \qquad [3.11]$$

Après dérivation on obtient :

$$\begin{array}{rcl} U_{xx}\, A + U_{xy}\, B + U_x\, C & = & -U_{xxx} - U_{xyy} \\ U_{xy}\, A + U_{yy}\, B + U_y\, C & = & -U_{xxy} - U_{yyy} \\ U_x\, A + U_y\, B + n\, C & = & -U_{xx} - U_{yy} \end{array} \qquad [3.12]$$

Avec :

$$U_{xy} = \sum_{i=1}^{n} x_i y_i$$

et des définitions similaires pour U_x, U_{xxy}, U_{xyz}, etc.

On obtient donc un système linéaire à trois inconnues, A, B et C, à partir desquelles on retrouve facilement les paramètres du cercle.

3.4. Algorithme de découpage récursif

Supposons pour l'instant qu'on veuille approximer une chaîne de points avec des segments de droite. Dans ce cas un algorithme de découpage récursif peut être le suivant :

1. pour une chaîne de points: est-ce un segment de droite ?
2. si oui – aller en fin ;
3. sinon – diviser la chaîne en deux sous-chaînes et répéter
l 'algorithme pour chaque sous-chaîne ;
4. fin.

Avec les notations précédentes, considérons une sous-chaîne s_k dont les extrémitées sont m_k et m_{k+1}. Si cette sous-chaîne n'est pas correctement approximée par un segment de droite (au sens du critère quadratique énoncé précédemment), une façon "naturelle" d'améliorer cette approximation est de chercher parmi les points de la sous-chaîne s_k celui qui est le plus éloigné de la corde définie par les extrémités de s_k. On casse alors s_k en deux sous-chaînes, s'_k et s''_k, et on leur applique le même traitement...

La figure 3.3. illustre une implantation efficace de l'algorithme de découpage récursif. La procédure utilise deux tableaux, CLOSED et OPEN. Initialement ces tableaux contiennent respectivement le premier et dernier points de la chaîne, A et B. Si la chaîne comprise entre A et B ne peut être approximée par un segment de droite (ce qui est le cas), le point le plus éloigné de la corde AB (C) est choisi comme point de cassure. Ce point est rajouté dans le tableau OPEN. A chaque pas d'itération de la procédure on considère la chaîne comprise entre le dernier point entré dans CLOSED et le dernier point entré dans OPEN. Lorsqu'un point de cassure est trouvé, il est rajouté à la fin du tableau OPEN. Lorsqu'une chaîne peut être approximée par un segment, le dernier point entré dans OPEN passe dans CLOSED.

CLOSED	OPEN	Segment considéré	Sommet créé
A	B	AB	C
A	B, C	AC	D
A	B, C, D	AD	–
A, D	B, C	DC	E
A, D	B, C, E	DE	–
A, D, E	B, C	EC	F
A, D, E	B, C, F	EF	–
A, D, E, F	B, C	FC	–
A, D, E, F, C	B	CB	G
A, D, E, F, C	B, G	CG	–
A, D, E, F, C, G	B	GB	H
A, D, E, F, C, G	B, G	GH	–
A, D, E, F, C, G, H	B	HB	–
A, D, E, F, C, G, H, B	–	–	–

Figure 3.3. *Un exemple de procédure de découpage récursif.*

3.5. Algorithme de découpage et union

Supposons maintenant qu'on veuille approximer une chaîne par une séquence de segments de droite et d'arcs de cercle. L'algorithme de découpage récursif devient alors :

1. pour une chaîne de points : est-ce un segment de droite ?

2. si oui – aller en fin ;

3. sinon – est-ce un cercle ?

4. si oui – aller en fin ;

5. sinon – diviser la chaîne en deux sous-chaînes et répéter l'algorithme pour chaque sous-chaîne ;

6. fin.

Cet algorithme pose trois problèmes :

1. étant donné un ensemble de points approximé correctement (au sens des moindres carrés) aussi bien par une droite que par un cercle, quelle approximation choisir ? Ce problème est illustré par la figure 3.4. Dans cet exemple nous sommes tentés de choisir un cercle. Nous introduisons un facteur d'évaluation e :

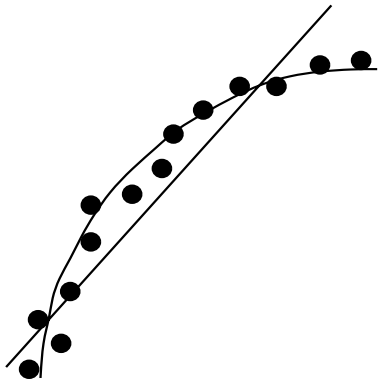


Figure 3.4. *Cette chaîne peut aussi bien être une droite qu'un cercle.*

$$e = \frac{k}{n}\,(1 - \frac{p}{n}) \qquad [3.13]$$

avec :

- n est le nombre de points de la chaîne ;
- k est le nombre de changements de signe de la séquence des vecteurs d'erreur ;
- p est la longueur de la plus grande séquence de vecteurs d'erreurs ayant le même signe.

Dans l'exemple traité, k est grand et p est petit pour un cercle, alors que c'est le contraire pour une droite. Dans ce cas on aura donc :

$$e_{cercle} > e_{droite}$$

2. les points de cassure choisis par l'algorithme de découpage récursif peuvent couper un arc de cercle en deux parties, figure 3.5. Il faudrait donc accompagner la procédure de découpage par une procédure d'union de deux chaînes adjacentes. Une procédure de découpage-et-union est donnée en exemple sur la figure 3.6.

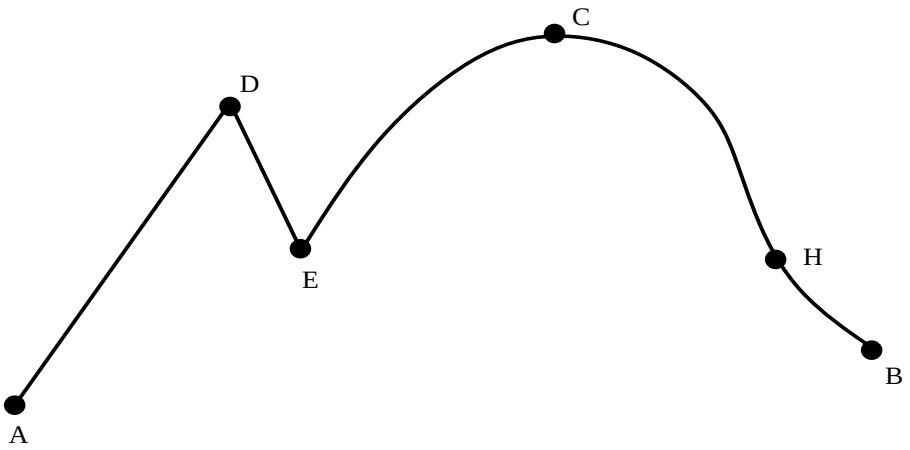


Figure 3.5. *La procédure de découpage récursif a coupé un arc de cercle en deux morceaux.*

3. lors du choix d'un point de cassure, sa position peut être mauvaise par rapport à une position théorique. C'est souvent le cas lorsqu'une droite est tangente à un cercle, le point de tangence étant difficile à localiser, figure 3.7. Soit E_k et E_{k+1} les erreurs d'approximation associées à deux segments adjacents et soit m_{k+1} leur point de cassure. On veut trouver la position d'un nouveau point de cassure m'_{k+1} tel que :

$$\begin{aligned} |m'_{k+1} - m_{k+1}| &< D_{max} \\ E'_k &< E_k \\ E'_{k+1} &< E_{k+1} \end{aligned}$$

CLOSED	OPEN	Segments considérés	Sommet enlevé
A	D, E, C, H, B	ADE	–
A, D	E, C, H, B	DEC	–
A, D, E	C, H, B	ECH	C
A, D, E	H, B	EHB	–
A, D, E, H	B	–	–
A, D, E, H, B	–	–	–

Figure 3.6. *Un exemple de découpage-et-union.*

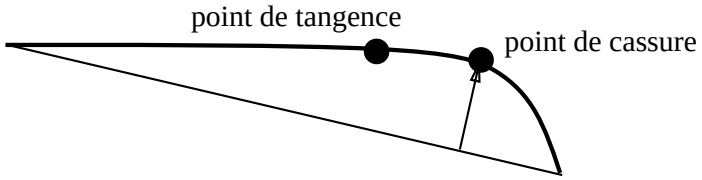


Figure 3.7. *Une droite tangente à un cercle.*

On définit donc un intervalle autour du point de cassure initial à l'intérieur duquel on doit se déplacer. Le nouveau point de cassure doit être tel que les erreurs associées aux deux segments adjacents soient diminuées.

3.6. Un exemple

Nous allons illustrer à l'aide d'un exemple simple la segmentation en lignes droites obtenues par un algorithme de découpage et union tel qu'il vient d'être décrit. La figure 3.8. montre une image comportant beaucoup de lignes droites (gauche) ainsi que le résultat de l'extraction de contours (droite). La figure 3.9. montre le résultat de l'approximation polygonale. On remarque que dans ce cas le choix de l'erreur maximale d'approximation n'est pas du tout critique.

3.7. La transformée de Hough

Il est possible de "grouper" des points de contour colinéaires sans passer par l'étape de chaînage. On peut utiliser pour cela une méthode qui consiste à faire

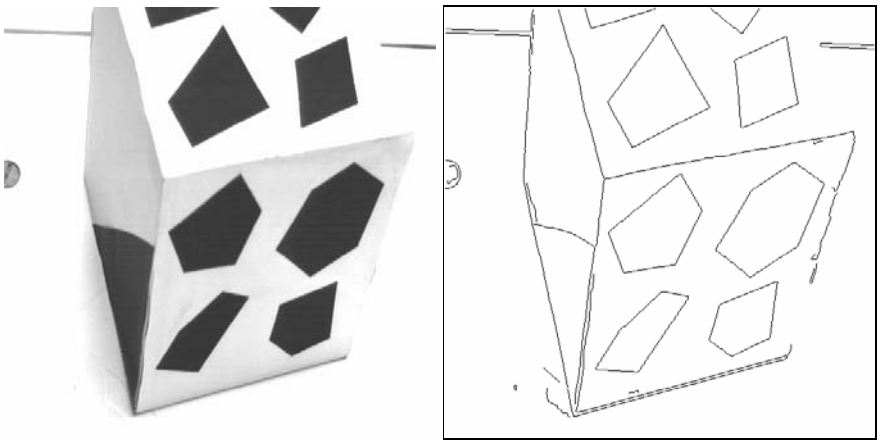


Figure 3.8. *Une image à niveaux de gris et les contours extraits de cette image utilisant le filtre de Deriche.*

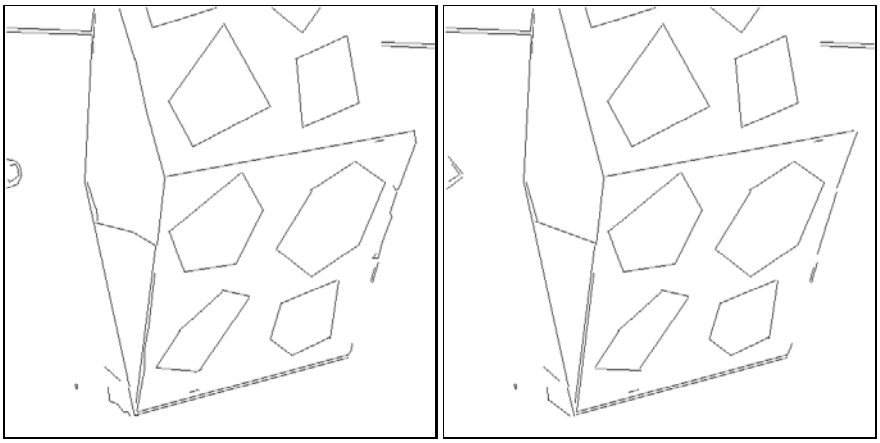


Figure 3.9. *Les segments de droite obtenus avec un algorithme de decoupage et union. Dans l'image de gauche nous avons utilisé une erreur maximale d'approximation de 0.4 pixels alors que dans l'image de droite nous avons utilisé une erreur de 5 pixels.*

correspondre à des points de contour du plan image des points d'un espace de paramètres. Cette technique s'appelle "transformée de Hough", du nom de son inventeur. Conceptuellement elle peut s'étendre pour détecter des courbes de forme quelconque. On trouvera un traitement complet de la tranformée de Hough dans [14].

Soit l'équation d'une droite telle que nous l'avons déjà utilisée :

$$x \sin \phi + y \cos \phi = d$$

Si de plus $\phi \in [0, \pi]$, les paramètres de la droite sont uniques. A chaque droite du plan $x - y$ correspond un point dans le plan $\phi - d$.

Considérons maintenant un point de contour du plan image. Ce point est caractérisé par sa position dans l'image, $[x_i, y_i]$ *et* par son orientation ϕ_i. On peut donc définir une droite qui passe par ce point et qui a la direction du contour :

$$x_i \sin \phi_i + y_i \cos \phi_i = d_i$$

A chaque point-contour du plan image correspond donc un point dans le plan $\phi - d$. Soient n points-contour colinéaires. On aura : $\phi_1 = \cdots = \phi_n$ et $d_1 = \cdots = d_n$. A tous ces points colinéaires il correspond *le même point* dans le plan des paramètres.

Afin d'utiliser pratiquement cette technique nous devons discrétiser le plan des paramètres en N×M cellules. A chaque point-contour du plan image correspond donc une cellule du plan $\phi - d$. De plus on associe un compteur à chaque cellule. Chaque fois qu'une cellule reçoit un "vote" de l'image on incrémente son compteur. Des points de contour colinéaires voterons pour la même cellule, le compteur correspondant contiendra par conséquent une valeur élevée par rapport au compteur d'une cellule qui ne correspond pas à des points de contour colinéaires, figure 3.10.

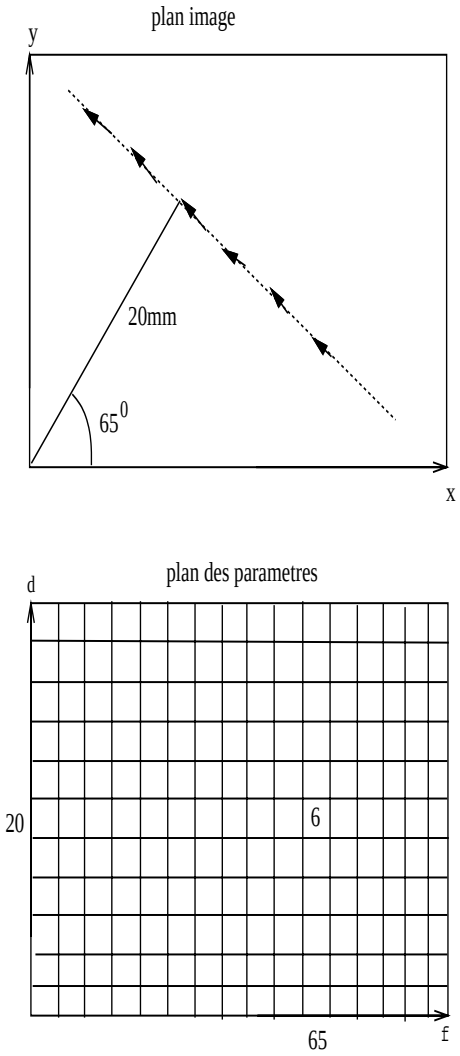


Figure 3.10. *Détection de droites avec la transformée de Hough.*

Chapitre 4

Segmentation d'images en régions

L'objectif de la segmentation d'images en régions est de partitionner une image en zones d'intérêt correspondant à des objets de la scène d'où elle est issue. Elle peut être située dans le cadre plus général de la segmentation de données.

Le problème de base de la segmentation de données est le suivant :

On connaît :

- un ensemble d'**entités**
- un ensemble d'**attributs** caractérisant ces entités
- des **relations topologiques** entre ces entités
- des **attributs relationnels** entre ces entités

On cherche :

- une (ou des) **partition(s) de ces données** ayant des propriétés intéressantes par rapport aux attributs et aux relations topologiques.

Les principaux problèmes qui se posent sont :

- définir les **propriétés des partitions** que l'on cherche
- concevoir des **algorithmes** permettant l'obtention de **partitions optimisant ces propriétés**.

Pour le problème spécifique de la segmentation d'images :

- les **entités** sont des **points d'images** par exemple bidimensionnelles ou tridimensionnelles.
- les **attributs** sont la **position dans l'espace** et les **luminances**.
- les **relations topologiques** sont la 4-connexité ou la 8-connexité.

• les **attributs relationnels** sont les caractéristiques des **frontières** entre deux ensembles d'entités.

On peut donc caractériser un problème de **segmentation d'images** par :

• **un ensemble de critères d'homogénéité** déterminant les propriétés des partitions de l'image que l'on cherche : ***modélisation analytique***.

• un algorithme utilisant ces critères de manière à segmenter l'image : *modélisation algorithmique*.

On est donc confronté à deux problèmes de base distincts :

• la définition de **critères de segmentation** spécifiques à chaque problème particulier de segmentation.

• la définition d'une **stratégie d'utilisation des critères** de segmentation qui peut être définie de manière générale.

On peut noter que dans le cas où les propriétés d'homogénéité des régions s'expriment simplement, il est possible de dériver un opérateur d'extraction de contour qui détecte les frontières entre les régions. On peut ensuite déterminer les régions par fermeture des contours ainsi obtenus. Un exemple de ce type d'approche est présenté ultérieurement. Mais souvent la recherche de filtres aisément implantables permettant de détecter les ruptures d'une propriété d'homogénéité d'ordre supérieur à 0 reste complexe. Ce constat a motivé le nombre important de travaux consacrés à la segmentation d'images en régions.

Les méthodes existantes peuvent être rangées dans deux catégories : les méthodes de "classification", et les méthodes de "croissance de régions". La différence principale entre ces deux types d'approches réside dans la stratégie d'utilisation de l'espace des luminances et des relations spatiales existant entre les pixels. Les méthodes de classification déterminent d'abord une partition de l'espace des luminances et se servent ensuite des relations de connexité pour déterminer les régions. Les méthodes de croissance de régions utilisent de manière simultanée ces deux types d'information.

4.1. Problématique : un problème d'optimisation NP-difficile

L'objectif de cette section est de montrer que la segmentation en régions dans des images de dimension quelconque se ramène à un problème d'optimisation. Nous verrons qu'il est le plus souvent NP-difficile, ce qui nécessite l'introduction d'heuristiques auxquelles est consacrée la section suivante.

La segmentation d'une image I utilisant un prédicat d'homogénéité P est communément définie [184, 98, 148] comme une partition $S = R_1, R_2, \ldots R_n$ de I telle que :

1. $I = \cup\ R_i,\ \ i \in [1 \ldots n]$
2. R_i est connexe, $\forall i \in [1 \ldots n]$
3. $P(R_i) =$ vrai, $\forall i \in [1 \ldots n]$
4. $P(R_i \cup R_j) =$ faux, $\forall i \neq j$, pour tout couple (R_i, R_j) de régions connexes.

Il est important de remarquer que les conditions 1., 2., 3., 4. ne définissent pas, en général, une segmentation unique. Les résultats de segmentation dépendent par conséquent de l'ordre et de la manière avec lesquelles les données sont traitées et non pas seulement de l'information contenue dans l'image. Par exemple, dans beaucoup de méthodes une segmentation calculée sur la transposée d'une image n'est pas similaire à la transposée de la segmentation de l'image. Il est possible de réduire ce problème d'indétermination en ajoutant une contrainte d'optimisation d'une fonction C caractérisant la qualité d'une segmentation [127]. Soit Q une fonction caractérisant la qualité d'un élément de S (par exemple la variance des niveaux de gris des pixels d'une région). Soit C une fonction monotone et symétrique par rapport à la qualité de chaque élément de S : $C(S) = C(Q(R_1), \ldots Q(R_n))$ (par exemple la moyenne arithmétique de $Q(R_1), \ldots Q(R_n)$). Nous ajoutons aux axiomes 1. $\rightarrow$ 4. le suivant :

5. parmi toutes les segmentations possibles S vérifiant 1., 2., 3., nous recherchons la (ou une) segmentation S^* qui optimise la fonction de qualité c'est-à-dire telle que :

$$C(S^*) \leq C(S)\ \forall S \in S_P(I)$$

où $S_P(I)$ est l'ensemble des partitions de I, et C une fonction décroissante.

Très souvent C est additive, elle peut être par exemple dans le cas d'images 2-D la fonction C_1 définie comme suit :

$C_1(S) = \sum_{i=1,n} \sum_{(k,l) \in R_i} (I(k,l) - M_i)^2$; n étant le nombre des régions;

$M_i = \frac{\sum_{(k,l) \in R_i} I(k,l)}{Card(R_i)}$; $Card(R_i)$ étant le nombre de points d'une région R_i.

Il est important de remarquer que généralement les propriétés désirées pour les régions peuvent être décomposées hiérarchiquement de manière à simplifier leur optimisation [127]. Une suite $(P_1, C_1), \ldots (P_n, C_n)$ sera ainsi utilisée au lieu d'un simple couple (P, C). Une telle décomposition permet de réduire, en le divisant le problème d'optimisation. Cette nécessité peut s'illustrer par

un exemple simple. Supposons que nous voulions déterminer des angles droits à partir d'un ensemble de points. Dans une première étape, nous utiliserons un couple (P_1, C_1) permettant l'obtention de segments de droites à partir des points. Dans une deuxième étape, nous utiliserons un couple (P_2, C_2) permettant la formation d'angles droits avec les segments.

Il est clair que la condition 5. ne définit pas toujours une segmentation unique. En effet plusieurs segmentations ayant la même valeur minimale de C peuvent exister. Quoi qu'il en soit, cette condition permet d'obtenir une définition plus précise mais pose un problème algorithmique sous-tendant une complexité importante. C'est pourquoi la recherche d'algorithmes sous-optimaux est nécessaire.

4.2. Segmentation en régions par classification

Ces méthodes déterminent d'abord une partition de l'espace des luminances en utilisant les niveaux de gris présents dans l'image. On associe à chaque pixel la classe de niveaux de gris à laquelle il appartient. Les régions sont définies par les ensembles maximaux de pixels connexes appartenant à la même classe. Les prédicats d'homogénéité et les fonctions de qualité (voir paragraphe précédent) ne sont pas définis explicitement dans ce type d'algorithme. Le plus souvent la classification des luminances s'effectue à partir du calcul de l'histogramme de répartition dans l'image. On recherche les différents modes de l'histogramme et les "vallées" correspondantes. Les classes sont déterminées par les intervalles entre les vallées. Cette procédure fonctionne bien pour des images comprenant un nombre peu important d'objets ayant des niveaux d'intensité différents. Olhander [145] a amélioré cette idée en introduisant une classification récursive permettant d'opérer quand un nombre important d'objets est présent dans l'image. On définit un masque sélectionnant tous les pixels de l'image (un masque définit une zone de l'image). Pour chaque masque un histogramme de l'image masquée est calculé. Les modes de l'histogramme sont détectés permettant ainsi de segmenter l'espace des niveaux de gris. Les points de l'image sont étiquetés avec la classe correspondante. Les ensembles maximaux de pixels connexes appartenant à une même classe sont déterminés. Si l'histogramme comprend plus qu'un mode alors le masque est terminé. Sinon tous les ensembles connexes précédemment calculés sont utilisés pour générer des masques qui sont placés dans une pile de masques. Durant les itérations

successives le masque courant détermine les pixels utilisés pour le calcul d'histogrammes. La classification est répétée pour chaque nouveau masque jusqu'à ce que la pile soit vide. Le nombre des modes de l'histogramme déterminés à chaque étape est un paramètre de cette procédure ; par exemple si ce nombre est réduit à l'unité on détermine à chaque étape le meilleur pic. Ohta, Kanade et Sakai [144] ont adapté cet algorithme pour la segmentation d'images couleur. Dans le cas où l'image contient un objet se détachant sur un fond, des techniques de classification par seuillage adaptatif ont été développées [34], [174]-[176]. Ces méthodes combinent l'information spatiale de l'image avec l'information de luminance pour aider la détermination des seuils. L'itération de ce type d'algorithme peut permettre la segmentation d'images comprenant plusieurs objets.

Chow et Kaneko [34] calculent en chaque point un seuil dépendant de l'histogramme de répartition des luminances sur son voisinage. Des voisinages carrés 33×33 ou 65×65 sont par exemple utilisés pour la détermination de ces "histogrammes locaux". De manière à éviter le calcul d'un histogramme pour chaque point Chow et Kaneko partitionnent l'image en blocs, calculent l'histogramme pour chaque bloc, déterminent un seuillage approprié pour chaque histogramme, et ensuite interpolent spatialement les valeurs des seuils de manière à obtenir un seuil adaptatif en chaque pixel.

Weszca, Nagel et Rosenfeld [176] suggèrent de déterminer un histogramme en prenant seulement en compte les pixels de laplacien faible. Seules les luminances à l'extérieur des zones de fort gradient donc a priori à l'intérieur de zones homogènes peu bruitées sont ainsi prises en compte.

Watanabe [174] propose de choisir une valeur de seuil qui maximise la somme des gradients calculée sur tous les pixels dont le niveau de gris est égal à la valeur du seuil.

Ces méthodes de "segmentation par classification" se révèlent efficaces si la classification des luminances permet de mettre en évidence les différentes régions homogènes de l'image. Dans le cas d'images comprenant des objets de luminances différentes se détachant sur un fond, cette approche donne de bons résultats. Mais lorsque les images sont bruitées et contiennent un nombre important d'objets la classification se révèle peu utilisable. Dans ce cas il est nécessaire d'utiliser les relations spatiales tout au long du processus de segmentation. Cette idée a été développée dans les méthodes dites de "croissance de régions".

4.3. Segmentation par croissance de régions

Ces techniques consistent à regrouper itérativement des ensembles de points connexes en régions plus importantes en utilisant des conditions dépendant de propriétés d'homogénéité. L'idée de base de ce type d'approche est de définir des critères de regroupement des pixels permettant l'obtention de régions homogènes. Les relations spatiales sont ainsi utilisées tout au long du processus de segmentation. Ces méthodes peuvent être subdivisées en deux classes : "agrégation de points" et "regroupement itératif d'ensembles de points".

4.3.1. Agrégation de points

Ces algorithmes associent à chaque pixel un vecteur de propriétés. Deux pixels sont regroupés si leurs vecteurs de propriétés sont "suffisamment similaires". Le résultat de la segmentation est constitué par les composantes connexes déterminées. Divers vecteurs de propriétés et diverses mesures de similarité ont été proposées [6, 81].

L'heuristique la plus simple consiste à réunir deux pixels si leur différence de niveau de gris est suffisament faible. Bryant [30] normalise cette différence par la moyenne des différences calculées sur tous les couples de pixels voisins de l'image. Asano et Yokoya [6] regroupent deux pixels si leur différence de niveaux de gris est faible par rapport à la plus grande différence existant entre chacun des pixels et ses voisins pris dans un petit voisinage carré.

Nagao et Matsuyama utilisent ce type d'algorithme pour la segmentation d'images couleur [139]. Un pixel est regroupé avec un noyau de région si la norme SUP entre son vecteur de composantes (dans R, V, B) et un vecteur de composantes quelconques d'un point du noyau est majorée par un seuil. Une expansion point à point permet d'obtenir un ensemble de régions dont l'amplitude de variation des intensités des points selon les composantes rouge (R), vert (V), bleu (B) est majorée par le seuil.

Des critères de regroupement plus sophistiqués attachant à chaque pixel un vecteur de propriétés dépendant d'un voisinage $k \times k$ autour du pixel ont été employés. Souvent le vecteur propriété est issu d'un opérateur de détection de contour [81, 80]. Les pixels non séparés par un contour sont associés. Les résultats obtenus dépendent évidemment du détecteur de contour utilisé.

Les limitations de ce type d'approche par agrégation de points sont liées au

fait que les entités que l'on regroupe (les pixels) véhiculent peu d'informations. En effet ces méthodes consistent à déterminer des noyaux de régions et ensuite à regrouper itérativement des points à ces noyaux. Les seules informations dont on dispose pour décider d'un regroupement sont les caractéristiques du noyau de régions et la luminance du point. Les méthodes par regroupement itératif d'ensembles de points permettent d'utiliser des informations plus riches pour former les régions.

4.3.2. Regroupement itératif d'ensemble de points

L'idée de ces méthodes est de définir une succession de partitions de l'image par regroupement itératif de régions connexes. Cette stratégie permet de définir des heuristiques de regroupement dépendant de la similarité des propriétés des régions.

La première approche de ce type a été proposée par Muerle et Allen [138]. Ils suggèrent de regrouper deux régions adjacentes si leurs distributions de niveaux de gris sont suffisament similaires. Ils recommandent d'utiliser le test de Kolmogorov-Smirnov.

Brice et Fennema [29] effectuent la croissance de régions en partitionnant l'image en ensembles initiaux de points de même intensité. Ils regroupent ensuite séquentiellement les couples de régions adjacentes dont une fraction significative de la frontière a un contraste faible. Horowitz et Pavlidis utilisent un "quad tree" pour représenter l'image. Une stratégie de fusion et d'explosion ("split and merge") utilisant le quad-tree est ensuite mise en œuvre [98]. On obtient ainsi un ensemble de régions dont l'amplitude de variation des luminances est majorée par un seuil.

La méthode développée par Pong et al. [155] est basée sur l'utilisation séquentielle de plusieurs algorithmes : l'image est d'abord segmentée en utilisant un "sloped facet model" [82] ; ensuite la segmentation initiale sert d'entrée à un algorithme de croissance de régions. Deux régions adjacentes sont fusionnées si un critère de similarité est vérifié.

La plupart de ces méthodes regroupent itérativement tous les couples de régions adjacentes vérifiant certaines conditions. Le processus de croissance s'arrête quand plus aucun couple de régions adjacentes ne vérifie les conditions de fusion. Généralement le résultat obtenu dépend de l'ordre avec lequel les couples de régions sont regroupés. Ainsi le principe des méthodes les plus

efficaces est de définir une hiérarchie de fusion.

4.3.2.1. Croissance hiérarchique de régions

On décrit un algorithme sous-optimal illustrant la définition de la segmentation proposée précédemment. L'idée de base de cet algorithme est d'optimiser la qualité globale de la segmentation (S) par une optimisation locale [127, 65]. Il consiste à regrouper parmi tous les couples de régions dont la réunion vérifie le prédicat (P) celui ayant la meilleure qualité locale (Q). Il s'écrit comme suit :

S = I

tant qu'un couple de régions connexes dont l'union vérifie P existe faire :

- regrouper parmi tous les couples (R_i, R_j) dont l'union vérifie P celui minimisant $Q(R_i \cup R_j)$
- mettre à jour S

Cet algorithme de regroupement optimise l'estimation globale de la qualité de la segmentation par optimisation locale de Q. Le fait que le coût de fusion (Q) soit optimisé sur l'image entière permet au processus de fusion d'être fortement guidé par les données. On ne définit pas de sens de parcours ou de traitement des données. L'ordre avec lequel l'image est traitée n'est pas déterminé a priori comme dans la plupart des algorithmes de segmentation. Cet algorithme peut être itéré pour utiliser une suite de critères de fusion : $(P_1, Q_1) \ldots (P_n, Q_n)$.

Au premier abord cet algorithme semble sous-tendre une complexité importante. En effet le choix du meilleur couple de régions connexes par rapport à (P_i, Q_i) amène en général des implantations coûteuses. Or si certaines conditions relatives aux (P_i, Q_i) sont vérifiées, une implantation de faible complexité algorithmique est possible [127]. On va maintenant décrire ces propriétés. On donnera tout d'abord quelques définitions, et ensuite les hypothèses nécessaires sur les (P_i, Q_i).

4.3.2.1.1. Définitions

Soit $E = e_1, e_2, \ldots e_n$ une partition initiale de l'image ; E peut être l'image ou une segmentation initiale.

Soit $P(E)$ l'ensemble des sous-ensembles de E.

Soit R l'ensemble des réels.

Soit k le nombre d'attributs associés à un sous-ensemble de E.

Soit A une fonction de $P(E)$ vers R^k qui associe un vecteur d'attributs à un sous-ensemble de E (par exemple pour une image de niveaux de gris : l'aire, la moyenne ou la variance de niveau de gris, ...). Soit m le nombre d'attributs associés à un couple de sous-ensembles de E.

Soit G une fonction de $P(E)^2$ vers R^m qui associe un vecteur d'attributs à un couple de sous-ensembles de E (par exemple pour une image de niveaux de gris : le gradient moyen à la frontière).

Soit S une partition de $E, S = \{E_1, E_2, \ldots E_p\}$.

On associe à S l'ensemble des attributs de ses éléments et l'ensemble des attributs de ses paires d'éléments respectivement $V(S)$ et $W(S)$:

$$\begin{aligned} V(S) &= \{A(E_1), A(E_2), \ldots A(E_p)\} \\ W(S) &= \{G(E_r, E_s); r, s \in [1 \ldots p]\} \end{aligned}$$

Chaque triplet $(S, V(S), W(S))$ définit et caractérise une segmentation de E.

On définit l'action consistant à fusionner deux éléments de S E_i et E_j. Cette opération consiste à définir un nouveau triplet :

$$(S, V(S), W(S))_{(i,j)} = (R, V(R), W(R))$$

$$\begin{aligned} R &= \{E_1, \ldots, E_{i-1}, E_i \cup E_j, E_{i+1}, \ldots, E_{j-1}, E_{j+1}, \ldots, E_{p-1}\} \\ V(R) &= \{A(_1), \ldots, A(E_{i-1}), A(E_i \cup E_j), \ldots, A(E_{p-1})\} \\ W(R) &= \{G(E_p, E_q), p \neq i, q \neq j, G(E_q, E_i \cup E_j) \ldots\} \end{aligned}$$

4.3.2.1.2. Conditions pour une implantation rapide

On suppose que les conditions (A_1) et (A_2) suivantes sont vérifiées :
(A_1) *: La suite de couples : (Prédicat de fusion, Coût de fusion)* $= (P, Q)$ *définissant les heuristiques de regroupement est telle que :*

$\forall E, F \in S$ *nous avons :*

$$\begin{aligned} P(E \cup F) &= P(A(E), A(F), G(E, F)) \\ Q(E \cup F) &= Q(A(E), A(F), G(E, F)). \end{aligned}$$

Cela signifie que le prédicat d'homogénéité et la fonction de qualité de la réunion de deux régions peuvent être calculés à partir de leurs attributs.
(A_2) *: Il existe deux fonctions T et U telles que :*

T est une fonction de $(R^k)^2$ vers R^k telle que :

$$\forall S_1, S_2 \in P(E); A(S_1 \cup S_2) = T(A(S_1), A(S_2))$$

Cela signifie que les attributs de la réunion de deux régions peuvent être calculés à partir des attributs des deux régions.

U est une fonction de $(R^m)^2$ dans R^m telle que :

$$\forall S_1, S_2, S_3 \in P(E); G(S_1, S_2 \cup S_3) = U(G(S_1, S_2), G(S_1, S_3))$$

Cela signifie que les attributs du couple $(S_1, S_2 \cup S_3)$ peuvent être calculés à partir de ceux de (S_1, S_2) et (S_1, S_3).

Les fonctions T et U permettent de mettre à jour aisément $(S, V(S), W(S))$ après une fusion. Ainsi, si un triplet initial $(S, V(S), W(S))$ et les fonctions T et U sont données, nous pouvons effectuer une suite de fusions en mettant à jour itérativement le triplet courant.

4.3.2.1.3. Remarques

Les conditions (A_1) et (A_2) permettent de réaliser une implantation de l' algorithme dont la complexité est faible. Dans beaucoup de problèmes de segmentation on peut trouver des critères de segmentation qui vérifient ces conditions. Le problème principal posé par la condition (A_1) est que le prédicat ainsi que le coût de fusion sont uniquement des fonctions du couple de régions. En effet, aucune information contextuelle (concernant, par exemple, les attributs des régions voisines) ne peut être insérée dans un couple (P_i, Q_i) vérifiant (A_1). Pour le cas de l'optimisation d'une fonction globale de qualité Ci monotone et symétrique par rapport aux qualités locales Q_i, cette restriction n'est pas gênante. En effet, une telle estimation de la valeur d'une segmentation ne nécessite pas, pour être optimisée, d'information contextuelle.

La caractérisation des attributs induite par (A_2) implique une restriction sur la nature des attributs utilisés. L'existence de la fonction T signifie que

les attributs associés à une région peuvent être calculés à partir des attributs associés à un ensemble quelconque de régions formant une partition de la région. L'existence de la fonction U implique que les attributs associés à un couple de régions peuvent être déterminés à partir de ceux des couples du produit cartésien de deux partitions quelconques des deux régions. Pratiquement, quand un attribut intéressant pour un problème de segmentation ne vérifie pas la condition (A_2) nous cherchons à l'exprimer en fonction d'attributs la vérifiant. D'un point de vue théorique une telle décomposition existe toujours ; le pire des cas étant quand elle utilise tous les points de la région. Nous avons évidemment intérêt à employer des attributs qui peuvent être exprimés avec un nombre réduit d'attributs vérifiant (A_2). Nous allons illustrer ces remarques par le cas simple de la variance des niveaux de gris souvent utilisée pour la segmentation des images naturelles. La variance des niveaux de gris de deux régions R_1 et R_2 n'est pas uniquement fonction de la variance de R_1 et de la variance de R_2. Cependant elle peut être calculée si on connait l'aire, la somme des niveaux de gris, et la somme des carrés des niveaux de gris des deux régions R_1 et R_2. Ces trois attributs vérifient la propriété (A_2), en effet le vecteur attribut incluant respectivement ces trois attributs est associé à la fonction T suivante : $T((x_1, x_2, x_3), (y_1, y_2, y_3)) = (x_1 + y_1, x_2 + y_2, x_3 + y_3)$. Ainsi si nous voulons utiliser la variance des niveaux de gris, pour segmenter des images naturelles 2-D par exemple, nous inclurons dans les attributs associés aux régions : l'aire, la somme des niveaux de gris, et la somme des carrés des niveaux de gris. De même la matrice de covariance d'un ensemble de points 2-D ou 3-D peut être calculée aisément à partir des sommes des produits de coordonnées. Les mêmes remarques concernant la condition (A_2) peuvent être faites au sujet des attributs associés aux couples de régions tels que : nombre de points et gradient moyen le long de la frontière.

4.3.2.1.4. Implantation

On représente un triplet $(S, V(S), W(S))$ par un graphe valué. On associe à chaque nœud du graphe un élément de S et le vecteur attribut qui le caractérise. Nous associons à chaque arc du graphe un couple d'éléments de S et le vecteur attribut qui caractérise les relations existant entre les deux nœuds. Chaque nœud du graphe correspond à la réunion d'un ensemble de nœuds issus du graphe initial -le graphe initial étant obtenu à partir d'une segmentation initiale

(dans le pire des cas l'image). Pratiquement des arcs sont créés seulement entre des régions vérifiant des propriétés intéressantes par exemple de connexité ou de recouvrement [127, 130].

Le processus de croissance de régions consiste donc à calculer le graphe initial et à itérer les deux actions suivantes :

1. détermination du couple de nœuds à regrouper.

2. mise à jour du graphe après le regroupement des deux nœuds correspondant.

L'étape 1 nécessite un accès rapide au couple de nœuds vérifiant le prédicat courant et optimisant la fonction de coût. Il est donc nécessaire de définir une structure de données permettant de représenter efficacement un ensemble d'objets muni d'une relation d'ordre et sur lequel doivent être appliqué les opérations suivantes :

- insérer
- ôter le minimum

La structure de **tas (heap)** permet d'effectuer ces opérations avec un coût de l'ordre du logarithme du nombre d'éléments pris en compte. On trouvera une description du tas et de son implantation dans [177].
L'étape 2 nécessite un **accès rapide** aux arcs incluant un nœud donné ainsi qu'**aux attributs** correspondant. On peut donc choisir une représentation du graphe permettant un accès direct à l'ensemble des arcs comprenant un nœud donné ainsi qu'aux attributs correspondants.
Le graphe est représenté par les **listes d'arcs associées à chaque nœud et des listes d'attributs associées aux nœuds et aux arcs**.

On spécifie ainsi un **problème de segmentation** par :

- un **graphe initial** issu d'une partition initiale et les attributs associés à ses nœuds et à ses arcs.

- **deux fonctions de mise à jour** attachées respectivement aux attributs-nœuds et aux attributs-arcs (T et U).

- **une suite de couples (Prédicat de fusion, Coût de fusion)** qui définit la stratégie de regroupement $((P_1, Q_1), \ldots (P_n, Q_n))$.

4.3.2.1.5. Complexité

Discutons maintenant de la complexité algorithmique de cet algorithme. Les calculs provoqués par la fusion de deux nœuds sont donc les suivants :

• calcul des attributs du nouveau nœud en utilisant la fonction de mise à jour T.

• calcul des attributs des arcs incluant le nouveau nœud en utilisant la fonction de mise à jour U.

Nous devons ajouter aux calculs précédents ceux dus à la détermination du couple à fusionner et à la mise à jour du tas c'est-à-dire :

• prendre la tête du tas jusqu'à ce qu'un arc valide soit extrait.

• calcul du coût attaché aux nouveaux arcs et rangement des pointeurs correspondants dans le tas.

Plus formellement, la complexité algorithmique peut être calculée comme suit :

Soient N et B respectivement le nombre de nœuds et le nombre d'arcs du graphe initial.

Soit F le nombre de fusions effectuées durant la segmentation.

Chaque nœud est connecté en moyenne avec $V = \frac{2B}{N}$ nœuds ; on suppose que V reste à peu près constant durant le processus.

Soit T_1 le nombre d'opérations requises pour le calcul de la fonction T.

Soit T_2 le nombre d'opérations requises pour le calcul de la fonction U.

Soit T_3 le nombre d'opérations requises pour le calcul du prédicat et de la fonction de coût.

Soit M le nombre moyen de nœuds adjacents vérifiant le prédicat de fusion pour un nœud donné du graphe initial ; on suppose que M reste constant durant le processus de regroupement.

Le calcul nécessité par la mise à jour de graphe lors des fusions est : $FT_1 + FT_2V$.

Pour mettre à jour **tas** lors d'une fusion nous insérons dans le tas en moyenne V nouveaux arcs et ainsi nous calculons V nouvelles valeurs de prédicats et de coût. De plus nous insérons dans le tas les nouveaux arcs qui vérifient le prédicat. Nous pouvons raisonnablement supposer que le nombre maximum d'éléments dans le tas est à peu près $V.M$.

Le nombre d'opérations requises pour la mise à jour du tas lors des fusions est donc approximativement :

$$FT_3V + FMV\ O(LOG(VM))$$

Le second terme de la somme ci-dessus correspond au coût de l'insertion des valeurs des nouveaux arcs créés par le regroupement.

La complexité totale de notre algorithme est donc :

$$O(FT_1 + FT_2V + FT_3V + FMVLOG(VM))$$

Ce résultat montre l'importance de V et M pour la complexité de notre algorithme. M caractérise l'état du graphe initial par rapport au prédicat de fusion. On a donc intérêt à utiliser une bonne méthode (!) pour déterminer une segmentation initiale. Nous remarquons que la complexité est une fonction de la "densité d'arc" du graphe. Nous aurons donc intérêt à limiter le nombre de relations entre les nœuds du graphe.

Cet algorithme de segmentation par croissance de régions s'applique pour des images de dimensions quelconques, monochromatiques et multichromatiques, ainsi que pour la segmentation de surfaces 3D [127, 130]. Il faut noter qu'il peut prendre en compte le résultat d'une détection de contours afin de contrôler la croissance des régions avec des propriétés de discontinuité [131]. Nous donnons un exemple d'application de cette méthode pour des images 2D monochromatiques [127].

4.3.2.1.6. Application aux images monochromatiques

De manière à appliquer cette méthode de segmentation aux images monochromatiques il faut établir une hiérarchie des propriétés désirées pour les régions. Ensuite, on attache à chaque propriété d'homogénéité un prédicat, une fonction globale de qualité, et une fonction locale de qualité. On détermine alors les attributs associés aux régions et aux couples de régions permettant de calculer le prédicat d'homogénéité et la fonction locale de qualité. On montre enfin que ces prédicats et ces fonctions vérifient les propriétés définies dans le paragraphe précédent.

La détermination de critères de segmentation s'appliquant à une classe importante d'images naturelles monochromatiques n'est pas aisée. En effet la

multiplicité des facteurs physiques intervenant dans la formation d'une image rend difficile le calcul de fonctions d'homogénéité permettant de différencier les projections des divers objets de la scène. L'expérimentation montre que des critères simples donnent le plus souvent des résultats comparables à ceux obtenus à l'aide de critères compliqués et que l'on maîtrise mal. Il est donc intéressant d'utiliser une suite de critères simples permettant de caractériser une région homogène au sens des niveaux de gris.

Le premier critère basé sur l'amplitude des variations des niveaux de gris à l'intérieur d'une région permet de déterminer les régions de luminance quasiment uniforme (cette amplitude de variation est obtenue en effectuant la différence entre le niveau de gris le plus haut et le niveau de gris le plus bas).

Le deuxième critère, basé sur la variance des niveaux de gris, engendre la formation de régions homogènes mais bruitées. Les régions de luminance à peu près constante sont ainsi regroupées afin de former des régions de variance faible.

Le troisième critère permet de fusionner les couples de régions voisines de luminances moyennes sensiblement différentes, mais présentant une frontière avec un faible gradient.

On remarquera que ces trois critères ont une complexité calculatoire croissante. De manière générale, plus on avance dans le processus de regroupement, plus les informations nécessaires à la croissance des régions deviennent complexes. Mais en contrepartie, le nombre de régions diminuant, le coût du processus de fusion n'est pas affecté. L'ordre d'application des critères est important car il correspond à une description hiérarchique des régions recherchées. Il serait par exemple illogique d'utiliser d'abord la variance des niveaux de gris et ensuite leur amplitude de variation. En effet, lorsque le processus de regroupement débute avec tous les pixels de l'image la variance des niveaux de gris d'une région est moins significative et plus coûteuse à calculer que l'amplitude de leurs variations. Lorsque les régions de luminance uniforme se sont formées, alors seulement la variance devient un critère intéressant. De même, lorsque les régions homogènes bruitées sont déterminées alors le gradient sur la frontière devient significatif. Les informations utilisées par le processus de croissance de régions évoluent donc au cours de celui-ci, correspondant à des niveaux de description différents de l'image.

Plus concrètement, on suppose qu'une région peut être hiérarchiquement décrite comme un ensemble connexe de points tel que :

1. la somme des gradients calculés sur ses points intérieurs est faible, i.e. des contours ne la traversent pas,

2. la région peut être partitionnée en sous-régions dont la variance des niveaux de gris est faible,

3. chaque sous-région de faible variance de niveau de gris peut être elle-même décomposée en sous-régions dont les pixels ont à peu près la même luminance.

La troisième propriété peut être caractérisée par le prédicat P_1, la fonction de qualité globale C_1 et la fonction de qualité locale Q_1 :

Soit $I(i,j)$ l'image initiale Soit $S = \{R_1, R_2, \ldots R_n\}$ une segmentation de I

$P_1(R_i) = [(MAX_i - MIN_i) < s_1]$; s_1 est un seuil ; MIN_i et MAX_i sont respectivement la plus petite et la plus grande valeur prises par I dans la région R_i.

$$C_1(S) = \sum_{i=1,n} \sum_{(k,l)\in R_i} ((I(k,l) - MAX_i)^2 + (I(k,l) - MIN_i)^2)$$

$$Q_1(R_i) = MAX_i - MIN_i$$

La seconde propriété peut être caractérisée par le prédicat P_2, la fonction de qualité globale C_2 et la fonction de qualité locale Q_2 :

$P_2(R_i) = [V(R_i) < s_2]$; s_2 est un seuil ; $V(R_i)$ est la variance des niveaux de gris des pixels de R_i.

s_2 est pris égal à s_1^2 de manière à ce que les régions issues de la première étape vérifient P_2 :

Notons $E(R_i)$ et $C(R_i)$ les moyennes respectives des niveaux de gris et du carré des niveaux de gris de la région R_i. Nous avons :

$MAX_i - MIN_i < s_1 \Rightarrow |\ I(k,l) - I(r,s)\ | < s_1; \forall (k,l)$ et (r,s) points de R_i (I est la fonction des niveaux de gris) $\Rightarrow (I(k,l) - I(r,s))^2 < s_1^2 \Rightarrow C(R_i) - E(R_i)^2 < s_1^2 \Rightarrow V(R_i) < s_1^2$

$C_2(S) = \sum_{(i,j)\in I}(I(i,j) - M_{(i,j)})^2$; $M_{(i,j)}$ est la moyenne des niveaux de gris de la région où (i,j) est inclus.

$$Q_2(R_i) = V(R_i)$$

La première propriété peut être exprimée par le prédicat P_3, la fonction de qualité globale C_3 et la fonction de qualité locale Q_3 :

Soient R_1 et R_2 deux ensembles de points connexes et S une segmentation.

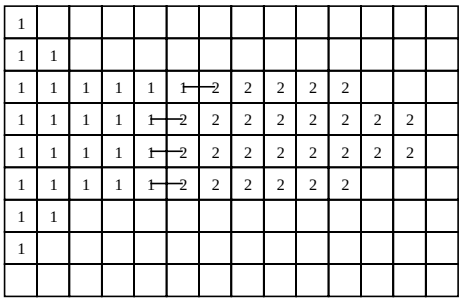


Figure 4.1. *Les couples de points déterminés par les extrémités des segments définissent la frontière entre la région "1" et la région "2" : F("1","2").*

Soit $F(R_1, R_2)$ l'ensemble des couples de points définissant la frontière entre R_1 et R_2 (voir figure 4.1.).

Soit $N_F(R_1, R_2)$ le nombre de couples de points de $F(R_1, R_2)$.

Soit $D(R_1, R_2)$ le gradient moyen à la frontière entre R_1 et R_2 :

$$D(R_1, R_2) = (\sum ((i,j),(k,l)) \in F(R_1, R_2) \frac{\mid I(i,j) - I(k,l) \mid)}{(N_F(R_1, R_2))}$$

$C_3(S) = \sum_{((i,j),(k,l)) \in X_s(I)} \mid I(i,j) - I(k,l) \mid$; $X_s(I)$ est l'ensemble des couples de points connexes de l'image I inclus dans une même région S.

Nous supposons que pendant le processus de fusion :

$P_3(R1 \cup R_2) = [D(R_1, R_2) < s_3]$; s_3 est un seuil.

s_3 est pris supérieur à la valeur maximum de $D(R_1, R_2)$ sur les couples de régions adjacentes issues de l'étape précédente.

$$Q_3(R_1 \cup R_2) = D(R_1, R_2)$$

Le choix des différents seuils s_1, s_2, s_3 implique $P_1 < P_2 < P_3$.

$<$ est le symbole de la relation d'ordre partiel sur les prédicats :

Soient P et Q deux prédicats définis sur un ensemble E $P < Q$ signifie que si P est vrai alors Q est vrai.

On peut montrer que les conditions (A_1) et (A_2) définies précédemment sont vérifiées [127].

On spécifie ainsi ce problème de segmentation par les couples :

$$(P_1, Q_1), (P_2, Q_2), (P_3, Q_3)$$

Les attributs attachés à chaque région sont donc : l'aire, la somme des niveaux de gris, la somme des carrés des niveaux de gris, le maximum et le minimum des niveaux de gris ; et les attributs associés à chaque couple de régions : le nombre de couples de points définissant la frontière et la somme des différences de niveaux de gris entre ces points.

Lors d'une première étape on détermine une segmentation initiale afin de diminuer le coût du processus de regroupement. Nous utilisons la procédure MERGE de Pavlidis avec le prédicat d'homogénéité P_1.

Ensuite on construit le graphe d'adjacence correspondant aux régions obtenues de manière à appliquer la stratégie de regroupement définie par :

$$(P_1, Q_1), (P_2, Q_2), (P_3, Q_3)$$

La première étape de fusion permet d'obtenir des régions de luminance à peu près uniformes. Lors de la seconde étape des régions de niveaux de gris homogènes mais bruités se forment, grâce à la troisième les régions de luminance sensiblement différentes mais dont le gradient à la frontière est faible sont fusionnées. On ajoute une dernière étape qui élimine les petites régions.

4.3.2.1.7. D'autres critères de segmentation

On peut aussi utiliser des critères de segmentation basés sur une approximation de la distribution des niveaux de gris dans une région par des polynômes d'ordre 2.

Si on s'intéresse à la discrimination de régions ayant les mêmes statistiques d'ordre 1 les matrices de co-occurrence peuvent aussi être prises en compte par le processus de regroupement.

La même stratégie de regroupement peut être mise en œuvre pour approximer une surface avec des quadriques par morceaux [52].

4.4. Détection de régions par fermeture de contours

Une idée classique, par exemple développée par R. Deriche et J.-P. Cocquerez [46], est de détecter les régions en identifiant les contours fermés constituant leurs frontières. On détermine alors les régions non pas en recherchant des zones homogènes mais en détectant les points frontières entre deux zones homogènes de caractéristiques différentes.

Ces points frontières correspondent aux points de contour et peuvent être calculés avec un algorithme de détection de contours.

Il faut noter que cette approche prend le contrepied de la méthode précédente fondée sur la détermination d'ensembles connexes de points possédant des propriétés d'homogénéité. Les contours issus d'un algorithme de détection de contours sont rarement fermés, aussi les difficultés des approches "détection des régions par fermeture des contours" résident justement dans la fermeture de ces contours. Le succès de ce genre de méthodes dépend donc généralement de la qualité du détecteur de contours utilisé.

R. Deriche et J-P. Cocquerez utilisent les résultats obtenus par le détecteur de contours de R. Deriche décrit précédemment [43]. On remarquera que ce choix conduit à l'obtention de régions dont la distribution des niveaux gris correspond à une fonction de Heavyside (marche d'escalier) additionnée d'un bruit blanc.

L'originalité de cet algorithme est de proposer une méthode de fermeture basée sur un opérateur simple qui permet de fermer les contours en suivant les crêtes de gradient. Les régions sont ensuite obtenues par détection des composantes connexes maximales n'incluant pas de points de contours. Cette méthode a un coût algorithmique très faible (linéaire selon le nombre de points de l'image).

4.4.1. Description d'un algorithme de fermeture de contours

Le principe de cet algorithme [46] est de détecter, par balayage de l'image ligne par ligne avec un opérateur 3×3, les extrémités des contours, puis de fermer ces contours en les prolongeant par suivi des crêtes du gradient.

L'extraction des maxima locaux de la norme du gradient dans la direction du gradient permet l'obtention de contours d'épaisseur 1, le plus souvent.

Ainsi, une extrémité de contour peut être identifiée par un simple examen de son voisinage 3×3.

La topologie de chaque extrémité permet de définir une direction d'exploration pour la fermeture et à chaque configuration d'extrémités est associée la liste des voisins à examiner. Pour élaborer le chemin de crête issu d'un point extrémité il suffit d'examiner 3 voisins en fonction de la topologie de l'extrémité.

L'image après détection de contours et fermeture comprend les pixels marqués à 1 (les contours) et d'autres pixels marqués à 0 qui représentent des

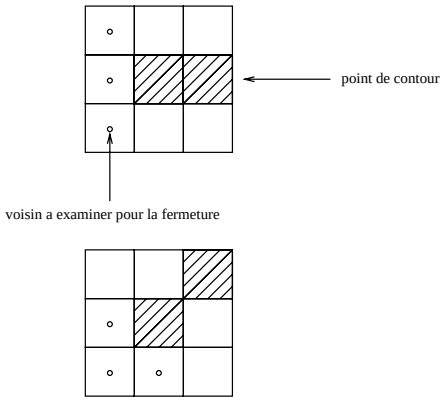


Figure 4.2. *Exemple de voisins à examiner pour la fermeture des contours, dans deux cas de configurations différentes.*

régions.

Le problème est donc d'affecter une même étiquette aux points marqués 0 appartenant à la même composante connexe de l'image. Cette opération est effectuée par simple balayage de l'image par un opérateur en L. Les classes d'équivalence sont mises à jour à chaque nouvelle application de l'opérateur.

4.5. Résultats expérimentaux

Nous présentons à titre indicatif des résultats obtenus avec l' algorithme de croissance optimale de régions. On remarque que les résultats obtenus avec ces deux types d'algorithmes (croissance de régions, fermeture de contours) sont comparables du moins *pour des images de scène d'intérieur*. On notera aussi que les contours des régions correspondent le plus souvent aux contours extraits par détection de contours. Les temps de calcul sur une station de travail SUN-3 sont pour une image 256×256 de l'ordre de 30 secondes CPU pour la croissance de régions.

Figure 4.3. *Image originale.*

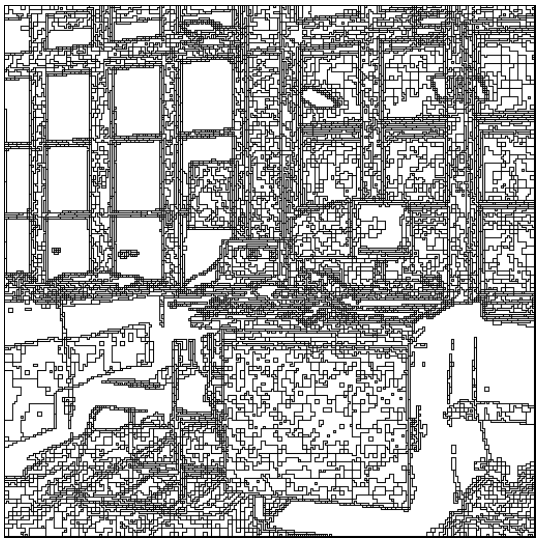


Figure 4.4. *Utilisation du critère max-min (croissance optimale de régions).*

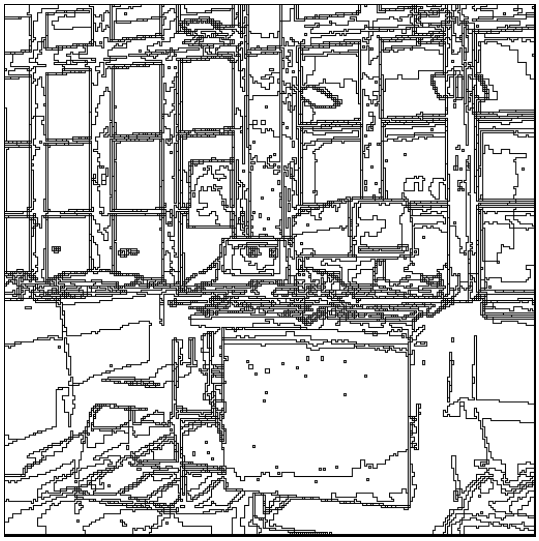


Figure 4.5. *Utilisation du critère moyenne (croissance optimale de régions).*

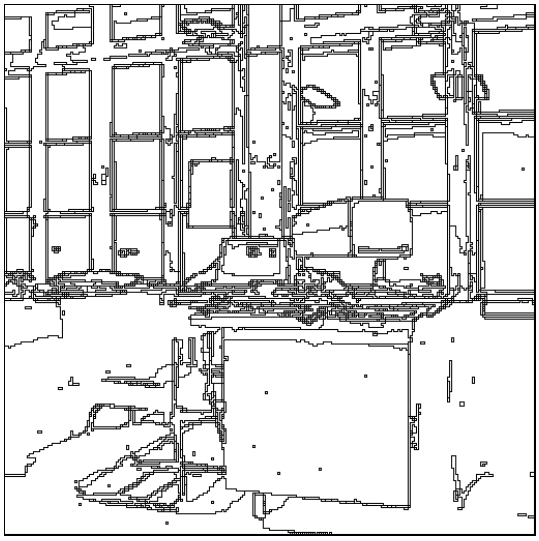


Figure 4.6. *Utilisation du critère gradient (croissance optimale de régions).*

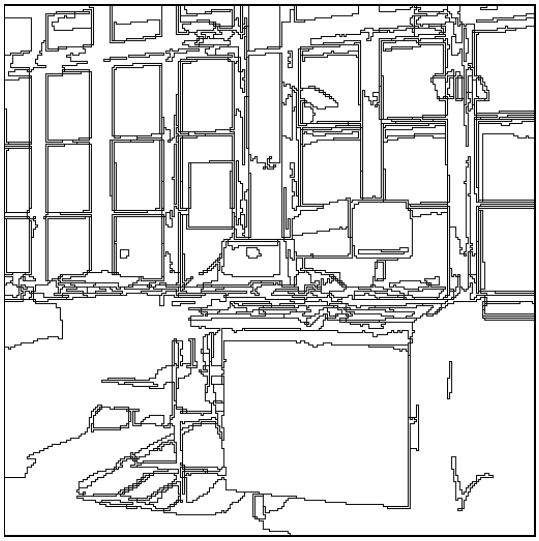


Figure 4.7. *Segmentation finale après élimination des petites régions (croissance optimale de régions).*

Figure 4.8. *Segmentation finale où les régions sont coloriées avec leurs moyennes de niveau de gris.*

Figure 4.9. *Image originale (coupe sagittale de tête obtenue par IRM).*

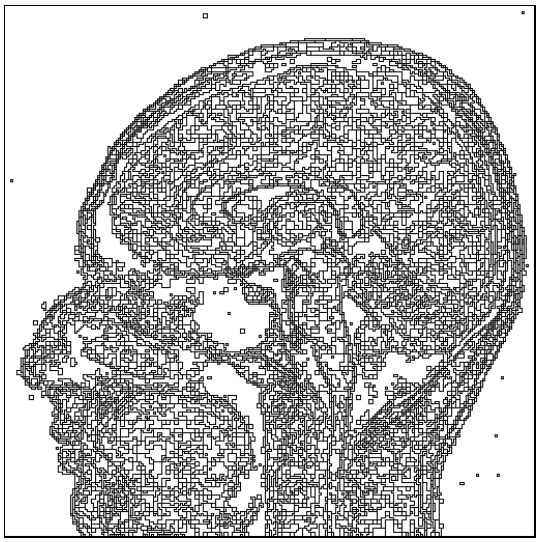


Figure 4.10. *Utilisation du critère max-min (croissance optimale de régions).*

Figure 4.11. *Utilisation du critère moyenne (croissance optimale de régions).*

Figure 4.12. *Utilisation du critère gradient (croissance optimale de régions).*

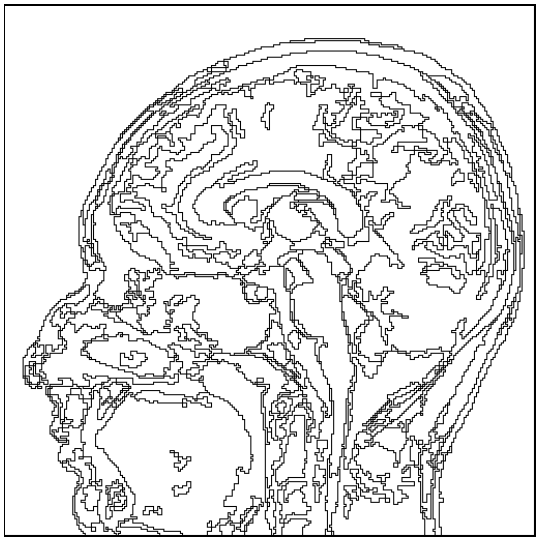


Figure 4.13. *Segmentation finale après élimination des petites régions (croissance optimale de régions).*

Figure 4.14. *Segmentation finale où les régions sont coloriées avec leurs moyennes de niveau de gris.*

Chapitre 5

Géométrie et calibration des caméras

5.1. Introduction

Dans ce chapitre nous allons brièvement décrire le modèle géométrique associé au processus de saisie d'image à l'aide d'une caméra. Ce modèle est caractérisé par un certain nombre de paramètres que nous allons estimer par étalonnage (ou calibration, ou calibrage). Les paramètres d'une caméra ou paramètres intrinsèques seront estimés en même temps que les paramètres extrinsèques, soit les paramètres qui lient un repère associé à la caméra au repère associé à l'objet étalon. Nous allons étudier en détail un modèle projectif de caméra. Nous allons voir par la suite qu'il existe un modèle simplifié – affine – qui peut être utile dans certaines configurations. Nous allons ensuite proposer une technique de calibration d'une caméra linéaire (une barette CCD).

Il est utile de signaler que lorsqu'on calibre une caméra, on s'intéresse à la modélisation de l'ensemble de la caméra et du convertisseur analogique/digital. Si on change de convertisseur, ou de caméra, il faut alors recalibrer l'ensemble [16].

Ensuite nous allons décrire un capteur comportant deux caméras (capteur stéréoscopique) et nous allons préciser quelques propriétés de ce capteur qui nous permettrons de faciliter la tâche de mise en correspondance. Un capteur composé de deux caméras est un capteur tri-dimensionnel puisqu'il nous permet, par *triangulation passive* de calculer la distance d'un objet au capteur. Nous allons étudier également un capteur tri-dimensionnel à *triangulation active* ainsi qu'une technique d'étalonnage d'un tel capteur.

5.2. Le modèle géométrique d'une caméra

Nous allons caractériser le modèle d'une caméra à l'aide de deux transformations :

1. une projection qui transforme un point de l'espace (3D) en un point image (2D) et

2. une transformation d'un repère *métrique* lié à la caméra à un repère lié à l'image.

5.2.1. La projection perspective

Soit un point O dans le plan image appelé point principal et soit une droite perpendiculaire au plan image passant par O, l'axe optique. Soit un point F placé sur l'axe optique à une distance f du plan image. Le point F est le centre de projection et f est la distance focale. On peut placer le centre de projection devant ou derrière le plan image, dans ce qui suit nous allons le placer derrière, comme sur la figure 5.1.

Un point B se projette dans le plan image le long d'une droite passant par B et F. Choisissons un système de coordonnées attaché à la caméra : le plan $x-y$ de ce repère est parallèle au plan image et l'axe des z est confondu avec l'axe optique. L'origine de ce repère se trouve en F. Soient x, y et z les coordonnées du point B dans le repère qu'on vient de décrire. Dans ce même repère, les coordonnées de la projection de B dans le plan image sont :

$$\begin{aligned} x' &= fx/z \\ y' &= fy/z \\ z' &= f \end{aligned}$$

On peut écrire cette transformation sous forme matricielle :

$$\mathbf{P} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f & 0 \end{pmatrix} \qquad [5.1]$$

Nous avons donc adopté les coordonnées homogènes. Les coordonnées homo-

gènes de B sont $(x, y, z, 1)$ et celles de b (sa projection) sont :

$$\begin{pmatrix} sx' \\ sy' \\ sz' \\ s \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Les coordonnées cartésiennes de b sont sx'/s, sy'/s et sz'/s.

5.2.2. Transformation caméra/image

Les points image sont mesurés en *pixels* dans un repère bi-dimensionnel $u-v$ associé à l'image, figure 5.1. Afin de pouvoir écrire la matrice de transformation du repère caméra au repère image nous devons introduire les paramètres suivants : u_0, v_0 et w_0 sont les coordonnées de F, dans le repère image (mesurées en pixels), k_u est le facteur d'échelle vertical (pixels/mm) et k_v est le facteur d'échelle horizontal. En effet, les pixels d'une caméra sont rarement carrés. La transformation du repère caméra au repère image s'écrit (pour le point b) :

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} k_u & 0 & 0 \\ 0 & k_v & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} + \begin{pmatrix} u_0 \\ v_0 \\ w_0 \end{pmatrix}$$

C'est une transformation affine représentant un changement d'échelle, une rotation et une translation. La composante w étant toujours nulle, on peut ignorer la troisième ligne et écrire cette transformation sous la forme d'une matrice 3×4. Cette transformation représente une application linéaire de l'espace projectif vers le plan projectif :

$$\mathbf{K} = \begin{pmatrix} -k_u & 0 & 0 & u_0 \\ 0 & k_v & 0 & v_0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad [5.2]$$

et

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \mathbf{K} \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$

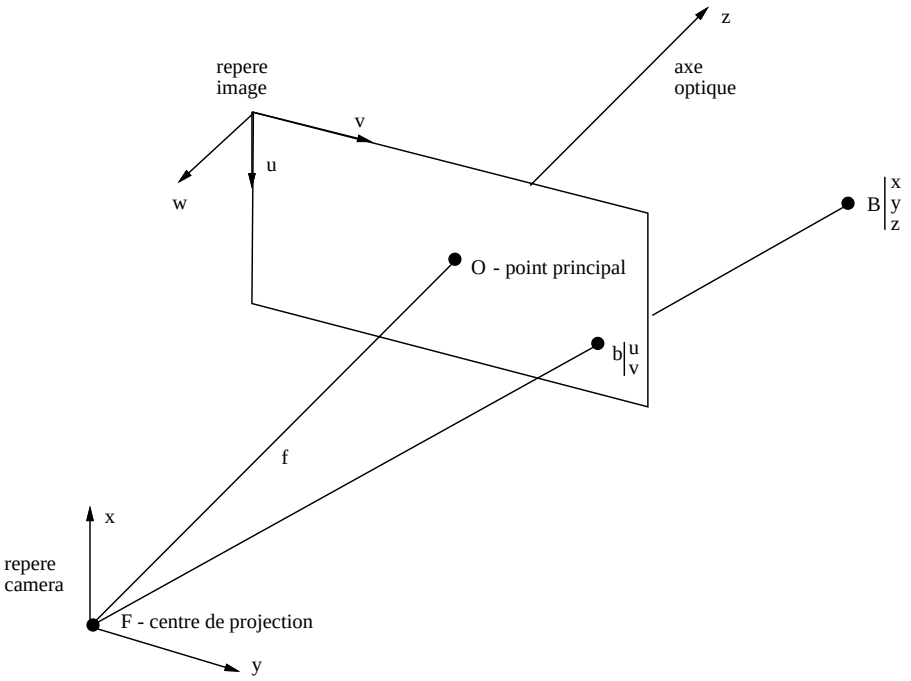


Figure 5.1. *Le modèle géométrique d'une caméra.*

5.2.3. Les paramètres intrinsèques

En multipliant les matrices **K**et **P**(projection perspective suivie d'une transformation affine) nous pouvons écrire les équations du modèle géométrique de la caméra, soit la relation entre les coordonnées caméra $(x\ y\ z)$ du point B et les coordonnées image $(u\ v)$ du point b :

$$\left\{ \begin{array}{rcl} u & = & -k_u f \frac{x}{z} + u_0 \\ v & = & k_v f \frac{y}{z} + v_0 \end{array} \right. \qquad [5.3]$$

En effet, le produit **KP** est :

$$\begin{pmatrix} -k_u & 0 & u_0/f & 0 \\ 0 & k_v & v_0/f & 0 \\ 0 & 0 & 1/f & 0 \end{pmatrix}$$

En multipliant tous les coefficients de la matrice par f (ce qui ne change pas le résultat puisque les coordonnées homogènes sont définies à un facteur

multiplicatif près) on obtient :

$$\mathbf{I}_c = \begin{pmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \qquad [5.4]$$

qui est une application linéaire de l'espace projectif vers le plan projectif exprimant la transformation perspective :

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \mathbf{I}_c \begin{pmatrix} x \\ y \\ z \\ s \end{pmatrix}$$

Ce modèle comporte 4 paramètres : $\alpha_u = -k_u f$, $\alpha_v = k_v f$, u_0 et v_0. Ce sont ces paramètres qui vont être estimés par calibrage. Notons que la distance focale ne peut être calculée explicitement. En introduisant des coordonnées caméra *sans dimension* telles que :

$$\begin{aligned} x_c &= x/z \\ y_c &= y/z \\ z_c &= 1 \end{aligned}$$

on peut maintenant écrire la relation entre les coordonnées image et les coordonnées caméra :

$$\begin{cases} u = \alpha_u x_c + u_0 \quad avec \quad \alpha_u < 0 \\ v = \alpha_v y_c + v_0 \end{cases} \qquad [5.5]$$

soit sous forme matricielle :

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \mathbf{C} \begin{pmatrix} x_c \\ y_c \\ 1 \end{pmatrix}$$

avec :

$$\mathbf{C} = \begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \qquad [5.6]$$

On peut maintenant décomposer la matrice $\mathbf{I}_c$ en une transformation affine caméra-image (soit la matrice $\mathbf{C}$) et une transformation projective :

$$\mathbf{I}_c = \begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

5.2.4. Les paramètres extrinsèques

Afin de déterminer les paramètres du modèle de la caméra, nous allons placer devant la caméra une mire (un objet étalon) : un ensemble de points dont les coordonnées sont parfaitement connues dans un repère de la mire qui est différent du repère caméra, figure 5.2. Chaque point de la mire se projette dans l'image et on mesure ses coordonnées dans le repère image. La transformation mire/image se décompose donc en une transformation mire/caméra suivie d'une projection et suivie enfin d'une transformation caméra/image. La transformation mire/caméra se compose d'une rotation et d'une translation :

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$$

La transformation rigide (rotation et translation) peut s'écrire sous la forme d'une transformation homogène :

$$\mathbf{A} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \qquad [5.7]$$

Cette matrice représente une transformation rigide (3 rotations et 3 translations) : ce sont les paramètres extrinsèques. La matrice inverse est donnée par l'expression suivante :

$$\mathbf{A}^{-1} = \begin{pmatrix} \mathbf{R}^{\,t} & -\mathbf{R}^{\,t}\mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \qquad [5.8]$$

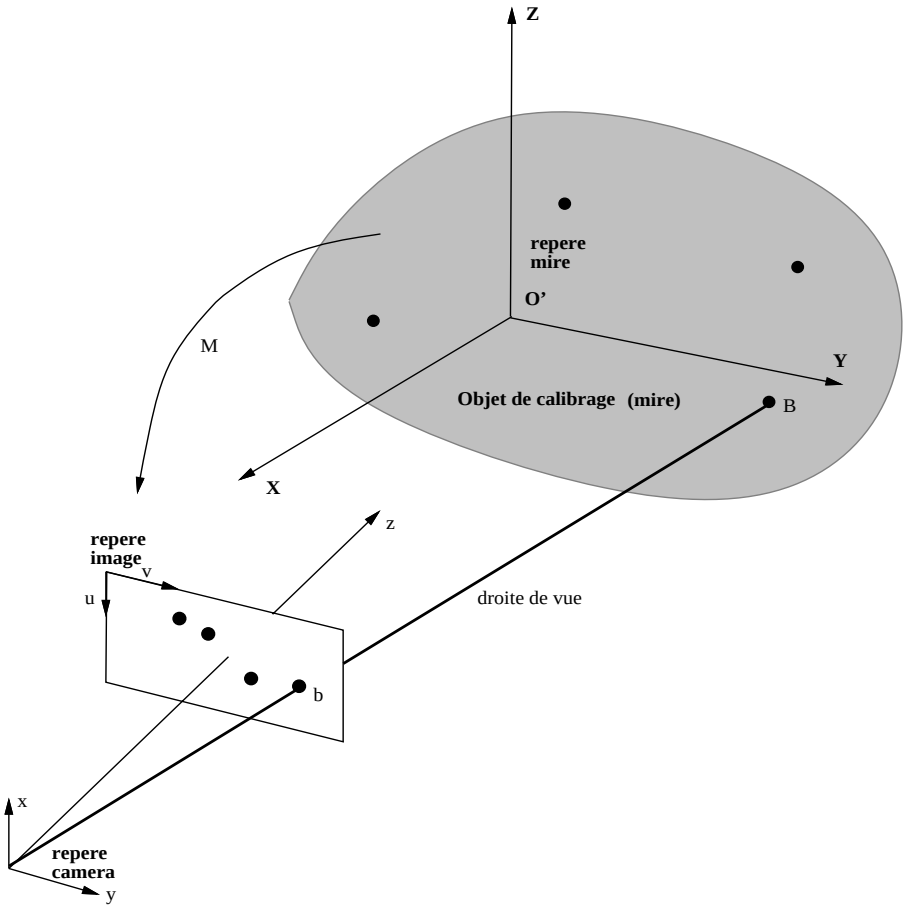


Figure 5.2. *Le principe de calibrage d'une caméra. Un point B dont les coordonnées sont exprimées dans le repère mire se projette en b dont les coordonnées sont exprimées dans le repère image.*

5.2.5. La transformation mire/image

Nous pouvons maintenant écrire la transformation mire/image sous la forme d'une matrice 3×4 appelée matrice de projection perspective et qui peut se décomposer comme suit :

$$\mathbf{M} = \mathbf{I}_c\mathbf{A}$$

$$= \begin{pmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad [5.9]$$

$$= \begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad [5.10]$$

$$= \begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{pmatrix} \quad [5.11]$$

$$= \begin{pmatrix} \alpha_u\mathbf{r}_1 + u_0\mathbf{r}_3 & \alpha_u t_x + u_0 t_z \\ \alpha_v\mathbf{r}_2 + v_0\mathbf{r}_3 & \alpha_v t_y + v_0 t_z \\ \mathbf{r}_3 & t_z \end{pmatrix} \quad [5.12]$$

Dans cette formule la matrice $\mathbf{A}$ a été écrite sous une forme plus compacte en utilisant la notation $\mathbf{r}_1 = (r_{11}\ r_{12}\ r_{13})$:

$$\mathbf{A} = \begin{pmatrix} \mathbf{r}_1 & t_x \\ \mathbf{r}_2 & t_y \\ \mathbf{r}_3 & t_z \\ \mathbf{0} & 1 \end{pmatrix}$$

$\mathbf{M}$ est la matrice de projection perspective et elle peut, en général, s'écrire sous la forme suivante :

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad [5.13]$$

Dans cette formule X, Y et Z sont les coordonnées d'un point B de la mire dans le repère mire. Cette matrice peut également s'écrire sous une forme plus simple :

$$\mathbf{M} = \begin{pmatrix} \mathbf{m}_1 & m_{14} \\ \mathbf{m}_2 & m_{24} \\ \mathbf{m}_3 & m_{34} \end{pmatrix}$$

avec $\mathbf{m}_i = (m_{i1} \ m_{i2} \ m_{i3})$. En identifiant $\mathbf{M}$ avec $\mathbf{I}_c\mathbf{A}$, en tenant compte des propriétés d'orthonormalité de la rotation et en remarquant que l'on doit obtenir une valeur négative pour α_u, on obtient un ensemble d'équations qui permettent de calculer les paramètres intrinsèques et extrinsèques en fonction des coefficients de $\mathbf{M}$. On a :

$$\begin{cases} \mathbf{r}_3 &= \mathbf{m}_3 \\ u_0 &= \mathbf{m}_1 \cdot \mathbf{m}_3 \\ v_0 &= \mathbf{m}_2 \cdot \mathbf{m}_3 \\ \alpha_u &= -\|\mathbf{m}_1 \wedge \mathbf{m}_3\| \\ \alpha_v &= \|\mathbf{m}_2 \wedge \mathbf{m}_3\| \\ \mathbf{r}_1 &= 1/\alpha_u(\mathbf{m}_1 - u_0\mathbf{m}_3) \\ \mathbf{r}_2 &= 1/\alpha_v(\mathbf{m}_2 - v_0\mathbf{m}_3) \\ t_x &= 1/\alpha_u(m_{14} - u_0 m_{34}) \\ t_y &= 1/\alpha_v(m_{24} - v_0 m_{34}) \\ t_z &= m_{34} \end{cases} \qquad [5.14]$$

Afin de trouver les paramètres intrinsèques on doit donc :

1. estimer les coefficients de la matrice de projection $\mathbf{M}$ et

2. extraire les paramètres de la caméra à partir de ces coefficients grâce aux formules données par l'equation [5.14].

5.2.6. Un modèle légèrement amélioré

Les équations précédentes montrent que l'on peut déterminer les 10 paramètres intrinsèques et extrinsèques à partir des coefficients de la matrice de projection perspective, soit 11 paramètres, cette matrice étant définie à un facteur multiplicatif près. Ceci implique qu'il y aurait un 11^e paramètre associé à la caméra. Lequel ? Il pourrait s'agir de l'angle entre l'axe des u et l'axe des v qui a été supposé égal à 90^0. Si ces deux axes ne sont pas perpendiculaires on

peut introduire comme 11^e paramètre θ, l'angle entre ces deux axes, figure 5.3. La relation entre le repère orthonormé $(u-v)$ et le nouveau repère $(u'-v')$

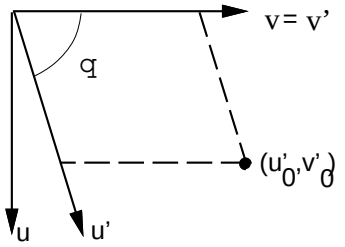


Figure 5.3. *Une image avec des axes non perpendiculaires.*

est :

$$\begin{cases} u' &= u\ \sin\theta + v\ \cos\theta \\ v' &= v \end{cases}$$

La matrice donnée par l'équation [5.6] devient dans ces conditions (on divise par $\sin\theta$) :

$$\mathbf{C}' = \begin{pmatrix} \alpha_u & \alpha_v \cot\theta & u'_0 \\ 0 & \alpha_v / \sin\theta & v'_0 \\ 0 & 0 & 1 \end{pmatrix} \qquad [5.15]$$

avec:

$$\begin{cases} u'_0 &= u_0 + v_0\ \cot\theta \\ v'_0 &= v_0 / \sin\theta \end{cases}$$

La matrice $\mathbf{M}$ s'écrit maintenant :

$$\mathbf{M} = \begin{pmatrix} \alpha_u \mathbf{r}_1 + (\alpha_v \cot\theta)\mathbf{r}_2 + u'_0 \mathbf{r}_3 & \alpha_u t_x + (\alpha_v \cot\theta) t_y + u'_0 t_z \\ (\alpha_v / \sin\theta)\mathbf{r}_2 + v'_0 \mathbf{r}_3 & (\alpha_v / \sin\theta) t_y + v'_0 t_z \\ \mathbf{r}_3 & t_z \end{pmatrix}$$

Les formules de l'équation [5.14] permettant d'exprimer les paramètres intrinsèques en fonction des coefficients de la matrice de projection perspective sont maintenant un peu plus complexes.

On obtient immédiatement u'_0 et v'_0 :

$$\begin{aligned} u'_0 &= \mathbf{m}_1 \cdot \mathbf{m}_3 \\ v'_0 &= \mathbf{m}_2 \cdot \mathbf{m}_3 \end{aligned}$$

En effectuant le produit vectoriel $\mathbf{m}_2 \wedge \mathbf{m}_3$ on obtient :

$$\frac{\alpha_v}{\sin\theta} = \|\mathbf{m}_2 \wedge \mathbf{m}_3\|$$

ainsi que :

$$\mathbf{r}_2 = \frac{\mathbf{m}_2 - (\mathbf{m}_2 \cdot \mathbf{m}_3)\mathbf{m}_3}{\|\mathbf{m}_2 \wedge \mathbf{m}_3\|}$$

Le produit scalaire $\mathbf{m}_1 \cdot \mathbf{m}_2$ nous permet d'obtenir le cosinus de l'angle entre les deux axes :

$$\cos\theta = \frac{(\mathbf{m}_1 \cdot \mathbf{m}_2) - (\mathbf{m}_1 \cdot \mathbf{m}_3)(\mathbf{m}_2 \cdot \mathbf{m}_3)}{\|\mathbf{m}_2 \wedge \mathbf{m}_3\|^2}$$

On effectue ensuite les produits vectoriels $\mathbf{m}_1 \wedge \mathbf{m}_3$ et $\mathbf{m}_1 \wedge \mathbf{m}_2$ et on obtient :

$$\alpha_u = \frac{\|\cos\theta(\mathbf{m}_1 \wedge \mathbf{m}_2) - (\cos\theta v_0' - u_0')(\mathbf{m}_1 \wedge \mathbf{m}_3)\|}{\|\frac{\alpha_v}{\sin\theta}\cos\theta\mathbf{m}_3 - u_0'\mathbf{r}_2\|}$$

$$\mathbf{r}_1 = \frac{1}{\alpha_u}(\mathbf{m}_1 - u_0'\mathbf{m}_3 - \frac{\alpha_v}{\sin\theta}\cos\theta\mathbf{r}_2)$$

Le vecteur de translation est égal à :

$$\begin{aligned} t_z &= m_{34} \\ t_y &= \frac{\alpha_v}{\sin\theta}(m_{24} - v_0' t_z) \\ t_x &= \frac{1}{\alpha_u}(m_{14} - u_0' t_z - \frac{\alpha_v}{\sin\theta}\cos\theta t_y) \end{aligned}$$

5.3. Calibrage d'une caméra

Utilisant l'équation [5.13] on peut écrire les coordonnées image d'un point de la scène. On obtient donc :

$$u = \frac{m_{11}X + m_{12}Y + m_{13}Z + m_{14}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}} \qquad [5.16]$$

$$v = \frac{m_{21}X + m_{22}Y + m_{23}Z + m_{24}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}} \qquad [5.17]$$

Notons au passage que ces deux équations décrivent la droite passant par le centre de projection et le point image (u, v), *dans le repère mire*, soit la droite passant par F et b sur la figure 5.2. Cette droite s'appelle également *la droite de vue* associée avec un point image de coordonnées (u, v).

Pour évaluer les coefficients de la matrice $\mathbf{M}$, il suffit d'écrire ce système d'équations pour les points de la mire (pour lesquels on mesure la projection dans l'image). Chaque point (X_i, Y_i, Z_i) se projetant en (u_i, v_i) fournit deux équations. Ces équations sont linéaires par rapport aux coefficients de la matrice. Il faut donc au moins 6 points pour déterminer les 12 coefficients de la matrice $\mathbf{M}$ qui nous intéresse.

Les équations [5.16] et [5.17] peuvent se réécrire comme une combinaison linéaire des m_{ij} :

$$\begin{aligned} & X_i m_{11} + Y_i m_{12} + Z_i m_{13} + m_{14} - \\ & - u_i X_i m_{31} - u_i Y_i m_{32} - u_i Z_i m_{33} \quad = \quad u_i m_{34} \end{aligned} \qquad [5.18]$$

$$\begin{aligned} & X_i m_{21} + Y_i m_{22} + Z_i m_{23} + m_{24} - \\ & - v_i X_i m_{31} - v_i Y_i m_{32} - v_i Z_i m_{33} \quad = \quad v_i m_{34} \end{aligned} \qquad [5.19]$$

On obtient donc $2n$ équations pour n points et on peut écrire ces équations sous forme matricielle (les indices indiquent la taille des matrices) :

$$\mathbf{K}_{2n \times 11} \, \mathbf{x}_{11} = \mathbf{u}_{2n} \qquad [5.20]$$

Soit, plus précisément :

$$\begin{pmatrix} & & & & & \vdots & & & & & \\ X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -u_i X_i & -u_i Y_i & -u_i Z_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -v_i X_i & -v_i Y_i & -v_i Z_i \\ & & & & & \vdots & & & & & \end{pmatrix} \begin{pmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \end{pmatrix} =$$

$$\begin{pmatrix} \vdots \\ u_i m_{34} \\ v_i m_{34} \\ \vdots \end{pmatrix}$$

5.3.1. Utilisation de la contrainte $m_{34} = 1$

Le système défini par l'équation [5.20] est un système homogène : afin d'obtenir une solution non triviale il faut fixer un des coefficients m_{ij}. On choisit $m_{34} = 1$, ce qui revient à diviser tous les coefficients de la matrice par m_{34} et à déterminer les paramètres de la caméra à un facteur près. On peut remarquer que m_{34} n'est autre que la composante en z du vecteur de translation entre le référentiel mire et le référentiel caméra et qu'on peut facilement s'arranger pour que cette composante ne soit pas nulle. Avec ce choix, la solution de l'équation [5.20] est donnée par :

$$\mathbf{x}_{11} = (\mathbf{K}^t\mathbf{K})^{-1}\mathbf{K}^t\,\mathbf{u} \qquad [5.21]$$

Cette solution n'est autre que la solution au sens des moindres carrés d'un système d'équations linéaires lorsque le nombre d'équations est supérieur au nombre d'inconnues. En effet, on peut écrire l'équation [5.20] sous la forme suivante :

$$\mathbf{K}\,\mathbf{x} - \mathbf{u} = \mathbf{e}$$

$\mathbf{e}$ représente un vecteur erreur. Autrement dit, on cherche une solution approximative. La meilleure solution ($\mathbf{x}$) est celle qui minimise le module du vecteur erreur. On cherche donc $\mathbf{x}$ tel que $\|\mathbf{e}\|^2 = \mathbf{e}^t\mathbf{e}$ soit minimum. On a :

$$\begin{aligned}
\mathbf{e}^t\mathbf{e} &= (\mathbf{K}\mathbf{x} - \mathbf{u})^t(\mathbf{K}\mathbf{x} - \mathbf{u}) \\
&= \mathbf{x}^t\mathbf{K}^t\mathbf{K}\mathbf{x} - \mathbf{u}^t\mathbf{K}\mathbf{x} - \mathbf{x}^t\mathbf{K}^t\mathbf{u} + \mathbf{u}^t\mathbf{u} \\
&= \mathbf{x}^t\mathbf{K}^t\mathbf{K}\mathbf{x} - 2\mathbf{x}^t\mathbf{K}^t\mathbf{u} + \mathbf{u}^t\mathbf{u}
\end{aligned}$$

En différenciant par rapport à X on obtient :

$$\mathbf{K}^t\mathbf{K}\mathbf{x} - \mathbf{K}^t\mathbf{u} = 0$$

La valeur de $\mathbf{x}$ qui minimise l'erreur est bien celle fournie précédemment. La matrice $(\mathbf{K}^t\mathbf{K})^{-1}\mathbf{K}^t$ est appelée pseudo-inverse de $\mathbf{K}$. La matrice $(\mathbf{K}^t\mathbf{K})^{-1}$ est une matrice 11×11 symétrique et positive. En pratique on utilise des techniques numériques (élimination Gauss-Jordan, décomposition LU ou QR ou décomposition en valeurs singulières) pour calculer une solution approximative aux moindres carrés pour l'équation [5.20]. Voir [156], pages 528–539 pour résoudre cette équation.

Il faut toutefois remarquer que le fait de choisir $m_{34} = 1$ introduit un inconvénient. En effet, les paramètres du modèle de la caméra ne sont calculés qu'à un facteur multiplicatif près, soit $t_z = m_{34}$.

5.3.2. Utilisation de la contrainte $\|\mathbf{m}_3\| = 1$: méthode Faugeras-Toscani

En calculant explicitement les coefficients de $\mathbf{M}$ en fonction des coefficients des matrices qui la composent ($\mathbf{I}_c$ et $\mathbf{A}$) on trouve en particulier (voir équation [5.14]) :

$$m_{31} = r_{31} \quad m_{32} = r_{32} \quad m_{33} = r_{33}$$

D'autre part il est facile de vérifier qu'on a : $r_{31}^2 + r_{32}^2 + r_{33}^2 = 1$. On obtient donc :

$$\|\mathbf{m}_3\|^2 = m_{31}^2 + m_{32}^2 + m_{33}^2 = 1$$

Dans ce qui suit nous montrons comment on peut calculer la matrice $\mathbf{M}$ en tenant compte de cette contrainte. Ce résultat est dû à Faugeras & Toscani [57, 58].

L'équation [5.20] peut maintenant se réécrire :

$$\mathbf{B}_{2n\times 9}\mathbf{x}_9 + \mathbf{C}_{2n\times 3}\mathbf{x}_3 = 0 \qquad [5.22]$$

avec :

$$\mathbf{B}_{2n\times 9} = \begin{pmatrix} & & & & \vdots & & & & \\ X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -u_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -v_i \\ & & & & \vdots & & & & \end{pmatrix}$$

$$\mathbf{C}_{2n\times 3} = \begin{pmatrix} & \vdots & \\ -u_i\,X_i & -u_i\,Y_i & -u_i\,X_i \\ -v_i\,X_i & -v_i\,Y_i & -v_i\,X_i \\ & \vdots & \end{pmatrix}$$

On décompose par ailleurs $\mathbf{x}_{11}$ en deux inconnues (cette fois-ci on intègre m_{34} comme inconnue) :

$$\mathbf{x}_9 = \begin{pmatrix} \mathbf{m}_1 & m_{14} & \mathbf{m}_2 & m_{24} & m_{34} \end{pmatrix}^t$$

$$\mathbf{x}_3 = (\mathbf{m}_3)^t$$

Le critère à minimiser est le même que précédemment :

$$Q = \|\mathbf{B}_{2n\times 9}\mathbf{x}_9 + \mathbf{C}_{2n\times 3}\mathbf{x}_3\|^2 \qquad [5.23]$$

avec la contrainte :

$$\|\mathbf{x}_3\|^2 = 1$$

Le critère peut donc s'écrire de la façon suivante :

$$Q = \|\mathbf{B}\mathbf{x}_9 + \mathbf{C}\mathbf{x}_3\|^2 + \lambda(1 - \|\mathbf{x}_3\|^2)$$

En développant on obtient :

$$Q = \mathbf{x}_9^t\mathbf{B}^t\mathbf{B}\mathbf{x}_9 + \mathbf{x}_3^t\mathbf{C}^t\mathbf{C}\mathbf{x}_3 + \mathbf{x}_9^t\mathbf{B}^t\mathbf{C}\mathbf{x}_3 + \mathbf{x}_3^t\mathbf{C}^t\mathbf{B}\mathbf{x}_9 + \lambda(1 - \mathbf{x}_3^t\mathbf{x}_3) \qquad [5.24]$$

En imposant que les dérivées partielles par rapport à $\mathbf{x}_9$ et $\mathbf{x}_3$ soient nulles, on obtient les deux équations suivantes :

$$\begin{aligned} \mathbf{B}^t\mathbf{B}\mathbf{x}_9 + \mathbf{B}^t\mathbf{C}\mathbf{x}_3 &= 0 \\ \mathbf{C}^t\mathbf{C}\mathbf{x}_3 + \mathbf{C}^t\mathbf{B}\mathbf{x}_9 - \lambda\mathbf{x}_3 &= 0 \end{aligned}$$

D'où on obtient :

$$\begin{aligned} \mathbf{x}_9 &= -(\mathbf{B}^t\mathbf{B})^{-1}\mathbf{B}^t\mathbf{C}\mathbf{x}_3 \\ \mathbf{D}\mathbf{x}_3 &= \lambda\mathbf{x}_3 \\ \mathbf{D} &= \mathbf{C}^t\mathbf{C} - \mathbf{C}^t\mathbf{B}(\mathbf{B}^t\mathbf{B})^{-1}\mathbf{B}^t\mathbf{C} \end{aligned}$$

Finalement, en substituant dans [5.24] on obtient pour Q :

$$Q = \mathbf{x}_3^t\mathbf{D}\mathbf{x}_3 = \lambda\mathbf{x}_3^t\mathbf{x}_3 = \lambda$$

On peut remarquer que $\mathbf{D}$ est une matrice symétrique et positive 3×3. Elle a donc des valeurs propres réelles et positives. $\mathbf{x}_3$ est un vecteur propre de $\mathbf{D}$ associé à la valeur propre λ. Pour minimiser le critère il faut donc calculer les valeurs propres de la matrice $\mathbf{D}$, choisir la plus petite valeur propre (car c'est elle qui minimise le critère), calculer le vecteur propre qui lui est associé, soit $\mathbf{x}_3$, le normaliser, et finalement calculer $\mathbf{x}_9$. Les coefficients de $\mathbf{M}$ sont fournis par $\mathbf{x}_3$ et $\mathbf{x}_9$. Etant donné que le signe de $\mathbf{x}_3$, vecteur propre, n'est pas défini, on a deux solutions, $\mathbf{M}$ et $-\mathbf{M}$. On peut en choisir une parmi ces solutions en utilisant le fait que l'objet de calibrage se trouve devant la caméra et non pas derrière la caméra. Dans ce cas on doit avoir $m_{34} = t_z > 0$.

5.3.3. Détermination des paramètres du modèle

A partir des coefficients de la matrice $\mathbf{M}$ et grâce aux équations [5.14] on peut déterminer les paramètres intrinsèques et extrinsèques de la caméra. Nous allons nous concentrer sur les paramètres intrinsèques car ils caractérisent la caméra indépendamment de la mire utilisée pour le calibrage. Quant aux paramètres extrinsèques ils caractérisent la position et l'orientation d'une caméra par rapport à un objet (un ensemble de points 3D). La détermination de ces paramètres peut se faire grâce aux équations [5.14] ou par d'autres méthodes qui seront décrites au chapitre 8. Comme nous allons le voir, la détermination des position et orientation relatives entre une caméra et un objet est un problème central en vision et en robotique.

Revenons maintenant aux paramètres intrinsèques et notons le caractère non linéaire des équations [5.14]. En particulier, une petite erreur sur $\mathbf{m}_3$ a une influence considérable sur le calcul des paramètres de la caméra. Puget & Skordas [157] ont calculé l'erreur commise sur les paramètres de la caméra en fonction des erreurs commises sur les coefficients de la matrice $\mathbf{M}$.

L'erreur commise sur le produit scalaire de deux vecteurs $\mathbf{v}$ et $\mathbf{w}$ s'écrit :

$$\Delta(\mathbf{v} \cdot \mathbf{w}) = \Delta\mathbf{v} \cdot \mid \mathbf{w} \mid + \mid \mathbf{v} \mid \cdot \Delta\mathbf{w}$$

avec $\mid \mathbf{v} \mid = (\mid v_x \mid, \mid v_y \mid, \mid v_z \mid)^t$. L'erreur commise sur u_0 peut donc s'écrire :

$$\Delta u_0 = \Delta\mathbf{m}_1 \cdot \mid \mathbf{m}_3 \mid + \mid \mathbf{m}_1 \mid \cdot \Delta\mathbf{m}_3$$

En pratique on peut estimer l'erreur commise sur les m_{ij} en fonction des erreurs commises sur les mesures (les coordonnées dans l'image des points de la mire). Puget & Skordas ont calibré *plusieurs fois* la même caméra en plusieurs positions et ont obtenu des variations sur les valeurs de u_0 et v_0 allant jusqu'à 20 pixels ! Ils proposent d'améliorer cette précision en calibrant plusieurs fois.

Soit $\mathbf{M}^i$ la matrice obtenue avec la méthode Faugeras-Toscani pour la $i^{ème}$ position caméra/mire. Pour chaque position on obtient un jeu de paramètres intrinsèques. Les paramètres optimaux sont obtenus en minimisant un critère de la forme (pour N positions distinctes) :

$$C = \frac{1}{N} \sum_{i=1}^{i=N} (u_0 - u_0^i)^2$$

La valeur moyenne des u_0^i minimise ce critère :

$$u_0 = \frac{1}{N} \sum_{i=1}^{i=N} u_0^i \qquad [5.25]$$

Il y a des expressions équivalentes pour les autres paramètres intrinsèques.

5.4. Modélisation des distorsions

Jusqu'à présent nous avons supposé un modèle linéaire : l'image d'un point est obtenue en intersectant le plan image avec la droite passant par ce point et par le centre de projection. En réalité la lumière ne voyage pas toujours le long d'une droite et l'équation [5.5] doit être remplacée par :

$$\begin{cases} u &= \alpha_u x_c + u_0 + \Delta u \\ v &= \alpha_v y_c + v_0 + \Delta v \end{cases} \qquad [5.26]$$

Dans cette équation on a rajouté un terme de correction qui peut être non linéaire. Les nombreuses études sur ce sujet ont montré qu'une correction radiale est suffisante dans la plupart des cas [170] :

$$\begin{cases} \Delta u &= K_1 r^2 (u - u_0) \\ \Delta v &= K_1 r^2 (v - v_0) \end{cases}$$

avec $r^2 = (u - u_0)^2 + (v - v_0)^2$. Le modèle de la caméra a maintenant 5 et non plus 4 paramètres intrinsèques, à savoir α_u, α_v, u_0, v_0, et K_1. Comment peut-on estimer le nouveau modèle d'une caméra ? La réponse n'est pas simple car on ne peut plus linéariser le système d'équations comme nous l'avons fait en l'absence des distorsions. En effet, le système d'équations s'écrit maintenant de la façon suivante :

$$\begin{cases} \vdots \\ (u_i - u_0) - \alpha_u \frac{r_{11} X_i + r_{12} Y_i + r_{13} Z_i + t_x}{r_{31} X_i + r_{32} Y_i + r_{33} Z_i + t_z} - K_1 r^2 (u_i - u_0) &= 0 \\ \\ (v_i - v_0) - \alpha_v \frac{r_{21} X_i + r_{22} Y_i + r_{23} Z_i + t_y}{r_{31} X_i + r_{32} Y_i + r_{33} Z_i + t_z} - K_1 r^2 (v_i - v_0) &= 0 \\ \vdots \end{cases}$$

Pour résoudre ce système d'équations il faut faire appel à des techniques d'optimisation non linéaire et le résultat dépend du choix des valeurs initiales des

paramètres. Une solution consiste à évaluer les paramètres (sauf pour K_1) utilisant, par exemple, une des techniques linéaires décrites auparavant et à les utiliser par la suite comme valeurs initiales.

Une autre solution consiste à déterminer la correction non linéaire indépendamment du modèle de la caméra. Une telle solution a été recemment proposée dans [28]. Soient (u, v) la position d'un point image et (u', v') la position corrigée :

$$\begin{cases} u' &= u - \Delta u(u, v) \\ v' &= v - \Delta v(u, v) \end{cases}$$

Les termes de correction Δu et Δv sont fonction de la position du pixel dans l'image. La figure 5.4. montre un exemple de distorsion. Les cercles illustrent les pixels tels qu'ils apparaissent dans l'image, (u et v dans la formule précédente) tandis que les carrés illustrent les positions corrigées de ces pixels (u' et v'). Le vecteur de correction ($\Delta u\ \Delta v$) est illustré par un segment fléché.

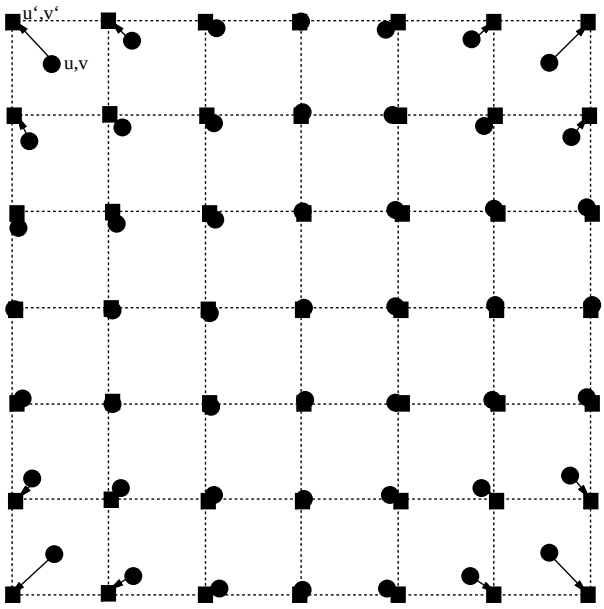


Figure 5.4. *Un exemple de distorsion. Les pixels "ronds" doivent être corrigés pour obtenir les pixels "carrés".*

5.5. Mise en œuvre du calibrage

Il est utile de rappeler que la calibration d'une caméra consiste en réalité en la calibration de l'ensemble objectif optique, caméra, convertisseur analogique/digital. Toute variation d'ouverture, de mise au point ou de focale modifie la configuration géométrique du capteur. Il est donc important que ces paramètres ne varient pas lors du calibrage. Bien évidemment, si l'on modifie un de ces paramètres il faut recalibrer l'ensemble du capteur. Les mêmes remarques s'appliquent au convertisseur analogique digital : tout changement concernant les paramètres du convertisseur entraîne un changement géométrique du capteur.

En outre, on doit disposer d'un objet de calibrage fournissant des "points" dont la position doit être connue avec une très bonne précision dans le repère de l'objet (repère de calibrage). Disons que, pour obtenir un bon calibrage, cette précision doit être inférieure ou égale au 1/10 de millimètre. La figure 5.5. montre l'objet de calibrage que nous utilisons au LIFIA[1] Les points de calibrage sont les sommets d'un ensemble de carrés formant une mire plane. Ce motif a été choisi pour que sa projection dans l'image puisse être mesurée avec une grande précision. L'origine se trouve au sommet du premier carré en haut et à gauche de l'image. Une table à déplacement micrométrique (1/100 millimètre dans notre cas) munie d'une équerre assure un mouvement perpendiculaire au plan de la mire. On obtient ainsi des coordonnées tri-dimensionnelles des points de mesure dans un repère lié au plan de la mire.

Le traitement de l'image a pour rôle de détecter avec précision les projections des points de la mire. Ces points étant, dans notre cas, les sommets de carrés alignés, l'image d'un sommet sera obtenue en intersectant des droites passant par les arêtes des carrés. Ce principe est illustré sur la figure 5.6. La projection d'un point de calibrage est obtenue en calculant l'intersection de deux droites passant "au mieux" par les arêtes des projections de carrés alignés. Un exemple permettra de mieux comprendre ce principe.

La figure 5.7. montre une image de la mire de calibrage (gauche) ainsi que les segments de droites extraits de cette image (droite). Pour extraire ces segments de droite nous avons tout d'abord extrait les contours qui ont été ensuite

[1] Laboratoire d'Informatique Fondamentale et d'Intelligence Artificielle. Unité associée au CNRS (URA 394), 46 avenue Félix Viallet, 38000 Grenoble.

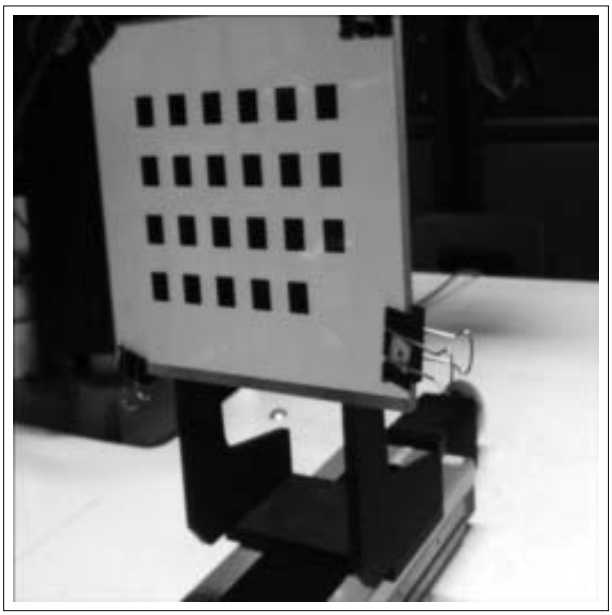


Figure 5.5. *Une vue générale de l'objet de calibrage utilisé. Une mire plane est fixée sur une équerre elle-même fixée sur une table à déplacement micrométrique.*

approximés par des segments de droite (voir chapitre 3). Une méthode basée sur la transformée de Hough (voir chapitre 3) nous a ensuite permis d'extraire deux familles de droites parallèles, familles montrées sur la figure 5.8. Chacune de ces familles est ensuite divisée en un ensemble de segments de droites colinéaires. Finalement, tous les points de contours appartenant au même ensemble de droites colinéaires sont approximés par une droite au sens des moindres carrés. On obtient une "grille", chaque arête de cette grille approximant les points de contours formant un ensemble d'arêtes colinéaires. Les points finalement retenus comme points de projection de la mire sont les intersections des arêtes de cette grille. La figure 5.9. (gauche) montre la grille ainsi obtenue.

Pour calibrer avec les méthodes que nous avons décrites il faut disposer de points de calibrage non coplanaires. Pour cela, on déplace la mire perpendiculairement à son plan sans bouger le capteur. Une fois qu'on a mesuré les projections de plusieurs positions de la mire et qu'on a ainsi obtenu plusieurs centaines de correspondances mire/image, on peut estimer la matrice de pro-

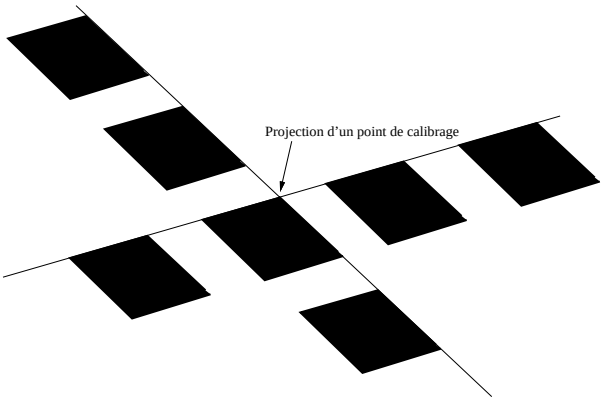


Figure 5.6. *L'image d'un point de calibrage est obtenue en intersectant deux droites passant au mieux par un ensemble d'arêtes alignées.*

jection perspective. La figure 5.9. (droite) montre la superposition de la grille avec des points obtenus en projetant les points de la mire grâce à la matrice de projection qu'on vient de calculer.

5.6. Caméra affine

Jusqu'à présent nous avons étudié le modèle projectif de caméra. Il est cependant utile dans certains cas de considérer un modèle simplifié. On remplace alors la transformation projective 3D/2D par une transformation affine. Le modèle affine de caméra est valide lorsque la taille des objets que l'on observe et faible par rapport à la distance moyenne de ces objets au centre de projection de la caméra.

Soit le schéma de la figure 5.2. et les mêmes notations que celles qui ont été utilisées auparavant. Nous designons par x et y les coordonnées d'un point de l'image dans le repère de la caméra. Ce point s'obtient en projetant un point de la scène dont les coordonnées sont exprimées dans le repère de la scène (ou de calibrage). Les équations de la projection peuvent s'écrire :

$$x = \frac{\mathbf{r}_1 \cdot \overrightarrow{O'B} + t_x}{\mathbf{r}_3 \cdot \overrightarrow{O'B} + t_z} \qquad [5.27]$$

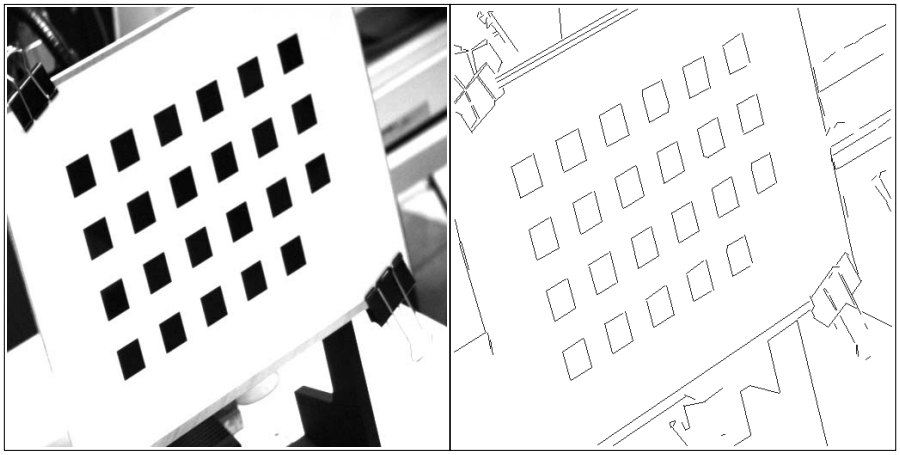


Figure 5.7. *L'image de la mire et les segments de droite extraits de cette image.*

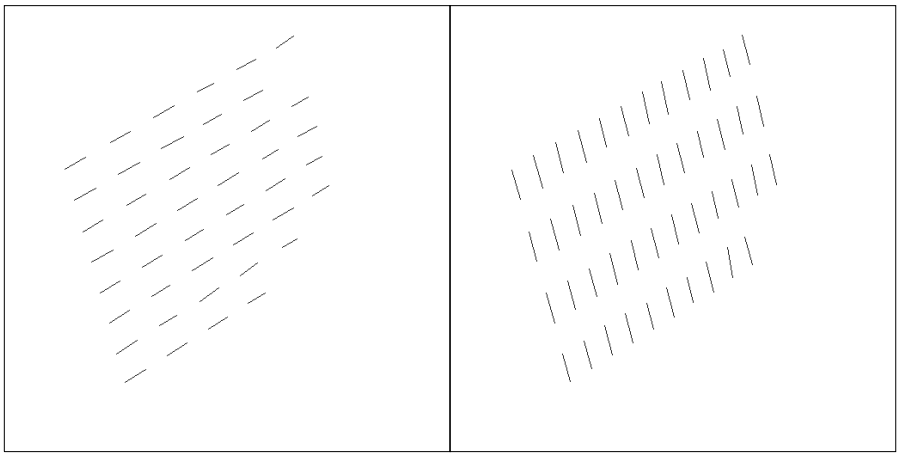


Figure 5.8. *Deux familles de droites parallèles détectées parmi l'ensemble des segments de droites extraits de l'image.*

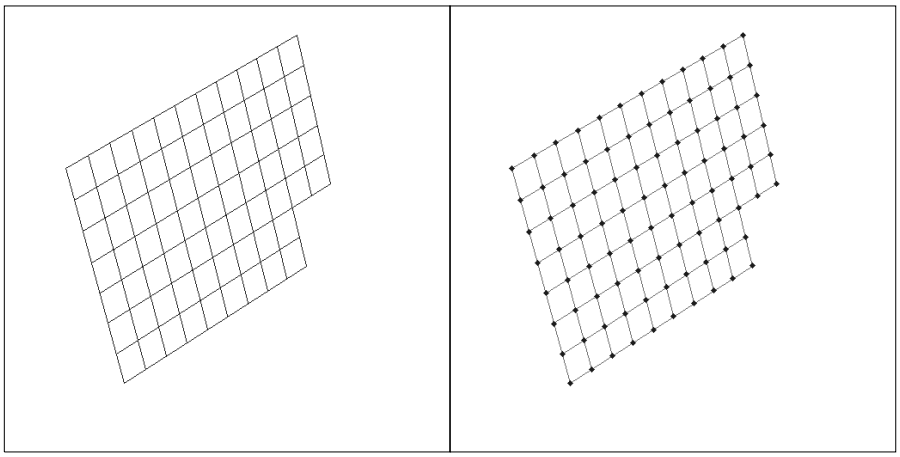


Figure 5.9. *Les sommets de cette grille sont les points image servant au calibrage. L'image de droite montre la superposition sur cette grille des points de la mire projetés grâce à la matrice de projection perspective obtenue par calibration.*

$$y = \frac{\mathbf{r}_2 \cdot \overrightarrow{O'B} + t_y}{\mathbf{r}_3 \cdot \overrightarrow{O'B} + t_z} \quad [5.28]$$

Divisons ces deux expressions par t_z. On peut écrire la projection sous la forme suivante :

$$x = \frac{s\mathbf{r}_1 \cdot \overrightarrow{O'B} + x'}{1 + \varepsilon} \quad [5.29]$$

$$y = \frac{s\mathbf{r}_2 \cdot \overrightarrow{O'B} + y'}{1 + \varepsilon} \quad [5.30]$$

- x' et y' sont les coordonnées de la projection de l'origine O′ du repère scène,
- $s = 1/t_z$ est un facteur d'echelle et
- $\varepsilon = \mathbf{r}_3 \cdot \overrightarrow{O'B}/t_z$ est un rapport qui est petit si l'objet est distant de la caméra ou si la taille de l'objet observé est petite par rapport à la distance à la caméra.

On peut donc obtenir un modèle affine de caméra en négligeant la valeur de

ε par rapport à 1 :

$$\frac{1}{1+\varepsilon} \approx 1$$

Avec cette approximation le modèle de caméra qu'on obtient est le suivant :

$$\begin{aligned} x - x' &= s\mathbf{r}_1 \cdot \overrightarrow{O'B} \\ y - y' &= s\mathbf{r}_2 \cdot \overrightarrow{O'B} \end{aligned}$$

On peut maintenant introduire à nouveau les coordonnées des points de l'image dans le repère image. On a :

$$\begin{pmatrix} u - u' \\ v - v' \end{pmatrix} = \begin{pmatrix} \alpha_u & 0 \\ 0 & \alpha_v \end{pmatrix} \begin{pmatrix} x - x' \\ y - y' \end{pmatrix}$$

En substituant dans l'équation précédente on obtient successivement :

$$\begin{pmatrix} u - u' \\ v - v' \end{pmatrix} = \begin{pmatrix} s\alpha_u & 0 \\ 0 & s\alpha_v \end{pmatrix} \begin{pmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \quad [5.31]$$

$$= \begin{pmatrix} a_u & 0 \\ 0 & a_v \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \mathbf{r}_3 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \quad [5.32]$$

$$= \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \quad [5.33]$$

$$= \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \quad [5.34]$$

Les paramètres intrinsèques d'une caméra affine sont a_u et a_v qui ont la même signification géométrique (à un facteur d'échelle près) que α_u et α_v dans le cas d'une caméra projective. On peut remarquer, que dans le cas d'une caméra affine on ne peut plus parler de centre de projection. Les paramétres extrinsèques sont constitués par une matrice de rotation.

Pour calibrer une telle caméra il suffit de disposer d'une mire de calibrage suffisamment loin de la caméra pour annuler les effets de perspective, et de choisir un des point de la mire comme origine du repère de la scène. Pour une

correspondance mire/image i on obtient les deux équations suivantes :

$$\begin{pmatrix} X_i & Y_i & Z_i & 0 & 0 & 0 \\ 0 & 0 & 0 & X_i & Y_i & Z_i \end{pmatrix} \begin{pmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \end{pmatrix} = \begin{pmatrix} u_i - u' \\ v_i - v' \end{pmatrix} \qquad [5.35]$$

On a donc besoin, au minimum, de 4 points de correspondance (3 points courants et l'origine) pour calibrer une caméra affine. On obtient facilement les paramètres intrinsèques et extrinsèques de cette caméra :

$$\begin{aligned} a_u &= \|\mathbf{a}_1\| \\ a_v &= \|\mathbf{a}_2\| \\ \mathbf{r}_1 &= \mathbf{a}_1/\|\mathbf{a}_1\| \\ \mathbf{r}_2 &= \mathbf{a}_2/\|\mathbf{a}_2\| \\ \mathbf{r}_3 &= \mathbf{r}_1 \wedge \mathbf{r}_2 \end{aligned}$$

Remarquons pour finir que le modèle affine peut également s'écrire sous la forme d'une matrice 3×4 :

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & u' \\ a_{21} & a_{22} & a_{23} & v' \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

5.7. Caméra linéaire (barette CCD)

Dans beaucoup d'applications de la vision par ordinateur les caméras linéaires (barettes CCD) peuvent remplacer les caméras matricielles qu'on vient d'étudier. L'inspection utilise abondamment ce type de capteurs car on a en même temps besoin de précision et de vitesse de calcul. Le signal fourni par une caméra linéaire est plus simple donc plus rapide à traiter que les images 2D. De plus, une barette CCD peut avoir jusqu'à 4096 pixels à un prix raisonnable.

Les techniques de calibrage des caméras matricielles qu'on vient de décrire sont nécessaires mais pas suffisantes pour calibrer une caméra linéaire [93], [94].

Dans cette section nous proposons une solution en deux étapes pour calibrer une telle caméra. La première étape utilise les mêmes techniques que les caméras matricielles. La deuxième étape utilise un objet de calibrage spécialement conçu à cet effet en combinaison avec le birapport, un invariant projectif qui a déjà été utilisé en vision, [126].

La meilleure façon d'imaginer une caméra linéaire est de penser à une caméra matricielle pour laquelle une seule ligne de pixels est active. Un point de l'espace se projette sur cette ligne suivant l'équation [5.16]. Par ailleurs, le point de l'espace se projetant sur la ligne image est contraint d'appartenir dans un *plan de vue*, soit le plan défini par le centre de projection et par la ligne de pixels, figure 5.10. L'équation du plan de vue peut être écrite comme suit :

$$X = pY + qZ + r \qquad [5.36]$$

En substituant l'équation [5.36] dans l'équation [5.16], en renommant les

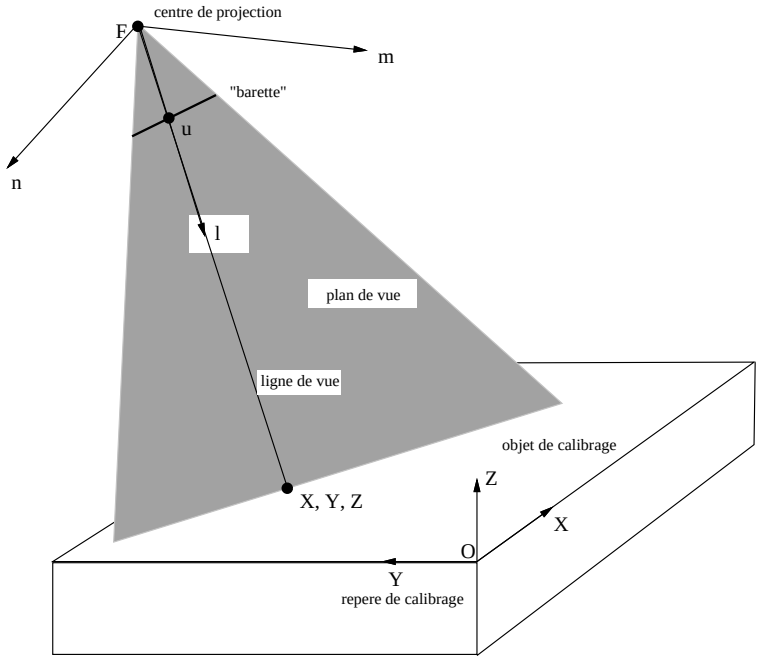


Figure 5.10. *Une vue générale d'une caméra linéaire, un objet de calibrage et les systèmes de coordonnées qui leur sont associés.*

variables et en notant que l'équation [5.16] est définie à un facteur multiplicatif

près, on obtient :

$$u = \frac{n_1 Y + n_2 Z + n_3}{n_4 Y + n_5 Z + 1} \qquad [5.37]$$

Les équations [5.36] et [5.37] définissent le modèle de la caméra linéaire, autrement dit elles décrivent l'équation de la droite passant par le centre de projection et un pixel, dans le repère de calibrage. Cette droite est la *droite de vue*.

Ce modèle a 8 paramètres : n_1, n_2, n_3, n_4, n_5, et p, q, r. Le problème de calibrage de la caméra linéaire est donc le problème d'estimation de ces paramètres. Ce problème peut donc se décomposer en deux étapes :

- estimer les paramètres n_1, n_2, n_3, n_4, n_5. Si au moins 5 correspondances sont disponibles, ce problème est un problème d'optimisation linéaire similaire au cas de la calibration matricielle.

- estimer p, q, r. Ces paramètres peuvent être estimés si et seulement si on possède des points de l'espace appartenant au plan de vue.

5.7.1. Calibration en deux étapes

On commence par décrire la structure de l'objet de calibrage utilisé. La raison du choix de cette structure sera justifiée plus bas. L'objet de calibrage est montré sur la figure 5.11. Il consiste en quatre droites coplanaires, D_1, D_2, D_3, D_4. Les trois premières sont parallèles entre elles et la quatrième fait un angle aigu avec la direction des trois autres. Les équations de ces droites sont données dans le repère de calibrage qui est défini comme suit : l'axe des X est confondu avec D_1 et l'axe des Y est perpendiculaire à D_1. L'origine se trouve quelque part le long de D_1. L'axe des Z est perpendiculaire au plan contenant l'objet de calibrage. Dans ce repère et dans le plan $Z = 0$ les équations des quatre droites s'écrivent :

$$\begin{aligned} Y &= 0 && (D_1) \\ Y &= \alpha && (D_2) \\ Y &= \beta && (D_3) \\ Y &= \gamma X + \delta && (D_4) \end{aligned}$$

les paramètres α, β, γ, et δ sont fixés et ils déterminent la structure de l'objet de calibrage.

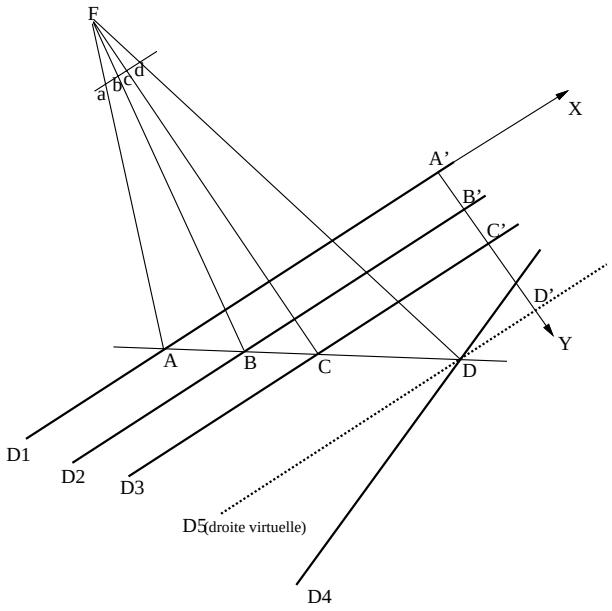


Figure 5.11. *La structure de l'objet de calibrage et la façon dont il est observé par la caméra linéaire.*

Lorsque la caméra linéaire "observe" cet objet elle "voit" quatre points, a, b, c, d qui sont les projections de A, B, C, D soient les intersections du plan de vue avec les quatre droites D_1, D_2, D_3, D_4, figure 5.11. Remarquez que l'on ne doit pas connaître les positions de ces points le long des droites formant l'objet. Quelle que soit l'orientation du plan de vue par rapport à l'objet de calibrage, les coordonnées Y et Z des points A, B, C sont connues dans le repère de calibrage. On peut donc établir des correspondances du type $\{Y_A, Z_A, u_a\}$, $\{Y_B, Z_B, u_b\}$ et $\{Y_C, Z_C, u_c\}$ pour chaque position de l'objet par rapport à la caméra. En faisant bouger l'objet de calibrage dans les directions Y et Z on peut établir trois nouvelles correspondances dans chacune des nouvelles positions. Chaque correspondance i vérifie l'équation [5.37] qui peut être écrite sous la forme suivante :

$$Y_i n_1 + Z_i n_2 + n_3 - u_i Y_i n_4 - u_i Z_i n_5 = u_i \qquad [5.38]$$

Pour n telles correspondances on obtient ($n \geq 5$) :

$$\begin{pmatrix} Y_1 & Z_1 & 1 & -u_1Y_1 & -u_1Z_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ Y_n & Z_n & 1 & -u_nY_n & -u_nZ_n \end{pmatrix} \begin{pmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \\ n_5 \end{pmatrix} = \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix} \qquad [5.39]$$

Il s'agit d'un système formé de n équations linéaires en 5 inconnues de la forme : $AX = B$. La solution est du même type que l'équation [5.21].

Afin d'estimer p, q, et r on doit avoir à notre disposition un ensemble de points appartenant au plan de vue dont on connaît les coordonnées dans le repère de calibrage. La structure de l'objet de calibrage est telle que l'on peut calculer pour des points de l'espace se trouvant sur cet objet et dans le plan de vue leurs coordonnées 3D dans le repère de calibrage.

Rapelons la définition du birapport de quatre points colinéaires :

$$\begin{aligned} \text{birapport}_{A,B,C,D} &= (A, B, C, D) \\ &= (CA/CB)/(DA/DB) \end{aligned}$$

CA étant la distance algébrique de C à A. La propriété fondamentale du birapport est qu'il est invariant par projection perspective ou orthographique. On a donc :

$$(A, B, C, D) = (a, b, c, d)$$

Considérons une droite *virtuelle* D_5 parallèle et coplanaire avec les droites D_1, D_2, D_3 et passant par le point D. Ces quatre droites parallèles (D_1, D_2, D_3 et D_5) intersectent l'axe des Y en A', B', C' et D'. Compte tenu de la propriété d'invariance du bi-rapport on a :

$$(A, B, C, D) = (A', B', C', D')$$

et on obtient :

$$(C'A'/C'B')/(D'A'/D'B') = (ca/cb)/(da/db)$$

Si la caméra linéaire observe simultanément D_1, D_2, D_3 et D_4 (en réalité la caméra ne voit que les points A, B, C et D) on peut alors calculer le bi-rapport r, $r = (a, b, c, d)$ dans l'image et on peut alors déterminer la position du point

D' le long de l'axe Y, soit $Y = \lambda$. En effet on a :

$$\begin{aligned} r &= (C'A'/C'B')/(D'A'/D'B') \\ &= \left(\frac{\beta}{\beta-\alpha}\right) / \left(\frac{\lambda}{\lambda-\alpha}\right) \end{aligned}$$

et on obtient :

$$\lambda = \frac{\alpha\beta}{r\alpha + (1-r)\beta}$$

Les coordonnées du point D seront donc données par l'intersection des droites D_4 et D_5 (qui est une droite virtuelle) :

$$\begin{aligned} Y &= \gamma X + \delta \\ Y &= \lambda \\ Z &= 0 \end{aligned}$$

Le point D appartient forcément au plan de vue et sa position dans le repère de calibrage peut être calculée à partir de mesures images mais sans calibrer la caméra. Encore une fois, en déplaçant l'objet de calibrage dans les directions Y et Z on peut déterminer un ensemble de points appartenant tous au plan de vue, chaque point j de cet ensemble vérifiant l'équation du plan de vue :

$$Y_j p + Z_j q + r = X_j$$

En combinant ces équations pour k points on obtient :

$$\begin{pmatrix} Y_1 & Z_1 & 1 \\ \vdots & \vdots & \vdots \\ Y_k & Z_k & 1 \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix} = \begin{pmatrix} X_1 \\ \vdots \\ X_k \end{pmatrix} \qquad [5.40]$$

Il s'agit une fois de plus d'un système d'équations linéaires à trois inconnues.

5.8. Capteurs stéréoscopiques passifs

Nous sommes maintenant en mesure d'étudier un capteur stéréoscopique. Il s'agit tout simplement d'utiliser deux caméras qui "observent" la même scène. On récupère ainsi deux projections de chaque point de la scène. Considérons le schéma de la figure 5.12. qui représente deux caméras. A chaque caméra est associé un repère. Soit P un point de la scène et soient p et p′ les deux

projections dans les images de gauche et de droite. Nous pouvons donc écrire l'équation de la droite passant par le centre focal (F) de la caméra de gauche et le point p. De même on peut écrire l'équation de la droite passant par F′ et p′. Le point de la scène P se trouve à l'intersection de ces droites. Afin de pouvoir calculer cette intersection et donc de déterminer la position de P il faut pouvoir exprimer les deux équations (de la droite Fp et de la droite F′p′) *dans le même repère.*

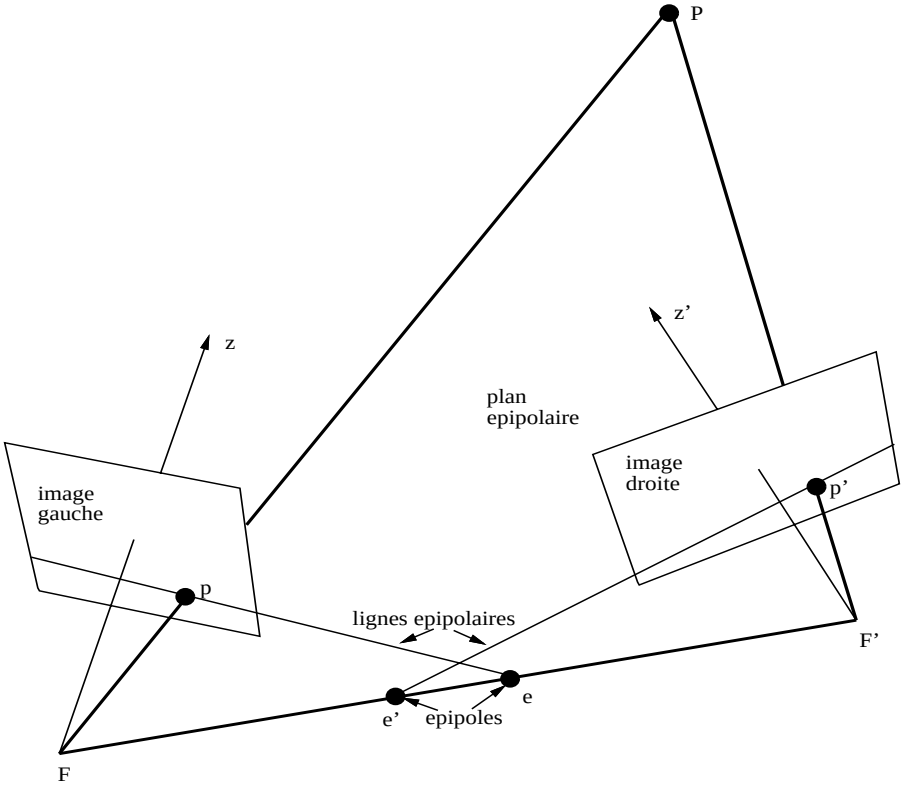


Figure 5.12. *La géométrie d'un capteur stéréoscopique passif.*

5.8.1. Calibrage stéréoscopique

Le calibrage stéréoscopique consiste donc à déterminer la matrice de transformation entre le repère caméra gauche et le repère caméra droite. Soit A_s

cette matrice qui est une matrice composée d'une matrice de rotation et une matrice de translation du même type que la matrice A de l'équation [5.7]. Cette matrice est également représentée sur la figure 5.13. Cette figure suggère également la démarche à suivre pour calibrer un tel capteur, soit déterminer A_s :

- on commence par calibrer chaque caméra par rapport à une mire unique. Ceci nous fournit les coefficients de deux matrices, $\mathbf{M}$ et $\mathbf{M}'$.
- on extrait ensuite les paramètres intrinsèques et extrinsèques de chaque caméra.
- à l'aide des paramètres extrinsèques on construit deux matrices, $\mathbf{A}$ et $\mathbf{A}'$, la première étant la transformation du repère mire au repère de la caméra gauche, la deuxième étant la transformation du repère mire au repère de la caméra droite.
- on calcule ensuite $\mathbf{A}_s$:

$$\mathbf{A}_s = \mathbf{A}'\mathbf{A}^{-1}$$

Cette matrice s'écrit comme auparavant :

$$\mathbf{A}_s = \begin{pmatrix} r_{11} & r_{12} & r_{13} & b_x \\ r_{21} & r_{22} & r_{23} & b_y \\ r_{31} & r_{32} & r_{33} & b_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{r}_1 & b_x \\ \mathbf{r}_2 & b_y \\ \mathbf{r}_3 & b_z \\ \mathbf{0} & 1 \end{pmatrix}$$

Le vecteur $\mathbf{b} = (b_x b_y b_z)^t$ est le vecteur allant de F′ à F. Il représente les coordonnées de F dans le repère de la caméra de droite, figure 5.12.

5.8.2. Relation droite-gauche

Nous allons maintenant établir une relation simple entre un point de l'image de gauche et un point de l'image de droite. Dans ce qui suit nous allons nous placer dans le repère caméra et non pas dans le repère image. Rappelons qu'à partir d'un pixel image de coordonnées u et v on peut facilement déduire ses coordonnées caméra (équation [5.6]) :

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathbf{C}^{-1} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \qquad [5.41]$$

et on a une expression similaire pour la caméra droite :

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \mathbf{C}'^{-1} \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix}$$

Exprimons maintenant un point P de la scène à la fois dans les deux repères. Soient (X, Y, Z) ses coordonnées dans le repère gauche et (X', Y', Z') ses coordonnées dans le repère droit. Les deux systèmes de coordonnées sont reliés par la formule :

$$\begin{pmatrix} X' \\ Y' \\ Z' \\ 1 \end{pmatrix} = \mathbf{A}_s \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Les coordonnées de p, la projection de P dans l'image de gauche sont $(x, y, 1)$ où $x = X/Z$ et $y = Y/Z$. On a également p', la projection dans l'image de droite de coordonnées $(x', y', 1)$. x' et y' peuvent s'écrire :

$$x' = \frac{X'}{Z'} = \frac{r_{11}X + r_{12}Y + r_{13}Z + b_x}{r_{31}X + r_{32}Y + r_{33}Z + b_z}$$

$$y' = \frac{Y'}{Z'} = \frac{r_{21}X + r_{22}Y + r_{23}Z + b_y}{r_{31}X + r_{32}Y + r_{33}Z + b_z}$$

En remarquant que $X = xZ$ et $Y = yZ$ et avec la notation $\mathbf{p} = (x\ y\ 1)^t$ on peut simplifier ces formules :

$$x' = \frac{Z\ \mathbf{r}_1 \cdot \mathbf{p} + b_x}{Z\ \mathbf{r}_3 \cdot \mathbf{p} + b_z} \qquad [5.42]$$

$$y' = \frac{Z\ \mathbf{r}_2 \cdot \mathbf{p} + b_y}{Z\ \mathbf{r}_3 \cdot \mathbf{p} + b_z} \qquad [5.43]$$

On peut donc exprimer la position d'un point de l'image de droite en fonction de *son correspondant* dans l'image de gauche, des *paramètres du capteur* et de la *profondeur* Z du point P.

5.8.3. La contrainte épipolaire

En éliminant Z entre les équations [5.42] et [5.43] on obtient une relation linéaire entre x' et y' :

$$(b_z\ \mathbf{r}_2 \cdot \mathbf{p} - b_y\ \mathbf{r}_3 \cdot \mathbf{p})\ x' + (b_x\ \mathbf{r}_3 \cdot \mathbf{p} - b_z\ \mathbf{r}_1 \cdot \mathbf{p})\ y' = b_x\ \mathbf{r}_2 \cdot \mathbf{p} - b_y\ \mathbf{r}_1 \cdot \mathbf{p} \quad [5.44]$$

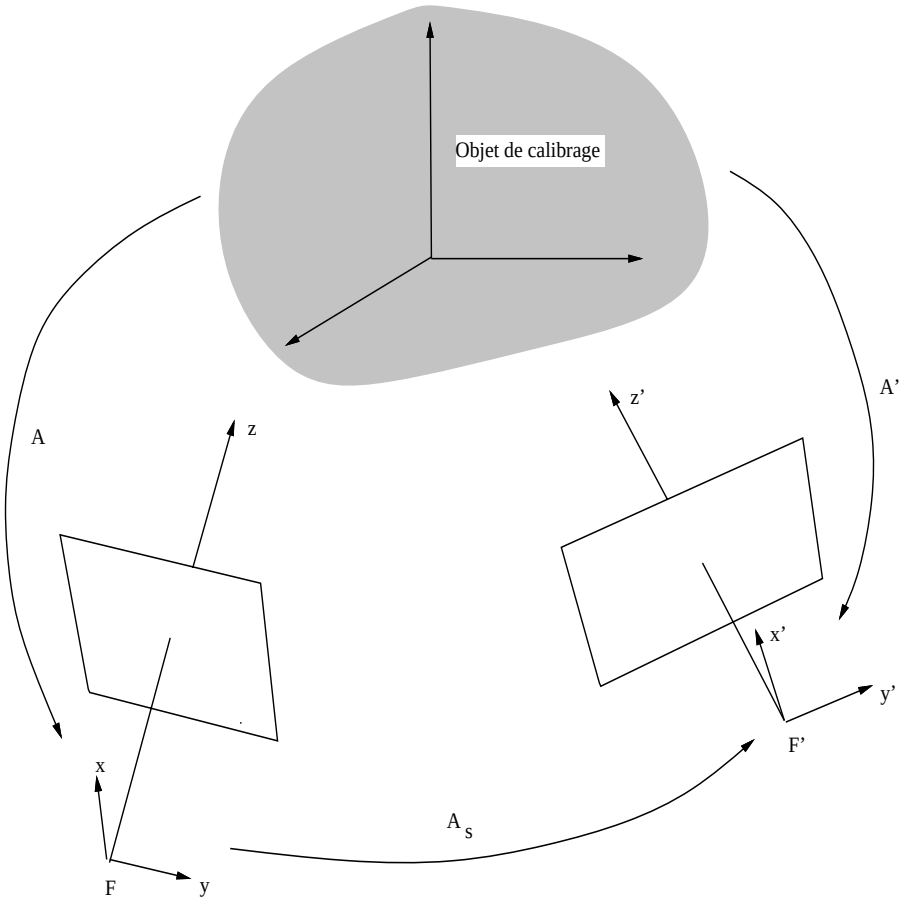


Figure 5.13. *Le calibrage de deux caméras formant un capteur stéréo à partir d'une seule mire.*

Cette équation décrit le lieu des points de l'image de droite pouvant correspondre à un point p de l'image de gauche : c'est une ligne qui s'appelle la ligne épipolaire droite. Il y a pour chaque point de l'image de gauche une telle ligne épipolaire (et réciproquement pour chaque point de l'image de droite il y a une ligne épipolaire gauche). Il est facile de remarquer que toutes les lignes épipolaires droites forment un faisceau. Le point commun de toutes ces lignes s'appelle l'épipôle droit et il s'obtient facilement en prenant $Z = 0$ dans les équations [5.42] et [5.43]. On obtient les coordonnées de l'épipôle droit, e' dans le repère de droite :

$$\begin{cases} x'_e &= \frac{b_x}{b_z} \\ y'_e &= \frac{b_y}{b_z} \end{cases}$$

L'interprétation géométrique est immédiate. En effet le centre de projection F de la caméra gauche a comme coordonnées dans le repère de droite $(b_x b_y b_z)^t$ (le vecteur translation de la transformation gauche-droite). L'épipôle droit n'est autre que la projection de F dans l'image de droite. On peut également définir un épipôle gauche. La figure 5.14. montre les deux faisceaux de lignes épipolaires.

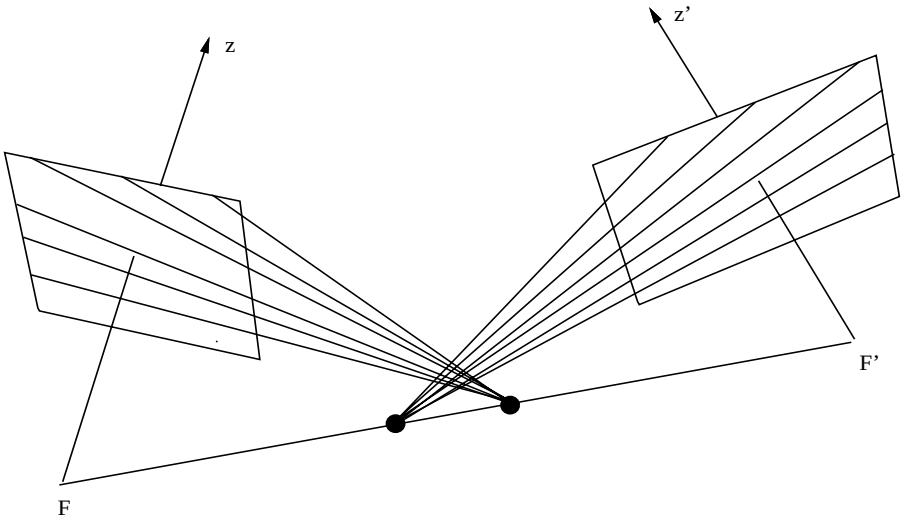


Figure 5.14. *Les deux faisceaux de lignes épipolaires et les deux épipôles.*

5.8.4. Rectification épipolaire

Il existe une configuration géométrique du capteur particulièrement intéressante : c'est lorsque les deux caméras sont disposées de telle façon que leurs axes optiques (z et z') sont parallèles et lorsque la droite FF$'$ est confondue avec les axes horizontaux (y et y') des deux caméras. La matrice de transformation gauche/droite devient :

$$\mathbf{A}_b = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad [5.45]$$

Dans ce cas l'équation d'une ligne épipolaire droite devient tout simplement (en substituant dans l'équation [5.44]) :

$$x' = x$$

Autrement dit, dans ce cas les lignes épipolaires sont parallèles. A un point de l'image de gauche se trouvant sur la ligne x correspond la ligne x' de l'image de droite. Dans ce cas les deux épipôles se trouvent à l'infini. Cette configuration est illustrée sur la figure 5.15.

Les lignes épipolaires jouent un rôle fondamental en vision stéréoscopique, comme nous allons le voir au chapitre 6. En effet, lorsqu'on cherche pour un point de l'image de gauche un correspondant dans l'image de droite, on peut limiter cette recherche le long de la ligne épipolaire correspondante. Il est donc important d'avoir les expressions mathématiques les plus simples pour ces épipolaires. On ne peut pas toujours disposer les deux caméras de façon que leur matrice gauche-droite associée soit réduite à une translation le long de FF$'$.

Cependant on peut *rectifier* les deux images, soit faire subir à chaque image une transformation de façon à obtenir une paire d'images stéréo coplanaires et parallèles à la droite passant par les centres de projection des deux caméras, [8] [118]. La position des caméras pour des images rectifiées est telle que :

- les deux centres de projection (F et F$'$) restent inchangés. De cette façon on pourra reconstruire la scène à partir des images rectifiées,
- les axes Fy et F$'y'$ des caméras dans la position rectifiée sont parallèles et confondus avec la droite FF$'$.

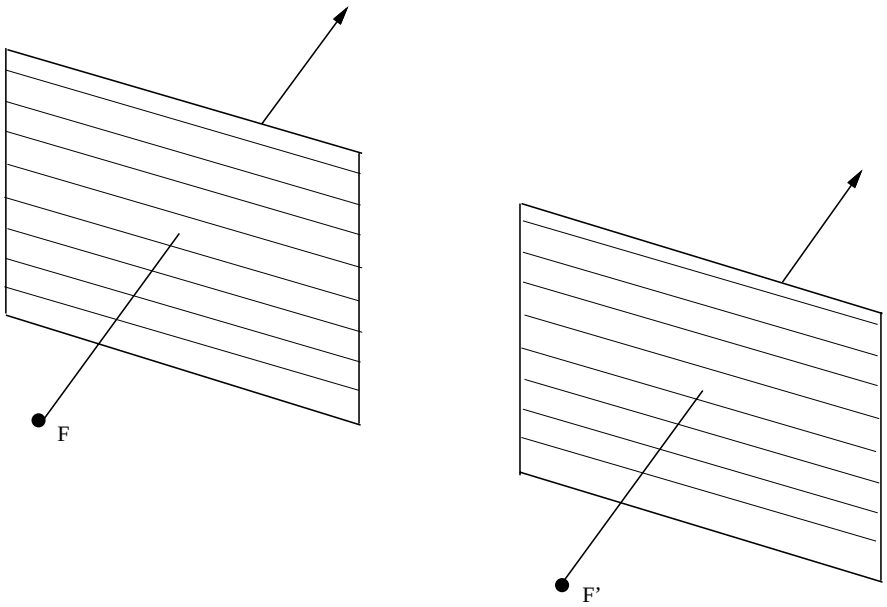


Figure 5.15. *La rectification épipolaire ramène le capteur stéréo dans une configuration telle que les lignes épipolaires sont parallèles aux axes Fy et $F'y'$. Dans cette configuration les deux épipôles sont rejetés à l'infini.*

La figure 5.16. illustre le principe de la rectification épipolaire. Un point P de la scène se projette en p et p′ sur les images initiales et en q et q′ sur les images rectifiées. Comme les centres de projections restent les mêmes, on peut reconstruire le point P, soit à partir de p et p′, soit à partir de q et q′. On peut noter que les paramètres intrinsèques de la projection ne sont pas modifiés. Ce qui change est la relation spatiale entre le repère de calibrage et la caméra. La transformation entre les deux positions (position initiale d'une caméra et position rectifiée de la même caméra) est donc une rotation dans l'espace.

Afin de rectifier une paire d'images stéréo, il faut donc appliquer à chacune des caméras une transformation spatiale. Les centres de projection restant inchangés, cette transformation est une rotation, figure 5.17.

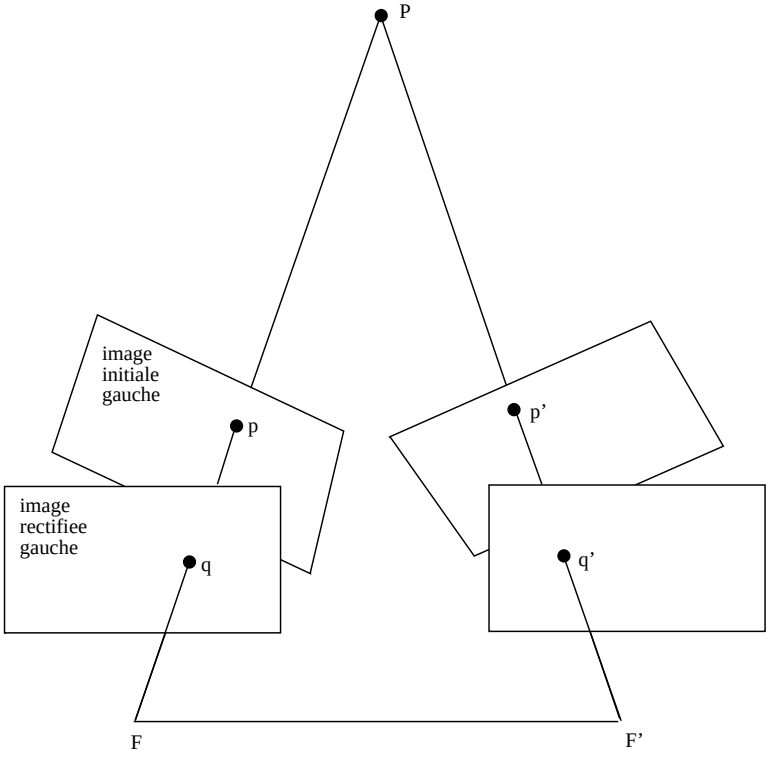


Figure 5.16. *Le principe de rectification épipolaire.*

Si on se place dans le repère de calibrage, soit le repère $OXYZ$ de la figure 5.17., la calibration du capteur stéréo nous fournit les transformations

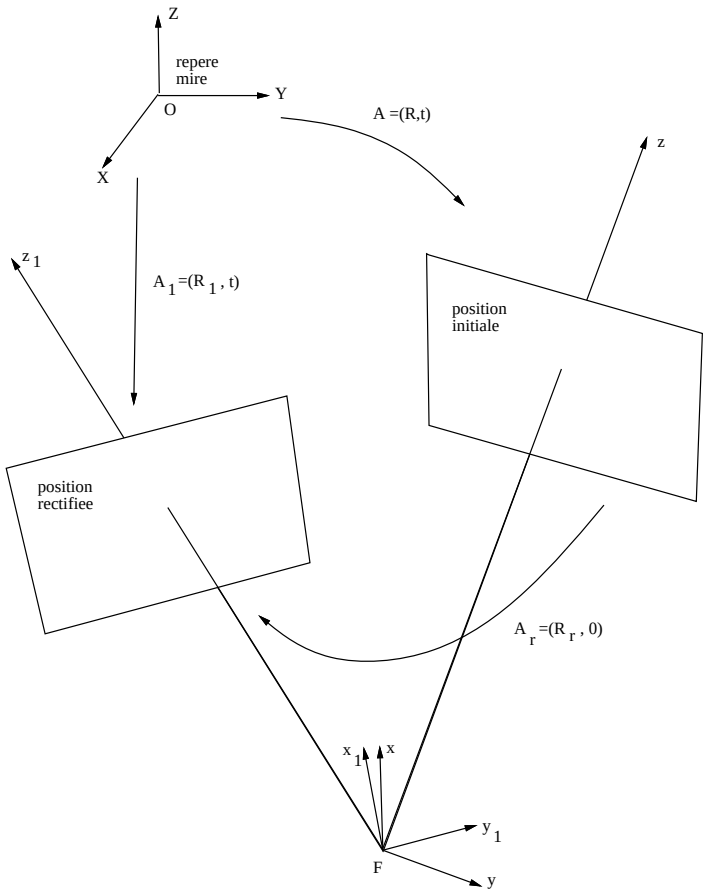


Figure 5.17. *Une image rectifiée s'obtient en appliquant à la caméra correspondante une rotation spatiale.*

$\mathbf{A}$ et $\mathbf{A}'$ du repère de calibrage vers les repères des caméras gauche et droite respectivement. Chacune de ses transformations se compose d'une rotation et d'une translation :

$$\mathbf{A} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix}$$

$$\mathbf{A}' = \begin{pmatrix} \mathbf{R}\,' & \mathbf{t}' \\ \mathbf{0} & 1 \end{pmatrix}$$

On cherche à déterminer la transformation du repère de calibrage au repère de la caméra gauche dans sa position rectifiée. Soit $\mathrm{F}x_1y_1z_1$ le repère de la caméra gauche dans sa position rectifiée.

Les coordonnées de F dans le repère de calibrage s'obtiennent en remarquant que F est l'origine du repère de la caméra de gauche. On aura donc :

$$\mathbf{R}\,\overrightarrow{OF} + \mathbf{t} = 0$$

ainsi qu'une expression similaire pour la caméra de droite :

$$\mathbf{R}\,'\overrightarrow{OF}' + \mathbf{t}' = 0$$

et on obtient ainsi la direction de la droite FF′ :

$$\begin{aligned} \overrightarrow{FF'} &= \overrightarrow{OF'} - \overrightarrow{OF} \\ &= \mathbf{R}^{-1}\mathbf{t} - \mathbf{R}\,'^{-1}\mathbf{t}' \end{aligned}$$

Cette droite n'est pas modifiée lors de la rectification. Choisissons les axes des caméras rectifiées. Pour la caméra de gauche ces axes sont $\mathrm{F}x_1$, $\mathrm{F}y_1$ et $\mathrm{F}z_1$. L'axe $\mathrm{F}y_1$ doit être parallèle à la droite FF′. On obtient donc pour le vecteur directeur de cet axe :

$$\mathbf{j}_1 = \frac{\overrightarrow{FF'}}{\|\overrightarrow{FF'}\|}$$

Les deux autres axes doivent être dans un plan perpendiculaire au vecteur $\mathbf{j}_1$. On peut par exemple choisir un de ces axes, soit l'axe optique, dans un plan contenant O (l'origine du repère de calibrage), F et F′. Le vecteur directeur de l'axe optique sera :

$$\mathbf{k}_1 = \frac{\mathbf{K}_1}{\|\mathbf{K}_1\|}$$

avec :

$$\mathbf{K}_1 = (\overrightarrow{OF} \wedge \overrightarrow{OF'}) \wedge \overrightarrow{FF'}$$

Le sens de $\mathbf{K}_1$ peut être déterminé par la contrainte :

$$\mathbf{K}_1 \cdot \mathbf{K} > 0$$

où $\mathbf{K}$est le vecteur directeur de l'axe optique de la caméra gauche dans sa position initiale. L'axe restant sera défini par :

$$\mathbf{i}_1 = \mathbf{j}_1 \wedge \mathbf{k}_1$$

La transformation du repère mire au repère de la caméra gauche rectifiée est donc :

$$\mathbf{A}_1 = \begin{pmatrix} \mathbf{R}_1 & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix}$$

avec :

$$\mathbf{R}_1 = \begin{pmatrix} \mathbf{i}_1 & \mathbf{j}_1 & \mathbf{k}_1 \end{pmatrix}$$

la translation étant la même que pour la transformation $\mathbf{A}$.

La rectification de l'image gauche est obtenue en appliquant une rotation au repère initial de la caméra gauche pour la ramener dans la position rectifiée :

$$\mathbf{R}_r = \mathbf{R}_1 \mathbf{R}^{-1}$$

Les coordonnées rectifiées s'obtiennent en appliquant aux coordonnées initiales cette rotation :

$$\begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = \mathbf{R}_r \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Bien sûr, le même raisonnement s'applique à la caméra de droite. Les deux images rectifiées étant parallèles, la transformation de rectification de la caméra de droite est :

$$\mathbf{R}'_r = \mathbf{R}_1 \mathbf{R}'^{-1}$$

Une fois que les deux images ont été rectifiées on a une géométrie épipolaire telle qu'elle a été décrite au début de ce paragraphe. La relation gauche/droite devient celle décrite par la transformation donnée par l'équation [5.45] :

$$\begin{aligned} \mathbf{A}_b &= \mathbf{A}'_1 \mathbf{A}_1^{-1} \\ &= \begin{pmatrix} \mathbf{R}_1 & \mathbf{t}' \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{R}_1 & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix}^{-1} \end{aligned}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

avec $b = (b_x^2 + b_y^2 + b_z^2)^{1/2}$.

5.9. Capteurs stéréoscopiques actifs

Nous pouvons combiner une caméra avec une source de lumière afin de mesurer les coordonnées tri-dimensionnelles de points sur la surface d'un objet. Un faisceau laser (lumière monochromatique) éclaire une scène, plus précisément éclaire une *tranche* de la scène qui correspond à l'intersection du plan lumineux et de la scène. La figure 5.18. illustre un capteur actif. Une caméra observe cette tranche de lumière qui se projette dans l'image. A un instant donné la seule information disponible dans l'image est la projection de cette tranche de lumière. Un point P de la scène se trouvant sur cette tranche se projette dans l'image en p.

Dans ce qui suit on va voir comment on peut calculer les coordonnées de P dans le repère mire connaissant :

- les coordonnées u et v de sa projection dans le plan image (après correction éventuelle de la distorsion) ;
- l'équation du plan de lumière dans le repère scène et
- la matrice de calibrage $\mathbf{M}$ de la caméra utilisée, en supposant que le même repère mire est utilisé pour calibrer la caméra et le capteur actif.

Comme nous allons le voir il est important de choisir le repère mire de façon que l'axe des Z soit perpendiculaire au plan de travail.

Le principe de calibrage qui sera décrit par la suite est dû à Bolles, Kremers et Cain, [23]. Les équations décrivant la droite passant par P, p et F sont (dans le repère mire) :

$$\begin{aligned} (m_{11} - um_{31})\, X + (m_{12} - um_{32})\, Y + (m_{13} - um_{33})\, Z &= -(m_{14} - um_{34}) \\ (m_{21} - vm_{31})\, X + (m_{22} - vm_{32})\, Y + (m_{23} - vm_{33})\, Z &= -(m_{24} - vm_{34}) \end{aligned}$$

Ce sont exactement les mêmes équations que les formules [5.16] et [5.17]. A ces équations on rajoute l'équation du plan de lumière (qu'on ne connaît pas

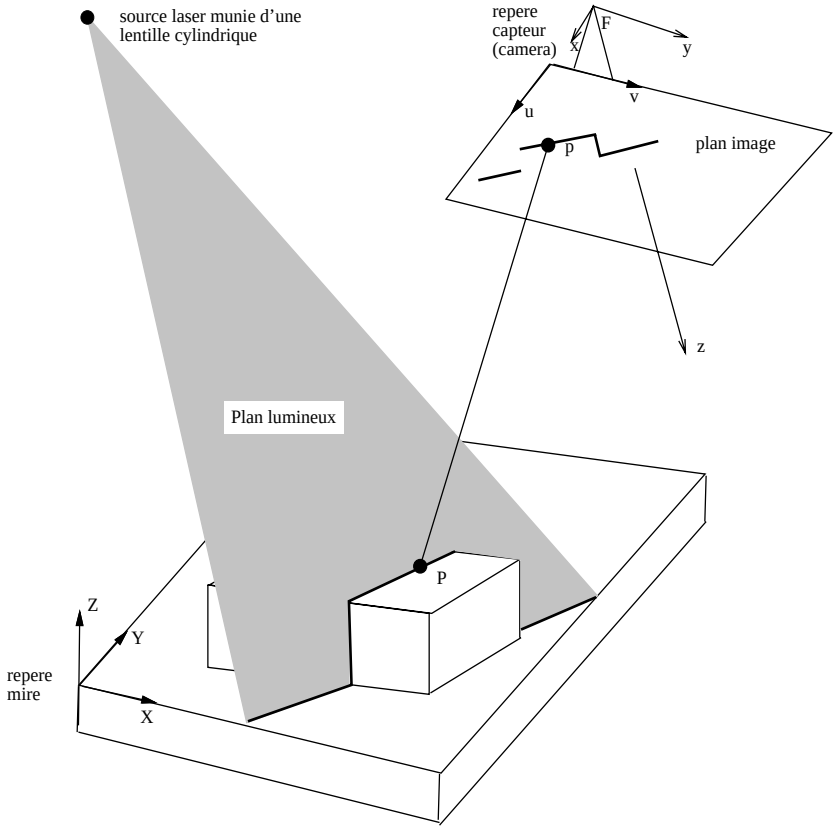


Figure 5.18. *Un capteur actif muni d'une caméra et d'un plan de lumière.*

encore) :

$$b_1X + b_2Y + b_3Z + b_4 = 0 \qquad [5.46]$$

Ces trois équations peuvent se combiner en une notation matricielle.

$$\mathbf{N} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} -b_4 \\ -(m_{14} - um_{34}) \\ -(m_{24} - vm_{34}) \end{pmatrix}$$

où :

$$\mathbf{N} = \begin{pmatrix} b_1 & b_2 & b_3 \\ m_{11} - um_{31} & m_{12} - um_{32} & m_{13} - um_{33} \\ m_{21} - vm_{31} & m_{22} - vm_{32} & m_{23} - vm_{33} \end{pmatrix}$$

En inversant on peut donc calculer les coordonnées X, Y et Z de P. Cependant les calculs risquent d'être lourds car ils faut inverser une matrice 3×3 pour chaque point. Il existe une autre solution qui consiste à inverser la matrice 3×3 symboliquement (utilisant un système de calcul formel – *Maple*, *Mathematica*, etc.) et à la multiplier par le vecteur de droite. Le calcul de X, Y et Z peut alors se mettre sous la forme suivante :

$$\begin{pmatrix} sX \\ sY \\ sZ \\ s \end{pmatrix} = \begin{pmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \\ t_{41} & t_{42} & t_{43} \end{pmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \qquad [5.47]$$

Ce qui est équivalent à :

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \mathbf{N}^{-1} \begin{pmatrix} -b_4 \\ -(m_{14} - um_{34}) \\ -(m_{24} - vm_{34}) \end{pmatrix}$$

La matrice $\mathbf{T} = (t_{ij})$ est la matrice de calibrage du capteur actif. Ses coefficients dépendent de la matrice de calibrage de la caméra et de l'équation du plan de lumière. Les coefficients de $\mathbf{T}$ s'obtiennent facilement par identification et sont (les m_{ij} ont été divisés par m_{34}) :

$$\begin{aligned} t_{11} &= (b_4m_{22} - b_2m_{24})m_{33} + (b_3m_{24} - b_4m_{23})m_{32} + (b_2m_{23} - b_3m_{22}) \\ t_{12} &= (b_2m_{14} - b_4m_{12})m_{33} + (b_4m_{13} - b_3m_{14})m_{32} + (b_3m_{12} - b_2m_{13}) \\ t_{13} &= (b_2m_{13} - b_3m_{12})m_{24} + (b_4m_{12} - b_2m_{14})m_{23} + (b_3m_{14} - b_4m_{13})m_{22} \end{aligned}$$

$$
\begin{aligned}
t_{21} &= (b_1 m_{24} - b_4 m_{21}) m_{33} + (b_4 m_{23} - b_3 m_{24}) m_{31} + (b_3 m_{21} - b_1 m_{23}) \\
t_{22} &= (b_4 m_{11} - b_1 m_{14}) m_{33} + (b_3 m_{14} - b_4 m_{13}) m_{31} + (b_1 m_{13} - b_3 m_{11}) \\
t_{23} &= (b_3 m_{11} - b_1 m_{13}) m_{24} + (b_1 m_{14} - b_4 m_{11}) m_{23} + (b_4 m_{13} - b_3 m_{14}) m_{21}
\end{aligned}
$$

$$
\begin{aligned}
t_{31} &= (b_4 m_{21} - b_1 m_{24}) m_{32} + (b_2 m_{24} - b_4 m_{22}) m_{31} + (b_1 m_{22} - b_2 m_{21}) \\
t_{32} &= (b_1 m_{14} - b_4 m_{11}) m_{32} + (b_4 m_{12} - b_2 m_{14}) m_{31} + (b_2 m_{11} - b_1 m_{12}) \\
t_{33} &= (b_1 m_{12} - b_2 m_{11}) m_{24} + (b_4 m_{11} - b_1 m_{14}) m_{22} + (b_2 m_{14} - b_4 m_{12}) m_{21}
\end{aligned}
$$

$$
\begin{aligned}
t_{41} &= (b_2 m_{21} - b_1 m_{22}) m_{33} + (b_1 m_{23} - b_3 m_{21}) m_{32} + (b_3 m_{22} - b_2 m_{23}) m_{31} \\
t_{42} &= (b_1 m_{12} - b_2 m_{11}) m_{33} + (b_3 m_{11} - b_1 m_{13}) m_{32} + (b_2 m_{13} - b_3 m_{12}) m_{31} \\
t_{43} &= (b_2 m_{11} - b_1 m_{12}) m_{23} + (b_1 m_{13} - b_3 m_{11}) m_{22} + (b_3 m_{12} - b_2 m_{13}) m_{21}
\end{aligned}
$$

5.9.1. Calibrage d'un capteur actif

La calibration du capteur actif consiste donc à calculer les valeurs numériques des coefficients de $\mathbf{T}$. Or nous avons déjà calculé les coefficients m_{ij}. Il nous reste à calculer l'équation du plan de lumière.

Pour cela on procède de la façon suivante. *On place devant le capteur des objets parallélipipédiques dont la hauteur est connue dans le repère de la mire de calibrage.* Un point P_i appartenant à une face de l'objet à la hauteur $Z = H_i$ a comme coordonnées mire X_i, Y_i et H_i et il se projette dans l'image en u_i, v_i. X_i et Y_i peuvent être déterminés grâce aux équations [5.16] et [5.17] qui peuvent s'écrire :

$$
\begin{aligned}
(m_{11} - u_i m_{31})\, X_i + (m_{12} - u_i m_{32})\, Y_i &= -(m_{13} - u_i m_{33})\, H_i + m_{14} - u_i m_{34} \\
(m_{21} - v_i m_{31})\, X_i + (m_{22} - v_i m_{32})\, Y_i &= -(m_{23} - v_i m_{33})\, H_i + m_{24} - v_i m_{34}
\end{aligned}
$$

Par ailleurs, le point P_i appartient au plan de lumière. En divisant par b_3 on obtient une relation linéaire entre b_1, b_2 et b_4 :

$$b_1 X_i + b_2 Y_i + b_4 = -H_i \qquad [5.48]$$

Pour n points appartenant à des faces à différentes hauteurs on obtient n

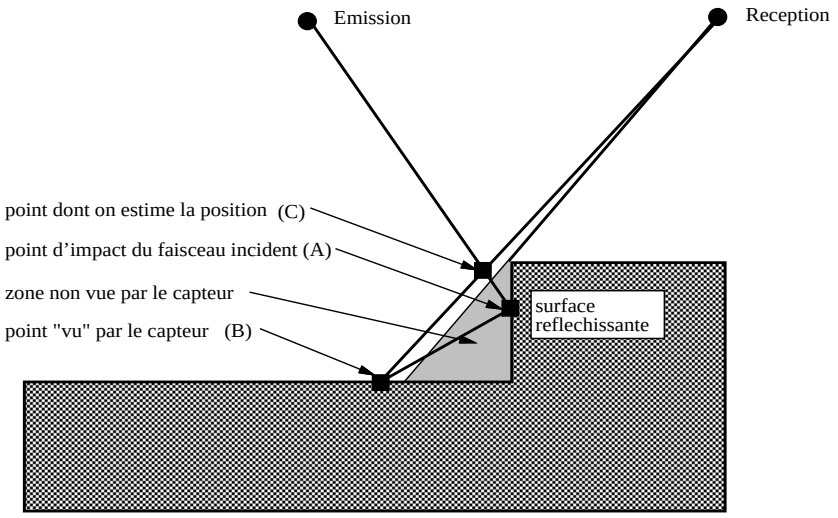


Figure 5.19. *Une configuration montrant un cas typique de la limite d'utilisation d'un capteur actif.*

équations qu'on peut écrire sous forme matricielle :

$$\begin{pmatrix} X_1 & Y_1 & 1 \\ \vdots & \vdots & \vdots \\ X_n & Y_n & 1 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_4 \end{pmatrix} = \begin{pmatrix} -H_1 \\ \vdots \\ -H_n \end{pmatrix} \qquad [5.49]$$

Ceci peut s'écrire sous forme matricielle :

$$\mathbf{A}_{n\times 3}\mathbf{b}_3 = \mathbf{c}_n$$

Les paramètres du plan de lumière sont la solution aux moindres carrés de l'équation [5.49], soit :

$$\mathbf{b} = (\mathbf{A}^t\mathbf{A})^{-1}\mathbf{A}^t\mathbf{c}$$

5.9.2. Limites du capteur actif

L'avantage du capteur actif par rapport à un couple de caméras est évident : on évite le problème combinatoire de la mise en correspondance en "allumant" les points et en calculant directement leur position dans un repère tri-dimensionnel. Cependant ce capteur n'a pas que des avantages.

Le premier inconvénient est qu'on doit bouger soit l'objet soit le capteur lui-même pour pouvoir faire l'acquisition d'une carte dense $Z = f(X, Y)$ où f est un tableau rectangulaire correspondant à la projection au sol de la zone vue par le capteur.

Le deuxième inconvénient est difficilement contournable car il s'agit de propriétés intrinsèques des surfaces qui sont nuisibles à la bonne mesure. La figure 5.19. montre un objet dont la surface est réfléchissante : le plan de lumière éclaire un pan de l'objet qui ne peut être vu par la caméra. Cependant le rayon de lumière se réfléchit pour finalement éclairer un autre point de l'objet qui, lui, est vu par la caméra. Une inspection de la figure montre clairement que le point pour lequel on calcule la position ne se trouve pas sur la surface de l'objet. La figure permet également de comprendre pourquoi certains points d'un objet ne peuvent être mesurés : soit ils sont vus par la caméra mais ils ne sont pas éclairés par le plan lumineux, soit ils sont éclairés mais ne peuvent être vus.

Un troisième inconvénient est illustré sur la figure 5.20. En effet, le faisceau laser étant conique, il y a une imprécision de mesure lorsque le faisceau rencontre une discontinuité sur l'objet analysé.

Enfin, un quatrième inconvénient est illustré sur la figure 5.21. Lorsque le faisceau fait avec la normale de la surface observée un angle d'incidence proche de 90^0, l'énergie du faisceau réfléchissant qui repart vers la caméra est faible. Dans ce cas la mesure peut être entachée d'erreurs.

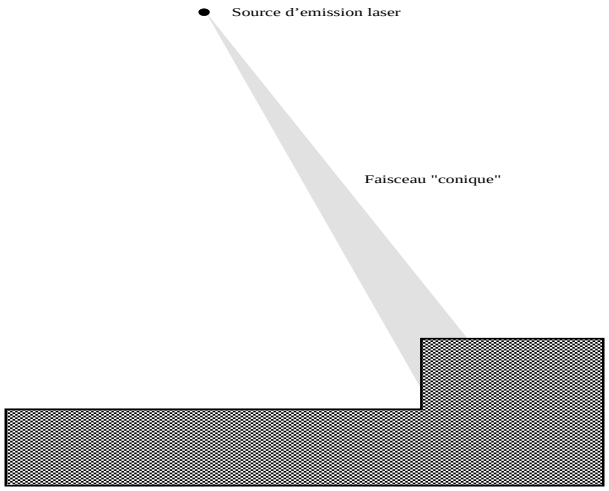


Figure 5.20. *Le faisceau laser étant conique, il y a des problèmes d'imprécision lorsque le faisceau rencontre une discontinuité de l'objet observé.*

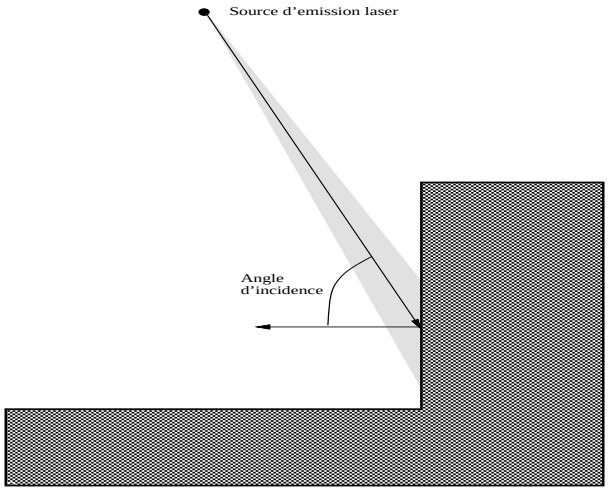


Figure 5.21. *Le faisceau laser a un angle d'incidence proche de 90^0, mesuré par rapport à la normale de la surface mesurée.*

Chapitre 6

Vision stéréoscopique

6.1. Introduction

Un des objectifs de la vision par ordinateur est de reconstruire la structure tri-dimensionnelle (3D) de l'espace à partir d'une ou plusieurs images. La vision stéréoscopique utilise deux images prises avec deux caméras. Connaissant le modèle de projection de chaque caméra et la relation spatiale entre les deux caméras, il s'agit de calculer les coordonnées 3D d'un point à partir de ses deux projections dans les deux images.

Les problèmes d'estimation du modèle de projection et d'estimation de la relation spatiale entre les deux caméras ont été abordés au chapitre 5. Dans ce chapitre nous allons nous concentrer sur les problèmes d'appariement et de reconstruction.

Le problème d'appariement ou de mise en correspondance consiste à trouver un ensemble de couples ou de paires, chaque couple étant formé d'un point d'une image apparié avec un point de l'autre image. Ce problème a une nature combinatoire et la seule façon de réduire cette combinatoire est de définir une mesure de ressemblance entre un point d'une image et un point de l'autre image et de mettre en œuvre des contraintes qui ont trait à la géométrie du capteur stéréoscopique et à la structure de la scène et des objets observés.

Le problème de reconstruction est directement lié au problème de calibration du capteur stéréo qui a été abordé au chapitre 5.

Considérons par exemple la paire d'images stéréoscopiques de la figure 6.1. Chacune de ces deux images représente une projection différente de la même scène 3D. Les propriétés métriques ne se conservant pas par projection il n'est

pas facile de comparer les éléments d'une image à ceux de l'autre image afin de décider si on peut les mettre en correspondance ou non. Il faut également tenir compte du phénomène d'occultation : un élément d'une image peut ne pas être vu dans l'autre image (et *vice versa*) soit par ce qu'il est caché par un autre élément, soit parce qu'il est dans le champ de vue d'une caméra et hors du champ de vue de l'autre caméra, soit enfin à cause de phénomènes physiques tels que les ombres et les reflets qui dépendent des positions relatives de la source, de l'objet observé et de la caméra.

Figure 6.1. *Une paire d'images stéréo. Cet exemple illustre les difficultés de mise en correspondance et notamment le phénomène d'occultation.*

La démarche calculatoire sera la suivante :

- établir un ensemble d'appariements entre les éléments des deux images. Chacun de ces appariements forme une mise en correspondance *locale* ;
- utiliser des contraintes afin de réduire l'espace de recherche d'une mise en correspondance *globale* ;
- mettre en œuvre une méthode de mise en correspondance globale.

Une mise en correspondance locale est tout simplement un appariement qui satisfait une mesure de ressemblance et qui est cohérent avec les contraintes découlant de la calibration du capteur. Une mise en correspondance globale est un ensemble d'appariements qui satisfont à d'autres contraintes qui seront explicitées dans ce chapitre à la section 6.5.

Dans la section suivante nous allons discuter du choix quant aux éléments image à mettre en correspondance. Ce choix dépend d'un certain nombre de

critères qui seront explicités. Ensuite nous allons consacrer une section à la géométrie de la stéréo et à la reconstruction. Cette section reprend la formulation du chapitre 5 en introduisant une méthode permettant le calcul de la géométrie épipolaire sans passer par la calibration des deux caméras.

Nous allons ensuite détailler dans la section 6.4. toutes les contraintes qui peuvent réduire la taille de l'espace de recherche d'une mise en correspondance globale.

Finalement nous allons étudier quelques méthodes de mise en correspondance ainsi que quelques cas particuliers tels que la vision trinoculaire (stéréo avec trois caméras) et la détection d'obstacles avec un système stéréo.

6.2. Primitives stéréoscopiques

Le choix des éléments (ou primitives) image à mettre en correspondance est crucial. La primitive "idéale" est telle que (i) ces propriétés intrinsèques permettent une mesure de ressemblance fortement discriminante, (ii) elle permette la mise en œuvre efficace de contraintes stéréoscopique et (iii) la reconstruction 3D soit possible. Les techniques de segmentation d'images que nous avons étudiées permettent l'extraction d'une gamme variée de primitives telles que :

- *des points* (pixels, points d'intérêt, éléments de contour, points caractéristiques le long d'un contour, jonctions, etc.) ;
- *des segments* (segments de droite, arcs de cercle, portions de conique, etc.) ;
- *des régions.*

Sauf quelques cas particuliers, les régions sont mal adaptées à la vision stéréo et cela pour plusieurs raisons. Une région se trouvant sur la surface d'un objet donnera naissance, dans les deux images, à deux régions de taille et de forme différentes, ceci étant dû au phénomène de distorsion projective. On ne peut donc pas comparer deux régions dans deux images différentes provenant de la même région 3D, figure 6.2. Par ailleurs, en l'absence d'un appariement terme à terme entre des points à l'intérieur des deux régions, on ne pourra pas décider s'il s'agit d'une surface 3D gauche ou planaire, figure 6.2.

Les segments de droite ont été utilisés avec quelque succès mais leur utilisation est limitée aux scènes polyédriques [11], [8], [166] et [96]. Il est à noter qu'un segment de droite a la même représentation géométrique qu'un point de

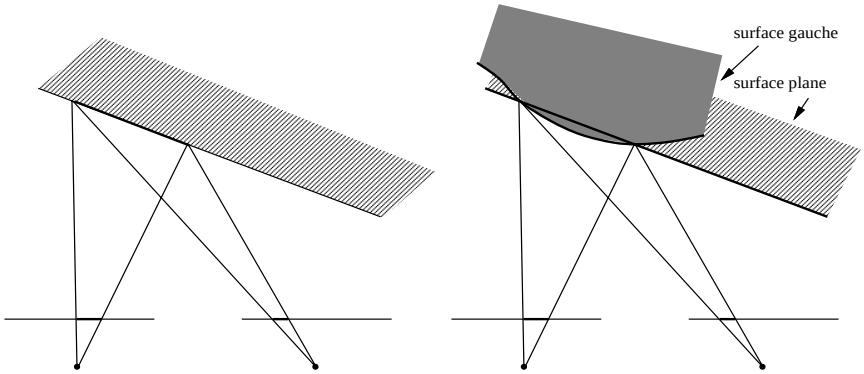


Figure 6.2. *Un objet incliné va donner naissance dans les deux images à deux régions de tailles différentes. Comment peut-on comparer ces deux régions ? (figure de gauche). En l'absence d'un appariement entre les points des deux régions on ne peut décider si cela correspond à une surface gauche ou à une surface plane (figure de droite). Ces phénomènes mettent sérieusement en échec les méthodes de mise en correspondance de régions.*

contour (position dans l'image et direction, figure 6.5.) et les mêmes contraintes stéréoscopiques s'appliquent également aux segments de droite et aux points de contour.

Bien que plus "pauvres" quant au contenu sémantique, les points s'avèrent finalement être les primitives les mieux adaptées pour la stéréo. La plupart des contraintes que nous allons étudier s'appliquent aux points, la contrainte figurale étant réservée aux points se trouvant le long d'un contour. Le seul vrai problème que pose l'utilisation des points comme primitives pour l'appariement est le choix d'une mesure de similarité (ressemblance) qui ne soit pas perturbée par la distorsion projective. A cette distorsion près, on peut comparer deux points (ou plus précisément deux régions carrées de taille égale) en utilisant une mesure de corrélation. Soit $\mathbf{p}$ un point d'une image de coordonnées $(u\ v)$ et $\mathbf{p}'$ un point de l'autre image de coordonnées $(u'\ v')$. Une mesure de similarité est le degré de corrélation entre les deux régions carrées entourant chacun de ces points :

$$c(\mathbf{p},\mathbf{p}') = \sum_{x=-M}^{x=M} \sum_{y=-N}^{y=N} \left[I(u-x,v-y) - I'(u'-x,v'-y)\right]^2 \qquad [6.1]$$

pour une région de taille $2M \times 2N$. $I(u,v)$ est la valeur de niveau de gris au pixel de coordonnées u et v. La corrélation est nulle pour une ressemblance parfaite et strictement positive sinon. En développant l'expression de la corrélation de l'équation [6.1] et en négligeant les termes constants on obtient l'expression du produit de corrélation entre deux régions de deux images différentes centrées aux points $\mathbf{p}$ et $\mathbf{p}'$:

$$C(\mathbf{p},\mathbf{p}') = \sum_{x=-M}^{x=M} \sum_{y=-N}^{y=N} [I(u-x, v-y) I'(u'-x, v'-y)] \qquad [6.2]$$

Le produit de corrélation est maximal pour une ressemblance parfaite. Généralement le produit de corrélation est appliqué aux images brutes (I et I') ayant éventuellement subi une opération de normalisation car la dynamique des deux caméras n'est pas forcément identique. Voir à ce propos l'étude comparative qui a été menée dans [7]. Une amélioration de la mise en correspondance par corrélation qui prend en ligne de compte une déformation affine est proposée dans [158].

Quel que soit le type de primitive choisi, une propriété importante est la fréquence de sa présence dans les deux images. La présence d'un grand nombre de primitives augmente la combinatoire mais en même temps *densifie* la reconstruction 3D qui en découle. Lorsque les primitives ne sont pas nombreuses la combinatoire est réduite mais la reconstruction sera d'autant moins bonne. Une représentation multi-résolution des images résout ce paradoxe de la façon suivante. Chaque image est décrite par un nombre croissant de primitives au fur et à mesure qu'on augmente la résolution. A faible résolution la combinatoire est réduite mais on a une reconstruction pauvre. L'appariement obtenu à faible résolution peut contraindre la combinatoire à haute résolution pour obtenir une reconstruction plus riche. Ce concept sera détaillé à la fin de ce chapitre (section 6.5.1.).

6.3. Géométrie épipolaire et reconstruction

Reprenons la présentation d'un capteur stéréo faite au chapitre 5. En particulier nous avons établi les équations [5.42] et [5.43]. Ces équations expriment la relation entre les coordonnées d'un point p′ de l'image de gauche, de son correspondant p dans l'image de droite, la matrice de transformation gauche/droite et la profondeur Z du point P de la scène qui se projette en p et p′.

En éliminant Z entre ces deux équations on obtient (rappelons que $\mathbf{p}$ est le vecteur de coordonnées $(x\ y\ 1)$ correspondant au point p) :

$$(b_z\ \mathbf{r}_2 \cdot \mathbf{p} - b_y\ \mathbf{r}_3 \cdot \mathbf{p})\ x' + (b_x\ \mathbf{r}_3 \cdot \mathbf{p} - b_z\ \mathbf{r}_1 \cdot \mathbf{p})\ y' = b_x\ \mathbf{r}_2 \cdot \mathbf{p} - b_y\ \mathbf{r}_1 \cdot \mathbf{p} \quad [6.3]$$

qui est l'équation d'une droite de la forme :

$$a'x' + b'y' + c' = 0 \quad [6.4]$$

Les $'$ indiquent qu'il s'agit d'une droite exprimée dans le repère de la caméra droite. Cette droite n'est rien d'autre que la contrainte épipolaire étudiée également au chapitre 5. On peut remarquer par ailleurs que les paramètres de cette droite, a', b' et c', ont les expressions suivantes :

$$\begin{aligned} a' &= (b_y r_{31} - b_z r_{21})x + (b_y r_{32} - b_z r_{22})y + (b_y r_{33} - b_z r_{23}) \\ b' &= (b_z r_{11} - b_x r_{31})x + (b_z r_{12} - b_x r_{32})y + (b_z r_{13} - b_x r_{33}) \\ c' &= (b_x r_{21} - b_y r_{11})x + (b_x r_{22} - b_y r_{12})y + (b_x r_{23} - b_y r_{13}) \end{aligned}$$

6.3.1. La matrice essentielle

Les expressions que nous venons d'établir pour les paramètres a', b' et c' peuvent se mettre sous forme matricielle :

$$\begin{pmatrix} a' \\ b' \\ c' \end{pmatrix} = \begin{pmatrix} 0 & -b_z & b_y \\ b_z & 0 & -b_x \\ -b_y & b_x & 0 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad [6.5]$$

C'est donc la transformation épipolaire qui, à un point de l'image de gauche $(x\ \ y\ \ 1)^t$, fait correspondre une droite de l'image de droite décrite par ses paramètres $(a'\ b'\ c')$:

$$\begin{pmatrix} a' \\ b' \\ c' \end{pmatrix} = \mathbf{E} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad [6.6]$$

L'équation de la droite épipolaire s'écrit :

$$(x'y'1) \begin{pmatrix} a' \\ b' \\ c' \end{pmatrix} = 0$$

ou encore :

$$(x'y'1)\mathbf{E}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = 0$$

soit

$$\mathbf{p}'^t\mathbf{E}\mathbf{p} = 0 \qquad [6.7]$$

La matrice $\mathbf{E}$ décrit la *transformation épipolaire* gauche-droite – elle permet de calculer l'équation d'une droite épipolaire passant par l'image de droite associée à un point de l'image de gauche. On remarque que la transformation épipolaire droite-gauche est donnée par la matrice transposée :

$$\mathbf{p}^t\mathbf{E}^t\mathbf{p}' = 0 \qquad [6.8]$$

La matrice $\mathbf{E}$ s'appelle la *matrice essentielle*. Elle est le produit de deux matrices : une matrice antisymétrique (de rang 2) et une matrice orthonormée (de rang 3). On peut remarquer que cette matrice peut être calculée à partir des paramètres b_x, b_y, b_z et $\mathbf{r}_1$, $\mathbf{r}_2$, $\mathbf{r}_3$. Ces paramètres sont fournis lors de la calibration du capteur stéréoscopique.

6.3.2. La matrice fondamentale

La relation entre les coordonnées caméra et les coordonnées image est donnée par (pour la caméra de gauche) :

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

ou :

$$\mathbf{m} = \mathbf{C}\mathbf{p}$$

On a la même expression pour la caméra de droite :

$$\mathbf{m}' = \mathbf{C}'\mathbf{p}'$$

Les éléments des matrices $\mathbf{C}$ et $\mathbf{C}'$ sont les paramètres intrinsèques des deux caméras. En remplaçant dans l'équation [6.7] on obtient :

$$\mathbf{m}'^t(\mathbf{C}'^{-1})^t\mathbf{E}\mathbf{C}^{-1}\mathbf{m} = 0$$

La matrice :

$$\mathbf{F} = (\mathbf{C}'^{-1})^t \mathbf{E} \mathbf{C}^{-1} \qquad [6.9]$$

est la matrice fondamentale qui décrit la géométrie épipolaire. L'équation :

$$\mathbf{m}'^t \mathbf{F} \mathbf{m} = 0 \qquad [6.10]$$

n'est rien d'autre que l'équation d'une droite épipolaire dans le repère image (et non pas dans le repère caméra comme dans l'équation [6.7]).

Comme la matrice essentielle, la matrice fondamentale peut se calculer pour un capteur stéréo préalablement calibré.

6.3.3. Estimation de la matrice fondamentale

La contrainte épipolaire, ou plus généralement la géométrie épipolaire, joue un rôle fondamental en vision stéréoscopique. Il serait intéressant d'analyser dans quelle mesure il est possible d'estimer la matrice fondamentale *sans calibrer* le capteur stéréo, à partir de correspondances gauche-droite. En effet la matrice fondamentale peut s'estimer grâce à l'équation [6.10].

La méthode la plus directe est une méthode linéaire [83]. Pour chaque correspondance ($\mathrm{m} \leftrightarrow \mathrm{m}'$) on a une équation linéaire du type :

$$f_{11}uu' + f_{12}vu' + f_{13}u' + f_{21}uv' + f_{22}vv' + f_{23}v' + f_{31}u + f_{32}v + f_{33} = 0$$

Si l'on dispose de n correspondances gauche-droite on obtient l'equation matricielle suivante :

$$A\mathbf{f} = 0 \qquad [6.11]$$

Dans cette équation A est une matrice de taille $n \times 9$ et $\mathbf{f}$ est un vecteur dont les composantes sont les 9 éléments de la matrice fondamentale. Afin d'éviter la solution triviale on peut rajouter la contrainte suivante :

$$\|\mathbf{f}\| = 1$$

Il est donc possible de résoudre l'équation [6.11] avec 8 correspondances. On obtient alors une solution exacte. Avec plus de 8 correspondances ($n > 8$) on obtient une solution en minimisant un critère quadratique.

Si les données dont on dispose sont parfaites alors la matrice A est de rang 8. Il a été, en effet, montré que sauf configurations exceptionnelles, la matrice

A a un rang égal à 8 [117]. En pratique il y a du bruit est le rang de A est égal à 9. On peut alors estimer $\mathbf{f}$ en minimisant le critère suivant :

$$\min_{\mathbf{f}} \left(\|A\mathbf{f}\|^2 + \lambda(\|\mathbf{f}\|^2 - 1) \right) \qquad [6.12]$$

La solution de ce problème de minimisation est bien connue (voir plus loin l'équation [7.24]) et $\mathbf{f}$ est donné par le vecteur propre correspondant à la plus petite valeur propre de la matrice A^tA.

Une propriété importante de la matrice fondamentale (et de la matrice essentielle) est qu'elle a un rang égal à 2. La façon la plus pratique d'introduire cette contrainte est de remplacer la matrice $\mathbf{F}$ estimée comme auparavant, par une matrice $\mathbf{F}'$ qui a un rang égal à 2. Pour faire ceci, considérons une décomposition en valeurs singulières de $\mathbf{F}$:

$$\mathbf{F} = UD(r, s, t)V^t$$

Dans cette expression D est une matrice diagonale 3×3 qui contient les valeurs singulières (r, s, et t) de $\mathbf{F}$, avec $r \geq s \geq t$. On peut alors forcer le rang de $\mathbf{F}'$ à être égal à 2 :

$$\mathbf{F}' = UD(r, s, 0)V^t$$

Dans un article paru récemment Hartley [83] montre comment on peut rendre la solution linéaire encore plus robuste numériquement tout simplement en opérant un changement de repère sur les données. Une solution linéaire qui tient explicitement compte de la géométrie du problème et qui donne également des résultats très satisfaisants a été recemment proposée par Boufama et Mohr [25].

La matrice fondamentale peut également être estimée par des méthodes non-linéaires. On peut, par exemple, minimiser la somme des carrés des distances d'un point à la droite épipolaire qui est censée passer par ce point. En effet, l'équation [6.10] exprime le fait que le point m′ se trouve sur la droite épipolaire dont les paramètres sont donnés par :

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{pmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$$

Plus généralement, la distance d'un point m′ à cette droite est donnée par la forme quadratique suivante :

$$d(\mathbf{m}', \mathbf{F}\mathbf{m}) = \mathbf{m}'^t\mathbf{F}\mathbf{m}$$

Les paramètres a et b de la droite étant :

$$a = f_{11}u + f_{12}v + f_{13}$$

$$b = f_{21}u + f_{22}v + f_{23}$$

En pratique, Faugeras, Luong & Maybank [56] proposent de considérer simultanément la distance de $\mathbf{m}'$ à la droite $\mathbf{F}\mathbf{m}$ et la distance de $\mathbf{m}$ à la droite $\mathbf{F}^t\mathbf{m}'$ afin de réduire les différences entre la géométrie épipolaire gauche-droite et droite-gauche. Le critère à minimiser est donc le suivant :

$$\min_{\mathbf{F}} \left(\sum_{i=1}^{i=n} \frac{(\mathbf{m}_i'^t \mathbf{F} \mathbf{m}_i)^2}{a_i^2 + b_i^2} + \sum_{i=1}^{i=n} \frac{(\mathbf{m}_i^t \mathbf{F}^t \mathbf{m}_i')^2}{a_i'^2 + b_i'^2} \right) \qquad [6.13]$$

pour n correspondances gauche-droite. Les paramètres a' et b' sont donnés par :

$$a' = f_{11}u' + f_{21}v' + f_{31}$$

$$b' = f_{12}u' + f_{22}v' + f_{32}$$

Comme la matrice essentielle, la matrice fondamentale est de rang 2. On peut alors écrire cette matrice, sans perte de généralité, de la façon suivante (les trois lignes de la matrice sont linéairement dépendantes) :

$$\mathbf{F} = \begin{pmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7x_1 + x_8x_4 & x_7x_2 + x_8x_5 & x_7x_3 + x_8x_6 \end{pmatrix}$$

Cette fois-ci la minimisation de Q (équation [6.13]) fournira donc une estimation des paramètres x_1 à x_8 à partir desquels on déduit facilement les coefficients de la matrice fondamentale. Cette minimisation implique l'utilisation de techniques d'optimisation non-linéaires telles que Newton, Gauss-Newton ou Levenberg-Marquardt.

L'estimation non linéaire de la matrice fondamentale est moins sensible au bruit que la méthode linéaire. Les deux méthodes (linéaire et non linéaire) supposent cependant un appariement parfait de points entre les deux images. S'il y a des appariements incorrects, aucune des méthodes ne donne des résultats satisfaisants. Pour pallier au problème d'appariements incorrects, des méthodes statistiques robustes peuvent être utilisées, comme dans [183].

6.3.4. Reconstruction

La reconstruction tri-dimensionnelle à partir d'une paire d'images suppose que l'on dispose de correspondances entre les deux images. Le type de reconstruction qu'on peut alors effectuer dépend du type de calibrage dont on dispose. On peut distinguer les cas suivants :

1. le capteur stéréoscopique est calibré et on dispose alors de paramètres internes de chaque caméra ainsi que de la transformation rigide entre les deux caméras – dans ce cas on obtient une reconstruction euclidienne dans le repère de calibrage ;

2. les paramètres internes de chaque caméra sont connus mais la transformation rigide entre les deux caméras n'est pas connue. Il faut alors estimer la matrice essentielle [55] à partir de laquelle on peut extraire la transformation rigide entre les deux caméra *à un facteur d'echelle près* – dans ce cas on obtient une reconstruction euclidienne dans le repère de l'une ou l'autre des deux caméras ;

3. si aucun calibrage n'est disponible (ni les paramètres internes des caméras, ni la transformation rigide entre les deux caméras) il faut alors estimer la matrice fondamentale comme il a été expliqué auparavant. A partir de la matrice fondamentale on peut obtenir une reconstruction *projective* tridimensionnelle [54].

Si on se place dans le premier de ces cas et si le point p de l'image de gauche a été mis en correspondance avec le point p′ de l'image de droite, on a alors deux ensembles d'équations [5.16] et [5.17] :

$$\begin{aligned}
u &= \frac{m_{11}X + m_{12}Y + m_{13}Z + m_{14}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}} \\
v &= \frac{m_{21}X + m_{22}Y + m_{23}Z + m_{24}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}} \\
u' &= \frac{m'_{11}X + m'_{12}Y + m'_{13}Z + m'_{14}}{m'_{31}X + m'_{32}Y + m'_{33}Z + m'_{34}} \\
v' &= \frac{m'_{21}X + m'_{22}Y + m'_{23}Z + m'_{24}}{m'_{31}X + m'_{32}Y + m'_{33}Z + m'_{34}}
\end{aligned}$$

Les coordonnées X, Y et Z du point P reconstruit, dans le repère de calibrage, se calculent en résolvant ce système de 4 équations linéaires.

On peut également reconstruire le point P dans le repère de la caméra de gauche en utilisant une ou l'autre des équations [5.42] ou [5.43]. Les coordonnées

X, Y et Z du point P seront données dans ce cas par :

$$\begin{aligned} x' &= \frac{(r_{11}x + r_{12}y + r_{13})Z + b_x}{(r_{31}x + r_{32}y + r_{33})Z + b_z} \\ X &= xZ \\ Y &= yZ \end{aligned}$$

Ces équations nous permettent de constater que le déplacement entre la caméra gauche et la caméra droite doit compter une translation. Si la transformation gauche-droite est une rotation pure, la reconstruction n'est pas possible.

Finalement on peut remarquer que lorsque les images ont été rectifiées la matrice décrivant la transformation gauche-droite, équation [5.45], est réduite à une translation et les équations [5.42] et [5.43] deviennent :

$$\begin{aligned} x' &= x \\ y' &= \frac{Zy + b}{Z} \end{aligned}$$

Le point P peut alors être reconstruit dans le repère rectifié de la caméra gauche grâce aux équations suivantes :

$$\begin{aligned} Z &= \frac{b}{y' - y} \\ X &= xZ \\ Y &= yZ \end{aligned}$$

et on remarque que, d'un point de vue géométrique, la reconstruction n'est rien d'autre qu'une triangulation.

En notant :

$$D = y' - y \qquad [6.14]$$

on définit la notion de *disparité* (différence de position entre les deux projections d'un point de la scène) qui est abondamment utilisée en stéréoscopie. Dans la section suivante on développera une contrainte d'appariement en relation avec la variation de la disparité.

6.4. Contraintes pour la mise en correspondance

La question à laquelle nous essayons de répondre dans cette section est la suivante : quels sont les critères à mettre en œuvre pour décider qu'une paire

de primitives gauche/droite est correcte ou pas ? En essayant de répondre à cette question, nous allons développer deux types de contraintes. Un premier type de contraintes nous permettra de valider un appariement primitive-gauche/primitive-droite. Il s'agira des contraintes épipolaire et d'orientation. Un deuxième type de contraintes nous permettra de valider la compatibilité entre deux appariements satisfaisant le premier type de contraintes. Il s'agit des contraintes d'ordre, d'unicité, de gradient de la disparité et figurale.

Supposons par exemple qu'on veuille apparier le point p_1 de l'image de gauche avec le point p'_1 de l'image de droite et le point p_2 de l'image de gauche avec le point p'_2 de l'image de droite. Nous devrons procéder de la façon suivante :

- on vérifie tout d'abord que les points p_1/p'_1 et p_2/p'_2 sont en correspondance épipolaire ;

- on vérifie ensuite grâce à l'équation [6.2] que les régions entourant ces paires de points se ressemblent ;

- finalement on vérifie que les deux appariements sont compatibles au sens des autres contraintes.

6.4.1. La contrainte épipolaire

La contrainte épipolaire est la seule contrainte géométrique intrinsèque au capteur stéréoscopique. En pratique nous avons étudié deux méthodes permettant de calculer la géométrie épipolaire d'un couple de caméras : soit en calibrant les deux caméras par rapport à une mire commune, soit en estimant la matrice fondamentale à partir d'appariements gauche-droite.

La contrainte épipolaire peut s'appliquer aux points et aux segments de droite. Soit P un point de la scène et soient p et p′ ses projections dans les deux images.

Comment se déplace p' le long de la ligne épipolaire lorsque P se trouve plus ou moins loin du capteur ? Puisque le point p est fixe, P se déplace le long de la droite de vue passant par F et p, figure 6.3. Lorsque le point P se trouve près du capteur tout en restant visible dans les deux images, il se projette en a'. Lorsque P se trouve très loin du capteur, il se projette en q' défini tel que la droite $F'q'$ soit parallèle à la droite Fp. Analytiquement, la position de q'

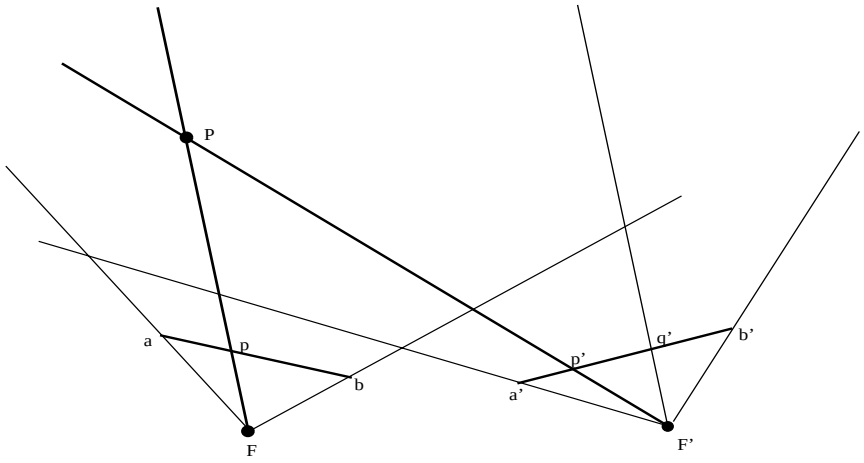


Figure 6.3. *Une vue d'un plan épipolaire : le point p' est contraint d'appartenir au segment $a'q'$.*

s'obtient des équations précédentes en faisant tendre Z vers l'infini :

$$\lim_{Z\to\infty} x' = \frac{\mathbf{r}_1 \cdot \mathbf{p}}{\mathbf{r}_3 \cdot \mathbf{p}}$$

$$\lim_{Z\to\infty} y' = \frac{\mathbf{r}_2 \cdot \mathbf{p}}{\mathbf{r}_3 \cdot \mathbf{p}}$$

p' et donc non seulement contraint le long de la ligne épipolaire mais il appartient également au segment $a'q'$.

Une analyse de la figure 6.3. révèle que les points de l'image de gauche voisins du point a ont intrinsèquement moins de correspondants que les points de l'image de gauche près du point b. Bien sûr, l'analyse gauche/droite que nous venons de faire est valide dans le cas droite/gauche.

La contrainte épipolaire s'applique facilement à des segments de droite. Il suffit pour cela de construire les droites épipolaires correspondant aux deux extrémités du segment. Remarquons toutefois la difficulté à définir une contrainte épipolaire pour des segments contenus dans le plan épipolaire, figure 6.4.

6.4.2. La contrainte d'orientation

Pour les primitives images auxquelles on s'intéresse plus particulièrement (des points de contours et des segments de droite), il y a, en plus de l'informa-

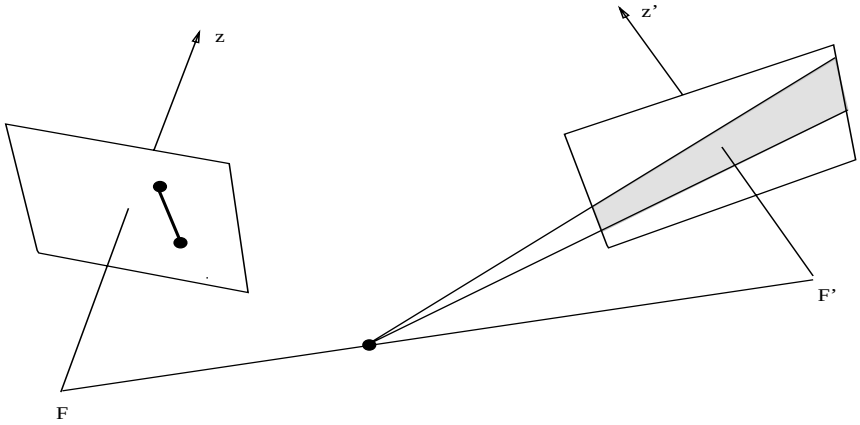


Figure 6.4. *Les deux lignes épipolaires associées aux deux extrémités du segment de l'image de gauche définissent une région dans l'image de droite.*

tion de position dans l'image, une information relative à son orientation par rapport au repère de l'image. Il s'agit de l'angle entre le vecteur directeur associé à cette primitive (dans le cas des points de contour il s'agit de la tangente au contour en ce point) et un des axes du repère de l'image. La valeur de cet angle peut être définie à 2π près en orientant le vecteur directeur par rapport aux valeurs moyennes des niveaux de gris de part et d'autre du contour, figure 6.5.

Soit maintenant une primitive spatiale définie dans le repère de gauche par sa position (X, Y et Z) et son orientation donnée par son vecteur directeur $\mathbf{V} = (V_x, V_y, V_z)^t$. La projection de cette primitive (point et vecteur) dans une image donne naissance à un point p et un vecteur $\mathbf{v} = (v_x, v_y, 0)$. On peut remarquer que le centre de projection F forme avec le point P et le vecteur $\mathbf{V}$ un plan, qu'on appellera *plan d'interprétation*. Le vecteur $\mathbf{v}$ est l'intersection de ce plan avec le plan image, figure 6.6. :

$$\mathbf{v} = \mathbf{k} \wedge (\mathbf{V} \wedge \overrightarrow{FP})$$

Le vecteur $\mathbf{k}$ est le vecteur normal au plan image. Avec le choix de coordonnées qui a été fait ce vecteur n'est autre que la direction de l'axe optique. En développant ce double produit vectoriel et tenant compte du fait que $X = xZ$ et $Y = yZ$ on obtient :

$$v_x = Z(V_x - xV_z)$$

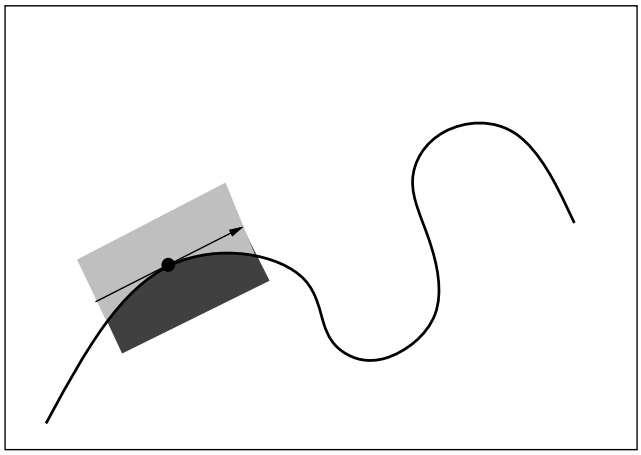


Figure 6.5. *Un point de contour est défini par sa position dans l'image, et la direction de la tangente au contour en ce point. Cette direction peut être définie à 2π près en tenant compte de la moyenne des niveaux de gris de part et d'autre du contour aux environs du point.*

$$\begin{aligned} v_y &= Z(V_y - yV_z) \\ v_z &= 0 \end{aligned}$$

La pente de cette primitive (v) est mesurable dans l'image et est donnée par :

$$\tan\theta = \frac{v_x}{v_y}$$

En exprimant la même chose dans le repère de l'image de droite on obtient une formule semblable pour la pente d'une primitive dans l'image de droite :

$$\tan\theta' = \frac{v'_x}{v'_y}$$

La relation entre les coordonnées du vecteur $\mathbf{V}$ exprimées dans les deux repères s'écrit ($\mathbf{A}$ est la matrice de transformation entre les deux caméras) :

$$\begin{pmatrix} V'_x \\ V'_y \\ V'_z \\ 0 \end{pmatrix} = \mathbf{A} \begin{pmatrix} V_x \\ V_y \\ V_z \\ 0 \end{pmatrix}$$

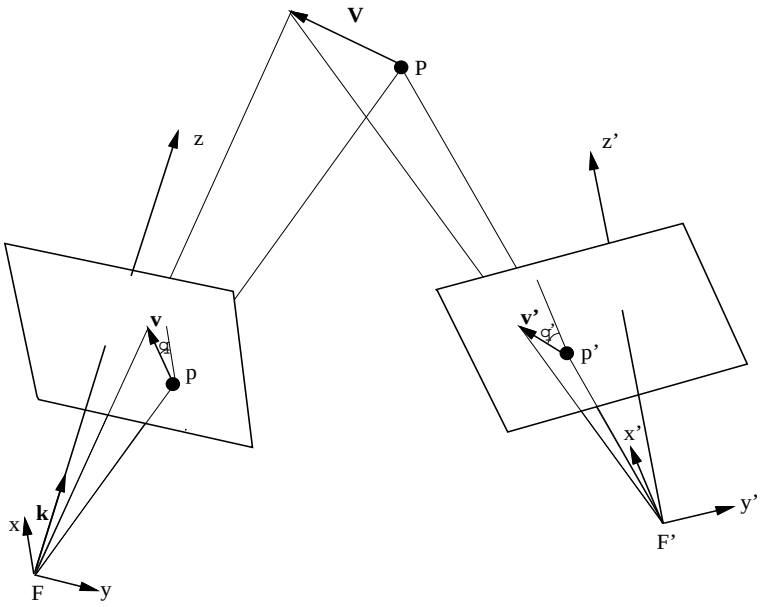


Figure 6.6. *La géométrie de la projection d'un vecteur dans les images.*

De plus on peut exprimer x' et y' en fonction de x et y grâce aux equations [5.42] et [5.43]. On peut ainsi obtenir une relation entre tan θ' et tan θ. Dans le cas général cette relation est inexploitable. Si les plans images et axes optiques sont parallèles (cas des images rectifiées) on a :

$$\begin{aligned} x' &= x \\ y' &= y + b/Z \\ V'_x &= V_x \\ V'_y &= V_y \\ V'_z &= V_z \end{aligned}$$

On obtient :

$$\tan\theta' = \frac{\tan\theta}{1 - \frac{b}{Z}\frac{V_z}{V_y - yV_z}}$$

ou plus simplement :

$$\tan\theta' = \frac{v_x}{v_y - bV_z}$$

On peut remarquer que si le vecteur $\mathbf{V}$ est parallèle au plan image, soit pour $V_z = 0$ on obtient :

$$\theta' = \theta$$

Dans ce cas la recherche d'un appariement gauche/droite est triviale. En effet, seules les primitives ayant la même inclinaison dans les deux images peuvent être appariées.

On peut également remarquer que plus le vecteur $\mathbf{V}$ sera incliné par rapport au plan des images (plus V_z est grand) et plus il y aura de différence entre θ et θ'.

Dans le cas géneral, on veut trouver une constante ε qui sera la borne de la différence des deux orientations :

$$\mid \theta - \theta' \mid < \varepsilon$$

Imposer une limite à la différence d'inclinaison entre les vecteurs se correspondant dans les deux images revient à rejeter les appariements qui correspondent à des vecteurs $\mathbf{V}$ fortement inclinés par rapport au plan des images. Cette limitation de la différence d'orientation dans les deux images est cohérente avec la limite du gradient de la disparité qui interdit également des surfaces trop inclinées par rapport au plan des images.

La valeur de la constante ε peut se calculer en fonction de la distribution des inclinaisons spatiales de $\mathbf{V}$ et pour une valeur de b (distance entre les centres de projection) donnée [4]. Choisissons une primitive de l'image de gauche. Plus ε sera grand, plus il y aura des appariements possibles entre cette primitive et des primitives de l'image de droite. Plus il y aura d'appariements gauche/droite ainsi formés plus la recherche d'une mise en correspondance globale sera complexe.

6.4.3. La contrainte d'ordre

Les contraintes épipolaires et d'orientation nous ont permis de réduire le nombre d'appariements entre des primitives de l'image de gauche et des primitives de l'image de droite. Soient maintenant deux appariements établis sur la base de ces contraintes : (g_1, d_1) et (g_2, d_2). Dans quelle mesure ces appariements sont compatibles entre eux ? La première contrainte qu'on va étudier est la contrainte d'ordre.

Nous allons nous intéresser à des primitives points. Soit P_1 un point et soient p_1 et p'_1 ses projections dans les deux images. Les directions de projection de ce point partitionnent le plan épipolaire en deux régions, une région (hachurée) intérieure et une région extérieure, figure 6.7. Soient P_2 et P_3 deux points de l'espace et soient p_2, p_3 et p'_2, p'_3 leurs projections respectives dans l'image de gauche et de droite. Le point P_2 se trouve dans la région intérieure alors que le point P_3 se trouve dans la région extérieure. Regardons l'ordre de projection de ces points dans les deux images.

- dans l'image de gauche l'ordre est p_3, p_1, p_2.
- dans l'image de droite l'ordre est p'_3, p'_2, p'_1.

Si on prend P_1 comme référence, les projections de P_3 apparaissent dans le même ordre dans les deux images alors que l'ordre de projection de P_2 est inversé.

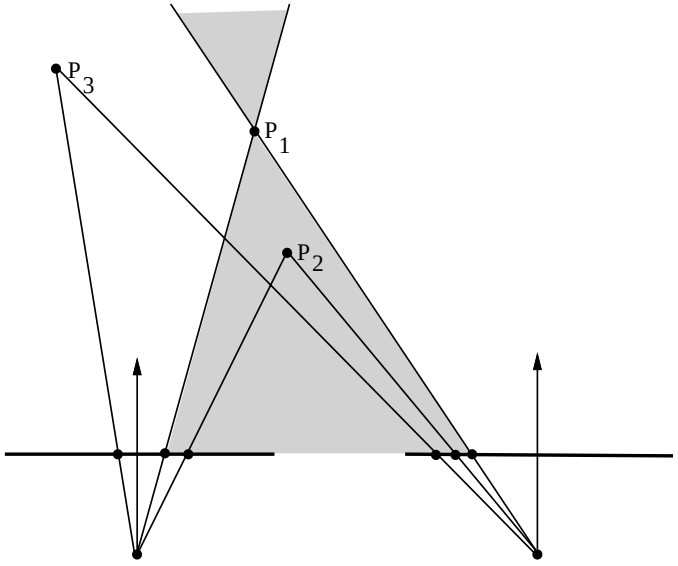


Figure 6.7. *P_1 et P_3 se projettent dans le même ordre dans les deux images alors que l'ordre des projections de P_1 et P_2 est inversé.*

Supposons maintenant que P_1 et P_3 se trouvent sur la surface d'un objet. Le fait que P_3 se trouve dans la région extérieure de P_1 correspond au fait que l'objet est relativement peu incliné par rapport au plan des images (il s'agit de

l'objet 1 de la figure 6.8.). Soit maintenant l'objet 2 de la même figure sur lequel se trouvent les points P_1 et P_2. Le fait que le point P_2 se trouve dans la région intérieure de P_1 correspond au fait que l'objet 2 est très incliné par rapport au plan des images. Si cet objet est opaque les projections des deux points ne peuvent pas être vues à la fois dans les deux images. Si l'objet est transparent, les projections seront vues dans les deux images mais dans un ordre inversé.

En conclusion, la seule configuration pour laquelle l'ordre des projections est inversé est le cas de *surfaces transparentes fortement inclinées* par rapport au plan des images. Si on suppose donc que la scène observée contient peu (ou pas) de surfaces transparentes fortement inclinées, l'ordre des projections est le même dans les deux images.

Si on revient à l'exemple précédent et si on respecte la contrainte d'ordre on aura : l'appariement (p_1, p'_1) est compatible avec (p_3, p'_3) et *incompatible* avec (p_2, p'_2).

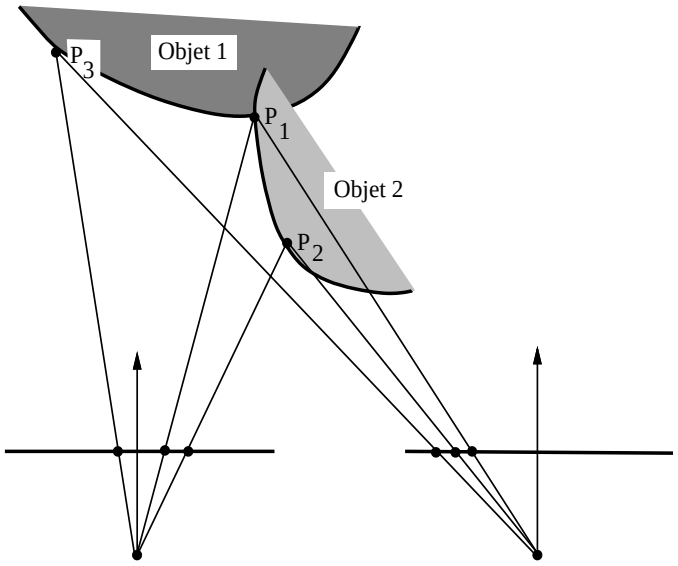


Figure 6.8. *Lorsque deux points se trouvent sur la surface d'un objet peu incliné (1) l'ordre de leurs projections est le même dans les deux images. Si l'objet est fortement incliné (2) les projections ne sont pas vues simultanément dans les deux images si l'objet est opaque ou l'ordre est inversé si l'objet est transparent.*

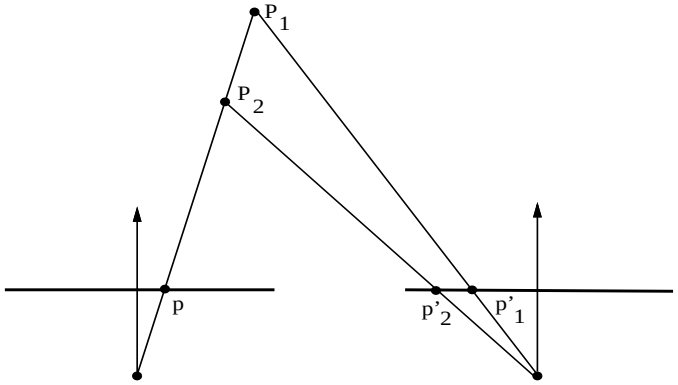


Figure 6.9. *La contrainte d'unicité n'est pas respectée lorsque deux points se projettent en un même point dans une image et en deux points différents dans l'autre image.*

La contrainte d'ordre a été utilisée pour la première fois par Baker et Binford [13] pour simplifier la combinatoire de la mise en correspondance. Une caractérisation analytique de cette contrainte a été proposée par Yuille et Poggio [182]. Cette caractérisation consiste essentiellement à interdire la compatibilité entre deux appariements qui placerait un point dans la région intérieure de l'autre point. Nous verrons plus loin que la contrainte d'ordre peut être exprimée dans le contexte plus général de la limite du gradient de disparité.

6.4.4. La contrainte d'unicité

Considérons maintenant un cas limite de la contrainte d'ordre. Ce cas est illustré par la figure 6.9. Le point P_2 se trouve sur la frontière des régions extérieure et intérieure du point P_1. Les deux points se projettent en un même point dans l'image de gauche et en deux points différents dans l'image de droite. Si on désigne par p la projection commune de P_1 et de P_2 dans l'image de gauche et par p'_1 et p'_2 leurs projections dans l'image de droite on doit avoir en même temps les appariements (p, p'_1) ET (p, p'_2). Accepter que ces deux appariements soient compatibles revient à augmenter la difficulté de la mise en correspondance car, à l'exception de l'alignement décrit plus haut, un point

d'une image devrait avoir un et un seul correspondant dans l'autre image.

En pratique nous verrons au paragraphe suivant que la contrainte d'unicité peut, comme la contrainte d'ordre, être exprimée analytiquement dans le cadre de la limite du gradient de disparité.

6.4.5. La limite du gradient de disparité

Nous avons vu que la contrainte d'ordre est directement associée à l'inclinaison d'une surface observée par rapport au plan des deux images. On peut exprimer cette contrainte d'une autre manière. Reprenons pour cela la formule qui permet de calculer la profondeur d'un point en fonction de la disparité :

$$Z = \frac{b}{D} = \frac{b}{y' - y}$$

Soient deux points sur la surface d'un objet, P et Q et soient Z_P et Z_Q leurs profondeurs. Une autre façon d'exprimer le fait que la surface est peu inclinée est d'imposer que la différence entre ces deux profondeurs est faible. La relation avec la contrainte d'ordre est triviale. Les projections des points se trouvant sur un objet peut incliné (objet 1 de la figure 6.8.) ont le même ordre dans les deux images alors que les projections des points se trouvant sur un objet fortement incliné sont inversées dans les deux images.

La profondeur d'un point est associée à la disparité (la différence entre les positions des deux projections dans les deux images). La variation de la profondeur est donc associée à la variation de la disparité. Cependant la relation entre la variation de la disparité et la variation de la profondeur n'est pas facile à exploiter en pratique car la profondeur est inversement proportionnelle à la disparité. Burt et Julesz [31] ont défini le *gradient de disparité* et Pollard et al. [152], [153] ont établi une caractérisation analytique intéressante grâce à la limite du gradient de disparité.

Soient p et p' les projections du point P et q et q' les projections du point Q, P et Q ne se situant pas nécessairement dans le même plan épipolaire, figure 6.10. Le gradient de disparité est défini comme la différence des disparités de deux couples de points image divisée par la *séparation cyclopéenne*. Soit un repère commun aux deux images et soient les couples de points (p, p') et (q, q'). $\mathbf{p}$, $\mathbf{p}'$, $\mathbf{q}$ et $\mathbf{q}'$ sont les vecteurs associés à ces points dans le repère commun. Le

gradient de disparité est défini par l'expression suivante :

$$G_d = \frac{\|(\mathbf{p}' - \mathbf{p}) - (\mathbf{q}' - \mathbf{q})\|}{\|\frac{1}{2}(\mathbf{p}' + \mathbf{p}) - \frac{1}{2}(\mathbf{q}' + \mathbf{q})\|} \quad [6.15]$$

Le gradient de disparité peut s'écrire sous la forme suivante :

$$\frac{G_d}{2} = \frac{\|(\mathbf{p}' - \mathbf{q}') - (\mathbf{p} - \mathbf{q})\|}{\|(\mathbf{p}' - \mathbf{q}') + (\mathbf{p} - \mathbf{q})\|}$$

Le gradient de disparité est relié à la pente d'une droite passant par P et Q. La pente de cette droite rend finalement compte de l'inclinaison de la surface autour de ces points. Interdire des appariements gauche/droite qui correspondent à des surfaces fortement inclinées revient à imposer une limite supérieure au gradient de disparité. Si on choisit :

$$0 < \frac{G_d}{2} \leq k \text{ avec } 0 < k < 1 \quad [6.16]$$

on obtient l'inégalité suivante :

$$\|(\mathbf{p}' - \mathbf{q}') - (\mathbf{p} - \mathbf{q})\| < k\|(\mathbf{p}' - \mathbf{q}') + (\mathbf{p} - \mathbf{q})\|$$

Utilisant les propriétés classiques des inégalités :

$$\begin{aligned} \|a\| - \|b\| &\leq \|a - b\| \\ \|a + b\| &\leq \|a\| + \|b\| \end{aligned}$$

on obtient $(0 < k < 1)$:

$$\frac{\|\mathbf{p}' - \mathbf{q}'\|}{\|\mathbf{p} - \mathbf{q}\|} < \frac{1 + k}{1 - k} = \frac{1}{\lambda} \quad [6.17]$$

Pour un point P fixé et pour une valeur de k on va chercher le lieu des points Q satisfaisant la contrainte exprimée par l'équation [6.17]. On cherche d'abord le lieu des points Q satisfaisant l'égalité :

$$\frac{\|\mathbf{p}' - \mathbf{q}'\|}{\|\mathbf{p} - \mathbf{q}\|} = \frac{1}{\lambda} \quad [6.18]$$

Soit le repère commun aux deux images, le repère de l'image de gauche. Les points p et p' ont comme coordonnées : $\mathbf{p} = (x_1\ y_1)^t$ et $\mathbf{p}' = (x_1\ y_1')^t$. De

même on a : $\mathbf{q} = (x_2\ y_2)^t$ et $\mathbf{p}' = (x_2\ y_2')^t$ Les coordonnées de q et q' sont reliées aux coordonnées de $Q = (X\ Y\ Z)^t$ par les formules suivantes :

$$\begin{aligned} x_2 &= X/Z \\ y_2 &= Y/Z \\ y_2' &= y_2 + b/Z \end{aligned}$$

L'égalité précédente, soit l'équation [6.18] devient alors :

$$\frac{(x_1 Z - X)^2 + (y_1 Z - Y)^2}{(x_1 Z - X)^2 + (y_1' Z - (b + Y))^2} = \lambda^2$$

Le lieu des points Q est finalement un cône dont le sommet se trouve en P, figure 6.10. :

$$(1-\lambda^2)(X^2+Y^2) + (x_1^2 + y_1^2 - \lambda^2(x_1^2 + y_1'^2))Z^2 + 2(\lambda^2 - 1)x_1 XZ + 2(\lambda^2 y_1' - y_1)YZ - 2\lambda^2 bY + 2\lambda^2 y_1' bZ - \lambda^2 b^2 = 0 \qquad [6.19]$$

Revenant maintenant au lieu des points Q satisfaisant l'inégalité, soit l'équation [6.17], on peut voir facilement que ces points doivent se trouver à l'extérieur du cône. L'intérieur du cône est la zone *interdite* soit la région qui correspond à des surfaces trop inclinées par rapport au plan des images.

Pour conclure, on dira que deux appariements gauche/droite (p, p') et (q, q') sont compatibles si la valeur du gradient de disparité qui leur est associée est inférieure à 2. On peut vérifier que cette limite garantit également la contrainte d'ordre et la contrainte d'unicité.

6.4.6. La contrainte de continuité figurale

La contrainte de continuité figurale est une contrainte simple et efficace lorsque les points à mettre en correspondance se trouvent le long de contours.

La figure 6.11. montre deux images et des points le long de contours appartenant à ces images. Les lignes épipolaires sont également représentées sur cette figure. Les points appartenant à un contour de l'image de gauche ont été mis en correspondance avec des points de l'image de droite n'appartenant pas nécessairement au même contour.

Ce type d'erreur peut être facilement corrigé en imposant aux points de l'image de droite d'appartenir au contour possédant déjà une majorité de points

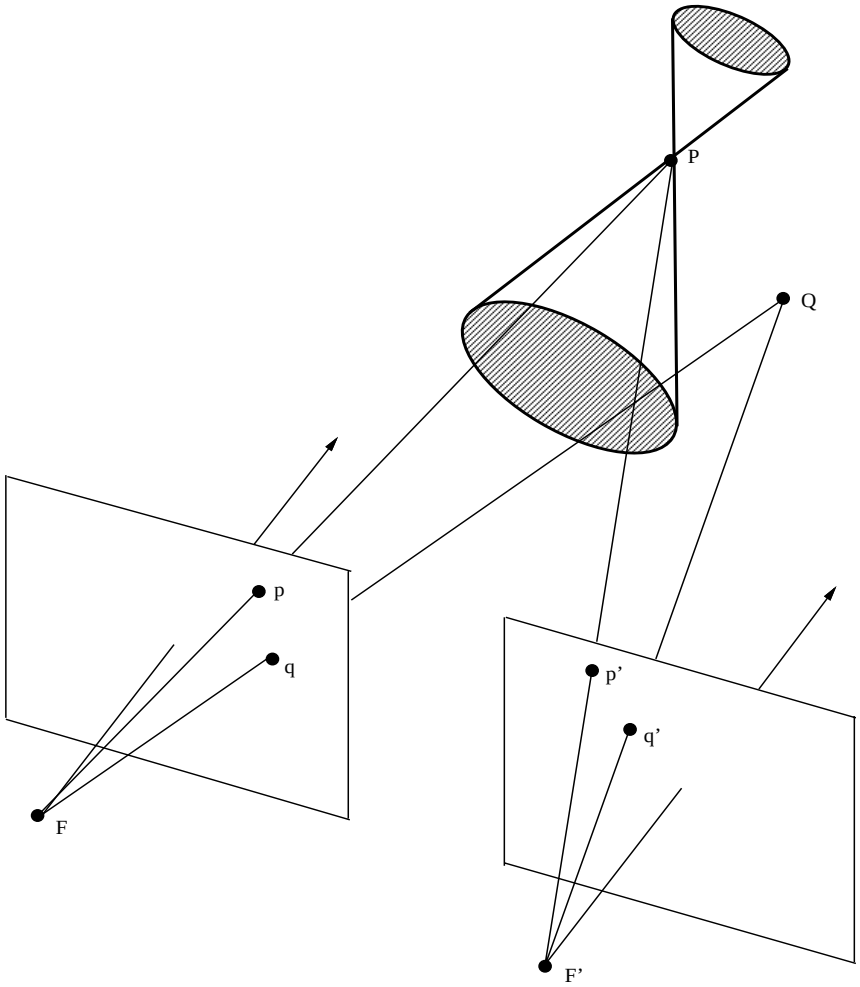


Figure 6.10. *Le lieu des points Q, tels que le gradient de disparité des appariements (p, p') et (q, q') est inférieur à 2, est la région extérieure à un cône dont le sommet se trouve en P.*

qui ont été mis en correspondance avec les points d'un contour de l'image de gauche.

Cependant Mohan, Medioni et Nevatia [125] remarquent à juste titre que ce genre de raisonnement *local* peut être source d'erreur. En effet, considérons la figure 6.12. Dans cette figure il y a une majorité de points appartenant à un contour de l'image de droite qui n'est pas le bon contour. La contrainte de continuité figurale doit donc être utilisée dans un contexte global et non pas local.

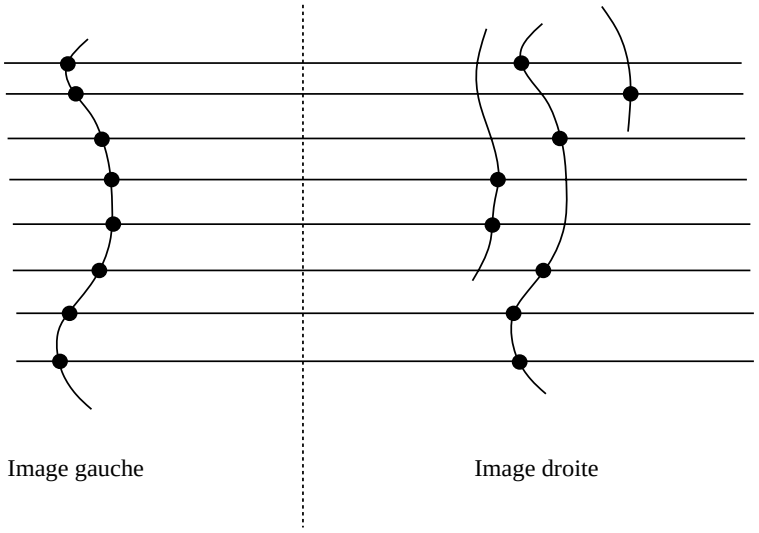


Figure 6.11. *L'erreur de mise en correspondance illustrée dans cette figure peut être corrigée en tenant compte de la continuité figurale.*

6.5. Mise en correspondance stéréo

Comme on peut s'en douter il n'existe pas actuellement une méthode de mise en correspondance suffisamment générale pour qu'elle puisse s'appliquer à une paire d'images stéréo indépendamment du contenu et du type de ces images. Des domaines d'application aussi variés que :

- reconstruction de terrain à partir d'images aériennes ;
- modélisation de scènes d'intérieur ;

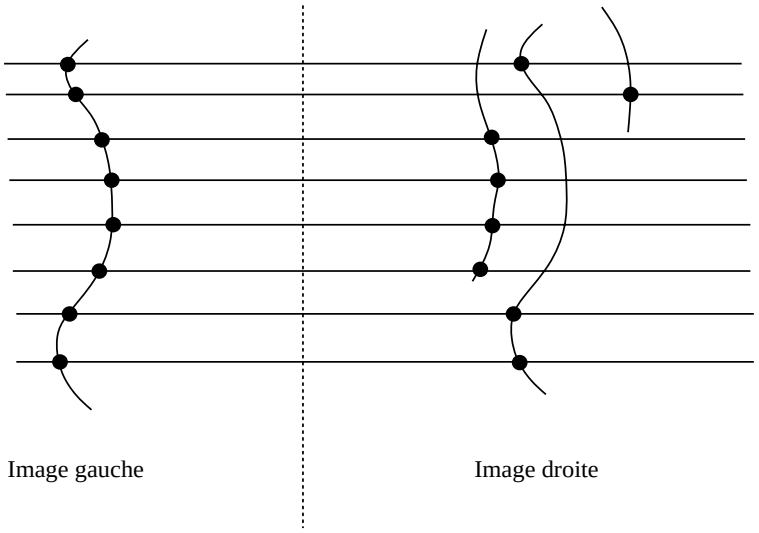


Figure 6.12. *La correction des erreurs de mise en correspondance selon un raisonnement local conduirait dans ce cas à un mauvais résultat.*

- modélisation de scènes d'extérieur ;
- images médicales et biomédicales ;

manipulent des images très différentes du point de vue de leur contenu.

Parmi le large spectre de méthodes proposées nous en avons sélectionné quatre que nous allons présenter en détail. Ces quatre méthodes sont les suivantes :

- *mise en correspondance hiérarchique.* Les images sont représentées à plusieurs niveaux de résolution, du niveau le plus "fin" au niveau le plus "grossier". Le résultat de la mise en correspondance au niveau le plus grossier est utilisé pour initialiser la mise en correspondance au niveau le plus fin.

- *mise en correspondance par programmation dynamique.* Cette méthode exploite trois contraintes parmi celles que nous avons développées : la contrainte épipolaire, la contrainte d'unicité et la contrainte d'ordre. Les points se trouvant le long de deux droites en correspondance épipolaire (qu'on appelle également deux droites épipolaires conjuguées) sont appariés d'une façon unique par programmation dynamique.

• *mise en correspondance par relaxation.* La contrainte épipolaire combinée avec une mesure de ressemblance permettent d'établir un certain nombre d'appariements. Un algorithme de relaxation permet ensuite de sélectionner, parmi cet ensemble d'appariements initiaux, un sous-ensemble d'appariements compatibles au sens des contraintes d'unicité et de limite du gradient de disparité.

• *mise en correspondance par isomorphisme de graphes.* Chaque image est vue comme un graphe. Le problème d'appariement se ramène alors à un problème d'appariement de deux graphes.

6.5.1. Mise en correspondance hiérarchique

Nous allons décrire une méthode qui a été développée à SRI International pour la reconstruction de terrain à partir de 2 vues aériennes [78], [79]. Les images utilisées ont une taille de 2048×2048 pixels et chaque image représente une surface au sol d'environ 2 kilomètres carrés. La hauteur du terrain varie entre 360 et 540 mètres. Il s'agit donc d'un terrain relativement plat.

A partir d'une paire d'images on produit une *hiérarchie* d'images. Une partie de cette hiérarchie est montrée sur la figure 6.13..

Cette hiérarchie consiste en deux *pyramides d'images*, une pyramide pour chaque image initiale [39]. Chaque pyramide contient l'image initiale ainsi que des images de taille 1024×1024, 512×512, 256×256, 128×128, 64×64, 32×32 et 16×16. Chaque pyramide a donc 8 niveaux, du niveau le plus fin (2048×2048) ou haute résolution au niveau le plus grossier (16×16) ou basse résolution. La construction de cette pyramide est simple : on réduit une image en moyennant la valeur des pixels dans un carré $N \times N$. Ainsi pour $N = 2$, un carré 2×2 d'une image est remplacé par un seul pixel dans l'image de résolution inférieure, et ainsi de suite. On peut remplacer ce moyennage par des techniques plus sophistiquées qui convoluent l'image avec des filtres gaussiens et en sous-échantillonnant pour réduire en même temps la taille de l'image (voir à ce sujet le chapitre 2).

La première étape de toute méthode de mise en correspondance est d'extraire de chaque image (gauche et droite) les points qui seront les candidats à l'appariement. La méthode, développée à SRI, extrait des points d'intérêt de l'image de gauche uniquement. Pour chacun de ces points on cherche un

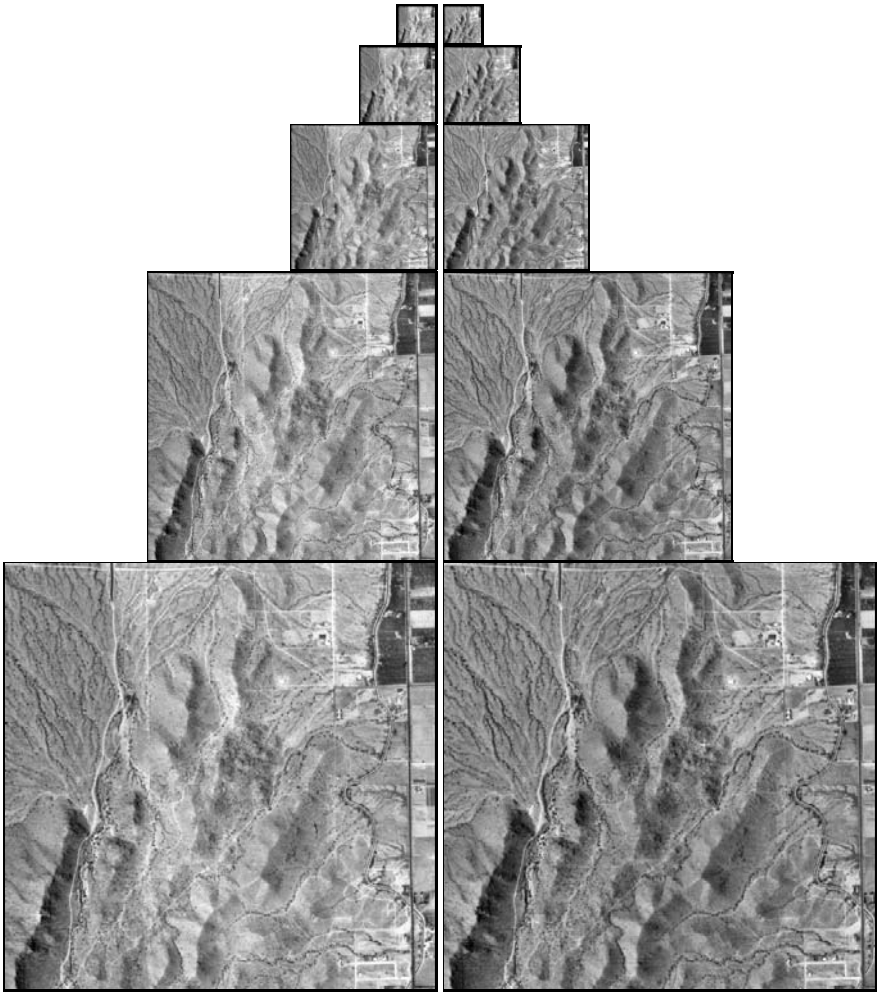


Figure 6.13. *Deux pyramides formées à partir d'une paire d'images à mettre en correspondance. Les images de cette figure sont de taille 512×512, 256×256, 128×128, 64×64 et 32×32.*

correspondant dans l'image de droite de la façon suivante (voir également la figure 6.14.)

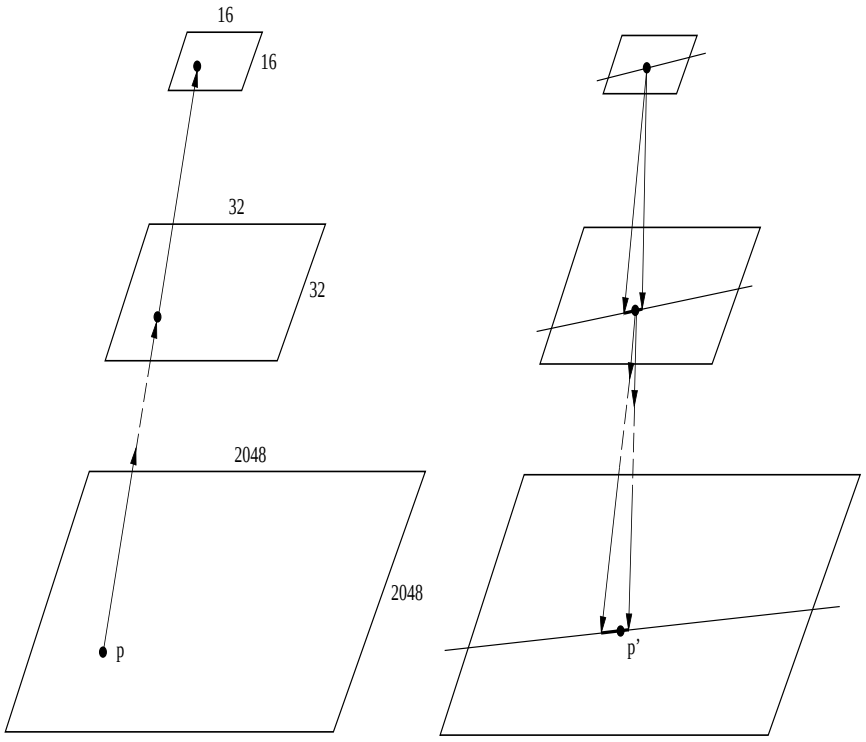


Figure 6.14. *L'algorithme de mise en correspondance hiérarchique développé à SRI utilise une représentation multi-résolution (pyramidale) des images, ce qui facilite la recherche d'appariements gauche/droite même lorsque la contrainte épipolaire est peu ou mal connue.*

Soit un point p de l'image de gauche haute résolution (2048×2048). Pour chaque image de la pyramide de gauche on calcule successivement les coordonnées de ce point, jusqu'à l'image de basse résolution (16×16). En pratique cela revient à diviser successivement par 2 les coordonnées du point p. On s'intéresse ensuite à la recherche d'une correspondance au niveau de résolution le plus faible. Pour cela on utilise la contrainte épipolaire et on cherche le long de la droite épipolaire le point de l'image de droite le plus ressemblant au point de l'image de gauche. On utilise le produit de corrélation comme mesure de

ressemblance, équation [6.2]. Les images étant de taille réduite, une recherche du point de corrélation maximum le long de la ligne épipolaire n'est pas très coûteuse.

La prochaine étape consiste à parcourir la pyramide droite de l'image 16×16 vers l'image 2048×2048 de façon à combiner la contrainte épipolaire avec la prediction fournie par l'image basse résolution. Le point selectionné dans l'image 16×16 correspond à une région (de taille 2×2 dans ce cas-ci) dans l'image 32×32. L'intersection de cette région avec la contrainte épipolaire constitue la zone d'initialisation pour la recherche du point le plus ressemblant (au sens du produit de corrélation). Cette stratégie est poursuivie jusqu'à atteindre l'image 2048×2048.

On peut remarquer que cette méthode peut rester efficace lorsque la contrainte épipolaire est peu ou mal connue. En effet cela entraîne une recherche quasi exhaustive du meilleur correspondant dans l'image 16×16 ce qui n'est pas très coûteux. On peut donc utiliser cette méthode pour trouver des appariements afin de déterminer la géométrie épipolaire conformément à la section 6.3.3. de ce chapitre.

6.5.2. Appariement par programmation dynamique

La programmation dynamique est une technique utile pour apparier deux séquences tout en respectant l'ordre des éléments à l'intérieur de chaque séquence. Soient par exemple les séquences $\{a_1\ a_2\ a_3\ a_4\}$ et $\{b_1\ b_2\ b_3\ b_4\}$. Si l'élément a_2 est apparié avec b_2, les éléments à droite de a_2 ne pourront être appariés qu'avec les éléments à droite de b_2.

L'algorithme de Viterbi est une méthode de programmation dynamique qui trouve le meilleur appariement entre deux séquences parmi tous les appariements possibles. Les éléments des séquences à apparier définissent les deux dimensions d'une matrice. Chaque élément de cette matrice représente donc le coût d'appariement de deux éléments, soit une paire d'éléments. Un *chemin* consistera en une séquence de telles paires, chaque paire correspondant à un élément de la matrice. L'objectif sera de trouver un chemin optimal dans cette matrice. Le coût d'un chemin est la somme des coûts de ses éléments. Afin de trouver le chemin optimal (de coût maximal) deux contraintes sont utilisées : (i) l'ordre doit être respecté et (ii) chaque élément d'une séquence n'intervient qu'une seule fois.

Dans le cas de la stéréo, les deux séquences correspondent à des points le long de deux droites épipolaires conjuguées. Le coût de mise en correspondance de deux points peut être calculé à partir d'une mesure de ressemblance telle que celle donnée par l'équation [6.2]. Il est possible qu'un point d'une image n'ait pas de correspondant dans l'autre image. Dans ce cas le coût associé à une paire contenant un tel point est très faible voire nul. Le chemin optimal est une séquence d'éléments de la matrice. Un chemin est formé de transitions verticales (un point sans correspondant), horizontales (idem) ou diagonales (un appariement).

L'algorithme de Viterbi n'est pas nouveau. Il a été proposé pour la première fois pour comparer deux signaux en reconnaissance de la parole. Pour la mise en correspondance stéréo il a été utilisé par Baker et Binford [13] et par Ohta et Kanade [143] pour ne citer que les travaux les plus connus. Outre la contrainte épipolaire, la contrainte d'ordre et la contrainte d'unicité, on peut prendre également en compte la contrainte de continuité figurale [143].

6.5.3. Appariement par relaxation

Le problème de mise en correspondance stéréo peut être vu comme un problème d'étiquetage : aux points d'une image on associe les points d'une autre image. Le problème étant combinatoire, on est amené à faire "coopérer" les points d'une image entre eux en mettant en œuvre des relations découlant des contraintes géométriques et/ou physiques. Cette coopération peut être réalisée dans le cadre d'un processus de relaxation qui peut être décrit de la façon suivante (voir [40] pour une introduction à la relaxation) :

- un ensemble d'étiquettes (appariements) est sélectionné pour chaque point. Une mesure de confiance est ensuite associée à chaque étiquette ;
- les différentes étiquettes (avec leurs confiances) possibles pour chaque point sont comparées avec celles des points voisins. Cette comparaison est faite sur la base des relations entre les points voisins. Les étiquettes ainsi que leurs confiances sont ensuite modifiées itérativement pour réduire les inconsistances d'étiquetage entre points voisins.

De nombreux travaux ont proposé la relaxation comme méthodologie de mise en correspondance stéréo. Un algorithme "générique" est proposé dans l'ouvrage de Ballard et Brown [15]. On peut également se référer à la thèse de

Pascale Long-Limozin [112] qui utilise la relaxation pour mettre en correspondance des segments de droite.

Nous décrivons ici l'algorithme PMF (Pollard-Mayhew-Frisby) mis au point à l'Université de Sheffield et développé dans le cadre de la thèse de Stephen Pollard [151]. Cette méthode utilise la limite du gradient de disparité et la contrainte d'unicité. PMF procède en deux étapes : (i) une initialisation et (ii) une phase itérative. Pendant l'initialisation on calcule la confiance associée à chaque appariement. Cette confiance est calculée sur la base du "support" que cet appariement reçoit plusieurs autres appariements voisins. Le processus itératif de la deuxième phase utilise la contrainte d'unicité pour finalement classer les appariements initiaux en *corrects* et *incorrects*.

6.5.3.1. La confiance d'un appariement

Soit p un point de l'image de gauche et p′ son correspondant dans l'image de droite. Bien évidemment le couple p/p′ est compatible avec la contrainte épipolaire. La confiance d'un appariement peut être définie par l'expression suivante (voir également la figure 6.15.) :

$$S(p/p') = \sum_{i \in N(p)} \left(\max_{j'} \left(\frac{C(i,j')G(p/p', i/j')}{d(p,i)} \right) \right) \qquad [6.20]$$

Dans cette expression :

- $N(p)$ est un voisinage du point p dans l'image de gauche et i est un point de ce voisinage qui a au moins un correspondant j' dans l'image de droite ;
- $C(i,j')$ est le produit de corrélation défini par l'équation [6.2] et détermine la "qualité" de l'appariement i/j' ou la ressemblance entre ces deux points ;
- $d(p,i)$ est une fonction proportionnelle à la distance entre les points p et i ;
- G est le support dû au gradient de disparité associé aux appariements p/p' et i/j' et se calcule grâce à l'expression suivante (voir la section 6.4.5.) :

$$G(p/p', i/j') = \begin{cases} 1 & \text{si} \quad G_d \leq 2k \\ 0 & \text{si} \quad G_d > 2k \end{cases}$$

où G_d et k sont définis par les équations [6.15] et [6.16].

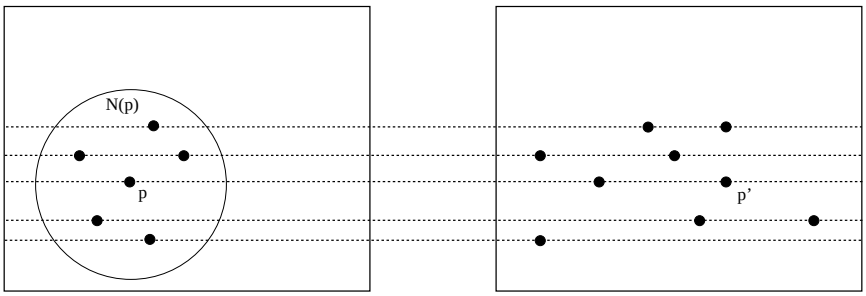


Figure 6.15. *La confiance d'un appariement p/p' peut se calculer à partir d'un voisinage $N(p)$ du point p et des appariements des points appartenant à ce voisinage. Les confiances respectives de p/p' (gauche-droite) et p'/p (droite-gauche) sont différentes dans le cas général car la structure locale des deux images n'est pas la même.*

On peut remarquer que l'expression définissant la confiance d'un appariement n'est pas symétrique, autrement dit dans le cas général on a :

$$S(p/p') \neq S(p'/p)$$

En effet, la configuration du voisinage de p dans l'image de gauche n'est pas la même que la configuration du voisinage de p' dans l'image de droite – surtout lorsque l'appariement p/p' est incorrect.

6.5.3.2. Appariement itératif

Les résultats expérimentaux obtenus par Stephen Pollard montrent que les confiances associées aux appariements corrects sont en général beaucoup plus élevées que celles associées aux appariements incorrects. Par conséquent, l'hypothèse faite dans PMF est que les appariements ayant une confiance maximale sont ceux qui sont physiquement corrects.

A chaque itération de l'algorithme PMF les appariements ayant la plus grande confiance pour les deux points les constituant – $S(p/p')$ et $S(p'/p)$ – sont choisis comme corrects. Ensuite, en accord avec la contrainte d'unicité, tous les autres appariements associés avec l'un ou l'autre des points des appariements choisis sont éliminés. Ceci permet à d'autres appariements qui on été ni acceptés ni été éliminés d'être choisis comme corrects parce qu'ils ont

maintenant la plus grande confiance pour leurs deux points les constituant. La convergence – lorsqu'il n'y a plus que des appariements corrects ou éliminés – a lieu au bout de 4 à 5 itérations. En pratique Stephen Pollard a observé que 73% des appariements sont choisis lors de la première itération et 11% lors des 3 itérations suivantes. Les points non appariés n'ont pas, en général, de correspondant.

L'algorithme PMF peut être vu comme une forme de relaxation car à l'initialisation chaque point de l'image de gauche a un certain nombre d'étiquettes (appariements) possibles et à chaque étiquette est associée une confiance. Au cours des itérations de l'algorithme les confiances évoluent et à la convergence chaque point n'a plus que deux étiquettes possibles : apparié ou éliminé.

Plus formellement, chaque pas d'itération de cette méthode de mise en correspondance stéréo se décompose en les étapes suivantes :

Etape 1 : Pour chaque point de l'image de gauche on choisit un appariement dans l'image de droite qui maximise son support :

$$\max_{p'} S(p/p')$$

On obtient ainsi un certain nombre d'appariements gauche-droite ;

Etape 2 : Pour chaque point de l'image de droite on choisit un appariement dans l'image de gauche qui maximise son support :

$$\max_{p} S(p'/p)$$

On obtient ainsi un certain nombre d'appariements droite-gauche ;

Etape 3 : On compare les appariements gauche-droite avec les appariements droite-gauche et on conserve ceux qui sont identiques.

Etape 4 : Les points qui viennent d'être appariés ne peuvent plus participer à d'autres appariements (en vertu de la contrainte d'unicité) et il faut donc éliminer de la liste initiale d'appariements ceux qui contiennent les points qui viennent d'être appariés. Ceci aura pour effet de modifier le support des appariements restants.

Il y a donc, au cours de l'algorithme trois types d'appariements : retenus, éliminés et non traités. L'algorithme s'arrête dès qu'il n'y a plus d'appariements non traités.

6.5.3.3. Prise en compte de la continuité figurale

Il est possible d'incorporer la contrainte de continuité figurale dans l'algorithme PMF. En effet, après chaque itération on considère les points appariés de l'image de gauche appartenant à un même contour. Leurs correspondants doivent également en principe appartenir au même contour de l'image de droite. Si ce n'est pas le cas, on sélectionne le contour qui contient la majorité de ces points et on élimine les quelques appariements qui ne satisfont pas cette contrainte – voir figure 6.11.

On peut remarquer que ce processus a un caractère moins local que le processus de relaxation puisqu'un contour peut traverser toute l'image.

6.5.4. Appariement par isomorphisme de graphes

Le problème de mise en correspondance stéréo peut être vu comme un problème d'isomorphisme de graphes. En effet l'ensemble de primitives extraites d'une image peut constituerl'ensemble des nœuds d'un graphe. Des relations entre ces primitives constitueront alors les arcs (ou arêtes) de ce graphe. C'est l'approche qui a été suivie dans la thèse de Thomas Skordas [166], [96]. Dans le cadre de cette méthode les primitives à mettre en correspondance sont des segments de droite et les relations entre ces segments sont : (i) à gauche de, (ii) à droite de, (iii) colinéaire avec et (iv) a une jonction commune avec. Chaque image est donc décrite par un réseau de segments de droite et de relations entre ces segments, soit un graphe relationnel. Le problème de mise en correspondance est donc le problème de trouver le meilleur appariement entre les nœuds et les arcs de ces deux graphes.

Cependant, pour un certain nombre de raisons ces deux graphes ne sont pas identiques :

- un segment vu dans une image peut être caché ou partiellement caché dans l'autre image ;
- un segment "entier" dans une image peut être coupé en plusieurs petits segments dans l'autre image ;
- deux segments peuvent former une jonction (ou être colinéaires) dans une image et non pas dans l'autre image ;

• la relation d'ordre peut parfois être violée : un segment se trouvant à gauche d'un autre segment dans une image peut se trouver à droite de ce même segment dans l'autre image.

En conclusion, le problème d'appariement des deux graphes n'est pas simplement un problème d'isomorphisme de graphes mais plutôt un problème d'isomorphisme de sous-graphes maximal. Ceci dit, le problème est de chercher tous les isomorphismes entre sous-graphes d'un graphe et sous-graphes de l'autre graphe et de sélectionner ensuite le plus grand isomorphisme – celui qui met en correspondance le plus grand nombre de nœuds.

L'approche suivie dans la méthode de Thomas Skordas pour chercher le meilleur appariement entre deux graphes consiste à construire un *graphe d'association* ou *graphe de correspondance*. Le graphe d'association est construit de la manière suivante. Les contraintes épipolaires, de position et d'orientation (voir sections 6.4.1. et 6.4.2.) sont tout d'abord utilisées pour permettre la construction d'un certain nombre d'appariements potentiels. Ces appariements segment/segment sont les nœuds du graphe d'association. Ensuite on construit un arc entre deux nœuds dès lors que les deux nœuds (ou les deux appariements sont compatibles). La notion de compatibilité entre deux appariements découle de l'utilisation des contraintes d'ordre, d'unicité et figurale (voir les sections 6.4.3., 6.4.4., 6.4.6.) ainsi que d'un certain nombre de propriétés projectives.

Dans ce contexte, la meilleure correspondance entre les graphes des deux images devient équivalente au plus grand ensemble de nœuds mutuellement compatibles dans le graphe d'association. Dans un graphe, un ensemble de nœuds mutuellement compatibles forme un sous-graphe complétement connecté, ou une *clique* – dans une clique chaque nœud est connecté aux autres nœuds de la clique. Une clique maximale est une clique qui ne peut être incluse dans une autre clique. Une clique maximale dans le graphe d'association correspond donc à un isomorphisme de sous-graphes. La plus grande clique maximale correspond au plus grand nombre d'appariements possible.

Les figures 6.16. et 6.17. montrent le résultat de mise en correspondance sur quelques segments de droite des images de la figure 6.1.. Pour plus de détails voir [166] et [96].

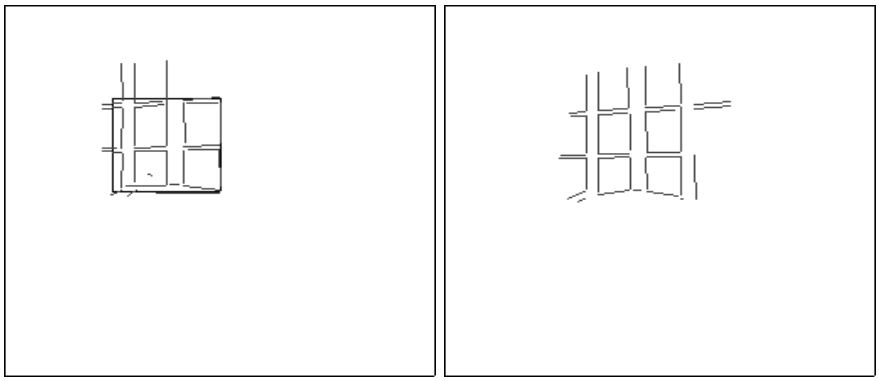


Figure 6.16. *Cette figure montre un ensemble de segments de l'image de gauche (segments se trouvant à l'intérieur d'un rectangle) ainsi que tous leurs correspondants potentiels dans l'image de droite.*

6.6. Systèmes stéréoscopiques particuliers

Les méthodes de mise en correspondance stéréoscopique que nous avons étudié comportent toutes une forme ou une autre de recherche des meilleurs appariements dans un ensemble fini. Malgré les contraintes mises en œuvre pour réduire au maximum l'espace de recherche, la complexité de ces méthodes reste prohibitive pour certaines applications où l'on ne dispose que d'une puissance de calcul limitée. Dans ce qui suit nous allons étudier deux cas particuliers de mise en correspondance stéréoscopique.

Dans le premier cas on utilise 3 caméras (et donc 3 images) au lieu de 2, ces 3 caméras étant "judicieusement" placées de façon que la recherche d'appariements devienne une simple vérification. La vision *trinoculaire* est couramment utilisée dans de nombreux projets de recherche et la présentation qui suit s'inspire des travaux de Nicholas Ayache et Francis Lustman [8], [118], [12].

Dans le deuxième cas on suppose que la scène observée est planaire et on peut alors décrire simplement la transformation qui permet de passer d'un point projeté dans une image à un point projeté dans l'autre image. Lorsqu'un objet est présent dans la scène (un obstacle posé sur un sol plat, par exemple) les projections des points de cet objet dans les deux images ne sont plus reliées par la transformation précédente. Encore une fois, une simple vérification permet de détecter des points de la scène qui n'appartiennent pas au plan de référence.

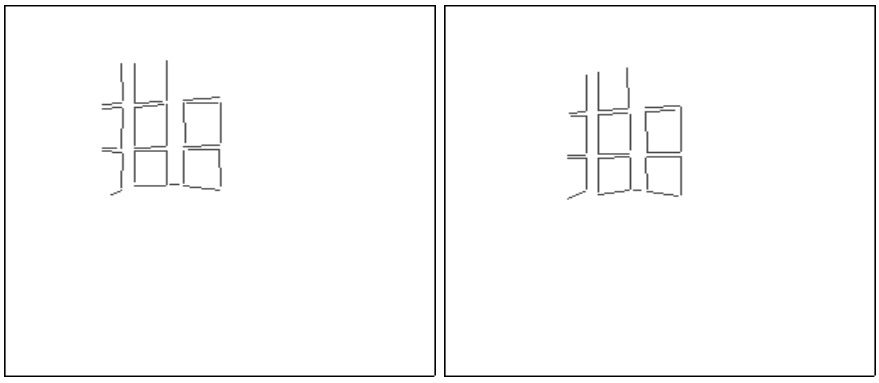


Figure 6.17. *Le résultat d'appariement obtenu sur les images de la figure précédente.*

6.6.1. Vision trinoculaire

La figure 6.18. illustre la géométrie épipolaire d'un capteur stéréoscopique avec 3 caméras. Un point P de la scène se projette en trois points p_1, p_2 et p_3 selon des droites passant par les centres de projection des caméras. Il y a 6 épipolaires associées à trois caméras et aux trois projections de P : l'épipolaire e_{ij} correspond à la projection de la droite Pp_i sur l'image j. Ces 6 épipolaires forment 3 paires d'épipolaires conjuguées : l'épipolaire e_{ij} est conjuguée avec l'épipolaire e_{ji}.

Lorsque p_1, p_2 et p_3 forment un appariement correct, le point p_i ($i \in \{1,2,3\}$) se trouve nécessairement à l'intersection des épipolaires e_{ki} et e_{ji}. Ceci nous conduit naturellement au processus d'appariement suivant : Soit un point q_1 de l'image 1 et un point q_2 de l'image 2. Si l'appariement q_1/q_2 est correct, alors q_3 (leur homologue dans l'image 3) se trouve précisément à l'intersection de deux épipolaires. On peut remarquer qu'un tel système est parfaitement symétrique et qu'on peut donc vérifier dans l'image k un appariement i/j.

6.6.2. Détection d'obstacles

Reprenons les équations [5.42] et [5.43]:

$$x' = \frac{Z\,\mathbf{r}_1 \cdot \mathbf{p} + b_x}{Z\,\mathbf{r}_3 \cdot \mathbf{p} + b_z} \qquad [6.21]$$

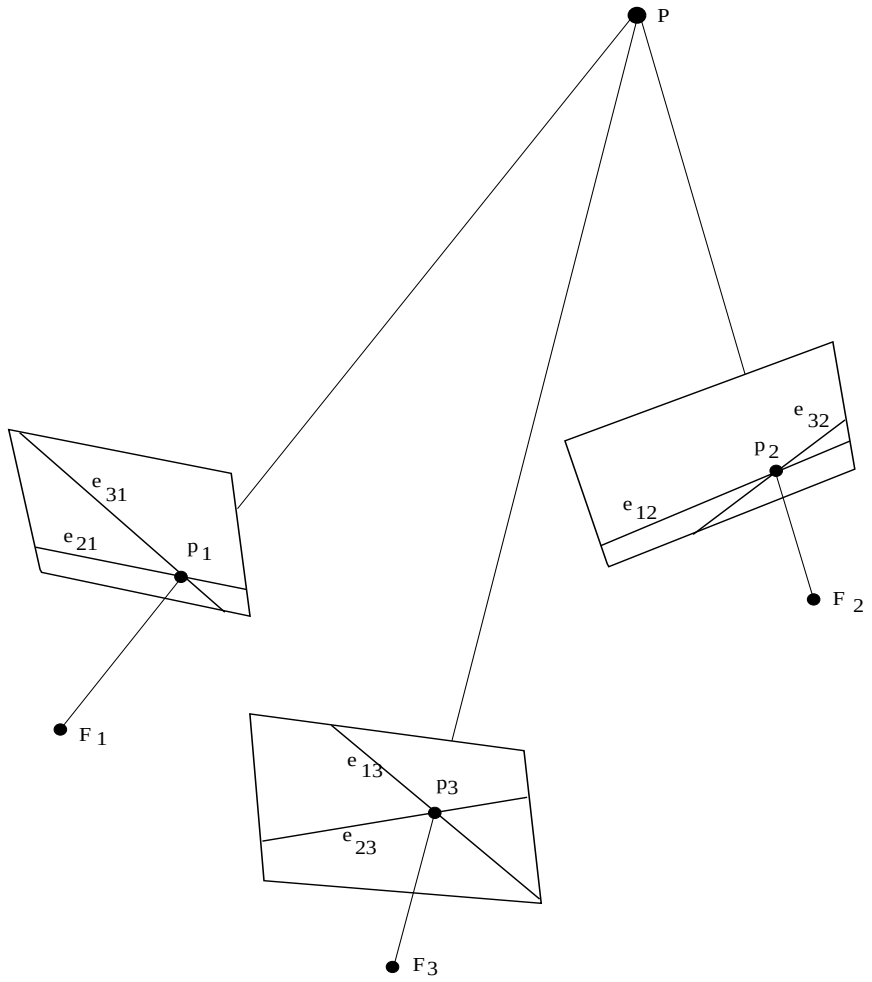


Figure 6.18. *Dans le cas d'un capteur stéréoscopique comportant 3 caméras on a 3 paires de droites épipolaires conjuguées et on peut ainsi réduire le problème de mise en correspondance de points à une vérification.*

$$y' = \frac{Z\,\mathbf{r}_2 \cdot \mathbf{p} + b_y}{Z\,\mathbf{r}_3 \cdot \mathbf{p} + b_z} \qquad [6.22]$$

qui expriment la relation entre les coordonnées x' et y' d'un point p' d'une image et un point p de l'autre image de coordonnées x et y et $\mathbf{p} = (x\ y\ 1)^t$.

Supposons de plus que la scène est une surface plane. Les coordonnées du point P vérifient alors l'équation :

$$\alpha X + \beta Y + \gamma Z = 1$$

et de plus on a $x = X/Z$ et $y = Y/Z$ en substituant on obtient :

$$\frac{1}{Z} = \alpha x + \beta y + \gamma = \mathbf{n} \cdot \mathbf{p}$$

avec $\mathbf{n} = (\alpha\ \beta\ \gamma)^t$.

Les équations [6.21] et [6.22] peuvent maintenant s'écrire :

$$x' = \frac{\mathbf{r}_1 \cdot \mathbf{p} + b_x(\mathbf{n} \cdot \mathbf{p}}{\mathbf{r}_3 \cdot \mathbf{p} + b_z(\mathbf{n} \cdot \mathbf{p})}$$

$$y' = \frac{\mathbf{r}_2 \cdot \mathbf{p} + b_y(\mathbf{n} \cdot \mathbf{p})}{\mathbf{r}_3 \cdot \mathbf{p} + b_z(\mathbf{n} \cdot \mathbf{p})}$$

Ceci peut s'écrire sous forme matricielle de la façon suivante ($\mathbf{n}^t = (n_x\, n_y\, n_z)$) :

$$\begin{pmatrix} sx' \\ sy' \\ s \end{pmatrix} = \left(\mathbf{R}\ + \mathbf{b}\mathbf{n}^t\right) \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

soit plus simplement :

$$\begin{pmatrix} sx' \\ sy' \\ s \end{pmatrix} = \mathbf{N} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

La matrice $\mathbf{N}$ est une matrice 3×3 exprimant la transformation projective entre un point d'une image et son correspondant dans l'autre image dans l'hypothèse où le point P de la scène appartient à un plan. On a donc :

$$\mathbf{N} = \left(\mathbf{R}\ + \mathbf{b}\mathbf{n}^t\right) = \begin{pmatrix} n_{11} & n_{12} & n_{13} \\ n_{21} & n_{22} & n_{23} \\ n_{31} & n_{32} & n_{33} \end{pmatrix} \qquad [6.23]$$

Les coefficients de la matrice $\mathbf{N}$ peuvent se calculer, à un facteur multiplicatif près, avec seulement 4 points de correspondance. En effet, chaque correspondance fournit 2 contraintes linéaires quant aux coefficients de la matrice $\mathbf{N}$ et

pour n correspondances ($n \geq 4$) on a à résoudre un système linéaire surcontraint.

Supposons, par exemple, qu'on ait déterminé la matrice $\mathbf{N}$ pour un système stéréo et par rapport au plan du sol, figure 6.19. Deux points p et p' correspondant aux projections d'un point P appartenant au sol vérifient donc la relation :

$$p' = \mathbf{N}p$$

alors que les points q et q' qui ne correspondent pas à un point du sol ne la vérifient pas :

$$q' \neq \mathbf{N}q$$

Ceci fournit un moyen simple de détection d'obstacles pour un robot mobile se déplaçant sur un sol plat. Néanmoins cette méthode de détection d'obstacles suppose que la mise en correspondance ait été obtenue au préalable.

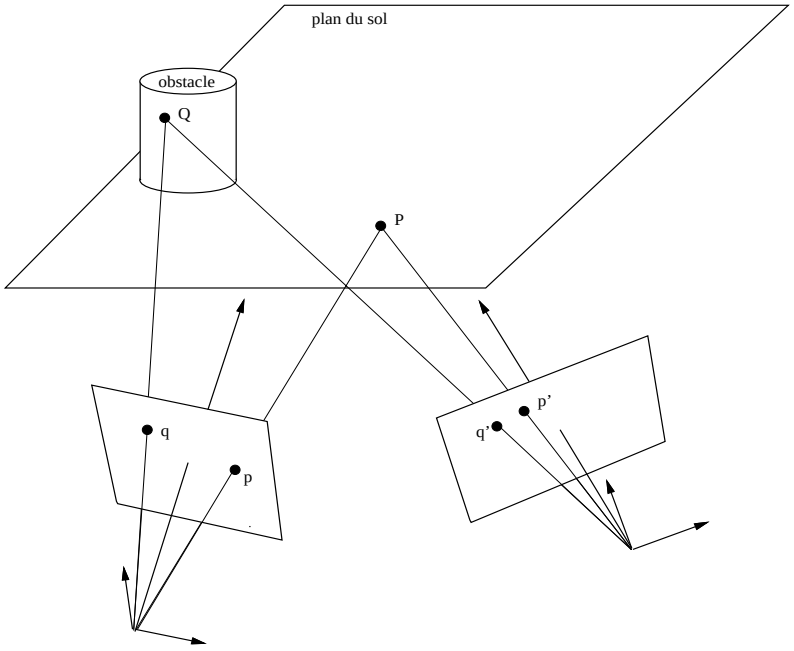


Figure 6.19. *En calibrant un système stéréo par rapport au plan du sol on peut détecter des obstacles à condition que l'appariement gauche/droite ait été préalablement établi.*

Chapitre 7

Localisation tri dimensionnelle

7.1. Introduction

Aux chapitres précédents nous avons étudié l'acquisition d'informations tridimensionnelles (3D) grâce à la vision. A quoi pourraient bien servir ces informations, ou ces données 3D ? Le premier exemple d'utilisation de telles données est fourni par un robot exécutant une tâche d'assemblage. Afin d'assembler deux objets, un robot doit être capable de *localiser* chaque objet par rapport à son référentiel de travail. Il pourra alors seulement saisir un objet et l'assembler avec l'autre objet. Le deuxième exemple est fourni par un robot mobile qui doit se rendre quelque part pour une opération de chargement/déchargement. Lorsque le robot se trouve à proximité du lieu où l'opération doit être effectuée, sa position par rapport aux objets environnants n'est pas connue. Il s'agit une fois de plus de localiser ces objets par rapport au référentiel du robot. Enfin, le troisième exemple est fourni par la robotique chirurgicale assistée par ordinateur. Il s'agit de mettre "en correspondance" des examens (radiologiques, échographiques, etc.) effectués lors de l'intervention (per-opératoires) avec des examens effectués avant l'intervention (pré-opératoires), [124].

D'abord nous allons décrire formellement le problème de localisation et ensuite nous allons proposer quelques solutions. Les solutions qui permettent de résoudre le problème de localisation 3D/3D peuvent également être utilisées pour résoudre le problème de calibration entre une caméra et un robot, lorsque la caméra est montée sur l'organe terminal du robot [91]. Nous verrons également que le problème de localisation est intimement lié au problème de reconnaissance qui sera traité indépendamment au chapitre 9. Le problème de

reconnaissance possède quelques similarités avec le problème de mise en correspondance stéréo. Quelques-unes des solutions que nous avons proposées pour résoudre le problème de mise en correspondance stéréo pourront être utilisées pour résoudre le problème de reconnaissance.

7.2. Définition du problème

Soit un ensemble de points 3D dans le référentiel d'un capteur visuel. Les coordonnées de ces points ont été obtenues, par exemple, par stéréo passive ou active. Soient $C_1, C_2, \ldots, C_n$ ces points. Par ailleurs, soit un ensemble de points à la surface d'un objet qui sont décrits dans le référentiel propre à cet objet : $M_1, M_2, \ldots, M_n$, figure 7.1. Si le capteur est en train d'observer l'objet en question, les points C_i correspondent aux points M_i. Il s'agit purement et simplement du **même** ensemble de points décrits dans deux référentiels différents.

La transformation qui permet de passer du repère de l'objet à celui du capteur est composée d'une matrice de rotation $\mathbf{R}$ et d'un vecteur de translation $\mathbf{t}$. Nous pouvons donc écrire pour tout i (soit pour toute mise en correspondance $C_i \leftrightarrow M_i$) :

$$\mathbf{c}_i = \mathbf{R}\ \mathbf{m}_i + \mathbf{t}$$

$\mathbf{m}_i$ étant le vecteur associé au point M_i. En pratique, à cause du bruit, il n'y aura pas de superposition parfaite entre les points objet et les points capteur. La distance entre un point objet transformé et le point correspondant dans le repère capteur sera :

$$d_i = \|\mathbf{c}_i - \mathbf{R}\ \mathbf{m}_i - \mathbf{t}\|$$

En sommant les carrés de ces distances on obtient un critère quadratique :

$$Q = \sum_{i=1}^{n} \|\mathbf{c}_i - \mathbf{R}\ \mathbf{m}_i - \mathbf{t}\|^2 \qquad [7.1]$$

Le problème de localisation 3D peut maintenant être défini de la façon suivante : sous l'hypothèse d'une mise en correspondance terme à terme de n points, trouver la transformation rigide (rotation et translation) qui minimise le critère fourni par l'équation [7.1].

En pratique la minimisation du critère Q est conditionnée par :

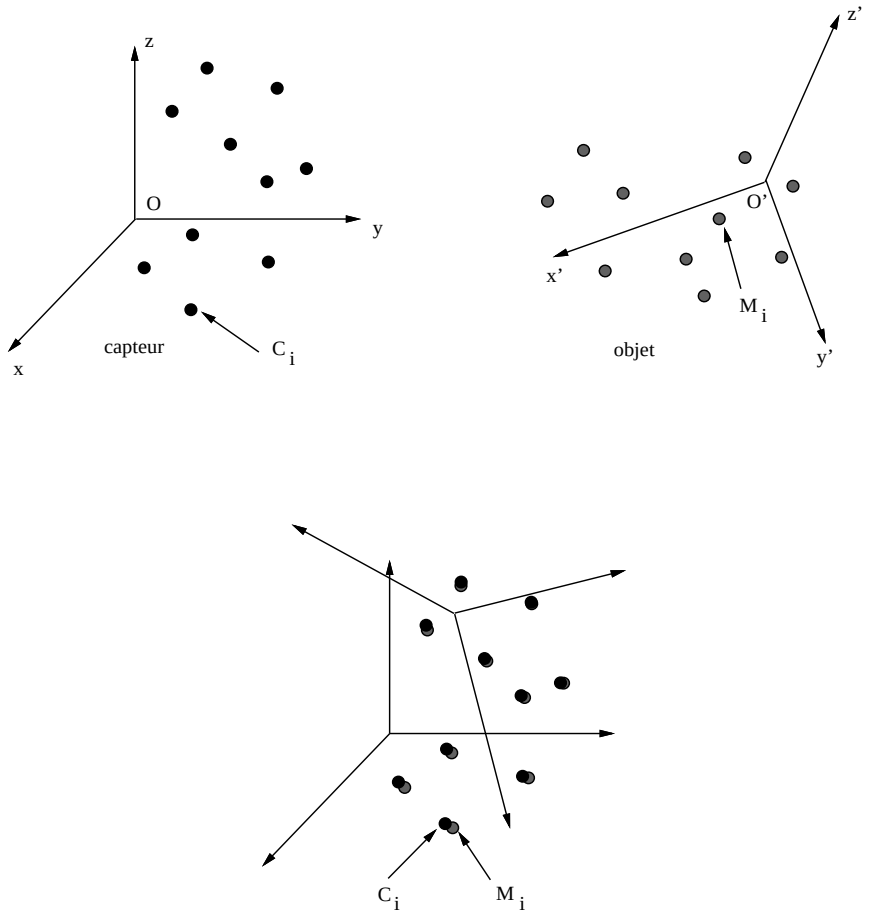


Figure 7.1. *Le même ensemble de points est décrit dans deux référentiels différents. En faisant surperposer les points on peut déterminer la rotation et la translation entre les deux référentiels.*

• les caractéristiques visuelles ne sont pas toujours des points. On aura à traiter aussi bien des points que des segments de droite, des facettes planes ou des primitives surfaciques ;

• la représentation matricielle de la rotation n'est pas la plus adéquate pour minimiser le critère Q.

Avant de proposer quelques solutions pour le problème de détermination de la transformation rigide optimale, nous allons décrire les représentations des caractéristiques telles que les droites et les plans ainsi que les expressions analytiques des transformations rigides des droites et des plans. Ensuite nous allons décrire deux représentations pour la rotation, une utilisant le calcul vectoriel et une autre utilisant les quaternions.

7.3. Représentation d'un plan

Un plan 3D peut être écrit sous la forme :

$$ax + by + cz = d$$

Si on impose la contrainte $a^2 + b^2 + c^2 = 1$, le vecteur $\mathbf{v}$ de coordonnées (a, b, c) normal au plan est de plus unitaire. Pour un point courant M de ce plan on a :

$$\overrightarrow{OM} \cdot \mathbf{v} = d$$

d est donc la projection de $\overrightarrow{OM}$ sur le vecteur $\mathbf{v}$, soit la distance de l'origine O à ce plan, figure 7.2.

Un plan est donc décrit par le couple formé d'un vecteur et d'un scalaire $(\mathbf{v}, d)$. Soit maintenant ce même plan décrit dans un autre repère :

$$\overrightarrow{O'M'} \cdot \mathbf{v}' = d'$$

Le changement de repère étant une rotation et une translation on obtient :

$$\begin{aligned} \overrightarrow{O'M'} &= \mathbf{R}\ \overrightarrow{OM} + \mathbf{t} \\ \mathbf{v}' &= \mathbf{R}\ \mathbf{v} \end{aligned}$$

De plus le point M' appartient au plan en question :

$$(\mathbf{R}\ \overrightarrow{OM} + \mathbf{t}) \cdot (\mathbf{R}\ \mathbf{v}) = d'$$

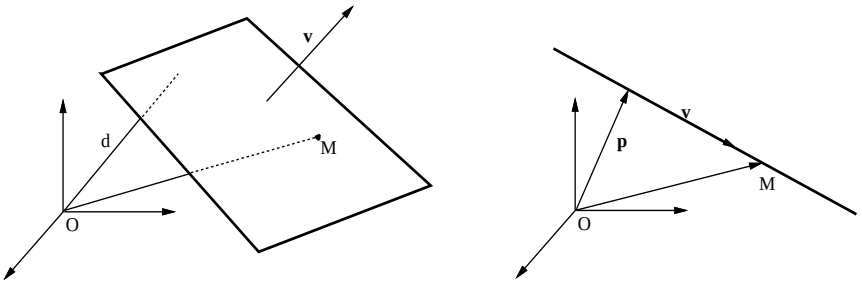


Figure 7.2. *Représentations d'un plan et d'une droite.*

En développant et en tenant compte du fait que la rotation conserve le produit scalaire on obtient les deux équations suivantes :

$$\mathbf{v}' = \mathbf{R}\ \mathbf{v} \qquad [7.2]$$

$$d' = d + (\mathbf{R}\ \mathbf{v}) \cdot \mathbf{t} \qquad [7.3]$$

7.4. Représentation d'une droite

La représentation la plus connue d'une droite est celle qui considère la droite comme l'intersection de deux plans :

$$\begin{aligned} a_1 x + b_1 y + c_1 z &= d_1 \\ a_2 x + b_2 y + c_2 z &= d_2 \end{aligned}$$

Cependant cette représentation n'est pas minimale (il faut 8 paramètres) et elle n'est pas non plus très pratique pour décrire la transformation rigide d'une droite. On choisira de préférence la représentation paramétrique :

$$\overrightarrow{OM} = \mathbf{p} + \lambda \mathbf{v}$$

$\mathbf{p}$ est le vecteur perpendiculaire à la droite dont la longueur est égale à la distance de l'origine à la droite et $\mathbf{v}$ est le vecteur directeur de la droite, figure 7.2. Par commodité on choisira un vecteur directeur unitaire. On a donc :

$$\mathbf{p} \cdot \mathbf{v} = 0$$

Une droite est donc représentée par un couple de deux vecteurs, $\mathbf{p}$ et $\mathbf{v}$, soit 4 paramètres car un des vecteurs est unitaire et les deux vecteurs sont perpendiculaires. Comme dans le cas du plan, étudions le changement de repère (rotation et translation) appliqué à une droite. Dans le nouveau repère la droite s'écrit :

$$\overrightarrow{O'M'} = \mathbf{p}' + \lambda'\mathbf{v}'$$

Calculons $\mathbf{p}'$ et $\mathbf{v}'$ en fonction de $\mathbf{p}$, de $\mathbf{v}$, de $\mathbf{R}$ et de $\mathbf{t}$. On a :

$$\begin{aligned} \overrightarrow{O'M'} &= \mathbf{R}\ \overrightarrow{OM} + \mathbf{t} \\ &= \mathbf{R}\ \mathbf{p} + \lambda \mathbf{R}\ \mathbf{v} + \mathbf{t} \end{aligned}$$

En identifiant on obtient :

$$\begin{aligned} \mathbf{v}' &= \mathbf{R}\ \mathbf{v} \\ \mathbf{p}' &= \mathbf{R}\ \mathbf{p} + \mathbf{t} + (\lambda - \lambda')\mathbf{v}' \end{aligned}$$

En tenant compte de la contrainte :

$$\mathbf{p}' \cdot \mathbf{v}' = 0$$

on obtient :

$$\lambda - \lambda' = -(\mathbf{R}\ \mathbf{v}) \cdot \mathbf{t}$$

Et finalement le changement de repère s'écrit :

$$\mathbf{v}' = \mathbf{R}\ \mathbf{v} \qquad [7.4]$$

$$\mathbf{p}' = \mathbf{R}\ \mathbf{p} + \mathbf{t} - ((\mathbf{R}\ \mathbf{v}) \cdot \mathbf{t})\mathbf{R}\ \mathbf{v} \qquad [7.5]$$

Une troisième façon de représenter une droite est d'utiliser les coordonnées de Plucker, [53]. Soient P_1 et P_2 deux points appartenant à une droite. Les coordonnées de Plucker sont données par les deux vecteurs suivants, O étant l'origine du repère cartésien :

$$\begin{aligned} \mathbf{v}_1 &= OP_1 - OP_2 \\ \mathbf{v}_2 &= OP_1 \wedge OP_2 \end{aligned}$$

Dans ce cas le changement de repère s'écrit :

$$\mathbf{v}_1' = \mathbf{R}\ \mathbf{v}_1 \qquad [7.6]$$

$$\mathbf{v}_2' = \mathbf{R}\ \mathbf{v}_2 - \mathbf{t} \wedge (\mathbf{R}\ \mathbf{v}_1) \qquad [7.7]$$

7.5. Représentation d'une rotation

Nous allons étudier quatre façons de représenter une rotation :

- la représentation matricielle ;
- les angles d'Euler ;
- la représentation vectorielle ;
- la représentation par des quaternions unitaires.

7.5.1. Matrices de rotation

Comme nous l'avons déjà vu au chapitre 5 une rotation peut être vue comme la transformation pouvant faire superposer deux repères orthonormés. Une rotation est donc décrite par une matrice orthonormée :

$$\mathbf{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$

Il y a de plus 6 contraintes d'orthonormalité qui expriment que les vecteurs ligne (ou colonne) sont de norme égale à 1 et orthogonaux entre eux :

$$\sum_{i=1}^{3} r_{ij} r_{ik} = \delta_{kj} \quad \forall k, j \in \{1, 2, 3\}$$

La rotation est donc représentée par 9 paramètres et 6 contraintes non linéaires. Ce n'est donc pas la meilleure représentation à utiliser lorsqu'on veut calculer la rotation optimale pour faire coïncider des points, des plans ou des droites.

Notons au passage quelques propriétés intéressantes des matrices de rotations. L'ensemble des rotations est un sous-groupe du groupe euclidien couramment désigné par $E(3)$. Le groupe des rotations est non commutatif. La matrice $\mathbf{R}$ a les propriétés suivantes :

$$\mathbf{R}^{-1} = \mathbf{R}^{t}$$

et

$$det(\mathbf{R}) = 1$$

7.5.2. Angles d'Euler

Une autre représentation des rotations est celle utilisant une décomposition de la matrice de rotation en trois matrices de rotation autour de chacun des trois axes. Ces trois matrices sont :

$$\mathbf{R}_z = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad [7.8]$$

$$\mathbf{R}_y = \begin{pmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{pmatrix} \quad [7.9]$$

$$\mathbf{R}_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{pmatrix} \quad [7.10]$$

La matrice $\mathbf{R}$ est donnée par :

$$\mathbf{R} = \mathbf{R}_z \mathbf{R}_y \mathbf{R}_x$$

et les trois angles θ, ϕ et ψ s'appellent les angles d'Euler d'après le mathématicien Leonhard Euler (1707-1783) qui a étudié les lois régissant les déplacements des solides.

7.5.3. Axe et angle de rotation

Nous allons maintenant introduire la représentation vectorielle d'une rotation. Déjà Euler avait démontré que toute rotation infinitésimale a lieu autour d'un axe instantané. Cependant c'est Olinde Rodrigues (1794-1851) qui propose un traitement complet des déplacements des objets indépendamment des forces qui causent ces déplacements. En 1840 Rodrigues a proposé le *théorème fondamental* qui montre que tout déplacement fini peut être effectué par une rotation autour d'un axe combinée avec une translation le long de cet axe. Il s'agit donc de la représentation des déplacements en termes d'un *vissage*. Ici on s'intéresse uniquement à la représentation des rotations.

Remarquons tout d'abord quelques propriétés simples de la matrice de rotation (certaines de ces propriétés ont déjà été mentionnées) :

- le déterminant d'une matrice de rotation est égal à 1,
- l'inverse d'une matrice de rotation est égale à sa transposée ;
- les valeurs propres d'une matrice de rotation sont : 1, $e^{i\theta}$ et $e^{-i\theta}$ (complexes conjugués).

L'axe d'une rotation est tel qu'il n'est pas modifié par la rotation. C'est donc la direction propre associée à la valeur propre unitaire :

$$\mathbf{R}\ \mathbf{n} = \mathbf{n}$$

L'angle de rotation est fourni par les valeurs propres complexes conjugués.

En notation matricielle un vecteur $\mathbf{v}$ se transforme en un vecteur $\mathbf{v}'$ grâce à la formule suivante :

$$\mathbf{v}' = \mathbf{R}\ \mathbf{v}$$

En notation vectorielle, un vecteur subissant une rotation d'axe $\mathbf{n}$ ($\mathbf{n} \cdot \mathbf{n} = 1$) et d'angle θ devient un vecteur $\mathbf{v}'$ qui s'obtient par la formule de Rodrigues :

$$\mathbf{v}' = \mathbf{v} + \sin\theta\ \mathbf{n} \wedge \mathbf{v} + (1 - \cos\theta)\mathbf{n} \wedge (\mathbf{n} \wedge \mathbf{v}) \qquad [7.11]$$

Pour démontrer cette formule nous allons décomposer le vecteur $\mathbf{v}$ en une composante parallèle à $\mathbf{n}$ et une composante orthogonale à $\mathbf{n}$, figure 7.3. (voir également [8]) :

$$\begin{aligned} \mathbf{v} &= \mathbf{v}_{\parallel} + \mathbf{v}_{\perp} \\ &= (\mathbf{v} \cdot \mathbf{n})\mathbf{n} + (\mathbf{v} - (\mathbf{v} \cdot \mathbf{n})\mathbf{n}) \end{aligned}$$

$\mathbf{v}'$ sera donc obtenu en additionant $\mathbf{v}'_{\parallel}$ et $\mathbf{v}'_{\perp}$. En remarquant que $\mathbf{v}'_{\parallel} = \mathbf{v}_{\parallel}$ on aura :

$$\mathbf{v}' = \mathbf{v}_{\parallel} + \mathbf{v}'_{\perp}$$

$\mathbf{v}'_{\perp}$ est obtenu en faisant tourner $\mathbf{v}_{\perp}$ d'un angle θ dans un plan orthogonal à $\mathbf{n}$. Dans la base vectorielle $\mathbf{n}$, $\mathbf{v}_{\perp}$ et $\mathbf{n} \wedge \mathbf{v}_{\perp}$ on a :

$$\mathbf{v}'_{\perp} = \cos\theta\ \mathbf{v}_{\perp} + \sin\theta\ \mathbf{n} \wedge \mathbf{v}_{\perp}$$

Finalement on obtient pour $\mathbf{v}'$:

$$\begin{aligned} \mathbf{v}' &= \mathbf{v}_{\parallel} + \mathbf{v}'_{\perp} \\ &= \mathbf{v}_{\parallel} + \cos\theta\ \mathbf{v}_{\perp} + \sin\theta\ \mathbf{n} \wedge \mathbf{v}_{\perp} \\ &= (\mathbf{v} \cdot \mathbf{n})\mathbf{n} + \cos\theta(\mathbf{v} - (\mathbf{v} \cdot \mathbf{n})\mathbf{n}) + \sin\theta(\mathbf{n} \wedge (\mathbf{v} - (\mathbf{v} \cdot \mathbf{n})\mathbf{n})) \\ &= \cos\theta\ \mathbf{v} + \sin\theta\ \mathbf{n} \wedge \mathbf{v} + (1 - \cos\theta)(\mathbf{v} \cdot \mathbf{n})\mathbf{n} \end{aligned}$$

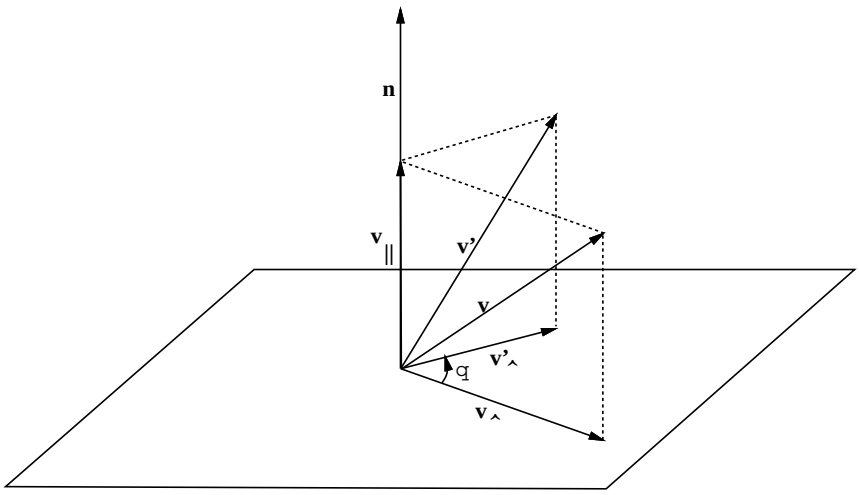


Figure 7.3. *Décomposition vectorielle permettant de démontrer facilement la formule de Rodrigues, d'après Ayache.*

Par ailleurs on a la relation vectorielle suivante :

$$\mathbf{n} \wedge (\mathbf{n} \wedge \mathbf{v}) = (\mathbf{v} \cdot \mathbf{n})\mathbf{n} - (\mathbf{n} \cdot \mathbf{n})\mathbf{v}$$

Mais comme $\mathbf{n}$ est unitaire on obtient :

$$(\mathbf{v} \cdot \mathbf{n})\mathbf{n} = \mathbf{v} + \mathbf{n} \wedge (\mathbf{n} \wedge \mathbf{v})$$

et en substituant dans l'équation précédente on obtient bien la formule de Rodrigues, soit l'équation [7.11].

7.5.4. De $(\mathbf{n}, \theta)$ à $\mathbf{R}$

La représentation d'une rotation par un axe et un angle est, comme nous allons le voir, très pratique pour résoudre le problème de la recherche d'une transformation rigide optimale. Cependant il est utile de pouvoir passer rapidement d'une représentation à une autre.

Remarquons tout d'abord qu'un produit vectoriel peut se mettre sous forme

matricielle :

$$\begin{aligned}\mathbf{n} \wedge \mathbf{v} &= \begin{pmatrix} -n_z v_y + n_y v_z \\ n_z v_x - n_x v_z \\ -n_y v_x + n_x v_y \end{pmatrix} \\ &= \begin{pmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix}\end{aligned}$$

$S(\mathbf{n})$ est la matrice antisymétrique associée au vecteur $\mathbf{n}$:

$$S(\mathbf{n}) = \begin{pmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{pmatrix}$$

On obtient ainsi la version matricielle de la formule de Rodrigues :

$$\mathbf{v}' = \left(\mathbf{I} + \sin\theta S(\mathbf{n}) + (1 - \cos\theta) S^2(\mathbf{n})\right) \mathbf{v}$$

Soit pour la matrice de rotation :

$$\mathbf{R} \ = \mathbf{I} + \sin\theta S(\mathbf{n}) + (1 - \cos\theta) S^2(\mathbf{n})$$

En explicitant les termes et avec les notations $s = \sin\theta$ et $c = \cos\theta$ on obtient pour $\mathbf{R}$:

$$\begin{pmatrix} c + (1-c)n_x^2 & -n_z s + (1-c)n_x n_y & n_y s + (1-c)n_x n_z \\ n_z s + (1-c)n_x n_y & c + (1-c)n_y^2 & -n_x s + (1-c)n_z n_y \\ -n_y s + (1-c)n_x n_z & n_x s + (1-c)n_z n_y & c + (1-c)n_z^2 \end{pmatrix} \quad [7.12]$$

Il est intéressant de remarquer que $(\mathbf{n}, \theta)$ représente la même rotation que $(-\mathbf{n}, -\theta)$ ou que $(-\mathbf{n}, 2\pi - \theta)$. Il suffit de substituer $\mathbf{n}$ par $-\mathbf{n}$ et θ par $-\theta$ ou par $2\pi - \theta$ dans l'équation précédente.

7.5.5. De R à $(\mathbf{n}, \theta)$

Il sera également utile de pouvoir extraire l'axe et l'angle de rotation à partir de la matrice de rotation. Pour cela il y a deux possibilités :

1. extraire les valeurs propres de cette matrice. Calculer le vecteur propre associé à la valeur propre 1 et normaliser ce vecteur. Cela donne la direction de l'axe de rotation. L'angle de rotation se calcule facilement à partir des valeurs propres complexes conjuguées.

2. identifier les éléments r_{ij} de la matrice de rotation avec l'expression de cette matrice telle qu'elle est fournie par la formule de Rodrigues, équation [7.12].

Le calcul par identification est trivial. En additionnant les termes diagonaux dans l'équation [7.12] on obtient :

$$r_{11} + r_{22} + r_{33} = 1 + 2\cos\theta$$

Par ailleurs on a les relations suivantes :

$$\begin{aligned} r_{21} - r_{12} &= 2n_z \sin\theta \\ r_{31} - r_{13} &= -2n_y \sin\theta \\ r_{32} - r_{23} &= 2n_x \sin\theta \end{aligned}$$

En tenant compte du fait que le vecteur $\mathbf{n}$doit être unitaire on obtient (rappelons-nous que le signe n'est pas très important) :

$$\begin{aligned} \cos\theta &= \frac{1}{2}(r_{11} + r_{22} + r_{33} - 1) \\ \sin^2\theta &= \frac{1}{4}\left((r_{21} - r_{12})^2 + (r_{31} - r_{13})^2 + (r_{32} - r_{23})^2\right) \\ n_x &= \frac{r_{32} - r_{23}}{2\sin\theta} \\ n_y &= -\frac{r_{31} - r_{13}}{2\sin\theta} \\ n_z &= \frac{r_{21} - r_{12}}{2\sin\theta} \end{aligned}$$

Il faut remarquer ici que le passage de $\mathbf{R}$ à $(\mathbf{n}, \theta)$ est ambigu car on peut choisir aussi bien $(\mathbf{n}, \theta)$ que $(-\mathbf{n}, -\theta)$ ou que $(-\mathbf{n}, 2\pi - \theta)$. La représentation $(\mathbf{n}, \theta)$ comporte 4 paramètres. Si de plus on choisit le module de $\mathbf{n}$tel qu'il soit égal à la valeur de l'angle de rotation :

$$\|\mathbf{n}\| = \theta$$

on n'a alors plus que 3 paramètres pour décrire une rotation et cette représentation est minimale. En pratique elle sera utilisée (sous une forme légérement modifiée) pour calculer la rotation optimale entre deux ensembles de caractéristiques 3D.

7.5.6. Rotation et quaternion unitaire

Une autre représentation possible des rotations est celle utilisant les quaternions. Dans un premier temps nous allons étudier quelques propriétés des quaternions, ensuite nous les utiliserons pour décrire une rotation et nous établirons également l'équivalence avec les autres représentations déjà utilisées. L'utilisation des quaternions se justifiera par une formulation très élégante du problème d'optimisation, [59].

7.5.6.1. Les quaternions

Les quaternions peuvent être vus comme des vecteurs de dimension 4 ou encore comme des nombre complexes à trois parties imaginaires :

$$\mathbf{q} = q_0 + iq_x + jq_y + kq_z$$

avec :

$$i^2 = j^2 = k^2 = ijk = -1$$

On peut remarquer qu'on obtient également :

$$ij = -ji = k$$
$$jk = -kj = i$$
$$ki = -ik = j$$

Grâce à ces formules on peut facilement calculer le produit de deux quaternions, noté "$*$" :

$$\mathbf{r} * \mathbf{q} = (r_0 + ir_x + jr_y + kr_z)(q_0 + iq_x + jq_y + kq_z)$$

Le produit de deux quaternions n'est pas commutatif. Comme le produit vectoriel, le produit de deux quaternions peut s'écrire sous forme matricielle :

$$\mathbf{r} * \mathbf{q} = Q(\mathbf{r})\mathbf{q}$$

avec $Q(\mathbf{r})$ donnée par :

$$Q(\mathbf{r}) = \begin{pmatrix} r_0 & -r_x & -r_y & -r_z \\ r_x & r_0 & -r_z & r_y \\ r_y & r_z & r_0 & -r_x \\ r_z & -r_y & r_x & r_0 \end{pmatrix} \qquad [7.13]$$

On peut également écrire le produit sous la forme suivante :

$$\mathbf{q} * \mathbf{r} = W(\mathbf{r})\mathbf{q}$$

Avec $W(\mathbf{r})$ défini par :

$$W(\mathbf{r}) = \begin{pmatrix} r_0 & -r_x & -r_y & -r_z \\ r_x & r_0 & r_z & -r_y \\ r_y & -r_z & r_0 & r_x \\ r_z & r_y & -r_x & r_0 \end{pmatrix} \qquad [7.14]$$

On peut facilement vérifier les propriétés suivantes quant à ces matrices :

$$\begin{aligned} Q(\mathbf{r})^t Q(\mathbf{r}) &= Q(\mathbf{r})Q(\mathbf{r})^t = \mathbf{r}^t\mathbf{r}I \\ W(\mathbf{r})^t W(\mathbf{r}) &= W(\mathbf{r})W(\mathbf{r})^t = \mathbf{r}^t\mathbf{r}I \\ Q(\mathbf{r})\mathbf{q} &= W(\mathbf{q})\mathbf{r} \\ Q(\mathbf{r})^t\mathbf{r} &= W(\mathbf{r})^t\mathbf{r} = \mathbf{r}^t\mathbf{r}\mathbf{e} \\ Q(\mathbf{r})Q(\mathbf{q}) &= Q(Q(\mathbf{r})\mathbf{q}) \\ W(\mathbf{r})W(\mathbf{q}) &= W(W(\mathbf{r})\mathbf{q}) \\ Q(\mathbf{r})W(\mathbf{q})^t &= W(\mathbf{q})^t Q(\mathbf{r}) \end{aligned}$$

e étant le quaternion unité : $e = (1\ 0\ 0\ 0)^t$.

Le produit scalaire de deux quaternions est donné par :

$$\mathbf{r} \cdot \mathbf{q} = r_0 q_0 + r_x q_x + r_y q_y + r_z q_z$$

On a bien évidemment :

$$\mathbf{q} \cdot \mathbf{q} = \|\mathbf{q}\|^2$$

Le quaternion conjugué de $\mathbf{q}$ est $\overline{\mathbf{q}}$ défini par :

$$\overline{\mathbf{q}} = q_0 - iq_x - jq_y - kq_z$$

Notons que si $Q(\mathbf{q})$ et $W(\mathbf{q})$ sont les matrices associées à $\mathbf{q}$, $Q(\mathbf{q})^t$ et $W(\mathbf{q})^t$ sont les matrices associées au quaternion conjugué $\overline{\mathbf{q}}$. On a :

$$\mathbf{q} * \overline{\mathbf{q}} = \mathbf{q} \cdot \mathbf{q} = \|\mathbf{q}\|^2$$

et on obtient ainsi le quaternion inverse de $\mathbf{q}$:

$$\mathbf{q}^{-1} = \frac{1}{\|\mathbf{q}\|^2}\,\overline{\mathbf{q}}$$

Pour un quaternion unitaire (de norme égale à 1) l'inverse est égal à son conjugué. Nous allons maintenant établir quelques propriétés du produit qui nous seront utiles par la suite. Ces propriétés sont facilement vérifiables si on utilise la notation matricielle. On a :

$$(\mathbf{q} * \mathbf{p}) \cdot (\mathbf{q} * \mathbf{r}) = (\mathbf{q} \cdot \mathbf{q})(\mathbf{p} \cdot \mathbf{r}) \qquad [7.15]$$

On en déduit facilement les propriétés suivantes :

$$\begin{aligned} (\mathbf{q} * \mathbf{p}) \cdot (\mathbf{q} * \mathbf{p}) &= (\mathbf{p} \cdot \mathbf{p})(\mathbf{q} \cdot \mathbf{q}) \\ \|\mathbf{p} * \mathbf{q}\|^2 &= \|\mathbf{p}\|^2 \|\mathbf{q}\|^2 \\ (\mathbf{p} * \mathbf{q}) \cdot \mathbf{r} &= \mathbf{p} \cdot (\mathbf{r} * \overline{\mathbf{q}}) \end{aligned}$$

Un vecteur de dimension 3 peut s'écrire comme un quaternion purement imaginaire :

$$\mathbf{q} = 0 + iq_x + jq_y + kq_z$$

et les matrices associées à un quaternion purement imaginaire sont des matrices antisymétriques. Par exemple on a :

$$Q(\mathbf{q}) = \begin{pmatrix} 0 & -q_x & -q_y & -q_z \\ q_x & 0 & -q_z & q_y \\ q_y & q_z & 0 & -q_x \\ q_z & -q_y & q_x & 0 \end{pmatrix}$$

avec les propriétés évidentes :

$$\begin{aligned} Q(\mathbf{q})^t &= -Q(\mathbf{q}) \\ W(\mathbf{q})^t &= -W(\mathbf{q}) \end{aligned}$$

7.5.6.2. Quaternion et rotation

On peut représenter une rotation avec un quaternion unitaire à condition de pouvoir trouver une transformation qui change un vecteur (un quaternion purement imaginaire) en un vecteur de façon que la transformation préserve la longueur du vecteur transformé ainsi que le produit scalaire et le signe du produit vectoriel.

Soit $\mathbf{r}$ un quaternion purement imaginaire et $\mathbf{q}$ un quaternion unitaire. On peut alors remarquer que le quaternion $\mathbf{r}'$ tel que :

$$\mathbf{r}' = \mathbf{q} * \mathbf{r} * \overline{\mathbf{q}} \qquad [7.16]$$

est un quaternion purement imaginaire et de plus on a :

$$\mathbf{q} * \mathbf{r} * \overline{\mathbf{q}} = (Q(\mathbf{q})\mathbf{r}) * \overline{\mathbf{q}} = (W(\mathbf{q})^t Q(\mathbf{q}))\mathbf{r}$$

Par ailleurs notons que :

$$(-\mathbf{q}) * \mathbf{r} * (-\overline{\mathbf{q}}) = \mathbf{q} * \mathbf{r} * \overline{\mathbf{q}}$$

ce qui implique que les quaternions $\mathbf{q}$ et $-\mathbf{q}$ représentent la même rotation. La matrice $W(\mathbf{q})^t Q(\mathbf{q})$ est une matrice orthonormée :

$$W(\mathbf{q})^t Q(\mathbf{q}) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & q_0^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y - q_0 q_z) & 2(q_x q_z + q_0 q_y) \\ 0 & 2(q_x q_y + q_0 q_z) & q_0^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z - q_0 q_x) \\ 0 & 2(q_x q_z - q_0 q_y) & 2(q_y q_z + q_0 q_x) & q_0^2 - q_x^2 - q_y^2 + q_z^2 \end{pmatrix}$$

On peut maintenant établir l'expression d'une matrice de rotation $\mathbf{R}$ en fonction d'un quaternion unitaire $\mathbf{q}$:

$$\mathbf{R} = \begin{pmatrix} q_0^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y - q_0 q_z) & 2(q_x q_z + q_0 q_y) \\ 2(q_x q_y + q_0 q_z) & q_0^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z - q_0 q_x) \\ 2(q_x q_z - q_0 q_y) & 2(q_y q_z + q_0 q_x) & q_0^2 - q_x^2 - q_y^2 + q_z^2 \end{pmatrix}$$

et on peut également écrire :

$$W(\mathbf{q})^t Q(\mathbf{q}) = \begin{pmatrix} 1 & 0^t \\ 0 & \mathbf{R} \end{pmatrix}$$

Nous pouvons également extraire un quaternion unitaire à partir d'une matrice de rotation. Soit r_{ij} un élément de la matrice de rotation. En considérant les termes diagonaux de la matrice précédente on obtient les combinaisons suivantes :

$$\begin{aligned} 1 + r_{11} + r_{22} + r_{33} &= 4q_0^2 \\ 1 + r_{11} - r_{22} - r_{33} &= 4q_x^2 \\ 1 - r_{11} + r_{22} - r_{33} &= 4q_y^2 \\ 1 - r_{11} - r_{22} + r_{33} &= 4q_z^2 \end{aligned}$$

On choisit la composante du quaternion ayant la valeur absolue la plus élevée. Ensuite on injecte la composante choisie dans trois parmi les six expressions :

$$\begin{aligned} r_{32} - r_{23} &= 4q_0q_x \\ r_{13} - r_{31} &= 4q_0q_y \\ r_{21} - r_{12} &= 4q_0q_z \\ r_{21} + r_{12} &= 4q_xq_y \\ r_{32} + r_{23} &= 4q_yq_z \\ r_{13} + r_{31} &= 4q_zq_x \end{aligned}$$

ce qui permet de trouver les composantes du quaternion unitaire correspondant à la rotation.

7.5.6.3. Equivalence avec la formule de Rodrigues

Nous venons d'établir l'équivalence entre la représentation matricielle et la représentation quaternion d'une rotation. Il y a une relation très simple entre un quaternion unitaire et l'axe et l'angle d'une rotation. Remarquons tout d'abord qu'un quaternion peut s'écrire comme la concaténation d'un réel avec un vecteur de dimension 3 :

$$\mathbf{q} = q_0 + iq_x + jq_y + kq_z = (q_0, \vec{\mathbf{q}})$$

et avec cette notation la multiplication de deux quaternions peut s'écrire :

$$\mathbf{p} * \mathbf{q} = (p_0q_0 - \vec{\mathbf{p}} \cdot \vec{\mathbf{q}},\ p_0\vec{\mathbf{q}} + q_0\vec{\mathbf{p}} + \vec{\mathbf{p}} \wedge \vec{\mathbf{q}})$$

L'équation [7.16] devient alors :

$$\mathbf{r}' = (0, (q_0^2 - \vec{\mathbf{q}} \cdot \vec{\mathbf{q}})\vec{\mathbf{r}} + 2q_0\vec{\mathbf{q}} \wedge \vec{\mathbf{r}} + 2(\vec{\mathbf{q}} \cdot \vec{\mathbf{r}})\vec{\mathbf{q}})$$

$\mathbf{r}'$ est donc un quaternion purement imaginaire et sa partie purement imaginaire est :

$$\vec{\mathbf{r}}' = (q_0^2 - \vec{\mathbf{q}} \cdot \vec{\mathbf{q}})\vec{\mathbf{r}} + 2q_0\vec{\mathbf{q}} \wedge \vec{\mathbf{r}} + 2(\vec{\mathbf{q}} \cdot \vec{\mathbf{r}})\vec{\mathbf{q}}$$

et on peut donc confondre un quaternion purement imaginaire avec un vecteur de dimension 3 :

$$\vec{\mathbf{r}'} = \mathbf{r}'$$

En remarquant que la formule de Rodrigues (équation [7.11]) peut se mettre sous la forme suivante :

$$\vec{\mathbf{r}}' = \cos\theta\ \vec{\mathbf{r}} + \sin\theta\ \vec{\mathbf{n}} \wedge \vec{\mathbf{r}} + (1 - \cos\theta)(\vec{\mathbf{n}} \cdot \vec{\mathbf{r}})\vec{\mathbf{n}}$$

et en identifiant les deux expressions précédentes de $\mathbf{r}'$ on obtient :

$$\begin{aligned} q_0 &= \cos\frac{\theta}{2} \\ \vec{\mathbf{q}} &= \sin\frac{\theta}{2}\vec{\mathbf{n}} \end{aligned}$$

Le quaternion représentant une rotation d'axe $\mathbf{n}$ et d'angle θ s'écrit donc :

$$\mathbf{q} = \cos\frac{\theta}{2} + \sin\frac{\theta}{2}\,(in_x + jn_y + kn_z) \qquad [7.17]$$

7.6. Transformation rigide optimale

7.6.1. Décomposition du critère d'optimalité

Nous sommes maintenant en mesure d'écrire à nouveau le critère permettant de déterminer la transformation rigide optimale dans les cas des points, des plans et des droites.

Pour des correspondances de points, le critère reste inchangé et le problème de minimisation correspondant est, conformément à l'équation [7.1] :

$$\min_{\mathbf{R},\mathbf{t}} \left(\sum_{i=1}^{n} \|\mathbf{c}_i - \mathbf{R}\ \mathbf{m}_i - \mathbf{t}\|^2 \right)$$

Quant aux correspondances de plans et conformément aux équations [7.2] et [7.3], le critère à minimiser se décompose en deux critères :

$$\min_{\mathbf{R}} \left(\sum_{i=1}^{n} \|\mathbf{v}'_i - \mathbf{R}\ \mathbf{v}_i\|^2 \right)$$

$$\min_{\mathbf{t}} \left(\sum_{i=1}^{n} \|d'_i - d_i - (\mathbf{R}^*\mathbf{v}_i) \cdot \mathbf{t}\|^2 \right)$$

$\mathbf{R}^*$ étant la rotation optimale trouvée grâce au premier de ces deux critères.

Quant aux correspondances de droites et pour la représentation paramétrique choisie on obtient, à partir des équations [7.4] et [7.5] les deux critères suivants :

$$\min_{\mathbf{R}} \left(\sum_{i=1}^{n} \|\mathbf{v}'_i - \mathbf{R}\ \mathbf{v}_i\|^2 \right)$$

$$\min_{\mathbf{t}} \left(\sum_{i=1}^{n} \|\mathbf{p}'_i - \mathbf{R}\ {}^*\mathbf{p}_i - t + ((\mathbf{R}\ {}^*\mathbf{v}_i) \cdot \mathbf{t})\mathbf{R}\ {}^*\mathbf{v}_i\|^2 \right)$$

Si on choisit les coordonnées de Plucker, à partir des équations [7.6] et [7.7] on obtient tout d'abord :

$$\min_{\mathbf{R}} \left(\sum_{i=1}^{n} \|\mathbf{w}'_{1i} - \mathbf{R}\ \mathbf{w}_{1i}\|^2 \right)$$

avec :

$$\mathbf{w}_{1i} = \frac{\mathbf{v}_{1i}}{\|\mathbf{v}_{1i}\|}$$

$$\mathbf{w}'_{1i} = \frac{\mathbf{v}'_{1i}}{\|\mathbf{v}'_{1i}\|}$$

ce qui permet de déterminer la rotation optimale $\mathbf{R}\ {}^*$.

Pour déterminer la translation optimale, on peut remarquer que pour tout i, le vecteur $\mathbf{v}'_{2i}$ doit être colinéaire au transformé du vecteur $\mathbf{v}_{2i}$, soit leur produit vectoriel est nul :

$$\mathbf{v}_{2i} \wedge (\mathbf{R}\ {}^*\mathbf{v}_{2i} + \mathbf{t} \wedge (\mathbf{R}\ {}^*\mathbf{v}_{1i})) = 0$$

En simplifiant cette expression et en sommant sur toutes les droites se correspondant, on obtient un critère pour trouver la translation optimale :

$$\min_{\mathbf{t}} \left(\sum_{i=1}^{n} \|\mathbf{v}'_{2i} \wedge (\mathbf{R}\ {}^*\mathbf{v}_{2i}) + r_i(\mathbf{v}'_{2i} \cdot \mathbf{t})\mathbf{v}'_{1i}\|^2 \right)$$

Pour résumer, dans les cas des droites et des plans nous avons décomposé le critère initial en deux critères : on estime d'abord la rotation optimale et ensuite on utilise cette rotation pour déterminer la translation optimale. Cette décomposition a des avantages et des inconvénients. L'avantage est que chaque critère est plus simple et, comme nous allons le voir, la solution au problème d'optimisation de la rotation est très élégante et très fiable d'un point de vue numérique. Le critère de translation se ramène à un problème d'optimisation

linéaire très classique. L'inconvénient est que les erreurs éventuelles associées au calcul de la rotation vont se répercuter sur le calcul de la translation.

Dans le cas des points nous pouvons également décomposer le critère initial en deux critères. En effet, soient $\mathbf{R}^*$ et $\mathbf{t}^*$ la rotation et translation optimales du critère dans le cas de la mise en correspondance de points, équation [7.1]. On a alors la propriété suivante : le centre de gravité des points C_i coïncide avec le centre de gravité des transformés des M_i [110]. Si C est le centre de gravité des points C_i et M est le centre de gravité des points M_i, on a d'après cette propriété :

$$\mathbf{c} = \mathbf{R}^*\mathbf{m} + \mathbf{t}^*$$

Choisissons C et M comme origines respectives des repères des données capteur et des données modèle. Cela revient à effectuer les changements de variables suivants :

$$\begin{aligned} \mathbf{q}_i &= \mathbf{c}_i - \mathbf{c} \\ k_i &= \mathbf{m}_i - \mathbf{m} \end{aligned}$$

avec :

$$\mathbf{c} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{c}_i$$

$$\mathbf{m} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{m}_i$$

On obtient :

$$\begin{aligned} \mathbf{c}_i - \mathbf{R}^*\mathbf{m}_i - \mathbf{t}^* &= \mathbf{q}_i + \mathbf{c} - \mathbf{R}^*k_i - \mathbf{R}^*\mathbf{m} - \mathbf{t}^* \\ &= \mathbf{q}_i - \mathbf{R}^*k_i \end{aligned}$$

Finalement, dans le cas des points, la rotation et la translation optimales sont données par :

$$\min\left(\sum_{i=1}^{n}\|\mathbf{q}_i - \mathbf{R}\ k_i\|^2\right)$$

et par :

$$\mathbf{t}^* = \mathbf{c} - \mathbf{R}^*\mathbf{m}$$

7.6.2. La rotation optimale

En ce qui concerne la rotation, dans les trois cas étudiés (points, plans et droites), nous avons abouti au critère suivant :

$$\min_{\mathbf{R}} \left(\sum_{i=1}^{n} \|\mathbf{v}'_i - \mathbf{R}\ \mathbf{v}_i\|^2 \right) \qquad [7.18]$$

Dans cette équation $\mathbf{v}_i$ est un vecteur mis en correspondance avec le vecteur $\mathbf{v}'_i$. Il y a n telles correspondances et de plus les modules de $\mathbf{v}_i$ et de $\mathbf{v}'_i$ sont égaux pour tout i. La minimisation du critère de l'équation [7.18] dépend du choix d'une représentation pour la rotation. Nous avons étudié trois représentations possibles pour la rotation, nous avons donc trois possibilités :

- *matrice orthonormée.* Dans ce cas une rotation est décrite par 9 paramètres qui sont liés par les 6 contraintes d'orthonormalité. Un certain nombre de solutions a été proposé parmi lesquelles on peut citer [110], [5], [172].

- *angles d'Euler.* Certains auteurs ont proposé de trouver les angles d'Euler optimaux. Cette formulation conduit à un problème de minimisation non linéaire.

- *axe et angle.* Dans ce cas nous aurons à estimer 3 paramètres, soit les composantes du vecteur directeur de l'axe de rotation, la valeur de l'angle étant proportionnelle au module de ce vecteur. Il s'agit d'une représentation intéressante car minimale en ce qui concerne le nombre des paramètres à estimer. Nous allons décrire une solution ramenant le problème à la résolution d'un système linéaire surcontraint. Cette méthode s'apparente à la méthode de calibration caméra/pince proposée par Tsai [171].

- *quaternion unitaire.* La représentation par des quaternions unitaires est également intéressante car, d'une part, il n'y a que 4 paramètres à estimer et, d'autre part, le calcul se ramène à la détermination de la plus petite valeur propre d'une matrice symétrique de taille 4 [59], [53].

7.6.3. Axe et angle de rotation optimaux

Le critère de l'équation [7.18] signifie que nous cherchons une solution approchée d'un système de n équations où chaque équation du système est du type :

$$\mathbf{v}'_i = \mathbf{R}\ \mathbf{v}_i$$

Soit $\mathbf{n}$ le vecteur directeur de l'axe de rotation et utilisons le fait que la rotation conserve le produit scalaire. On obtient :

$$\mathbf{n} \cdot \mathbf{v}_i = (\mathbf{R}\ \mathbf{n}) \cdot (\mathbf{R}\ \mathbf{v}_i)$$

et si on se souvient que $\mathbf{n}$ est un vecteur propre de $\mathbf{R}$ associé à la valeur propre 1 on obtient :

$$\mathbf{n} \cdot \mathbf{v}_i = \mathbf{n} \cdot \mathbf{v}'_i$$

d'où on déduit la propriété de perpendicularité suivante :

$$\mathbf{n} \cdot (\mathbf{v}_i - \mathbf{v}'_i) = 0 \qquad [7.19]$$

Par ailleurs le module d'un vecteur est conservé par rotation. On a donc, pour tout i :

$$\|\mathbf{v}'_i\| = \|\mathbf{v}_i\|$$

On déduit facilement une deuxième propriété de perpendicularité :

$$(\mathbf{v}_i + \mathbf{v}'_i) \cdot (\mathbf{v}_i - \mathbf{v}'_i) = 0 \qquad [7.20]$$

Puisque le vecteur $\mathbf{v}_i - \mathbf{v}'_i$ est perpendiculaire aux vecteurs $\mathbf{n}$ et $\mathbf{v}_i + \mathbf{v}'_i$ il est donc proportionnel à leur produit vectoriel :

$$(\mathbf{v}_i + \mathbf{v}'_i) \wedge \mathbf{n} = k(\mathbf{v}_i - \mathbf{v}'_i) \qquad [7.21]$$

Afin de déterminer la valeur de k nous allons, comme lorsque nous avons démontré la formule de Rodrigues, décomposer les vecteurs v_i et $\mathbf{v}'_i$ selon deux directions, une direction colinéaire à l'axe de rotation $\mathbf{n}$ et une autre direction orthogonale à cet axe de rotation, figure 7.4. :

$$\begin{aligned} \mathbf{v}_i &= \mathbf{v}_{i\perp} + \mathbf{v}_{i\parallel} \\ \mathbf{v}'_i &= \mathbf{v}'_{i\perp} + \mathbf{v}'_{i\parallel} \end{aligned}$$

En substituant ces formules dans l'équation [7.21] on obtient :

$$(\mathbf{v}'_{i\perp} + \mathbf{v}'_{i\parallel} + \mathbf{v}_{i\perp} + v_{i\parallel}) \wedge \mathbf{n} = k(\mathbf{v}_{i\perp} + \mathbf{v}_{i\parallel} - \mathbf{v}'_{i\perp} - \mathbf{v}'_{i\parallel})$$

En développant et en remarquant que $\mathbf{v}'_{i\parallel} = v_{i\parallel}$ on obtient :

$$(\mathbf{v}_{i\perp} + \mathbf{v}'_{i\perp}) \wedge \mathbf{n} = k(\mathbf{v}_{i\perp} - \mathbf{v}'_{i\perp})$$

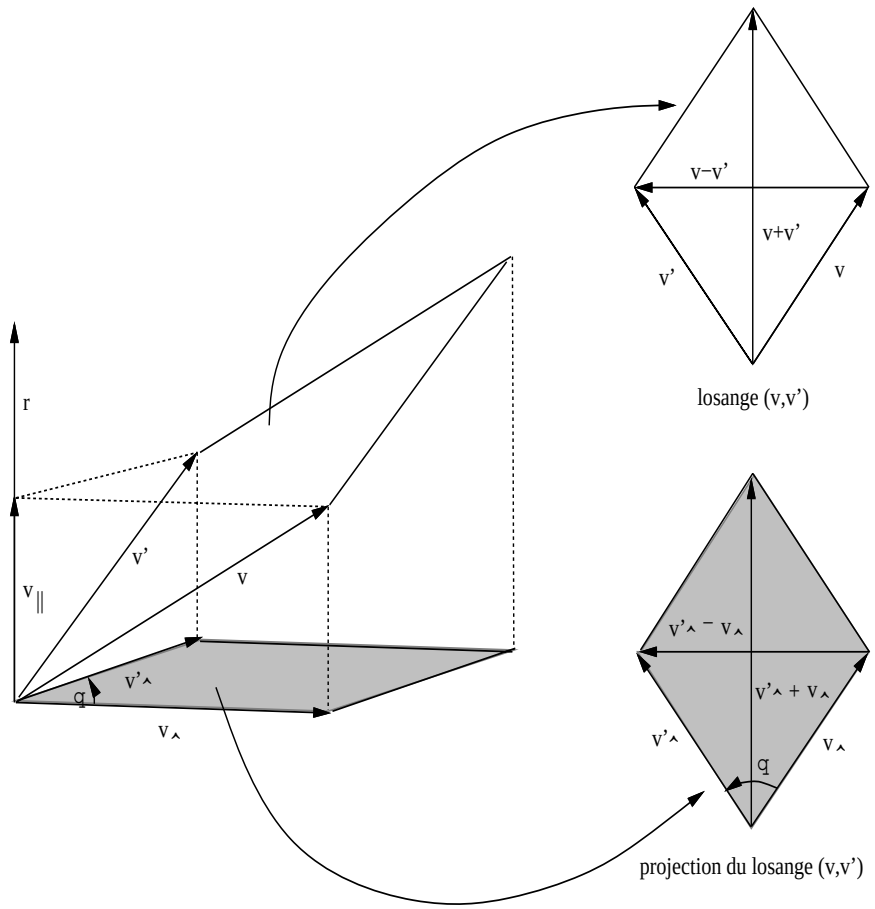


Figure 7.4. *Calcul de la rotation optimale utilisant l'axe et l'angle de rotation.*

On peut remarquer que les vecteurs $\mathbf{v}_{i\perp} + \mathbf{v}'_{i\perp}$ et $\mathbf{v}_{i\perp} - \mathbf{v}'_{i\perp}$ sont orthogonaux et par ailleurs ils sont tous les deux orthogonaux au vecteur unitaire $\mathbf{n}$. On obtient donc k comme le rapport suivant :

$$k = \frac{\|\mathbf{v}_{i\perp} + \mathbf{v}'_{i\perp}\|}{\|\mathbf{v}_{i\perp} - \mathbf{v}'_{i\perp}\|}$$

Or, l'angle entre les vecteurs $\mathbf{v}_{i\perp}$ et $\mathbf{v}'_{i\perp}$ est l'angle de la rotation que l'on cherche, soit θ. D'après la figure 7.4. on remarque qu'il s'agit d'un des angles d'un losange qui a comme diagonales $\mathbf{v}_{i\perp} + \mathbf{v}'_{i\perp}$ et $\mathbf{v}_{i\perp} - \mathbf{v}'_{i\perp}$. La relation entre cet angle et le rapport des longueurs de ces diagonales est :

$$k = \frac{1}{\tan(\theta/2)} \qquad [7.22]$$

En posant :

$$\mathbf{N} = \tan\frac{\theta}{2}\ \mathbf{n}$$

on obtient un système de trois équations et trois inconnues pour chaque i :

$$S(\mathbf{v}_i + \mathbf{v}'_i)\mathbf{N} = \mathbf{v}'_i - \mathbf{v}_i \qquad [7.23]$$

Le rang de la matrice antisymetrique $S()$ est 2 et chaque mise en correspondance i fournit donc 2 équations et non pas 3 équations. Il faut donc au moins 2 mises en correspondance pour trouver un axe de rotation. En général on dispose de n mises en correspondance. On obtient alors un système de $2n$ équations à 3 inconnues. Si on considère les deux premières équations du système d'équations [7.23] on obtiendra :

$$\begin{pmatrix} \vdots & \vdots & \vdots \\ 0 & -(\mathbf{v}_{zi} + \mathbf{v}'_{zi}) & \mathbf{v}_{yi} + \mathbf{v}'_{yi} \\ \mathbf{v}_{zi} + \mathbf{v}'_{zi} & 0 & -(\mathbf{v}_{xi} + \mathbf{v}'_{xi}) \\ \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} \mathbf{N}_x \\ \mathbf{N}_y \\ \mathbf{N}_z \end{pmatrix} = \begin{pmatrix} \vdots \\ \mathbf{v}'_{xi} - \mathbf{v}_{xi} \\ \mathbf{v}'_{yi} - \mathbf{v}_{yi} \\ \vdots \end{pmatrix}$$

La solution optimale pour $\mathbf{N}$ est obtenue en résolvant ce système d'équations surcontraint. On obtient finalement le vecteur unitaire de l'axe de rotation et l'angle de rotation :

$$\begin{aligned} \mathbf{n} &= \frac{\mathbf{N}}{\|\mathbf{N}\|} \\ \theta &= 2\arctan(\|\mathbf{N}\|) \end{aligned}$$

et grâce à la formule de Rodrigues on obtient facilement la matrice de rotation à partir du vecteur unitaire $\mathbf{n}$ et de l'angle θ.

La solution que nous avons proposée ici est particulièrement simple car la rotation est représentée par un vecteur dont le module est proportionnel à l'angle de rotation. On a donc une représentation minimale particulièrement intéressante pour la recherche d'une rotation optimale.

7.6.4. Quaternion unitaire optimal

Nous rappelons qu'un quaternion unitaire décrivant une rotation d'axe $\mathbf{n}$ et d'angle θ s'écrit sous la forme (équation [7.17]) :

$$\mathbf{q} = \cos\frac{\theta}{2} + \sin\frac{\theta}{2}\,(in_x + jn_y + kn_z)$$

et que la rotation d'un vecteur $\mathbf{v}$ s'écrit alors :

$$\begin{aligned} \mathbf{v}' &= \mathbf{R}\ \mathbf{v} \\ &= \mathbf{q} * \mathbf{v} * \overline{\mathbf{q}} \end{aligned}$$

Le critère de l'équation [7.18] devient :

$$\begin{gathered} \min_{\mathbf{q}} \left(\sum_{i=1}^{n} \|\mathbf{v}'_i - \mathbf{q} * \mathbf{v}_i * \overline{\mathbf{q}}\|^2\right) \\ \text{avec } \|\mathbf{q}\|^2 = 1 \end{gathered}$$

Compte tenu de l'équation [7.15] et des propriétés qui en découlent, on a successivement :

$$\begin{aligned} \|\mathbf{v}'_i - \mathbf{q} * \mathbf{v}_i * \overline{\mathbf{q}}\|^2 &= \|\mathbf{v}'_i - \mathbf{q} * \mathbf{v}_i * \overline{\mathbf{q}}\|^2 \|\mathbf{q}\|^2 \\ &= \|\mathbf{v}'_i * \mathbf{q} - \mathbf{q} * \mathbf{v}_i * \overline{\mathbf{q}} * \mathbf{q}\|^2 \\ &= \|\mathbf{v}'_i * \mathbf{q} - \mathbf{q} * \mathbf{v}_i\|^2 \end{aligned}$$

Le produit de deux quaternions peut s'écrire sous forme matricielle. On a :

$$\begin{aligned} \mathbf{v}'_i * \mathbf{q} &= Q(\mathbf{v}'_i)\mathbf{q} \\ \mathbf{q} * \mathbf{v}_i &= W(\mathbf{v}_i)\mathbf{q} \end{aligned}$$

ce qui donne :

$$\begin{aligned} \|\mathbf{v}'_i * \mathbf{q} - \mathbf{q} * \mathbf{v}_i\|^2 &= \left((Q(\mathbf{v}'_i) - W(\mathbf{v}_i))\mathbf{q}\right)^t \left((Q(\mathbf{v}'_i) - W(\mathbf{v}_i))\mathbf{q}\right) \\ &= \mathbf{q}^t A_i \mathbf{q} \end{aligned}$$

où A_i est la matrice symétrique de dimension 4 :

$$A_i = (Q(\mathbf{v}'_i) - W(\mathbf{v}_i))^t (Q(\mathbf{v}'_i) - W(\mathbf{v}_i))$$

Le critère de l'équation [7.18] devient donc :

$$\begin{aligned} \min_{\mathbf{q}} \left(\sum_{i=1}^{n} \mathbf{q}^t A_i \mathbf{q} \right) &= \min_{\mathbf{q}} \left(\mathbf{q}^t (\sum_{i=1}^{n} A_i) \mathbf{q} \right) \\ &= \min_{\mathbf{q}} \left(\mathbf{q}^t B \mathbf{q} \right) \end{aligned}$$

avec :

$$B = \sum_{i=1}^{n} A_i = \sum_{i=1}^{n} (Q(\mathbf{v}'_i) - W(\mathbf{v}_i))^t (Q(\mathbf{v}'_i) - W(\mathbf{v}_i))$$

étant une matrice symétrique positive. Sous la contrainte que le quaternion doit être unitaire, on obtient finalement le critère suivant :

$$\min_{\mathbf{q}} Q = \min_{\mathbf{q}} \left(\mathbf{q}^t B \mathbf{q} + \lambda (1 - \mathbf{q}^t \mathbf{q}) \right) \qquad [7.24]$$

En dérivant Q par rapport à $\mathbf{q}$on obtient :

$$B\mathbf{q} - \lambda \mathbf{q} = 0$$

et en substituant cette solution dans l'équation [7.24] on obtient :

$$Q = \lambda$$

Le quaternion unitaire qui minimise Q est donc le vecteur propre unitaire de la matrice B associé à la plus petite valeur propre de B, soit λ. Nous rappelons qu'une matrice symétrique a des valeurs propres réelles et une matrice symétrique positive a toutes ses valeurs propres positives. Soient alors les vecteurs propres de B qui forment une base orthogonale, $\mathbf{w}_1$, $\mathbf{w}_2$, $\mathbf{w}_3$ et $\mathbf{w}_4$. Si ces vecteurs sont unitaires, la base est orthonormée. Le quaternion $\mathbf{q}$ peut s'écrire dans cette base :

$$\mathbf{q} = \mu_1 \mathbf{w}_1 + \mu_2 \mathbf{w}_2 + \mu_3 \mathbf{w}_3 + \mu_4 \mathbf{w}_4$$

Pour tout i on a :

$$B\mathbf{w}_i = \lambda_i \mathbf{w}_i$$

λ_1, λ_2, λ_3 et λ_4 étant les valeurs propres de B avec :

$$\lambda_1 < \lambda_2 < \lambda_3 < \lambda_4$$

En tenant compte du fait que les vecteurs propres forment une base orthonormée, on peut écrire :

$$\mathbf{q}^t B\mathbf{q} = \mu_1^2\lambda_1 + \mu_2^2\lambda_2 + \mu_3^2\lambda_3 + \mu_4^2\lambda_4$$

Cette expression est minimale pour $\mu_1 = 1$ et $\mu_2 = \mu_3 = \mu_4 = 0$. On obtient donc :

$$\mathbf{q} = \mathbf{w}_1$$

et

$$\mathbf{q}^t B\mathbf{q} = \lambda_1$$

Le quaternion unitaire optimal est donc le vecteur propre unitaire de B associé à sa plus petite valeur propre.

7.6.5. La translation optimale

Une fois que nous avons estimé la rotation optimale $\mathbf{R}^*$, le calcul de la translation optimale est possible grâce à l'un des critères suivants :

$$\min_{\mathbf{t}} \left(\sum_{i=1}^{n} \| d'_i - d_i - (\mathbf{R}^* \mathbf{v}_i) \cdot \mathbf{t} \|^2 \right)$$

pour le cas des plans, ou :

$$\min_{\mathbf{t}} \left(\sum_{i=1}^{n} \| \mathbf{p}'_i - \mathbf{R}^* \mathbf{p}_i - \mathbf{t} + ((\mathbf{R}^* \mathbf{v}_i) \cdot \mathbf{t})\mathbf{R}^* \mathbf{v}_i \|^2 \right)$$

pour le cas des droites.

Dans ces deux cas, en dérivant par rapport aux trois composantes de $\mathbf{t}$ et en annulant ces dérivées, on obtient un système de trois équations linéaires en trois inconnues qui se résout sans aucune difficulté. Le cas des points est lui encore plus simple car il suffit de calculer le vecteur reliant les barycentres des deux ensembles de points :

$$\mathbf{t}^* = \mathbf{c} - \mathbf{R}^* \mathbf{m}$$

Notons cependant que, dans ce dernier cas, la solution obtenue pour la translation n'est pas très fiable en présence de bruit. Une alternative intéressante a été récemment proposée par Walker et al [173] : la rotation et la translation sont représentées par un quaternion dual. On peut alors estimer simultanément et d'une façon optimale les paramètres de rotation et translation.

Chapitre 8

Localisation caméra/objet

Dans ce chapitre on s'intéresse au problème de la détermination de la position et l'orientation d'un objet par rapport à une caméra. Ce problème est équivalent au problème de la détermination des paramètres extrinsèques d'une caméra lorsque les paramètres intrinsèques sont connus. Cependant il diffère du problème de calibrage car on n'a pas toujours à sa disposition un grand nombre de points de correspondance. En effet, un grand nombre de points de correspondance permet une approche linéaire qui est bien conditionnée. Lorsqu'il y a peu de points de correspondance l'approche linéaire est mal conditionnée. Nous allons donc introduire une technique non linéaire d'estimation des paramètres recherchés.

8.1. Introduction

Plus formellement, le problème est le suivant : étant donné un ensemble de points ou de droites 3D décrits dans un référentiel "objet" et leurs projections 2D décrites dans un référentiel "caméra" connaissant les paramètres de la caméra, il faut alors déterminer la transformation rigide (rotation et translation) entre le référentiel objet et le référentiel caméra.

Les solutions proposées pour résoudre ce problème peuvent se grouper en 2 catégories :

1. *solutions analytiques.* Lorsque le nombre de points de correspondance est petit (entre 3 et 6 correspondances) on peut trouver une solution analytique qui consiste essentiellement à résoudre un système d'équations non linéaires. Cependant, pour 3 points, il peut y avoir jusqu'à 4 solutions [62]. Des solutions

existent également pour 4 points coplanaires [101] ou pour 4 points non coplanaires [90], [86]. On a également proposé des solutions pour 3 droites [48], [33], ce qui est équivalent à 3, 4, 5 ou 6 points comme il est montré sur la figure 8.2.

2. *solutions numériques.* Lorsque le nombre de points de correspondance est grand, les solutions analytiques ne sont plus efficaces, car on doit résoudre un système d'équations non linéaires où le nombre d'équations est grand par rapport au nombre d'inconnues. Dans ce cas, des solutions numériques sont nécessaires. Nous allons discuter de quelques solutions proposées.

Comme nous allons le voir, le problème étant non linéaire, les choix relatifs quant à la formulation mathématique, quant à l'expression de la fonction d'erreur à minimiser et quant à la méthode de minimisation, sont cruciaux.

Ganapathy [66] présente une solution linéaire, la rotation étant représentée par une matrice 3×3, sans exprimer explicitement les contraintes d'orthonormalité de cette matrice. Cette méthode est extrêmement sensible au bruit.

Yuan [180] a proposé de séparer la rotation de la translation et il a concentré ses efforts sur la détermination de la rotation. Les six contraintes d'orthonormalité de la matrice de rotation donnent naissance à 6 contraintes quadratiques. La solution commune à ces 6 contraintes est trouvée utilisant la méthode de Newton (descente de gradient). L'auteur a remarqué que des minima locaux peuvent être trouvés avec cette méthode. Plusieurs minima locaux correspondent à la nature non linéaire du problème. Le minimum global peut être atteint dans ce cas, seulement si on fournit une initialisation proche de la solution. Lowe [114] a également utilisé la méthode de Newton. La fonction d'erreur à minimiser est la somme des carrés des distances entre chaque projection d'un point (ou d'une droite) de l'objet et son correspondant dans l'image. Comme dans le cas de la méthode de Yuan, Lowe a rencontré des problèmes avec cette méthode et dans un autre article, il a suggéré des modifications de la méthode pour résoudre les problèmes d'initialisation et de stabilité [115]. La méthode de stabilisation qu'il a suggérée n'est efficace qu'à condition que certains des paramètres soient correctement initialisés.

Liu et al. [111] ont examiné une méthode itérative, la rotation étant représentée par les angles d'Euler. D'abord, ils constatent que, si on met en correspondance des droites et non pas des points, la rotation est séparée de la translation et qu'une fois que l'on a déterminé la rotation, la détermination de la translation devient un problème linéaire. Cependant les auteurs consta-

tent que leur méthode n'est efficace que lorsque les angles de la rotation sont inférieurs à 30^0.

Utilisant la même représentation que Liu et al., Kumar et Hanson [108] ont examiné deux méthodes de minimisation : une technique itérative qui linéarise la fonction d'erreur et une technique combinatoire. Cette dernière technique permet, en principe, d'éliminer des mesures aberrantes mais elle est très lente, car elle considère tous les triplets possibles de correspondance.

Dans ce qui suit nous allons considérer des correspondances de droites et de points. Utilisant les quaternions unitaires pour décrire les rotations, les contraintes du problème deviennent quadratiques. La fonction d'erreur sera la somme des carrés de ces contraintes quadratiques. Nous allons ensuite introduire une méthode d'optimisation non linéaire dite de *région de confiance* [179], [150] ainsi qu'une méthode linéaire utilisant un modèle rapproché de caméra et un algorithme itératif. En effet, toute méthode de minimisation non-linéaire nécessite une initialisation de la solution. Plus cette solution initiale est proche de la solution recherchée, plus rapide et plus sure sera la convergence de l'algorithme de minimisation non linéaire.

8.2. Correspondances de droites

Nous considérons le modèle géométrique de caméra tel qu'il a été développé au chapitre 5. Un point m de l'image, qui a u et v comme coordonnées dans le repère image, aura x_c et y_c comme coordonnées caméra :

$$\begin{aligned} x_c &= (u - u_0)/\alpha_u \\ y_c &= (v - v_0)/\alpha_v \end{aligned}$$

où u_0, v_0, α_u, et α_v sont les paramètres intrinsèques de la caméra. Soient x, y et z les coordonnées dans le repère caméra d'un point 3D M se trouvant sur un objet et se projetant dans l'image en m. On a :

$$\left\{ \begin{array}{l} x_c = \frac{x}{z} \\ y_c = \frac{y}{z} \end{array} \right. \qquad [8.1]$$

De plus, nous allons contraindre le point m d'appartenir à une droite dans l'image dont l'équation s'écrit :

$$a x_c + b y_c + c = 0 \qquad [8.2]$$

En substituant l'équation [8.1] dans l'équation [8.2] on obtient :

$$ax + by + cz = \mathbf{n} \cdot \overrightarrow{FM} = 0 \qquad [8.3]$$

ceci n'est autre que l'équation du plan contenant le centre de projection, la droite de l'image et le point M ; $\mathbf{n}$ est le vecteur normal à ce plan, comme sur la figure 8.1.

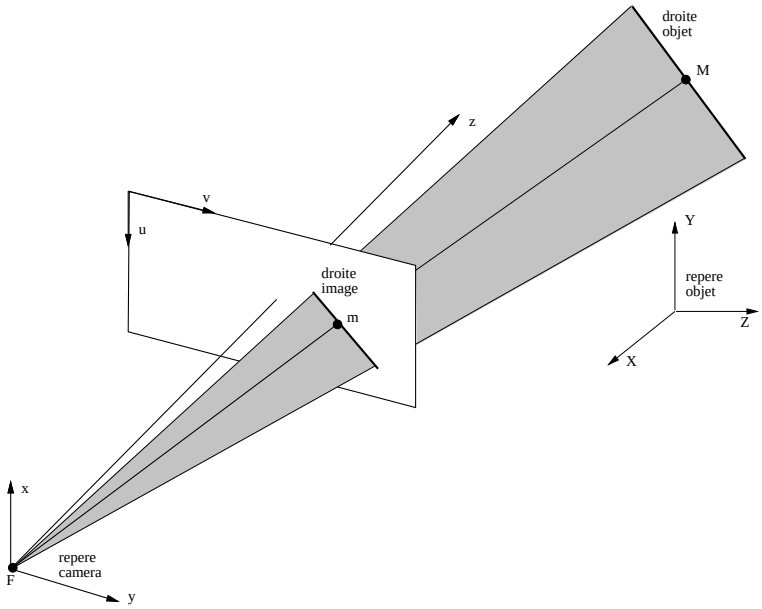


Figure 8.1. *La droite "objet", sa projection dans l'image et le centre de projection F sont coplanaires et ce plan est montré en gris. $\mathbf{n}$ est le vecteur unitaire orthogonal à ce plan.*

Considérons maintenant la droite objet décrite par le point M. Dans le référentiel de l'objet, cette droite peut être décrite par sa représentation paramétrique. Soient $\mathbf{v}$ son vecteur directeur (unitaire) et $\mathbf{p}$ son vecteur position. Cette droite peut également être exprimée dans le repère de la caméra :

$$\overrightarrow{FM} = \mathbf{p}' + \mu\mathbf{v}'$$

La relation entre les paramètres de la droite dans deux repères différents étant :

$$\begin{aligned} \mathbf{p}' &= \mathbf{R}\ \mathbf{p} + \mathbf{t} \\ \mathbf{v}' &= \mathbf{R}\ \mathbf{v} \end{aligned}$$

où $\mathbf{R}$ et $\mathbf{t}$ décrivent la rotation et la translation entre le référentiel objet et le référentiel caméra et contiennent justement les paramètres extrinsèques que l'on se propose de déterminer. La contrainte de correspondance exprime le fait que la droite objet appartient au plan défini auparavant, équation [8.3] :

$$\begin{cases} \mathbf{n} \cdot \mathbf{v}' = 0 \\ \mathbf{n} \cdot \mathbf{p}' = 0 \end{cases}$$

ce qui peut s'écrire :

$$\begin{cases} \mathbf{n} \cdot (\mathbf{R}\ \mathbf{v}) = 0 \\ \mathbf{n} \cdot (\mathbf{R}\ \mathbf{p} + \mathbf{t}) = 0 \end{cases} \qquad [8.4]$$

Par conséquent, chaque correspondance droite-objet/droite-image fournit 2 contraintes. Puisque le nombre de paramètres inconnus qu'il faut déterminer est 6 (3 pour la rotation et 3 pour la translation), le problème peut être résolu si au moins 3 correspondances sont disponibles. En définitive, 3 droites peuvent être construites grâce à 3, 4, 5, ou 6 points, comme sur la figure 8.2. La solution que nous présentons dans ce chapitre est donc valable pour des correspondances de droites et de points. Il faut avoir au minimum 3 correspondances point/point ou 3 correspondances droite/droite. Dans le cas général, lorsque N correspondances droite/droite sont disponibles, le problème est de résoudre $2N$ contraintes non linéaires, ce qui est équivalent au problème de minimiser la fonction d'erreur suivante :

$$f(\mathbf{R}\ , t) = \sum_{i=1}^{N} (\mathbf{n}_i \cdot (\mathbf{R}\ \mathbf{v}_i))^2 + \sum_{i=1}^{N} (\mathbf{n}_i \cdot (\mathbf{R}\ \mathbf{p}_i + \mathbf{t}))^2 \qquad [8.5]$$

8.3. La fonction d'erreur

Grâce à la représentation des rotations par des quaternions unitaires (section 7.5.6.), nous pouvons transformer la fonction d'erreur que l'on vient d'établir en une somme de carrés de contraintes quadratiques. Une approche analogue utilisant des quaternions duaux pour représenter les rotations et translations est proposée dans [150]. Rappelons que les vecteurs de dimension 3 vont être traités comme des quaternions purement imaginaires. La première contrainte de l'équation [8.4] peut s'écrire de la manière suivante :

$$\begin{aligned} \mathbf{n} \cdot (\mathbf{R}\ \mathbf{v}) &= \mathbf{n}^t (W(\mathbf{r})^t Q(\mathbf{r}) \mathbf{v}) \\ &= \mathbf{r}^t Q(\mathbf{n})^t W(\mathbf{v}) \mathbf{r} \end{aligned} \qquad [8.6]$$

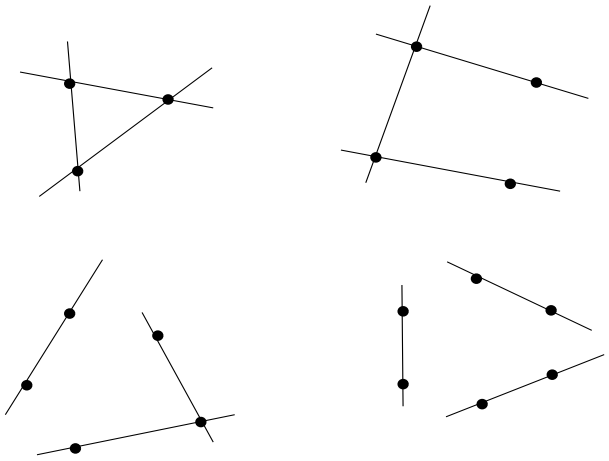


Figure 8.2. *Configurations de 3, 4, 5 ou 6 points qui peuvent être remplacées par 3 droites.*

où $\mathbf{r}$ est le quaternion unitaire associé à la matrice de rotation $\mathbf{R}$. La seconde contrainte de l'équation [8.4] devient :

$$\begin{aligned} \mathbf{n} \cdot (\mathbf{R}\ \mathbf{p} + \mathbf{t}) &= \mathbf{n}^t (W(\mathbf{r})^t Q(\mathbf{r})\mathbf{p} + \mathbf{t}) \\ &= \mathbf{r}^t Q(\mathbf{n})^t W(\mathbf{p})\mathbf{r} + \mathbf{n}^t \mathbf{t} \end{aligned} \qquad [8.7]$$

En d'autres termes, chaque correspondance droite/droite i fournit 2 contraintes quadratiques :

$$\mathbf{r}^t A_i \mathbf{r} = 0$$

et :

$$\mathbf{r}^t B_i \mathbf{r} + \mathbf{n}_i^t \mathbf{n} = 0$$

avec la contrainte $\mathbf{r}^t\mathbf{r} = 1$ et A_i, B_i étant 2 matrices 4×4 définies par :

$$\begin{aligned} A_i &= Q(\mathbf{n}_i)^t W(\mathbf{v}_i) \\ B_i &= Q(\mathbf{n}_i)^t W(\mathbf{p}_i) \end{aligned}$$

On peut maintenant écrire une nouvelle expression pour la fonction d'erreur associée à notre problème, soit l'équation [8.5] :

$$f(\mathbf{r},\ \mathbf{t}) = \sum_{i=1}^{N} ((\mathbf{r}^t A_i \mathbf{r})^2 + (\mathbf{r}^t B_i \mathbf{r} + \mathbf{n}_i^t \mathbf{t})^2) + \lambda(\mathbf{r}^t\mathbf{r} - 1)^2 \qquad [8.8]$$

où les paramètres à estimer sont les éléments du quaternion unitaire $\mathbf{r}$ et du vecteur de translation $\mathbf{t}$. Le dernier terme de la fonction d'erreur est un terme de pénalisation avec $\lambda > 0$. Plus λ sera grand et mieux on garantira que le quaternion sera unitaire. En pratique, la valeur de λ peut être comprise entre 50 et 100. Une valeur trop grande de λ peut considérablement ralentir la vitesse de convergence de l'algorithme d'optimisation.

On peut noter qu'une alternative consisterait à estimer la rotation et la translation séparément. On peut estimer tout d'abord la rotation en utilisant la fonction d'erreur suivante :

$$f(\mathbf{r}) = \sum_{i=1}^{N} (\mathbf{r}^t A_i \mathbf{r})^2 + \lambda (\mathbf{r}^t \mathbf{r} - 1)^2 \qquad [8.9]$$

Une fois que l'on a estimé la rotation optimale, le calcul de la translation optimale est un problème linéaire.

8.4. Correspondance de points

La fonction d'erreur que nous venons d'établir est valable pour des correspondances de droites ET pour des correspondances de points puisque un ensemble de 2 contraintes quadratiques peuvent être associées soit avec une droite soit avec 2 points. Dans cette section nous allons établir une fonction d'erreur directement à partir de correspondances de points : 2 contraintes pour chaque correspondance point/point. Comme dans le cas précédent, ces contraintes sont quadratiques et elles décrivent la colinéarité entre le point 3D, sa projection 2D et le centre de projection du modèle de la caméra.

Comme auparavant, on considère un point M_i de l'objet ayant comme coordonnées X_i, Y_i et Z_i dans un repère lié à l'objet, et soient x_i et y_i les coordonnées dans le repère de la caméra de sa projection m_i. La colinéarité entre M_i, m_i, et F (le centre de projection) s'exprime avec les deux équations suivantes :

$$x_i = \frac{r_{11}X_i + r_{12}Y_i + r_{13}Z_i + t_1}{r_{31}X_i + r_{32}Y_i + r_{33}Z_i + t_3} \qquad [8.10]$$

$$y_i = \frac{r_{21}X_i + r_{22}Y_i + r_{23}Z_i + t_2}{r_{31}X_i + r_{32}Y_i + r_{33}Z_i + t_3} \qquad [8.11]$$

Dans ces équations r_{ij} est un éléement de la matrice de rotation $\mathbf{R}$ et t_i est une composante du vecteur de translation $\mathbf{t}$. Ces équations peuvent également

s'écrire sous la forme matricielle suivante :

$$\begin{pmatrix} 1 & 0 & -x_i \end{pmatrix} \mathbf{R} \begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} + \begin{pmatrix} 1 & 0 & -x_i \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} = 0 \quad [8.12]$$

$$\begin{pmatrix} 0 & 1 & -y_i \end{pmatrix} \mathbf{R} \begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} + \begin{pmatrix} 0 & 1 & -y_i \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} = 0 \quad [8.13]$$

Soit $\mathbf{M}_i$ le quaternion purement imaginaire associé au vecteur point M_i et soient $\mathbf{m}_{x_i}$ et $\mathbf{m}_{y_i}$ deux quaternions purement imaginaires, chacun étant associé avec une "moitié" du vecteur point m_i :

$$\begin{aligned} \mathbf{M}_i &= (0\ X_i\ Y_i\ Z_i)^t \\ \mathbf{m}_{x_i} &= (0\ 1\ 0\ -x_i)^t \\ \mathbf{m}_{y_i} &= (0\ 0\ 1\ -y_i)^t \end{aligned}$$

Si on remplace la matrice de rotation $\mathbf{R}$ par sa représentation en terme d'un quaternion unitaire (voir la section 7.5.6.), on obtient pour la première contrainte :

$$\mathbf{m}_{x_i}^t W(\mathbf{r})^t Q(\mathbf{r}) \mathbf{M}_i + \mathbf{m}_{x_i}^t \mathbf{t} = \mathbf{r}^t Q(\mathbf{m}_{x_i})^t W(\mathbf{M}_i) \mathbf{r} + \mathbf{m}_{x_i}^t \mathbf{t}$$

Il y a une expression similaire pour la deuxième équation :

$$\mathbf{m}_{y_i}^t W(\mathbf{r})^t Q(\mathbf{r}) \mathbf{M}_i + \mathbf{m}_{y_i}^t \mathbf{t} = \mathbf{r}^t Q(\mathbf{m}_{y_i})^t W(\mathbf{M}_i) \mathbf{r} + \mathbf{m}_{y_i}^t \mathbf{t}$$

Finalement, les équations [8.10] et [8.11] deviennent :

$$\mathbf{r}^t C_i \mathbf{r} + \mathbf{m}_{x_i}^t \mathbf{t} = 0 \quad [8.14]$$

$$\mathbf{r}^t D_i \mathbf{r} + \mathbf{m}_{y_i}^t \mathbf{t} = 0 \quad [8.15]$$

Dans ces équations les matrices 4×4 C_i et D_i sont données par :

$$C_i = Q(\mathbf{m}_{x_i})^t W(\mathbf{M}_i)$$

$$D_i = Q(\mathbf{m}_{y_i})^t W(\mathbf{M}_i)$$

La fonction d'erreur dans le cas des points s'écrit comme la somme au carré de ces fonctions quadratiques sous la contrainte de quaternion unitaire pour $\mathbf{r}$:

$$f(\mathbf{r}, \mathbf{t}) = \sum_{i=1}^{N} \left((\mathbf{r}^t C_i \mathbf{r} + \mathbf{m}_{x_i}^t \mathbf{t})^2 + (\mathbf{r}^t D_i \mathbf{r} + \mathbf{m}_{y_i}^t \mathbf{t})^2 \right) + \lambda (\mathbf{r}^T \mathbf{r} - 1)^2 \quad [8.16]$$

8.5. Optimisation : la méthode de région de confiance

La minimisation de la fonction d'erreur décrite par les équations [8.8] et [8.9] est équivalente à celle du critère quadratique non linéaire suivant :

$$0 = \min\{f(x) = \frac{1}{2}\sum_{j=1}^{m} \Phi_j^2(x) : x \in I\!R^n\} \qquad [8.17]$$

Dans cette expression les $\Phi_j(x)$ sont des fonctions continues et deux fois dérivables de $I\!R^n$ dans $I\!R$. Dans notre cas x et soit de dimension 4 ($\mathbf{r}$) soit de dimension 7 ($\mathbf{r}$ et $\mathbf{t}$). On rappelle, par ailleurs, que le gradient $\nabla f(x)$ et le hessien $\nabla^2 f(x)$ peuvent être calculés comme suit :

$$g(x) = \nabla f(x) = \sum_{j=1}^{m} \Phi_j(x)\nabla\Phi_j(x) = J(x)^t\Phi(x), \qquad [8.18]$$

$$\begin{aligned} H(x) = \nabla^2 f(x) &= \sum_{j=1}^{m} \nabla\Phi_j(x)\nabla\Phi_j(x)^t + \sum_{j=1}^{m} \Phi_j(x)\nabla^2\Phi_j(x) \\ &= J(x)^t J(x) + \sum_{j=1}^{m} \Phi_j(x)\nabla^2\Phi_j(x) \qquad [8.19] \end{aligned}$$

où $\Phi(x) = (\Phi_1(x), \ldots, \Phi_m(x))^t$ et $J(x) = (\nabla\Phi_1(x), \ldots, \nabla\Phi_m(x))^t$ est la matrice jacobienne $m \times n$ de $\Phi(x)$. On peut facilement voir que $\Phi(x)$ est deux fois continue et différentiable de $I\!R^n$ dans $I\!R^m$ et que $J(x)$ est une matrice non singulière. Le hessien d'une telle fonction quadratique est une combinaison de termes du premier et deuxième ordre :

$$H(x) = J(x)^t J(x) + Q(x)$$

avec : $Q(x) = \sum_{j=1}^{m} \Phi_j(x)\nabla^2\Phi_j(x)$. En pratique l'approximation Gauss-Newton du hessien est utilisée, soit $H(x) = J(x)^t J(x)$. Cela est basé sur l'hypothèse que le terme du premier ordre va finalement dominer le terme du deuxième ordre $Q(x)$ [70].

On désigne par x_k l'estimation courante de la solution. Une variable avec l'indice k désigne cette variable évaluée à la k^e itération de l'algorithme. L'idée de base de la méthode de région de confiance est d'approcher successivement la fonction d'erreur par une *forme quadratique locale* dans un voisinage de la solution courante x_k :

$$f(x_k + d) \approx f(x_k) + q_k(d), \quad \|d\| \leq \delta_k$$

où :

$$q_k(d) = g(x_k)^t d + \frac{1}{2} d^t H(x_k) d \qquad [8.20]$$

La fonction d'erreur sera réduite selon la direction d_k, soit $x_{k+1} = x_k + d_k$, où d_k est la valeur de d minimisant la forme quadratique locale sur une région sphérique centrée sur x_k : cette sphère s'appelle la *région de confiance* et on a :

$$d_k = \min\{q_k(d) : \|d\| \leq \delta_k\} \qquad [8.21]$$

Le paramètre δ_k est le rayon de confiance et il est déterminé dynamiquement en utilisant une mesure de la qualité de l'approximation. Cette qualité est mesurée par un coefficient de qualité r_k :

$$r_k = \frac{f(x_k) - f(x_k + d_k)}{q_k(0) - q_k(d_k)} \qquad [8.22]$$

mesurant le rapport entre la *vraie réduction* de la fonction d'erreur lorsque l'on se déplace de x_k à $x_k + d_k$ et la *réduction prédite* par l'approximation quadratique. Si r_k est trop petit, cela veut dire que l'approximation n'est pas bonne et la taille de la région de confiance doit être réduite. Sinon, la région de confiance doit être augmentée en taille. Schématiquement, la stratégie suivante sera appliquée :

- si $r_k \geq r$ (avec $0 < r \leq 1$), alors l'itération doit être acceptée et $x_{k+1} = x_k + d_k$. De plus, la taille de la région de confiance est augmentée.
- si $r_k < r$, alors l'itération n'est pas acceptée et la taille de la région de confiance doit être diminuée. Ce processus est répété jusqu'à ce que la forme quadratique locale fournisse une approximation satisfaisante à l'intérieur de la région de confiance – ce qui va nécessairement se passer pour des rayons petits, car le gradient est connu et le terme du premier ordre de q_k devient dominant lorsque $\|q_k\| \neq 0$.

La forme quadratique locale dépend du gradient et du hessien de la fonction d'erreur. Ceci implique que le minimum ainsi trouvé a de "bonnes" propriétés du second ordre. La majeure difficulté de la méthode de région de confiance réside dans la minimisation de la forme quadratique locale. Les quelques versions de la méthode de région de confiance diffèrent par la méthode qui est utilisée pour minimiser la forme quadratique locale. La méthode que nous avons choisi d'utiliser est décrite en détail dans la section suivante.

Nous pouvons comprendre la différence entre les méthodes classiques de minimisation non linéaire et la méthode de région de confiance grâce à un

exemple de minimisation d'une fonction $f(x)$ à une variable. Une telle fonction est illustrée sur la figure 8.3. A partir d'un point initial x^0, une méthode du type Newton approxime la fonction avec une parabole utilisant un développement limité au deuxième ordre autour de x^0. Le point suivant, x^1 correspond au minimum de cette parabole mais cette valeur augmente la fonction $f(x)$. L'idée de la méthode de région de confiance est d'approximer la fonction avec une parabole également et de chercher le minimum sur cette parabole dans une zone restreinte plutôt que sur tout le domaine de définition de la fonction à minimiser. La figure 8.4. montre le comportement d'une telle méthode sur la même fonction. A partir du point x^0 le minimum de la parabole est recherché sur l'intervalle de taille $2r_0$. Le minimum est dans ce cas une des extrémités de la parabole : $x^1 = x^0 - r_0$. Autour de x^1 la fonction est à nouveau approximée par une parabole mais, cette fois-ci, sur un intervalle 2 fois plus large : $r_1 = 2r_0$. Le minimum de cette nouvelle parabole est en $x^2 = x^1 + r_1$. A partir de maintenant le comportement de l'algorithem de région de confiance est très similaire à la méthode de Newton et il converge vers le point x^*.

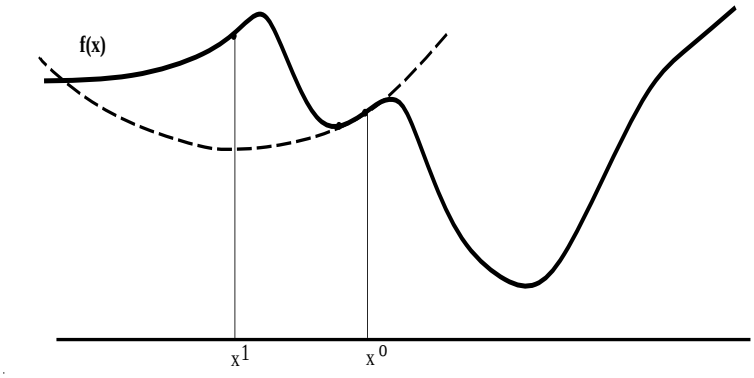


Figure 8.3. *Un exemple de minimisation lorsque plusieurs minima sont présent. Dans ce cas, la méthode de minimisation de Newton peut conduire à l'augmentation de la valeur de la fonction $f(x)$.*

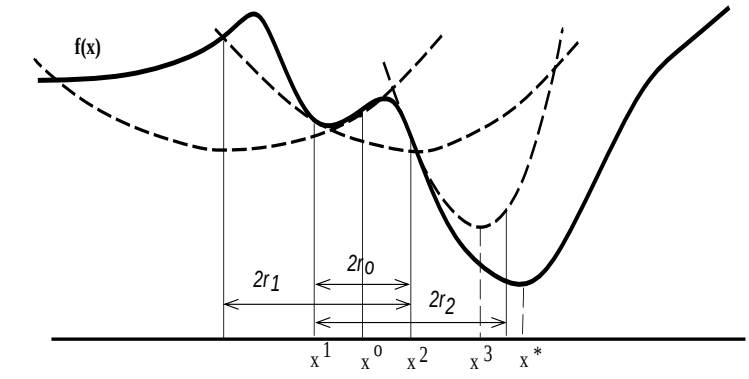


Figure 8.4. *Cette figure montre un exemple de minimisation de la fonction $f(x)$ avec la méthode de région de confiance. Cette méthode garantit que la valeur de la fonction à minimiser décroit.*

8.5.1. Minimisation de la forme quadratique locale

Il s'agit du problème de minimisation d'une forme quadratique à l'intérieur d'une boule :

$$\min\{\frac{1}{2}d^t H d + g^t d : \|d\| \leq \delta\} \qquad (LQP)$$

où g est un vecteur, H est une matrice symétrique et δ est un nombre positif. Puisque l'ensemble de définition est un compact, le problème a une solution. Toutes les méthodes existantes pour résoudre ce problème sont basées sur le théorème suivant ([69] et [137]) :

Théorème 8.1 *d^* est une solution de (LQP) si et seulement s'il existe $\mu \geq 0$ tel que :*

(i) $H + \mu I$ est positive semi-définie,

(ii) $(H + \mu I)d^* = -g$,

(iii) $\|d^*\| \leq \delta$ et $\mu(\|d^*\| - \delta) = 0$.

Un tel μ est unique.

Si μ peut être estimé, la solution optimale de notre problème est facile à calculer. En fait, la minimisation de (LQP) représente un cas intéressant d'optimisation non convexe et, dans ce qui suit, nous allons donner une caractérisation complète de la solution. La solution elle-même sera donnée par un algorithme dont nous allons donner la trame.

Pour $\mu > 0$, le problème (LQP) a une solution sur la frontière de son ensemble de définition, soit sur la frontière de la région de confiance :

$$\|d^*\| = \delta$$

et le problème se réduit à celui de trouver μ tel que :

$$\phi(\mu) = \|d(\mu)\| = \delta \qquad [8.23]$$

où $d(\mu)$ est solution de $(H + \mu I)d = -g$. Soient $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ les valeurs propres de H (une matrice symétrique) et soient $u_1, u_2, \ldots, u_n$ les vecteurs propres correspondants. La fonction ϕ peut être écrite explicitement en utilisant le système propre de H :

$$\phi(\mu) = \left(\sum_{i=1}^{n} \frac{(u_i^t g)^2}{(\lambda_i + \mu)^2} \right)^{1/2} \qquad [8.24]$$

et on peut vérifier les propriétés suivantes de ϕ :

(i) $\phi(\mu)$ est positive, strictement décroissante sur l'intervalle $]-\lambda_1, +\infty[$ et $\phi(\mu) \to 0$ lorsque $\mu \to \infty$.

(ii) $\phi^2(\mu)$ est une fonction rationnelle de μ, avec des pôles du deuxième ordre sur un sous-ensemble de l'ensemble $\{-\lambda_1, -\lambda_2, \ldots -\lambda_n\}$ des opposés des valeurs propres de H [84].

Hebden [84] a proposé un algorithme pour déterminer μ et d^* (la solution optimale) itérativement lorsque l'approximation Gauss-Newton du hessien est suffisante. Dans ce cas, H est symétrique, semi-définie et positive. Hebden a proposé de mettre à jour μ_{l+1} en fonction de μ_l comme suit :

$$\mu_{l+1} = \mu_l + \frac{\phi(\mu_l)}{\phi'(\mu_l)} \frac{(\delta - \phi(\mu_l))}{\delta}. \qquad [8.25]$$

En fait, l'algorithme de Hebden peut être vu comme une méthode de Newton pour trouver la solution de l'équation :

$$\psi(\mu) = \frac{1}{\delta} - \frac{1}{\phi(\mu)} = 0, \quad \text{pour } \mu \in]-\lambda_1, +\infty[\qquad [8.26]$$

La caractéristique la plus importante de l'équation [8.25] est que le nombre d'itérations normalement nécessaire pour aboutir à une bonne approximation de la solution μ^* est petit, car ψ est convexe, presque linéaire et strictement

décroissant sur l'intervalle $]-\lambda_1, +\infty[$. De plus, le calcul de la factorisation de Cholesky de $H+\mu I$ rend possible le calcul décrit par l'équation [8.25] sans connaissance explicite des éléments propres de H, dès lors que $\mu \in]-\lambda_1, +\infty[$. Plus précisément, il est facile de voir que :

$$\phi(\mu) = \|d(\mu)\|, \qquad \phi'(\mu) = -[1/\phi(\mu)]d(\mu)^t(H+\mu I)^{-1}d(\mu),$$

où $(H+\mu I)d(\mu) = -g$. Donc, si R^tR est la factorisation de Cholesky de $H+\mu_l I$, où R est une matrice triangulaire supérieure, alors nous avons :

$$\frac{\phi(\mu_l)}{\phi'(\mu_l)} = -\frac{\phi^2(\mu_l)}{d(\mu_l)^t(R^tR)^{-1}d(\mu_l)} = -\frac{\phi^2(\mu_l)}{\|(R^t)^{-1}d(\mu_l)\|^2}$$

En conclusion, l'algorithme de Hebden, qui trouve le minimum de la forme quadratique locale, peut être résumé comme suit :

- *initialisation* : Soit $\mu_o \geq 0$ et $\phi(\mu_o) > \delta$.
- *itération l=0,1,...*

l.1 factoriser $(H+\mu_l I) = R^tR$.

l.2 résoudre $R^tRp = -g$.

l.3 si $|\|p\| - \delta| < \varepsilon$, alors stop: μ_l est une approximation de μ^* et p est une solution de (LQP).

l.4 sinon, résoudre $R^tq = p$.

l.5 incrémenter

$$\mu_{l+1} = \mu_l + \left(\frac{\|p\|}{\|q\|}\right)^2 \frac{(\|p\| - \delta)}{\delta}. \qquad [8.27]$$

l.6 incrémenter l:=l+1 et retourner à l.1.

En pratique, cet algorithme converge après 4 itérations.

8.5.2. L'algorithme de région de confiance

Nous sommes maintenant en mesure de décrire l'algorithme de région de confiance qui est bien adapté au problème de détermination des paramètres extrinsèques. L'idée est d'utiliser l'approximation Gauss-Newton du hessien pour déterminer la direction de recherche à chaque itération. Dans ce cas, H est définie et positive, on a donc $\lambda_1 > 0$ et la condition (i) du théorème 8.1

est vraie pour tout $\mu \geq 0$. D'abord, on vérifie si $\mu = 0$ lors de la résolution de $H(x)d = -g$. Si $\|d\| \leq \delta$, alors d est la solution optimale pour (LQP). Sinon, $\mu > 0$ et on peut appliquer l'algorithme de Hebden. La trame de l'algorithme de région de confiance est la suivante :

- *initialisation* : Soient

- x_o, valeur initiale.

- δ_o, rayon de confiance initial.

- ε , test "zéro" général.

- ε_g, zéro pour $\|g\|$.

- ε_f, zéro pour f.

- $k := 0$.

- *itération k:=0,1,...*

k.1 calculer $f_k = f(x_k), g_k = \nabla f(x_k)$ et H_k.

k.2 si $\|g_k\| \leq \varepsilon_g$ ou $\delta_k \leq \varepsilon_\delta$ ou $f_k \leq \varepsilon_f$, alors stop: x_k est une solution.

k.3 soit d une solution du système

$$H_k d = -g_k$$

Si $\|d\| < \delta_k - \varepsilon$ alors $d_k = d$. Sinon, utiliser l'algorithme de Hebden pour trouver $\mu > 0$ tel que la solution de $(H_k + \mu I)d = -g_k$ satisfasse $|\|d\| - \delta_k| < \varepsilon$, et on a $d_k = d$.

k.4 calculer r_k et utilisant l'équation [8.22].

k.5 si $r_k \geq s$, alors

$$x_{k+1} := x_k + d_k$$

si $r_k \geq t$, alors

$$\delta_{k+1} := 2\delta_k$$

sinon

$$\delta_{k+1} := \delta_k$$

Faire $k := k + 1$ et retourner à k.1

k.6 si $r_k < s$ alors

$$\delta_k := \delta_k/2$$

et retourner à k.3.

Ceci est l'algorithme général de région de confiance. Le paramètre s doit appartenir à l'intervalle $[0.1, 0.3]$ et le paramètre t doit appartenir à l'intervalle $[0.5, 0.8]$, [179]. Dans notre cas, les valeurs de ces paramètres sont : $s = 0.25$ et $t = 0.75$.

8.6. Méthode linéaire avec une caméra affine

La méthode de minimisation non linéaire que nous venons de décrire nécessite une initialisation. Bien que l'algorithme de région de confiance ait des qualités de convergence qui sont meilleures que celles d'une simple méthode de Newton, il est important de pouvoir l'initialiser correctement afin de réduire le nombre d'itérations et s'assurer un temps de calcul réduit. Pour ce faire, nous allons introduire deux méthodes de calcul des paramètres de rotation et translation :

- une méthode linéaire qui utilise un modèle affine de caméra (voir la section 5.6.) et
- une méthode quasi linéaire qui utilise un algorithme itératif transformant le modèle affine de caméra en un modèle projectif.

Dans ce qui suit, nous allons traité le cas des correspondances de points – l'extension à des correspondances de droites est relativement aisée. Soit à nouveau les équations [8.10] et [eq:y-collinearity] de la projection d'un point M_i d'un objet sur l'image :

$$x_i = \frac{\mathbf{r}_1 \cdot \mathbf{M}_i + t_1}{\mathbf{r}_3 \cdot \mathbf{M}_i + t_3} \quad [8.28]$$

$$y_i = \frac{\mathbf{r}_2 \cdot \mathbf{M}_i + t_2}{\mathbf{r}_3 \cdot \mathbf{M}_i + t_3} \quad [8.29]$$

On divise les nominateurs et dénominateurs de ces fractions par t_3 et on obtient alors les equations suivantes :

$$x_i = \frac{\mathbf{I} \cdot \mathbf{M}_i + x_0}{1 + \varepsilon_i} \quad [8.30]$$

$$y_i = \frac{\mathbf{J} \cdot \mathbf{M}_i + y_0}{1 + \varepsilon_i} \quad [8.31]$$

Les nouvelles notations ont les significations suivantes :

- $\mathbf{I} = \mathbf{r}_1/t_3$

- $\mathbf{J} = \mathbf{r}_2/t_3$
- $x_0 = t_1/t_3$ and $y_0 = t_2/t_3$ sont les coordonnées de la projection dans l'image de m_0 qui est la projection de M_0 – l'origine du repère objet
- $\varepsilon_i = \mathbf{r}_3 \cdot \mathbf{M}_i/t_3$.

On peut également écrire ces équations comme suit :

$$x_i(1+\varepsilon_i) - x_0 = \mathbf{I} \cdot \mathbf{M}_i \quad [8.32]$$

$$y_i(1+\varepsilon_i) - y_0 = \mathbf{J} \cdot \mathbf{M}_i \quad [8.33]$$

Lorsque le rapport entre la taille de l'objet et la distance de l'objet à la caméra est grand, on peut négliger la valeur de ε par rapport à 1. On obtient dans ce cas le modèle de perspective faible et les équations précédentes deviennent :

$$\begin{aligned} x_i - x_0 &= \mathbf{I} \cdot \mathbf{M}_i \\ y_i - y_0 &= \mathbf{J} \cdot \mathbf{M}_i \end{aligned}$$

Si l'on dispose d'au moins 4 points non coplanaires, on peut facilement calculer les vecteurs $\mathbf{I}$ et $\mathbf{J}$ de la façon suivante. Soit $\mathbf{x}$ et $\mathbf{y}$ les vecteurs formés avec les coordonnés image des points, et soit P la matrice $N \times 3$ formée avec les coordonnées 3D des points de l'objet. Les deux équations précédentes s'écrivent alors :

$$\begin{aligned} P\mathbf{I} &= \mathbf{x} \\ P\mathbf{J} &= \mathbf{y} \end{aligned}$$

Si les points de l'objet ne sont pas coplanaires, la matrice P est de rang égal à 3 et on peut alors calculer facilement $\mathbf{I}$ et $\mathbf{J}$:

$$\begin{aligned} \mathbf{I} &= (P^t P)^{-1} P^t \, \mathbf{x} \\ \mathbf{J} &= (P^t P)^{-1} P^t \, \mathbf{y} \end{aligned}$$

On en déduit facilement les paramètres de la rotation et translation entre le repère objet et le repère de la caméra :

$$\begin{aligned} t_3 &= \frac{1}{2}\left(\frac{1}{\|\mathbf{I}\|} + \frac{1}{\|\mathbf{J}\|}\right) \\ t_1 &= x_0 \, t_3 \\ t_2 &= y_0 \, t_3 \end{aligned}$$

$$\begin{aligned} \mathbf{r}_1 &= \frac{\mathbf{I}}{\|\mathbf{I}\|} \\ \mathbf{r}_2 &= \frac{\mathbf{J}}{\|\mathbf{J}\|} \\ \mathbf{r}_3 &= \mathbf{r}_1 \wedge \mathbf{r}_2 \end{aligned}$$

8.7. Méthode quasi linéaire

La méthode linéaire qui vient d'être décrite permet une initialisation des paramètres lorsque l'on se trouve dans des conditions expérimentales proches du modèle affine de caméra. Que se passe-t-il si l'objet est relativement plus proche de la caméra. Dementhon [41] a proposé dans ce cas un algorithme qui applique le modèle précédent d'une manière itérative. On peut également étendre l'algorithme au cas des points coplanaires [142]. Une généralisation de ces algorithmes au cas de la paraperspective a été également proposée [89].

L'idée de base de tous ces algorithmes est de calculer itérativement des valeurs pour les ε_i dans les equations [8.32] et [8.33] jusqu'à l'obtention d'une solution stable. L'algorithme est le suivant :

1. pour tout i, $i \in \{1 \ldots N\}$, $N \geq 3$, $\varepsilon_i = 0$;
2. estimer les vecteurs $\mathbf{I}$ and $\mathbf{J}$,
3. calculer la position et l'orientation de l'objet par rapport à la caméra (voir les équations de la section précédente) ;
4. pour tout i, calculer:

$$\varepsilon_i = \frac{\mathbf{r}_3 \cdot \mathbf{M}_i}{t_3}$$

si les ε_i calculés à l'itération courante sont égaux à ceux calculés à l'itération précédente, arrêt de l'algorithme, sinon retourner à l'étape 2.

On peut se référer à [92] pour une analyse détaillé de cet algorithme ainsi que pour une étude expérimentale. Illustrons le comportement des algorithmes linéaire et quasi linéaire par les deux images de la figure 8.5.

8.8. Conclusion

Les méthodes que nous avons décrites dans ce chapitre sont particulièrement intéressantes lorsque l'on désire déterminer la position et l'orientation d'une

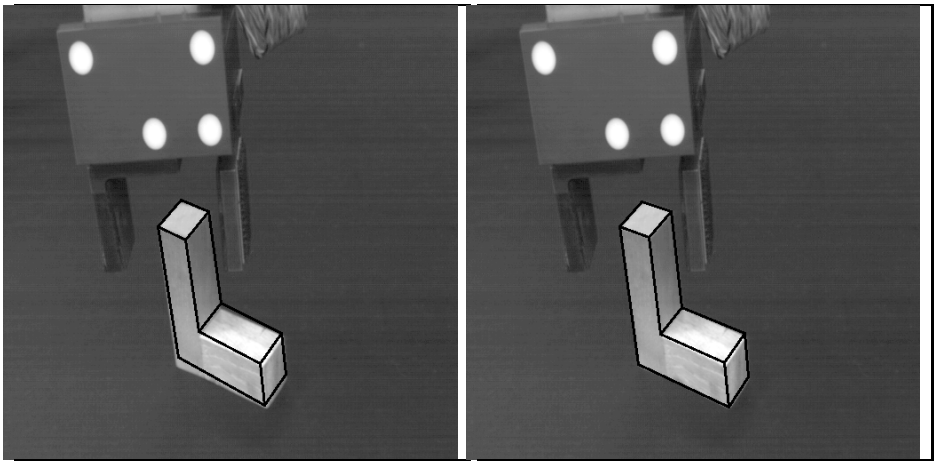


Figure 8.5. *Un exemple de calcul des paramètres d'orientation et position avec un algorithme linéaire (gauche) et avec un algorithme itératif, ou quasi linéaire (droite).*

caméra par rapport à un objet dans un contexte de reconnaissance d'objets. Ces méthodes doivent être distinguées des méthodes de calibration présentées au chapitre 5, qui mettaient l'accent sur l'obtention d'un modèle de capteur très précis en présence d'objets de calibrage spécialement construits à cet effet. Un certain nombre d'expérimentations décrites dans [149] et dans [150] permettent de comparer les résultats obtenus avec la méthode Faugeras-Toscani (chapitre 5 et [57]) avec la méthode de région de confiance décrite dans ce chapitre.

Une caractéristique importante de la méthode utilisant l'algorithme de région de confiance est que cette méthode n'est pas seulement robuste par rapport au bruit, mais également par rapport aux mesures aberrantes et par rapport aux erreurs d'appariement. La méthode linéaire du chapitre 5 nécessite au moins 6 points, alors que la méthode décrite ici peut en principe fonctionner avec seulement 3 points. En pratique la méthode de région de confiance est initialisée avec un algorithme linéaire ou avec un algorithme quasi linéaire et nécessite donc un minimum de 4 appariements.

Chapitre 9

Reconnaissance d'objets

9.1. Introduction

Dans les chapitres précédents nous avons proposé quelques méthodes pour résoudre le problème de la localisation d'un objet tri-dimensionnel à partir de données bi-dimensionnelles (chapitre 8) ou de données tri-dimensionnelles (chapitre 7). Plus précisément, toutes les méthodes que nous avons étudiées avaient ceci en commun qu'elles supposaient qu'un certain nombre de caractéristiques de l'objet étaient en correspondance terme à terme avec un nombre égal de caractéristiques issues des données. Sur la base d'appariements objets/données il a été possible de calculer une transformation entre le repère de l'objet et le repère des donnés. Le repère des données n'est autre que le repère du capteur qu'il s'agisse d'une seule caméra (données 2D) ou d'un capteur fournissant des données 3D.

Dans ce chapitre nous allons étudier une méthodologie permettant d'établir des appariements objet/données 3D/3D ou 3D/2D. Si un certain nombre de tels appariements existe on dira alors qu'une instance de cet objet se trouve dans les données, autrement dit, un objet de ce type est *vu* par le capteur ayant fourni les données. Ces appariements une fois établis on pourra alors localiser l'objet, soit calculer sa position et son orientation par rapport au repère du capteur. Le terme "reconnaissance d'objet" désignera en fin de compte l'ensemble des deux processus suivants :

Appariement – établir une correspondance terme à terme entre des caractéristiques objet et des caractéristiques données ;

Localisation – déterminer la position et l'orientation de l'objet dans le repère

des données (du capteur).

9.2. Complexité

La tâche qui consiste à établir des appariements est difficile. Soit par exemple le cas où aussi bien les données que l'objet sont décrits par des caractéristiques du même type et soient $\mathcal{O}$ l'ensemble de caractéristiques objet et $\mathcal{D}$ l'ensemble de caractéristiques données.

Si $\mathcal{O}$ et $\mathcal{D}$ ont le même nombre d'éléments (ce qui sera rarement le cas !) et si n est ce nombre, le nombre d'appariements possibles est :

$$n!$$

Cependant $\mathcal{O}$ et $\mathcal{D}$ ont rarement la même taille. Les données peuvent contenir des caractéristiques provenant d'autres objets. Soit m la taille de $\mathcal{D}$ et n la taille de $\mathcal{O}$ avec $m \geq n$. Le nombre d'appariement possible est alors :

$$C_m^n \;\; n!$$

Les caractéristiques d'un objet ne sont pas toutes présentes dans les données car :

- le capteur ne peut "voir" qu'une partie de l'objet (occlusions mutuelles entre les caractéristiques d'un même objet) ;
- l'objet peut être partiellement caché par un autre objet (occlusion mutuelle entre deux objets).

Dans ces conditions on peut espérer tout au mieux mettre en correspondance un sous-ensemble de $\mathcal{O}$ avec un sous-ensemble de $\mathcal{D}$. Le nombre d'appariements possibles devient alors beaucoup plus grand :

$$\sum_{i=1}^{n} C_n^i \;\; C_m^i \;\; i!$$

La reconnaissance est d'autant plus fiable que le nombre d'appariements est grand, voire le plus grand. Le problème à résoudre est donc le suivant : trouver le plus grand nombre d'appariements entre $\mathcal{O}$ et $\mathcal{D}$. Un tel appariement *maximal* ne peut être trouvé facilement. Même si on tient compte des propriétés des caractéristiques constituantes de $\mathcal{O}$ et $\mathcal{D}$, la complexité de ce problème est combinatoire. En pratique on se contentera d'une solution sous-optimale.

9.3. Appariement et isomorphisme de graphes

Une première approche possible consiste à traiter le problème d'appariement comme un problème d'isomorphisme de graphes. Un objet peut être décrit comme un ensemble de caractéristiques et des relations entre ces caractéristiques. L'objet peut alors être décrit par un graphe dans lequel un nœud représente une caractéristique et une arête représente une relation entre deux caractéristiques. Les données peuvent être décrites par un graphe de même nature. Le problème d'appariement de caractéristiqes devient alors un problème d'appariement de graphes, soit un problème d'isomorphisme de graphes.

Plus particulièrement il s'agira de résoudre le problème de trouver l'isomorphisme maximal de sous-graphes entre un sous-graphe de $G(\mathcal{O})$ et un sous-graphe de $G(\mathcal{D})$ où $G(\mathcal{O})$ est le graphe associé à l'ensemble $\mathcal{O}$. Si n est le nombre de nœuds du graphe objet et m est le nombre de nœuds du graphe données, $m \geq n$, la taille de l'espace des solutions est (comme auparavant) :

$$\sum_{i=1}^{n} C_n^i \;\; C_m^i \;\; i!$$

et le problème de trouver l'isomorphisme de sous-graphes maximal parmi tous ces isomorphismes de sous-graphes est un problème NP-complet [67].

Un certain nombre de travaux ont proposé des solutions au problème d'isomorphisme de graphes en général et au problème d'isomorphisme de sous-graphes maximal en particulier [71], [164]. Laurent Hérault [85] ainsi que d'autres auteurs proposent de traiter le problème d'isomorphisme de graphes comme un problème d'optimisation combinatoire – cela revient à chercher l'isomorphisme qui minimise une distance entre deux graphes ou deux sous-graphes. Si cette distance est assimilée à l'énergie potentielle d'un système physique on peut utiliser des algorithmes stochastiques qui convergent en un temps fini vers une configuration très proche de la configuration correspondant à l'énergie potentielle la plus basse.

Dans le cas de l'isomorphisme de graphes la distance a une expression simple. Dans le cas de l'isomorphisme de sous-graphes maximal, la distance a une expression plus complexe – il s'agit de la somme pondérée de plusieurs termes et en pratique à chaque cas particulier correspond une pondération différente. Des travaux de recherche sont en cours dans ce domaine.

9.4. Appariement et prédiction/vérification

Cette deuxième approche consiste à résoudre les problèmes d'appariement et de localisation *simultanément* dans le cadre du paradigme prédiction et vérification. Ce paradigme consiste essentiellement à faire une hypohtèse quant aux paramètres de position et d'orientation de l'objet par rapport au capteur et à vérifier que des appariements sont compatibles avec ces paramètres. Dans ses lignes les plus générales, ce paradigme peut être décrit comme suit :

- *Prédiction* :

– établir quelques appariements objet/données,

– calculer les paramètres de la transformation objet/données sur la base de ces appariements ainsi que les intervalles de confiance associés à ces paramètres,

– appliquer cette transformation à l'ensemble des caractéristiques objet de façon à pouvoir prédire lesquelles parmi ces caractéristiques sont susceptibles d'être vues par le capteur et quelle est leur position et orientation dans le repère du capteur.

- *Vérification* :

– pour chaque caractéristique objet ainsi prédite, sélectionner les caractéristiques données les plus proches et les plus ressemblantes. Former ainsi des nouveaux appariements objet/données tout en rejetant les appariements qui s'avèrent incorrects,

– fusionner chaque nouvel appariement avec les appariements précédents afin de mettre à jour les paramètres de la transformation objet/données ainsi que les intervalles de confiance associés.

La figure 9.1. illustre cette approche par un exemple lorsque les caractéristiques sont des points et lorsque les données sont 3D. L'objet à reconnaître (à droite) est un parallélipipède et les données (à gauche) contiennent une instance de cet objet caché partiellement par un autre objet, ainsi que du bruit. Dans ce cas il suffit de 3 appariements (en haut) pour estimer une transformation objet/données. Le choix de ces appariements initiaux est crucial. Ils ne peuvent être validés ... qu'en vérifiant que d'autres appariements produisent les mêmes paramètres de la transformation objet/données. Les points de l'objet sont transformés dans le repère des données (au milieu) sauf les points cachés

par l'objet lui-même. Dans ce cas l'objet a 8 points mais 7 points seulement sont prédits visibles dans les données. La position de ces 7 points est prédite dans les données. A chacune de ces positions est également associé une tolérance qui dans ce cas est une sphère de rayon r.

Soit $\mathbf{m}_i$ un point de l'objet et soit $\hat{\mathbf{m}}_i$ l'image de $\mathbf{m}_i$ par rotation et translation :

$$\hat{\mathbf{m}}_i = \mathbf{R}\ \mathbf{m}_i + \mathbf{t}$$

Les points données $\mathbf{c}_i$ qui satisfont à :

$$\|\hat{\mathbf{m}}_i - \mathbf{c}_i\| \leq r$$

sont des appariements potentiels de $\mathbf{m}_i$. Pour chaque point $\hat{\mathbf{m}}_i$ on sélectionne un point $\mathbf{c}_i$ parmi tous les candidats possibles. La figure 9.1. (en bas) montre 4 appariements parmi lesquels 3 seulement sont corrects (1, 3 et 4). En effet, l'appariement 2 correspond à un point de l'objet prédit visible mais caché en réalité par un autre objet. De ce fait le point a été incorrectement apparié avec un point des données appartenant à un autre objet. Ce genre de faux appariements arrive souvent lorsqu'on a affaire à des scènes complexes.

En pratique nous verrons comment on peut éviter ce genre de problème en effectuant à la fois une vérification *locale* de chaque appariement, une vérification *globale* d'un ensemble d'appariements ainsi que la mise en œuvre d'une structure algorithmique que recherche le meilleur ensemble d'appariements dans l'espace formé par tous les appariements.

Abordons d'un point de vue plus formel les phases de prédiction et de vérification pour les deux cas de reconnaissance auxquels on s'intéresse : 3D/3D et 3D/2D.

9.4.1. Prédiction et vérification 3D/3D

Dans ce cas aussi bien l'objet que les données sont tri-dimensionnels. Au chapitre 7 nous avons calculé la rotation et translation optimale entre deux ensembles 3D exprimés dans deux repères différents pour des points, des droites et des plans.

La phase de prédiction consiste donc dans ce cas à calculer les paramètres de rotation et translation utilisant un petit nombre d'appariements, le plus petit nombre étant 3 (pour les points et les plans) et 2 (pour les droites). Soient $\mathbf{R}$ et $\mathbf{t}$ la rotation et translation ainsi calculées.

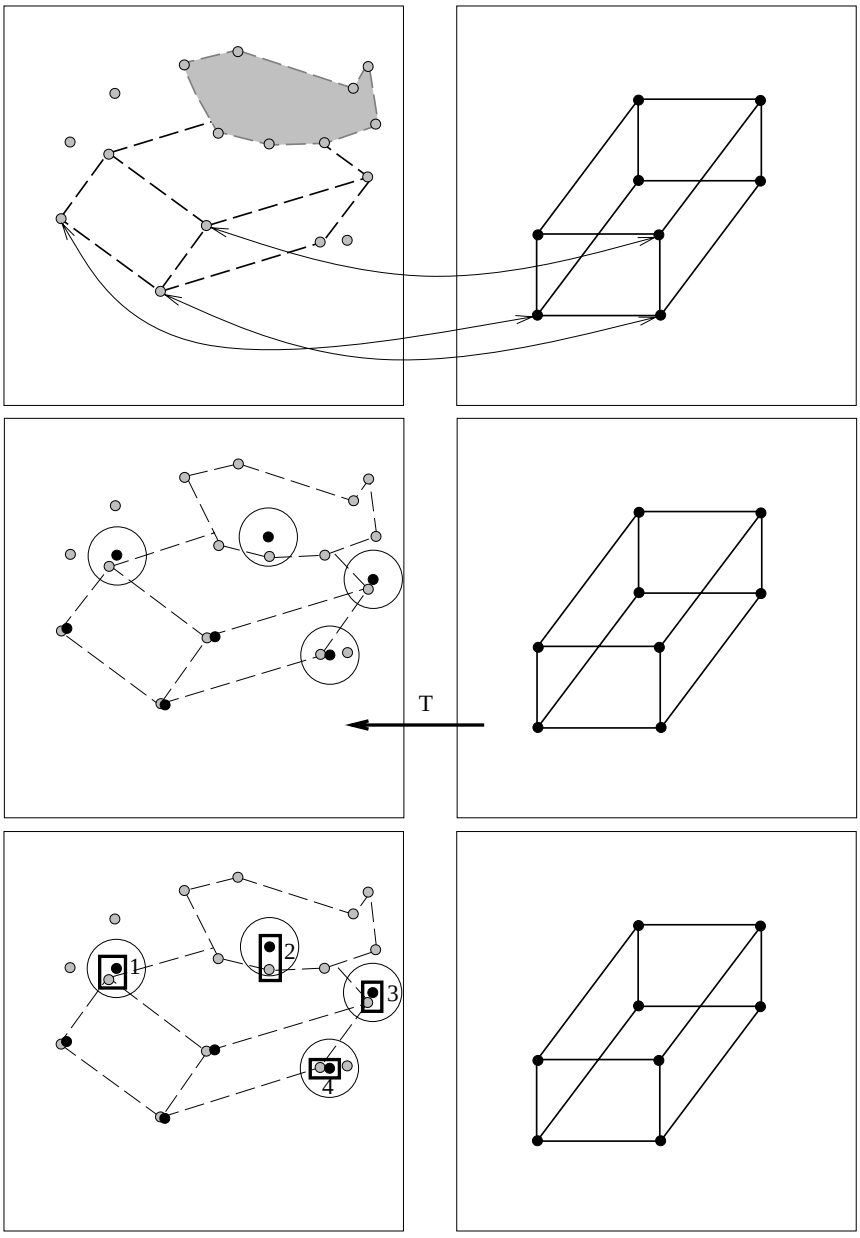


Figure 9.1. *Le principe de l'approche prédiction/vérification.*

La phase de vérification consiste à transformer d'autres caractéristiques de l'objet dans le repère des données en utilisant la transformation précédemment calculée et à confirmer ou rejeter la prédiction sur la base d'un certain nombre de mesures. Soit p le nombre d'appariements initial et soit k le nombre d'appariements à confirmer ou à rejeter. Si n est le nombre de caractéristiques de l'objet on a nécessairement $p \leq k \leq n$. Soit par exemple le cas où les caractéristiques sont des segments de droite. Deux vérification sont possibles: une vérification individuelle (ou locale) de chaque appariement et une vérification globale d'un ensemble d'appariements.

9.4.1.1. Vérification individuelle

Soient $\mathbf{v}_i$ et $\mathbf{p}_i$ les deux vecteurs décrivant la droite support du segment objet et $\mathbf{v}'_i$ et $\mathbf{p}'_i$ les vecteurs décrivant la droite support correspondante dans les données. Soit $\mathbf{m}_i$ le point milieu du segment objet et $\mathbf{m}'_i$ le point milieu du segment données. Soit l la longueur du segment objet et l' la longueur du segment données. Le segment objet est apparié avec le segment données si les critères suivants sont respectés (voir les équations [7.4] et [7.5] ainsi que la figure 9.2.) :

$$\begin{aligned} \|\mathbf{v}'_i - \mathbf{R}\ \mathbf{v}_i\|^2 &\leq \epsilon_1 \\ \|\mathbf{p}'_i - \mathbf{R}\ \mathbf{p}_i - \mathbf{t} + ((\mathbf{R}\ \mathbf{v}_i) \cdot \mathbf{t})\mathbf{R}\ \mathbf{v}_i\|^2 &\leq \epsilon_2 \\ \|\mathbf{m}'_i - \mathbf{R}\ \mathbf{m}_i - \mathbf{t}\|^2 &\leq \epsilon_3 \\ (l'_i - l_i)^2 &\leq \epsilon_4 \end{aligned}$$

Lorsque ϵ_1, ϵ_2, ϵ_3 et ϵ_4 sont petits ces équations garantissent une très bonne ressemblance entre le segment "données" et le segment "objet". Si ces critères sont satisfaits, on peut alors utiliser ce nouvel appariement pour incrémenter le nombre d'appariements et mettre à jour les paramètres de la transformation objet-données.

9.4.1.2. Vérification globale

Soient maintenant k appariements, chaque appariement individuel satisfaisant aux critères qu'on vient de décrire. La qualité de cet ensemble d'appari-

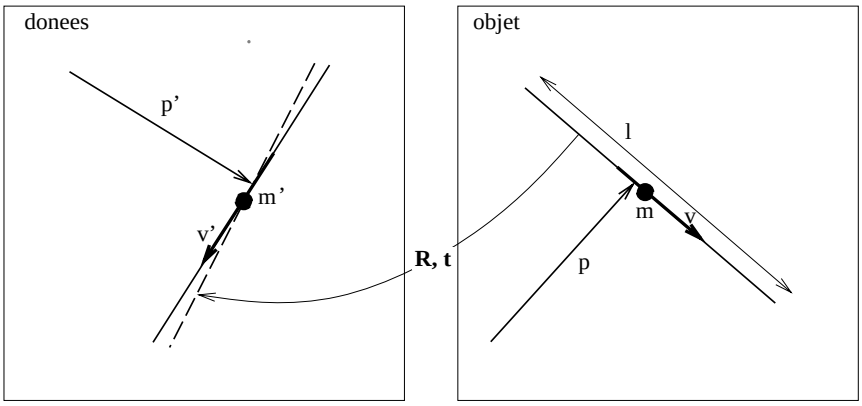


Figure 9.2. *Un segment de droite est représenté par un vecteur directeur* $\boldsymbol{v}$*, un vecteur de position* $\boldsymbol{p}$*, son point milieu* $\boldsymbol{m}$ *et sa longueur* l*. Moyennant la connaissance de la rotation et la translation on peut comparer un segment de l'objet avec un segment des données.*

ments peut être évalué en sommant sur les appariements :

$$
\begin{aligned}
q_1 &= \frac{1}{k}\sum_{i=1}^{k} \|\mathbf{v}'_i - \mathbf{R}^{\,k}\mathbf{v}_i\|^2 \\
q_2 &= \frac{1}{k}\sum_{i=1}^{k} \|\mathbf{p}'_i - \mathbf{R}^{\,k}\mathbf{p}_i - \mathbf{t}^k + ((\mathbf{R}^{\,k}\mathbf{v}_i)\cdot\mathbf{t}^k)\mathbf{R}^{\,k}\mathbf{v}_i\|^2 \\
q_3 &= \frac{1}{k}\sum_{i=1}^{k} \|\mathbf{m}'_i - \mathbf{R}^{\,k}\mathbf{m}_i - \mathbf{t}^k\|^2 \\
q_4 &= \frac{1}{k}\sum_{i=1}^{k} (l'_i - l_i)^2
\end{aligned}
$$

Dans ces équations $\mathbf{R}^{\,k}$ et $\mathbf{t}^k$ désignent la rotation et la translation correspondants à k appariements, différentes de la rotation et de la translation calculées lors de l'étape de prédiction.

9.4.2. Prédiction et vérification 3D/2D

Lorsque les données sont 2D (une image) on peut obtenir une solution pour la rotation et la translation avec des points et/ou des segments de droites. Il

faut un minimum de 3 points pour déterminer les paramètres de position et d'orientation de l'objet par rapport au capteur (la caméra) mais la solution n'est pas unique. Chaque solution ainsi obtenue constituera une prédiction qu'il faudra vérifier. De même, il faut au minimum 3 segments de droite pour déterminer la rotation et la translation mais là encore la solution n'est pas unique. Si l'on dispose d'un nombre de correspondances plus élevé la solution devient unique (voir chapitre 8) et on peut alors associer une seule prédiction à un ensemble de correspondances point/point ou droite/droite.

Même si une prédiction correspond à un nombre élevé de correspondances, rien ne prouve que la solution est correcte du point de vue de la reconnaissance d'objet (elle est correcte d'un point de vue numérique). Il faut donc envisager une méthode de vérification de la prédiction. Comme auparavant la vérification consiste à mesurer la distance entre un point de l'objet et son homologue dans l'image.

Soient $\mathbf{R}$ et $\mathbf{t}$ les paramètres associés avec une prédiction et soit $\mathbf{m}_i/\mathbf{m}'_i$ une correspondance point-objet/point-image n'appartenant pas à l'ensemble d'appariements associés avec la prédiction. Les coordonnées de la projection de $\mathbf{m}_i$ dans l'image sont données par :

$$\begin{pmatrix} su_i \\ sv_i \\ s \end{pmatrix} = \begin{pmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ 0^t & 1 \end{pmatrix} \begin{pmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{pmatrix}$$

où α_u, α_v, u_0 et v_0 sont les paramètres intrinsèques de la caméra. L'appariement $\mathbf{m}_i/\mathbf{m}'_i$ est vérifié si l'inégalité :

$$(u_i - u'_i)^2 + (v_i - v'_i)^2 \leq \epsilon$$

est vérifiée, ϵ étant petit. Pour vérifier k appariements il suffit de sommer la grandeur qu'on vient de calculer :

$$\sum_{i=1}^{k} ((u_i - u'_i)^2 + (v_i - v'_i)^2)$$

et de s'assurer qu'elle est inférieure à un seuil.

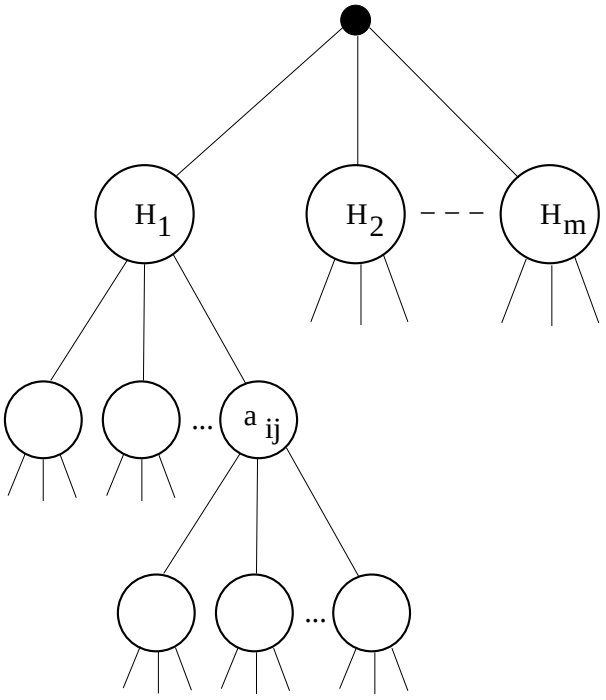


Figure 9.3. *La structure de l'arbre de recherche pour le paradigme prédiction/vérification.*

9.5. Arbre de recherche

La façon la plus répandue de réaliser en pratique le paradigme prédiction/vérification est par une recherche en *profondeur d'abord* d'une structure d'arbre (depth-first tree search) [10], [75], [76], [119], [21], [59], [113], [72], [87], [95]. La structure de cet arbre est celle de la figure 9.3.

Le premier niveau de cet arbre, juste après sa racine, est constitué de nœuds-hypothèses, soit H_1, H_2, ... H_m. Chaque hypothèse comporte en fait le nombre minimum d'appariements permettant de calculer une transformation objet-données. Puisqu'une telle hypothèse détermine entièrement une position et une orientation de l'objet par rapport au capteur, on peut lui associer "une vue" de l'objet, une vue étant définie par un ensemble de caractéristiques de l'objet susceptibles d'être visibles pour une orientation donnée. Ce sont ces caractéristiques visibles sous une hypothèse qui vont servir à la vérification de

cette hypothèse.

Les niveaux suivants de l'arbre sont constitués de nœuds tels que chaque nœud représente un seul appariement. Un nœud a_{ij} décrit une caractéristique m_i de l'objet mise en correspondance avec une caractéristique c_j des données. Si une caractéristique objet n'a pas de correspondant dans les données, soit parce qu'elle n'est pas visible, soit parce que les données sont défaillantes on notera dans l'arbre un nœud "nul".

A partir d'une hypothèse comportant p appariements on considère une nouvelle caractéristique objet, m_{p+1} ainsi que l'ensemble des caractéristiques données qui peuvent lui correspondre, c^1_{p+1}, ... c^k_{p+1}, figure 9.4. Soit le premier de ces appariements m_{p+1}/c^1_{p+1} grâce à qui on met à jour les paramètres de la transformation associée à l'hypothèse en cours ainsi que les intervalles de tolérance associés avec ces paramètres. Si cet appariement est accepté on monte dans l'arbre au niveau $p+2$ et ainsi de suite. Un chemin dans un arbre comporte un nœud par niveau. Un chemin représente donc une solution, soit un ensemble d'appariements :

$$m_1/c_1 - \ldots - nul - \ldots - m_i/c_i - \ldots$$

Lorsqu'un chemin n'est pas satisfaisant, la recherche fait un retour en arrière jusqu'au dernier point de choix (backtracking). Les points de choix apparaissent dès lors qu'il y a plusieurs interprétations possibles pour la même caractéristique objet.

Les nœuds "nuls" permettent de "sauter" une caractéristique objet soit parce que cette caractéristique n'a pas de correspondant, soit parce que le fait de lui attribuer un correspondant est trop pénalisant. Par exemple, sur la figure 9.1. le fait d'accepter l'appariement 2 (qui est faux) peut pénaliser fortement le coût de la reconnaissance. Dans cet exemple, la solution 1-nul-3-4 est peut-être meilleure que la solution 1-2-3-4.

La question cruciale qui se pose avec ce type de méthodes est de savoir sur quelles bases décide-t-on que le nombre d'appariements d'un chemin de l'arbre est une bonne solution. Une condition nécessaire est que les critères numériques développés auparavant (sections 9.4.1. et 9.4.2.) soient satisfaits. Une autre condition est que le nombre d'appariements soint le plus grand possible, voire le plus grand. C'est cette deuxième condition que nous allons analyser maintenant.

Soit H une hypothèse en cours de vérification et soit n_H le nombre de

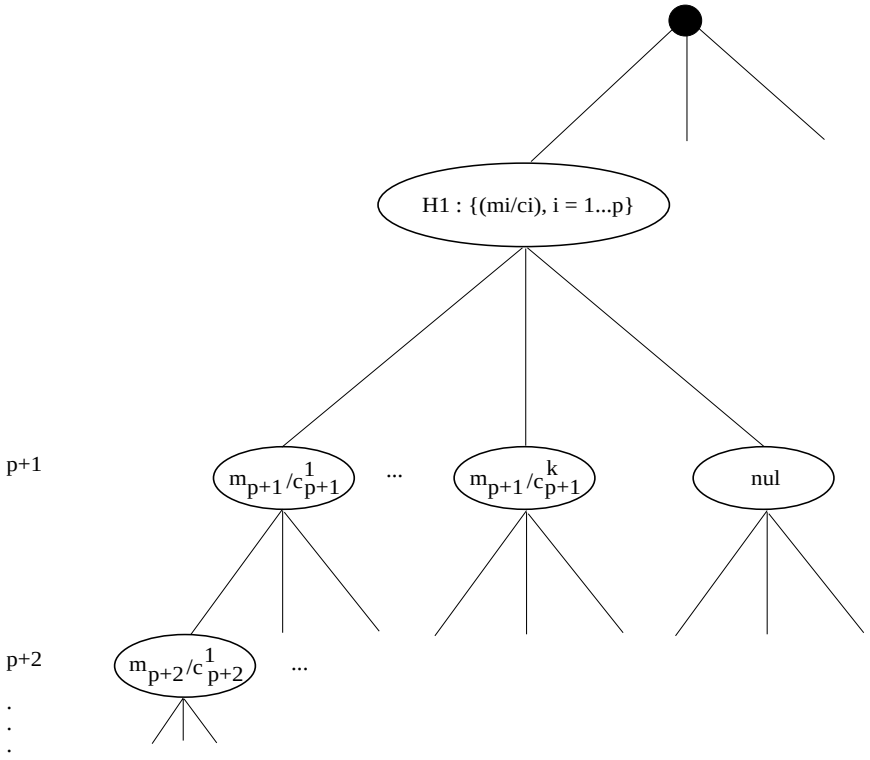


Figure 9.4. *A partir d'une hypothèse comportant p appariements, chaque niveau dans l'arbre de recherche correspond à une* $(p+i)^{ème}$ *caractéristique de l'objet en cours de vérification.*

caractéristiques objet prédites visibles grâce aux paramètres de cette hypothèse. Si n est le nombre total de caractéristiques de l'objet on a $n_H < n$. En pratique ces n_H caractéristiques ne seront pas toutes vérifiables dans les données pour des raisons que nous avons déjà discutées. On sera donc amené à accepter une solution telle que le nombre d'appariements k qui lui est associé est inférieur à n_H. Une solution sera acceptable si :

- k est supérieur à m_H, le nombre *minimum* de caractéristiques qui doivent être présentes pour accepter l'hypothèse H et k doit donc satisfaire à : $m_H \leq k \leq n_H$ et
- parmi ces k appariements il y a des caractéristiques objet "clés", c'est à dire des caractéristiques objet dont la présence dans l'image est indispensable.

Par la suite nous analysons comment on peut tenir compte de ces deux critères pour accélérer la recherche.

9.5.1. Abandon d'un chemin

A partir du premier critère qu'on vient d'établir on peut envisager une stratégie très simple pour accélérer la recherche dans l'arbre. L'idée est de pouvoir prédire le nombre d'appariements d'un chemin et d'abandonner le parcours de ce chemin s'il est peu probable d'améliorer la solution courante.

Remarquons d'abord qu'à chaque hypothèse correspond un sous-arbre. Soit donc un tel sous-arbre dont le nœud de départ est un nœud-hypothèse – l'hypothèse H. Si cette hypothèse comporte p appariements et si le nombre total de caractéristiques prédites visibles par cette hypthèse est de n_H, la profondeur du sous-arbre est alors précisément $n_H - p$. Supposons par ailleurs que p est inférieur à m_H.

Soit i la profondeur du nœud courant du sous-arbre, $p < i < n_H$ et supposons que l'on a déjà apparié s_i caractéristiques. s_i est inférieur ou égal à i car le chemin courant jusqu'à la profondeur i peut comporter des nœuds nuls (pas d'appariement). Le nombre de nœuds restant à parcourir le long de ce chemin est $n_H - i$ et on peut donc espérer, tout au mieux, trouver $n_H - i$ appariements. On peut donc estimer la taille de la meilleure solution le long de ce chemin, soit $[s_i + (n_H - i)]$ appariements. Si $s_i + (n_H - i) < m_H$ alors cette solution ne sera pas satisfaisante et ceci peut être prédit à la profondeur i sans perdre le temps d'aller jusqu'au bout du chemin. En conclusion :

Si, à un stade quelconque du parcours d'un sous-arbre associé à une hypothèse H, le nombre d'appariements (s_i) de la solution courante augmenté du nombre de caractéristiques qui restent à vérifier est inférieur à m_H, il faut alors abandonner le chemin courant et revenir au dernier point de choix car cette solution ne sera pas satisfaisante.

Ce concept d'abandon de parcours de chemin est illustré sur la figure 9.5. Le même concept peut être utilisé dès qu'on estime qu'on ne peut pas améliorer la meilleure solution courante. Soit en effet k_{max} le nombre d'appariements de la meilleure solution disponible à un moment donné au cours de la recherche dans l'arbre. On abandonne la recherche si on se rend compte que, lors du parcours d'un chemin, on ne pourra pas améliorer cette solution. On a donc :

Si, à un stade quelconque du parcours de l'arbre le nombre d'appariements (s_i) de la solution courante augmenté du nombre de caractéristiques qui restent à vérifier est inférieur à k_{max}, il faut alors abandonner le chemin courant et revenir au dernier point de choix car cette solution ne sera pas meilleure que la meilleure solution disponible.

9.5.2. Abandon d'une hypothèse

La recherche sera d'autant plus rapide que l'arbre aura un petit nombre de nœuds. Le nombre de nœuds d'un arbre dépend crucialement du nombre de nœuds de ses premiers niveaux. Le premier niveau étant constitué d'hypothèses, moins il y aura d'hypothèses prédites, plus la recherche sera efficace.

Soit une hypothèse courante H à qui sont associés des appariements ainsi que des paramètres de position et orientation. Ces derniers paramètres caractérisent géométriquement l'hypothèse. Si ces paramètres coïncident avec les paramètres d'une hypothèse déjà traitée, fut-elle bonne ou mauvaise, il y a de fortes chances que l'hypothèse H qu'on s'apprête à vérifier produise une solution équivalente. En effet plusieurs ensembles d'appariements différents peuvent produire la même solution géométrique : ceci est dû aux symétries inhérentes à beaucoup d'objets.

9.5.3. Ordonnancement des nœuds

L'ordre dans lequel on considère les caractéristiques de l'objet peut avoir une influence considérable sur la taille de l'espace de recherche. On peut par

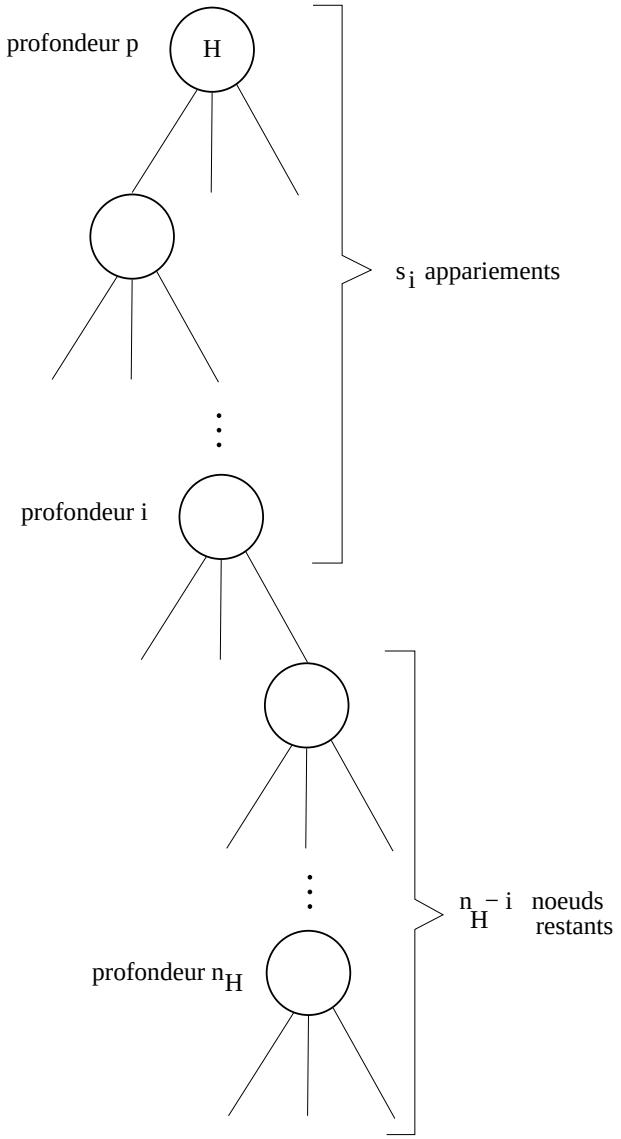


Figure 9.5. *A tout moment on peut abandonner le parcours d'un chemin si les nœuds restants ne sont pas en nombre suffisant pour constituer une solution ou pour améliorer la meilleure solution.*

exemple classifier les caractéristiques d'un objet de la plus discriminante à la plus ambiguë. Dans le cas de segments de droite, cela revient à classer les segments par leur longueur, du plus long au plus court. En effet, un segment long a plus de chances d'être vu dans l'image et a intrinsèquement moins de correspondants. Cette dernière propriété est due au phénomène d'obstruction. Un segment de l'objet, pouvant être partiellement caché, peut potentiellement correspondre à des segments donnés qui lui sont supérieurs ou égaux en longueur.

Supposons qu'on divise les segments de l'objet en deux classes : segments longs et segments courts et supposons que les segments longs constituent les premiers niveaux de l'arbre, juste au-dessus de la racine et les segments courts consituent les derniers niveaux de l'arbre. Soit i la profondeur de l'arbre à la jonction entre les segments longs et les segments courts. Si l'on a constaté que les segments longs sont présents dans les données (disons à 75%) cela veut dire qu'on est peut-être sur le bon chemin. Si la plupart de ces segments sont absents des données (disons à 50%) cela veut dire qu'on devrait abandonner cette hypothèse.

9.6. Un exemple : 3DPO

Le système 3DPO (Three Dimensional Part Orientation) a été développé à SRI International en 1983 par l'un des auteurs en collaboration avec R. Bolles et M. J. Hannah [22], [24], [21], [88]. Il s'agit d'une méthode pour reconnaître et localiser un objet 3D à partir de données 3D "denses" obtenues grâce à un capteur stéréoscopique actif, chapitre 5, section 5.9. et figure 5.18. 3DPO détermine la position et l'orientation d'un objet dans le repère du capteur ainsi que la position relative de cet objet par rapport à d'autres objets : quel objet se trouve au-dessus de la pile et est-il partiellement caché par d'autres objets ?

Pour répondre à cette question 3DPO procède en trois étapes. La première étape consiste en des traitements bas niveau des données afin d'extraire des caractéristiques propices pour la reconnaissance. La deuxième étape consiste en une procédure prédiction/vérification d'appariement entre les données et le modèle. La troisième étape détermine la configuration globale de la scène analysée en comparant les données avec une image synthétique produite à partir du modèle de l'objet et des paramètres de position et d'orientation précédemment calculés.

La figure 9.6. montre une image 3D d'un ensemble d'objets à reconnaître et

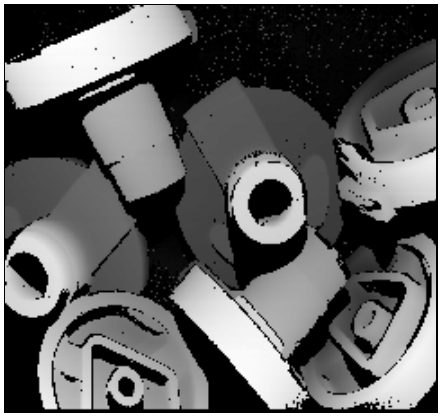


Figure 9.6. *Une image de profondeur obtenue grâce à un capteur stéréoscopique actif. Les pixels sombres codent des points éloignés et les pixels clairs codent des points proches du capteur.*

à localiser. Dans cette image les niveaux de gris codent la hauteur en chaque pixel, les pixels sombres correspondant à des points éloignés et les pixels clairs à des points proches du capteur. Cette image a été obtenue *tranche par tranche* ou ligne par ligne en déplaçant le plan de lumière du capteur transversalement par rapport à la scène.

Cette image est tout d'abord traitée comme une image 2D afin d'extraire les contours. A chaque point de contour on associe ensuite ses coordonnées 3D ce qui permet finalement d'extraire des arêtes 3D. La figure 9.7. montre les contours 2D et les contours 3D. Ces contours sont segmentés de façon à obtenir des arêtes circulaires et des arêtes droites, une arête étant définie comme l'intersection de deux surfaces (deux plans ou un plan et un cylindre). La figure 9.8. montre une arête droite et une arête circulaire. Plus précisément, une arête droite est définie par :

- longueur,
- position,
- orientation de ses facettes

et une arête circulaire est définie par :

- rayon,
- longueur,

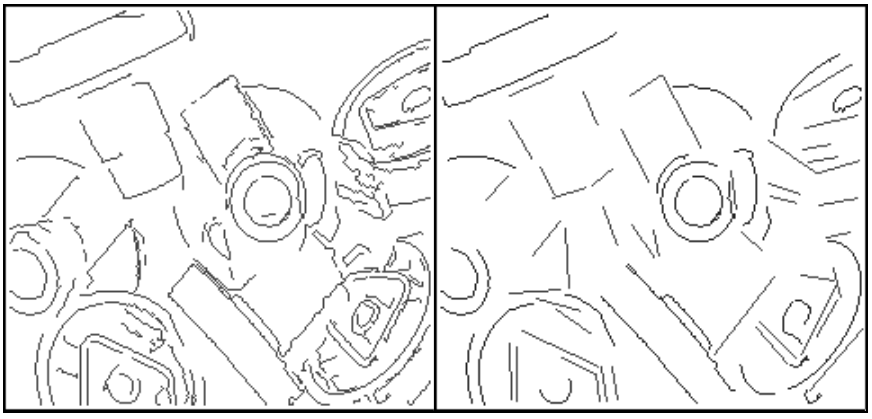


Figure 9.7. *Les contours 2D (gauche) et 3D (droite) extraits d'une image de profondeur.*

- position du centre,
- orientation du plan contenant l'arête.

La figure 9.9. montre des cercles et des droites correspondant à des arêtes circulaires et droites extraites le long des contours. Un système de modélisation géométrique a été utilisé pour décrire l'objet à identifier. Ce modèle contient des informations de type géométrique et topologique (ou relationnelles). Le modèle géométrique est tout simplement constitué d'une liste d'arête droites et circulaires telles qu'on vient de les décrire. Le modèle topologique décrit la manière dont ces arêtes sont reliées entre elles. La figure 9.10. montre le modèle de l'objet à reconnaître.

L'étape de reconnaissance est basée sur le paradigme prédiction/vérification qu'on vient de décrire. On peut remarquer qu'un appariement arête-modèle/arête-image définit 5 parmi les 6 paramètres de position et orientation associés avec une hypothèse. Dans le cas d'une arête droite la translation le long de l'arête n'est pas définie et dans le cas d'une arête circulaire la rotation autour de l'axe du cylindre correspondant n'est pas défini (figure 9.8.). A partir d'une telle hypothèse la vérification est matérialisée par une recherche en profondeur dans un arbre afin de confirmer ou d'infirmer une hypothèse. La figure 9.11. montre le résultat de la reconnaissance et de la localisation des 7 objets présents dans la scène. A partir de ces 7 objets prédits présents dans la scène, du modèle géométrique associé et de la position et l'orientation de chaque objet,

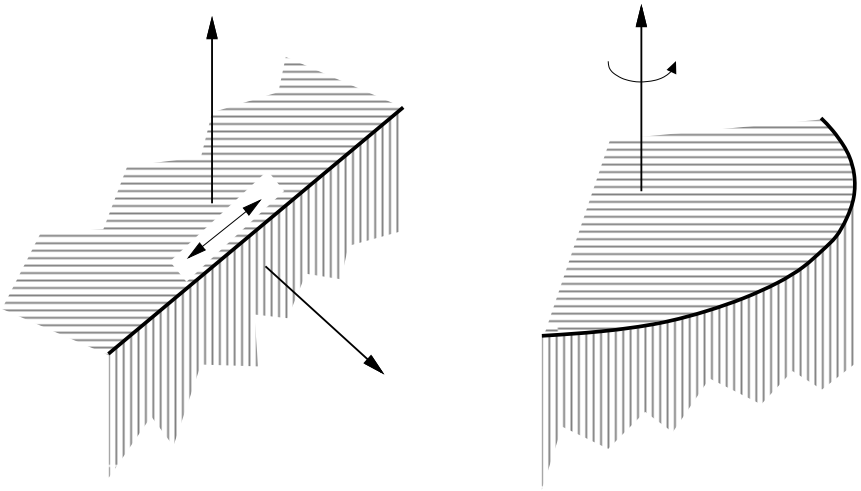


Figure 9.8. *Les deux caractéristiques extraites des images de profondeur par le système 3DPO.*

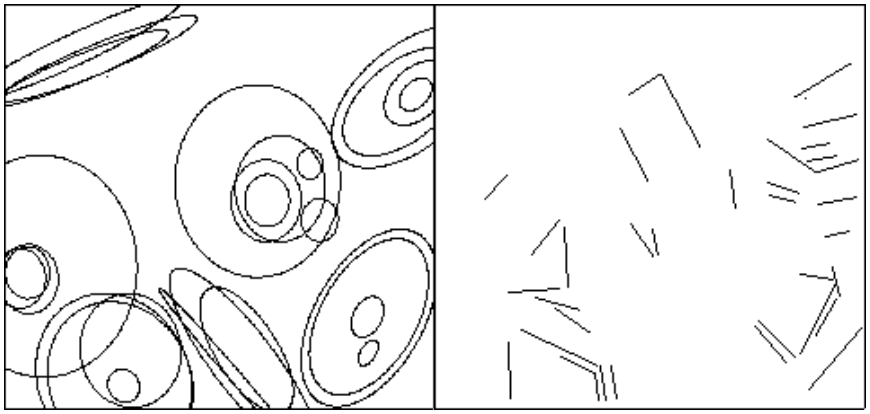


Figure 9.9. *Les arêtes circulaires et droites détectées dans l'image de profondeur.*

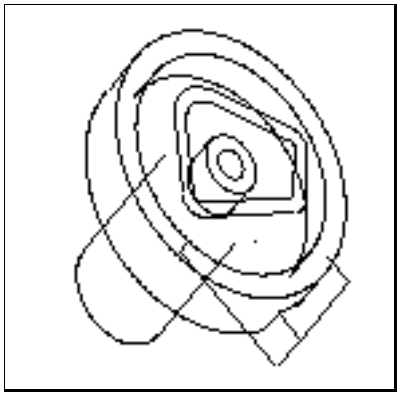


Figure 9.10. *Une vue "fil de fer" du modèle géométrique de l'objet à reconnaître. Toutes les arêtes de cet objet sont modélisées par des arcs de cercle et des segments de droite.*

on construit une image de profondeur *synthétique* qui est comparée à l'image de profondeur de la scène. Cette comparaison permet de décider pour chaque objet quels sont les objets qui le cachent. La figure 9.12. montre l'image synthétique ainsi obtenue.

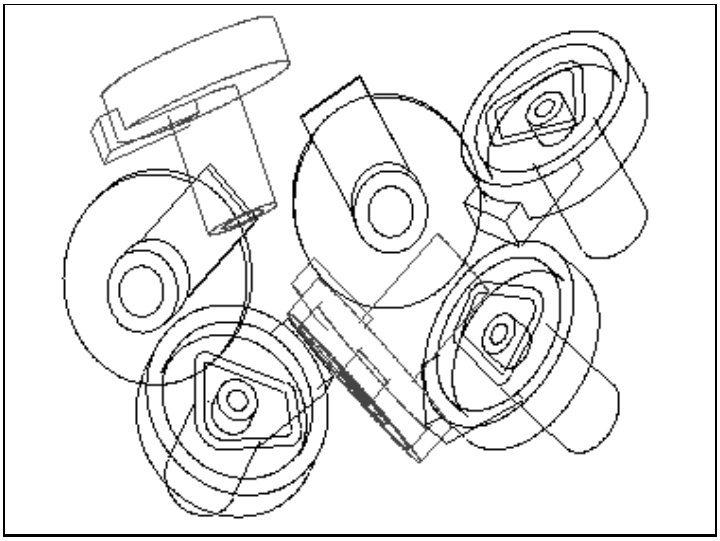


Figure 9.11. *Résultat de la localisation de 7 objets identiques dans la scène observée. La précision de la localisation dépend du nombre d'arêtes visibles de chaque objet ainsi que de la précision avec laquelle chaque arête est détectée dans l'image.*

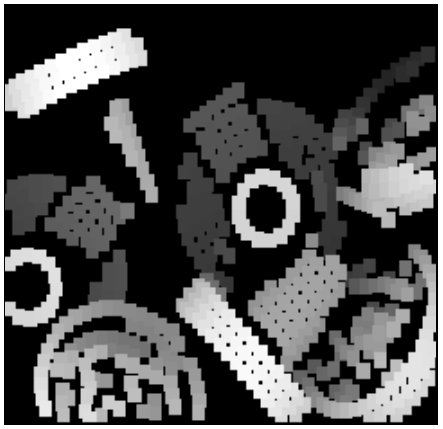


Figure 9.12. *Image synthétique obtenue en utilisant la position et l'orientation précédemment calculées et en éliminant les points d'un objet cachés par les points d'un autre objet.*

Chapitre 10

Des images 2D aux surfaces polyédriques

10.1. Introduction

On se pose le problème de la reconstruction d'une scène 3D à partir d'images 2D par vision stéréoscopique [61]. On considère la scène comme une surface polyédrique, les images comme ses projections, et on recherche un polyèdre approximant. *Ce chapitre présente des méthodes permettant de générer une approximation polyédrique des objets à partir des résultats de la vision stéréoscopique. Ces méthodes peuvent être utilisées en aval de la vision stéréo (voir chapitre 6) pour simplifier le problème de reconnaissance.*

10.2. Approche régions

L'idée développée par l'approche régions, initialisée dans [36], est de segmenter en zones homogènes les deux images d'une paire stéréoscopique. Si on suppose que la scène est un polyèdre dont les facettes ont des niveaux de gris homogènes, les régions correspondent à la projection de ces facettes planes. La mise en correspondance de ces régions permet l'obtention de couples de régions homologues, c'est à dire associées à la même facette plane. Un exemple d'algorithme de mise en correspondance de régions par recherche de cliques maximums dans le graphe d'association est décrit dans [36]. Les facettes planes 3D s'obtiennent en calculant la meilleure homographie mettant en correspondance les segments de contour formant les frontières des deux régions. Les avantages de cette approche sont :

• la mise en correspondance des régions s'effectue relativement aisément, à cause du grand nombre d'attributs discriminants des primitives. En effet, dans la pratique l'utilisation des caractéristiques des régions et des graphes d'adjacence suffit pour la mise en correspondance (les contraintes géométriques ne sont pas indispensables surtout pour des images couleurs) ;

• on obtient des morceaux de plans dans l'espace formant une approximation de la surface. Une étape ultérieure d'approximation n'est donc pas nécessaire.

Cette méthode comporte néanmoins des restrictions :

• la validité des régions obtenues dépend des critères d'homogénéité choisis [127]. Or, pour des images naturelles bruitées, les résultats de segmentation sont parfois instables ;

• les régions définissent peu d'invariants projectifs ce qui les rend difficiles à manipuler en dehors des bonnes hypothèses (projection d'une même facette plane). En fait il n'est pas aisé de relier les propriétés radiométriques des régions avec les surfaces associées.

Cette approche régions a été étudiée dans plusieurs laboratoires [37, 175, 178] mais se heurte, pour l'instant, au fait que les propriétés radiométriques des objets sont actuellement encore peu étudiées [107]. L'approche contour, à laquelle est consacrée la partie suivante, extrait des informations fiables mais réduites de l'image, et repousse une partie des problèmes sur la structuration de ces informations.

10.2.1. Mise en correspondance de régions

Les algorithmes de segmentation en régions (voir chapitre 4 "Segmentation d'images en régions") fournissent une partition S d'une image qui peut être représentée par un graphe d'adjacence $\Delta = (S, L)$ où :

$S = \{R_1, R_2, \ldots, R_n\}$ est l'ensemble des régions,

R_i est l'étiquette de la région qui a été donnée par la segmentation,

$L = < R_1, R_2 >, \ldots$ est l'ensemble des arêtes du graphe,

$< R_i, R_j >$ existe si R_i et R_j sont des régions voisines, ainsi les nœuds R_i et R_j sont adjacents.

Le graphe Δ est un graphe valué : un vecteur attribut $A(R_i)$ est associé à chaque région, soit à chaque nœud.

On associe aussi un vecteur attribut $G(R_i, R_j)$ à chaque arc $< R_i, R_j >$.
Soit $\Delta_L = (S_L, E_L)$ et $\Delta_R = (S_R, E_R)$ les graphes représentant les segmentations des images droites et gauches où :

$S_L = \{L_1, \ldots\ldots L_n\}$ est l'ensemble des régions de l'image gauche ;

$S_R = \{R_1, \ldots . R_m\}$ est l'ensemble des régions de l'image droite.

Dans le cas où les deux images correspondent à deux vues stéréoscopiques, il existe une disparité entre l'image droite et l'image gauche (voir chapitre 6 : "Vision stéréoscopique"). Ainsi, les partitionnements obtenus pour les deux images sont différents et leur mise en correspondance revient à apparier de manière inexacte deux graphes valués. Une méthode efficace est d'utiliser la notion de graphe d'association introduite par Ambler [3]. On décrit dans la suite un exemple d'algorithme basé sur le graphe d'association.

10.2.1.1. Mise en correspondance et graphe d'association

Soit $\Delta_L = (S_L, E_L)$ et $\Delta_R = (S_R, E_R)$ les deux graphes résultant de la segmentation des deux images de la paire d'images stéréoscopique.

Le graphe d'association $G = (S, L)$ se construit comme suit :

• si deux régions $L_i \in S_L$ et $R_j \in S_R$ sont compatibles alors l'arc $< L_i, R_j >$ est inclu dans l'ensemble S ;

• un arc $< L_i, R_j >$ est connecté à un autre arc $< L_m, R_n >$ du graphe d'association si l'appariement de L_i avec R_j est compatible avec l'appariement de L_m avec R_n.

On peut considérer que la meilleure mise en correspondance est associée au plus grand ensemble de nœud-correspondances qui sont mutuellement compatibles. Dans le graphe d'association cela correspond au plus grand ensemble de nœuds totalement connecté : c'est à dire une clique. Ainsi les meilleures mises en correspondance sont déterminées par les cliques maximums.

Comme la mise en correspondance intervient après la segmentation, nous n'avons pas de connaissances à priori sur les images, ainsi nous utilisons seulement des relations de bas niveau pour construire le graphe d'association. En conséquence le graphe d'association peut être incomplet, nous ne rechercherons donc pas la clique maximum mais simplement une composante connexe "la plus connectée possible".

10.2.1.2. Une méthode de mise en correspondance

Cette méthode décrite dans [36] comporte deux étapes principales.

La première S_1 est un processus d'initialisation qui construit des paires de régions homologues (région gauche et région droite).

La seconde S_2 utilise le résultat de l'initialisation pour construire des composantes connexes dans le graphe d'association, nous l'appellerons "mise en correspondance secondaire". Cette étape revient à propager la mise en correspondance en utilisant les relations de connexité.

10.2.1.2.1. Processus d'initialisation

Soient $\Delta_L = (S_L, E_L)$ et $\Delta_R = (S_r, E_R)$ deux graphes tels que ceux définis précédemment.
Chaque région $L_i \in S_L$ est associée à un vecteur attribut $A(L_i)$, de même pour chaque région $R_j \in S_R$.
Ce vecteur attribut peut par exemple comprendre les attributs définis au chapitre 4 : taille, niveau de gris moyen, variance, amplitude de variation des niveaux de gris.

Cette étape S_1 classe les régions L_i en fonction de la première composante du vecteur attribut qui peut être, par exemple, l'aire de la région. Ainsi, pour chaque région L_i prise dans cet ordre, on recherche une région R_j qui vérifie un nombre de prédicats, égal à la taille du vecteur attribut. Chaque prédicat P_k teste la kième composante du vecteur attribut. On définit P_k comme suit :

$$P_k(R_j) = \text{vrai} \Longleftrightarrow (r(A_k(L_i), A_k(R_j)) > \delta_k)$$

où

$$A_k(L_i) \text{est la kième composante de} A(L_i)$$

$$r(x,y) = \frac{MIN(x,y)}{MAX(x,y)}; x, y \in R^+ - \{0\}$$

$$\delta_k \text{ étant un seuil donné}$$

Si plusieurs régions vérifient les prédicats, le meilleur candidat pour la mise en correspondance est sélectionné par rapport à la valeur d'une fonction de coût f_1 donnée par :

$f_1 = \sum_k (1 - r(A_k(L_i), A_k(R_j)); k = 1, 2, 3, 4$ si la taille du vecteur attribut est 4.

Le processus S_1 met les paires de régions (L_i, R_j) dans une liste F_1 ordonnée par rapport à $A_1(L_i)$. On définit ainsi pour chaque région gauche L_i la région droite dont les attributs sont les plus proches (R_j).

10.2.1.2.2. Définitions

Avant de décrire la recherche de composantes connexes du graphe d'association, donnons quelques définitions :
Définition d'un anneau :
Soit $\Delta = (S, E)$ le graphe de régions de l'image segmentée.
Soit $X \in S$ une région, nous noterons $H_X = (V_X, E_X)$ le sous-graphe qui contient tous les sommets connectés à X, X est appelée la région milieu de l'anneau H_X.

Définition d'une fonction d'orientation :
Soit $\Delta = (S, E)$ un graphe, soit $X \in S$ et $Y \in V_X$ deux régions voisines. On notera $\theta(X, Y)$ la fonction suivante :

$$\theta(X, Y) : R \times R \rightarrow [0, 2\pi]$$

Soient M_x et M_Y le centre d'inertie de X et Y, la valeur de $\theta(X, Y)$ est donnée par l'angle formé par $M_X M_Y$ et l'horizontale.

Définition d'un secteur :
Soit $\Delta_L = (S_L, E_L)$ et $\delta_R = (S_R, E_R)$ les deux graphes d'une paire stéréoscopique segmentée. Soit $X \in S_L$ et $W \in S_R$ deux régions associées respectivement à leurs anneaux : $H_X = (V_X, E_X)$ et $H_W = (V_W, E_W)$. Soit $Y \in V_X$ une région, un secteur noté $S_{X,Y}$ est l'ensemble des régions définies comme suit (voir figure 10.1.) :

$$U \in V_W$$

$$\theta(W, U) \leq \theta(X, Y) + \delta_\theta$$

et

$$\theta(W, U) \geq \theta(X, Y) - \delta_\theta$$

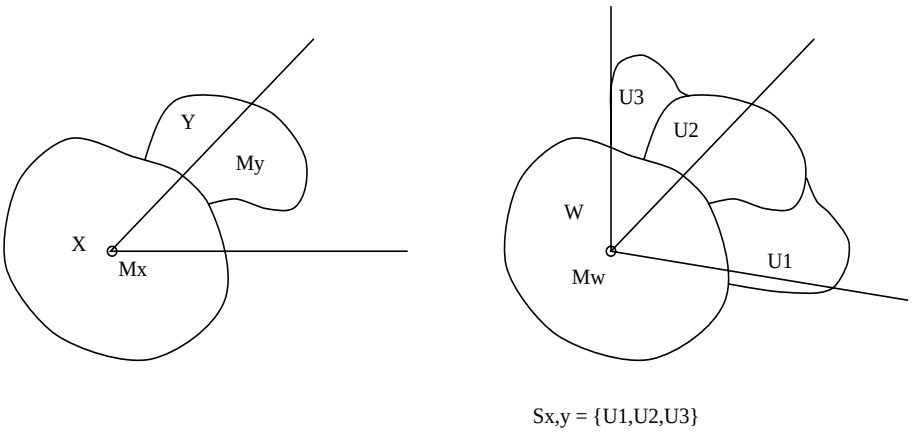


Figure 10.1. *Définition d'un secteur.*

10.2.1.3. Recherche de composantes connexes

Cette étape consiste à rechercher les composantes connexes les plus proches des "cliques maximums". On notera que seule une partie du graphe d'association est construite ce qui évite une complexité algorithmique trop importante. Cela est réalisé par le processus S_2 qui s'initialise en prenant le premier élément (X, W) de la liste F_1 pour l'insérer dans une seconde liste F_2 formant ainsi le premier nœud du graphe d'association. On exécute ensuite les étapes suivantes :

1. *mise en correspondance inexacte des anneaux, H_X et H_W :*
Les régions milieux de ces deux anneaux sont respectivement X et W. Cette étape fournit des nouveaux nœuds (Y,U) où $Y \in H_X$ et $U \in H_W$. Ces nouveaux nœuds sont connectés avec (X, W) et sont placés dans une liste F_2. Un coût de mise en correspondance $C_2(Y, U, X, W)$ est associé avec chaque nœud de F_2.

2. *propagation et ré-examen de vieilles paires* :
Les paires de régions obtenues à partir de l'étape 1 peuvent être utilisées comme régions milieux pour mettre en correspondance de nouveaux anneaux et pour construire de nouveaux nœuds du graphe d'association. En fait, si un candidat couple (Y, U) est proposé comme un nouveau nœud, les paires existant dans F_2 sont examinées pour vérifier si Y ou U ont déjà été associées avec d'autres régions. Dans ce cas, les fonctions coût de la nouvelle et de la vieille paire sont comparées. La meilleure paire est sélectionnée, si c'est la nouvelle on l'insère

dans la liste F_2 à la place de l'ancienne.

3. *fin du processus* S_2 :
L'étape 2 est exécutée jusqu'à ce qu'il n'y ait plus de propagation et de réexamen de paires de régions existantes. Cet état survient lorsque :

- toutes les paires de régions possibles ont été examinées,
- toutes les paires possibles de régions d'une composante connexe ont été examinées.

4. *restauration du processus* S_2 *ou fin de la mise en correspondance* :
S'il existe plusieurs composantes connexes dans le graphe d'association (voir figure 10.2.), le processus S_2 stoppe après la construction d'une composante. Alors la liste F_1 est utilisée pour restaurer le processus S_2. Le premier élément de la liste F_1 qui n'est pas inclu dans la liste F_2 est placé dans F_2, il constitue le premier nœud d'une nouvelle composante connexe. Ainsi l'étape C est répétée.

Si toutes les paires de la liste F_1 ont été examinées, le processus S_2 s'arrête.
Le processus S_2 s'arrête toujours car les deux graphes sont finis.

10.2.1.4. Correspondance durant la construction d'une composante connexe

La mise en correspondance est réalisée par le processus S_2. Deux régions mises en correspondance X et W sont considérées comme les régions milieux de deux anneaux H_X et H_W. Les candidats $U \in H_W$ pour un appariement avec une région $Y \in H_X$ sont choisis à l'intérieur du secteur $S_{X,Y}$. Ils doivent vérifier plusieurs prédicats de la même forme que ceux utilisés pour l'étape d'initialisation. On peut utiliser des prédicats prenant en compte l'aire, la moyenne des niveaux de gris et les dimensions du rectangle d'aire minimum des régions examinées.

Pour chaque paire (Y, U) une fonction de coût $C_2(Y, U, X, W)$ composée de deux parties est calculée :

$$C_2(Y, U, X, W) = d(Y, U) + f_a((X, Y), (W, U))$$

Le terme $d(Y, U)$ prend en compte la disparité des attributs associés aux régions Y et U et le terme $f_a((X, Y), (W, U))$ la disparité entre les arcs $< X, Y >$ et $< W, U >$ des deux graphes de régions.

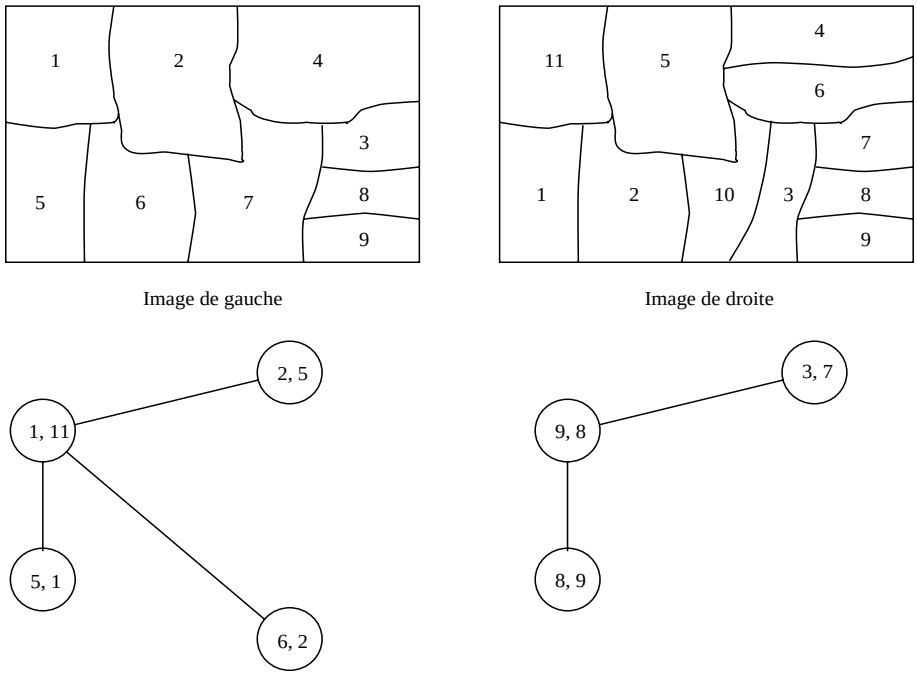


Figure 10.2. *Composantes connexes du graphe d'association.*

10.2.1.5. Résultats de mise en correspondance et conclusion

Cet algorithme a été testé sur des scènes d'intérieur à partir de segmentations obtenues par croissance optimale de régions (voir chapitre 4 "Segmentation d'images en régions") :. On obtient une mise en correspondance de deux graphes comprenant approximativement 150 régions en quelques secondes CPU sur une station SUN. On remarquera que dans l'implantation décrite précédemment seules les relations de connexité et les attributs des régions sont prises en compte. On peut cependant, sans rien changer à la structure algorithmique, rajouter des contraintes géométriques liées à la vision stéréoscopique comme la contrainte épipolaire (voir chapitre 6). On notera que l'algorithme décrit n'est qu'un exemple de méthode de mise en correspondance inexacte de graphe appliquée à l'appariement de régions. D'autres méthodes, au moins aussi performantes existent, Son principe est identique à celui de l'algorithme de mise en correspondance de segments décrit dans le chapitre 6, section 6.5.4. néanmoins elles reposent, pour la plupart, sur des principes similaires.

10.2.2. Reconstruction de facettes planes

Après l'obtention d'un ensemble de couples de régions homologues se , le problème se pose de savoir comment exploiter ces résultats pour la reconstruction de surfaces. Pour cela, on peut supposer que chaque couple de régions appariées est issu de la projection d'un même objet plan. Cette supposition est en particulier raisonnable dans le cas de scènes d'intérieur où il existe beaucoup de surfaces planes telles que les murs ou les plafonds. On peut attacher à chaque région les segments qui correspondent à l'approximation polygonale de ses points frontières. Ensuite on apparie les segments correspondant à deux régions homologues. Grâce au faible nombre de segments pour chaque frontière de région, ce problème est aisément résolu. La mise en correspondance des segments rend possible le calcul du meilleur plan 3D, au sens des moindres carrés, sur lequel la surface, dont la paire de régions est la projection, s'appuie.

Pour ce calcul, on utilise une méthode décrite dans [118] et dont les principes sont les suivants. Quand un point 3D se déplace dans un plan, les relations entre les coordonnées de ses projections sur les images gauche et droite sont linéaires en coordonnées projectives c'est à dire homographiques dans l'espace cartésien. Ainsi, si nous connaissons au moins quatre lignes non parallèles et

leurs correspondants, on peut calculer la meilleure matrice de transformation par le moyen d'une technique de moindres carrés. Enfin, de la matrice de transformation, on déduit les paramètres du plan 3D correspondant. Ces calculs sont décrits précisément dans [118].

On attache donc à chaque région un plan 3D sur lequel s'appuie la surface dont elle est la projection. On obtient ainsi des surfaces 3D planes existant dans la scène 3D.

On peut noter que l'utilisation de régions pour la vision stéréoscopique se justifie lorsqu'il est possible de déduire facilement des informations sur les surfaces 3D de la scène à partir de leurs projections sur les plans image. Dans le cas où une analyse avancée des scènes 2D est nécessaire pour relier les objets aux régions et les régions aux surfaces 3D constituant les objets, l'opportunité de ce type d'approche n'est pas évidente.

10.3. Approche contours

Le principe de l'approche contours est d'extraire dans les deux images stéréoscopiques des segments de points de contours correspondant à la projection des arêtes de la surface polyédrique [12]. Ces segments sont mis en correspondance de manière à reconstruire des segments 3D formant les arêtes du polyèdre (voir chapitre 6). On obtient ainsi des segments 3D répartis dans l'espace et on se pose le problème de reconstruire le polyèdre dont les arêtes sont ces morceaux de droite. Pour cela, on peut faire des hypothèses sur la surface et mettre en œuvre des méthodes de minimisation [74], avec la difficulté de déterminer une bonne fonctionnelle à optimiser incluant les discontinuités et un algorithme d'optimisation efficace...

Une autre alternative est de ramener ce problème d'approximation à un problème combinatoire, en le réduisant considérablement grâce à la contrainte de visibilité. En effet les triangles, définis par les centres optiques des caméras et les segments 3D, traversent l'espace libre (dans l'hypothèse de non-transparence des objets). Ces triangles forment un ensemble de rayons. On est donc ramené à rechercher les polyèdres, pas nécessairement convexes, dont les arêtes sont les segments et qui n'intersectent aucun rayon.

Des méthodes de géométrie algorithmique développées par ailleurs [19, 20] apportent des solutions élégantes et efficaces à ce problème. Deux approches de ce type, assez complémentaires, ont été proposées [18, 17].

L'approche présentée dans [18] consiste à définir un ensemble de plans de coupe des données (segments 3D), à déterminer dans chaque plan la section de la surface, puis à relier les sections des plans successifs. Si les plans contiennent les rayons de tous leurs points (intersection du plan et des segments) un algorithme permet de reconstruire une solution unique pour chaque section. Ensuite on obtient un polyèdre en reliant les polygones définissant les sections avec une méthode de triangulation optimale. Dans le cas d'une surface polyédrique simple cet algorithme fournit une solution exacte, sinon un algorithme de reconstruction par coupes basé sur la triangulation de Delaunay peut être mis en œuvre [20].

La force de cette méthode est essentiellement l'utilisation d'une méthode de reconstruction 2D non heuristique et de faible complexité algorithmique [1] pour obtenir des sections planes successives. On obtient ainsi un coût calculatoire très faible pour déterminer l'espace libre à partir d'un couple d'images stéréoscopiques. L'inconvénient réside dans la nature 2.5D de cet algorithme. En effet, dans le cas de plusieurs données stéréo, la reconstruction de la surface entre les plans est heuristique et la mise à jour de la représentation n'est pas immédiate.

Une méthode de reconstruction réellement 3D est décrite dans [17]. Le principe est de déterminer la triangulation de Delaunay 3D des extrémités des segments en imposant la contrainte que les segments soient tous inclus dans une arête de Delaunay. Ensuite, on supprime les facettes des tétraèdres traversées par un rayon (sculpture). L'inconvénient de cet algorithme par rapport au précédent est essentiellement une complexité algorithmique beaucoup plus importante liée en particulier à l'étape de sculpture. De plus l'aspect heuristique de la triangulation de Delaunay offre moins de possibilités pour induire des stratégies actives de prises de mesure [2].

En effet dans la réalité de la robotique le problème se pose souvent de la manière suivante : soit un capteur, par exemple une caméra vidéo, comment générer une suite de prises de mesure permettant de reconstruire à coup sûr un ensemble d'objets polyédriques ? Ce problème est en partie résolu dans le cas 2D [2] et on peut penser qu'une adaptation de ces algorithmes aux données stéréoscopiques serait possible.

10.3.1. Reconstruction à partir de méthodes de rayons

10.3.1.1. Rayons

On suppose donc que les segments 3D issus du processus de vision stéréoscopique [12] correspondent aux arêtes d'une surface polyédrique. Afin de réduire l'ensemble des solutions pour la reconstruction de la scène, on introduit une information supplémentaire : les rayons.

Les segments détectés sont observés depuis différents points de vue situés aux centres optiques des caméras. Les triangles pleins formés par un segment et un centre optique de caméra traversent donc l'espace libre et définissent des rayons (voir figure 10.3.). En effet, on suppose les objets opaques et la contrainte de visibilité respectée.

La reconstruction de la scène se ramène donc à la recherche de surfaces polyédriques dont les arêtes sont les segments et qui ne sont traversées par aucun rayon.

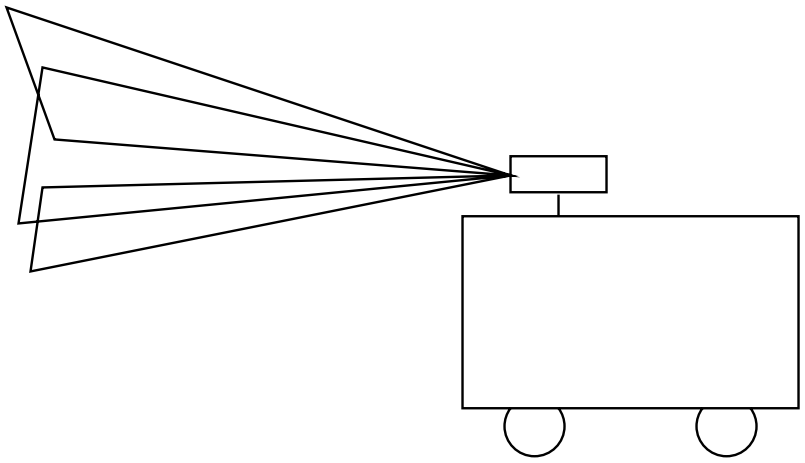


Figure 10.3. *Robot voyant des segments.*

10.3.1.2. Description de l'algorithme

Cette approche comporte trois étapes principales :

1. détermination d'un ensemble de plans de coupe : calcul d'une suite de sections planes des segments 3D.

2. reconstruction de polygones dans chaque plan de coupe : utilisation de méthodes de reconstruction de formes planes à partir de rayons.

3. reconstruction de surfaces entre deux plans successifs : connexion des polygones obtenus par des algorithmes de triangulation.

10.3.1.2.1. Détermination des plans de coupe

On détermine une suite de plans de coupe (P_i) telle que :

1. les sections des segments avec les (P_i) ne s'intersectent pas deux à deux : la détermination de la surface se réduit donc à sa reconstruction entre chaque couple de plans successifs.

2. chaque extrémité de segment est contenue dans un plan P_i pour une valeur de i : les parties de surfaces entre les plans consécutifs peuvent ainsi être déterminées sans ambiguité.

3. chaque rayon issu d'une extrémité de segment est contenu dans un plan P_i pour une valeur de i : la section de la surface dans chaque plan de coupe peut alors être obtenue par des méthodes de reconstruction de formes planes avec des rayons.

Il faut noter que si (1) et (2) peuvent être exactement vérifiés pour des données stéréoscopiques, il en est autrement pour (3). En effet l'existence d'une suite ordonnée de plans contenant tous les rayons des extrémités des segments n'est assurée que si les centres optiques des caméras sont alignés [18].

Ainsi deux cas se présentent :

1. les centres optiques des caméras sont alignés : on détermine un faisceau de plans passant par la droite définie par les centres optiques (C) et contenant les extrémités des segments (s_i) (voir figure 10.4.).

2. les centres optiques des caméras ne sont pas alignés : on détermine un ensemble de coupes parallèles contenant les extrémités des segments, et tels que les rayons peuvent être approximés par leur projection sur les plans (voir figure 10.5.).

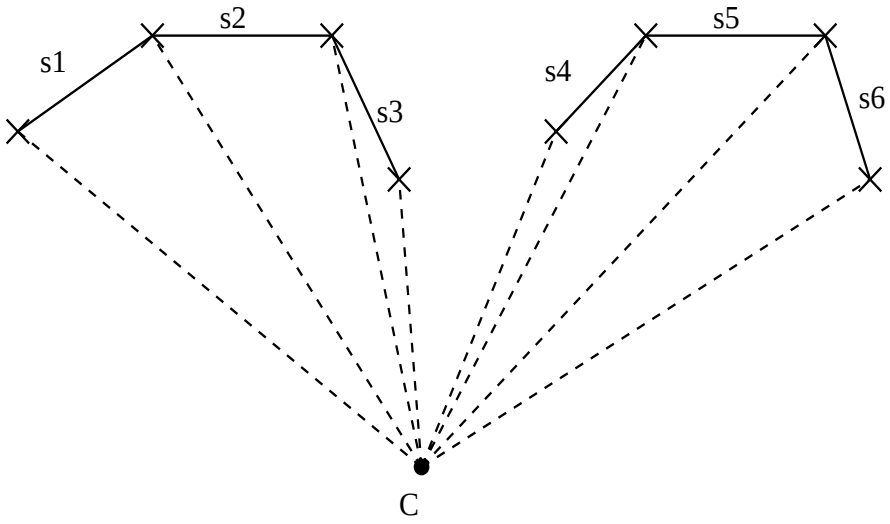


Figure 10.4. *Rayons et segments.*

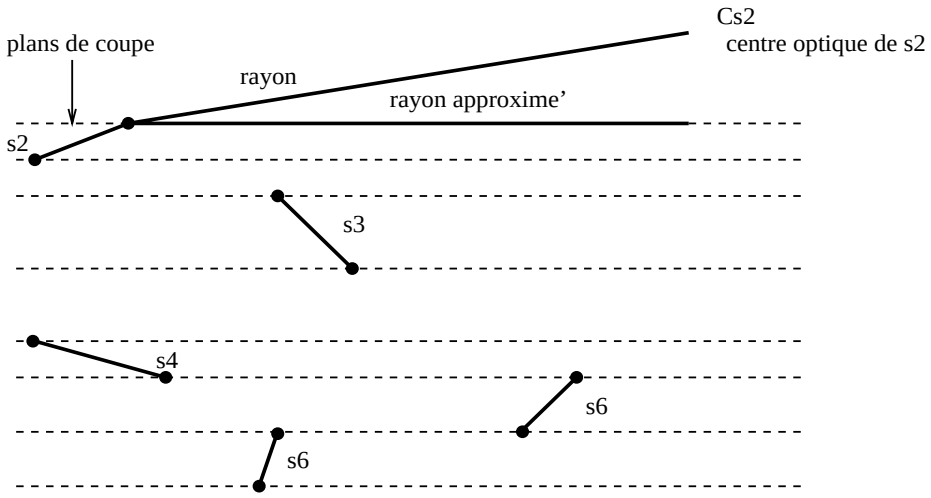


Figure 10.5. *Approximation des rayons, les plans sont en pointillés.*

10.3.1.3. Reconstruction dans les plans de coupes

Cette étape pose le problème de la reconstruction de polygones non convexes à partir de leurs sommets. En général, pour un ensemble donné de sommets on obtient plusieurs solutions pour le contour (voir figure 10.6.).

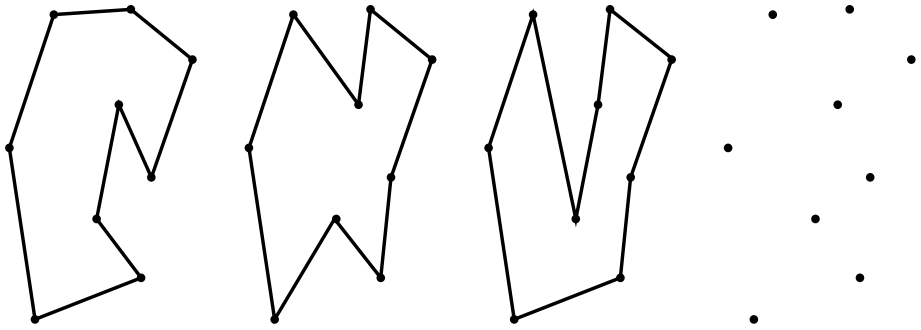


Figure 10.6. *Points mesurés et quelques solutions pour le contour de l'objet.*

Si on ajoute des rayons associés aux points mesurés on obtient une solution unique dans le cas d'un seul objet [1] (voir figures 10.9., 10.10. et 10.11.). La méthode décrite dans [1] permet de déterminer en temps optimal ($O(nlog(n))$ un ordre total sur l'ensemble des points mesurés, définissant le polygone solution dans le cas où ces points appartiennent à un objet unique et simplement connexe.

Le principe de cette méthode est basé sur l'algorithme suivant [1] :
Soit $p = \{p_1, p_2, \ldots p_n\}$ l'ensemble des points mesurés et $L = \{L_1, L_2, \ldots L_n\}$ l'ensemble des rayons correspondants. On suppose que les L_i sont des courbes semi-infinies ayant pour origines les points P_i. Le problème est de joindre les points de P par des segments de droite qui n'intersectent pas les rayons de L ce qui revient à trouver une approximation polygonale de l'objet (voir figure 10.7.).

On montre les deux théorèmes suivants [1] :
Théorème 1
Si on suppose, sans perte de généralité, qu'un sommet de l'enveloppe convexe des points mesurés (P) est choisi comme origine et un sens de parcours donné (par exemple le sens des aiguilles d'une montre). Alors les sommets de l'enveloppe convexe apparaissent dans le même ordre dans la solution polygonale du problème lorsque l'on parcourt les sommets de l'enveloppe convexe.

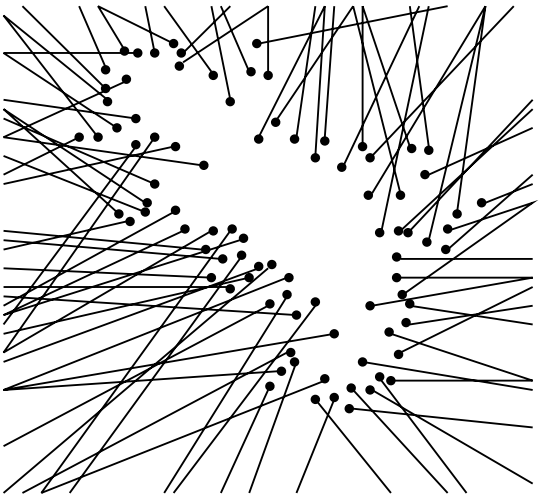


Figure 10.7. *Points et rayons.*

Théorème 2
Soient a_i et a_{i+1} deux sommets consécutifs de l'enveloppe convexe. Alors tout point C dont le rayon intersecte le segment $[a_i a_{i+1}]$ de l'enveloppe convexe est entre a_i et a_{i+1} dans l'ordre des points définissant la solution du problème.
Le problème de reconstruction se réduit donc à trier les classes de points dont les rayons intersectent la même facette de l'enveloppe convexe CH. Cela est réalisé avec la règle de comparaison suivante :
Règle de comparaison
c_1 est avant c_2 ($c_1 < c_2$) si une des deux propositions suivantes est vraie (voir figure 10.8.) :

1. i_{c_1} (intersection du rayon de c_1 avec $[a_i a_{i+1})$ est avant i_{c_2} sur l'arête orientée $a_i a_{i+1}$ et si le nombre d'intersection entre les rayons correspondants C_1 et C_2 dans l'enveloppe convexe est pair.

2. i_{c_1} est après i_{c_2} sur le contour orienté $a_i a_{i+1}$ et le nombre d'intersection des rayons C_1 et C_2 dans l'enveloppe convexe est impair.

L'algorithme de reconstruction se dérive directement soit :
Algorithme

1. pour chaque sommet c de $P - CH$ trouver la première intersection du rayon C associé à c avec l'enveloppe convexe et former les classes d'équivalence

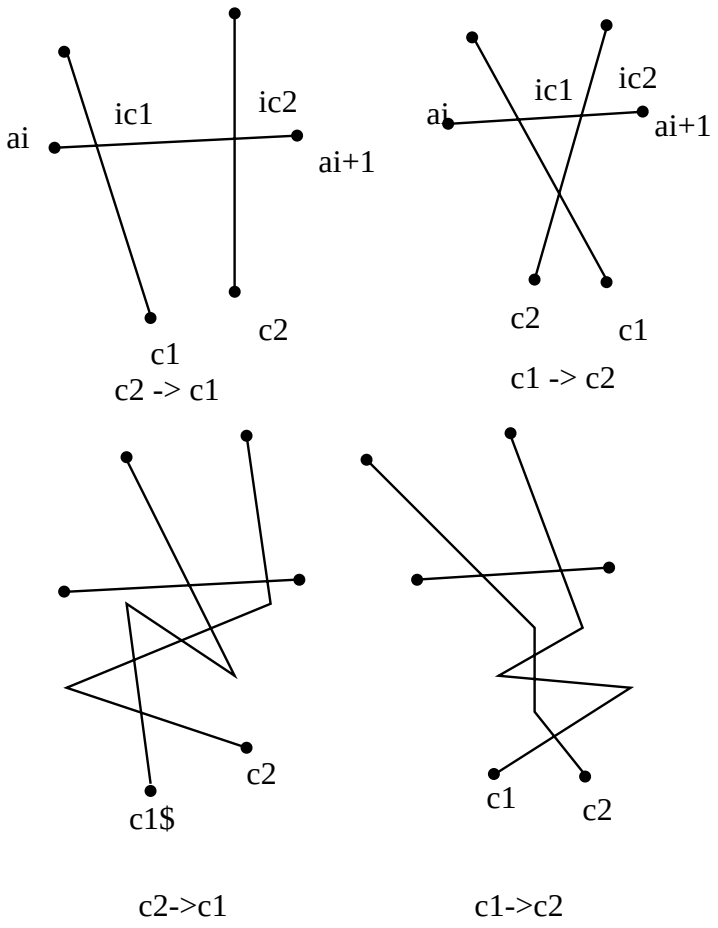


Figure 10.8. *Illustration de la règle de comparaison ; l'ordre correct est donné dans les différents exemples.*

correspondant à tous les sommets de l'enveloppe convexe.

2. trier chaque classe d'équivalence en utilisant par exemple l'algorithme *Quicksort* et la règle de comparaison.

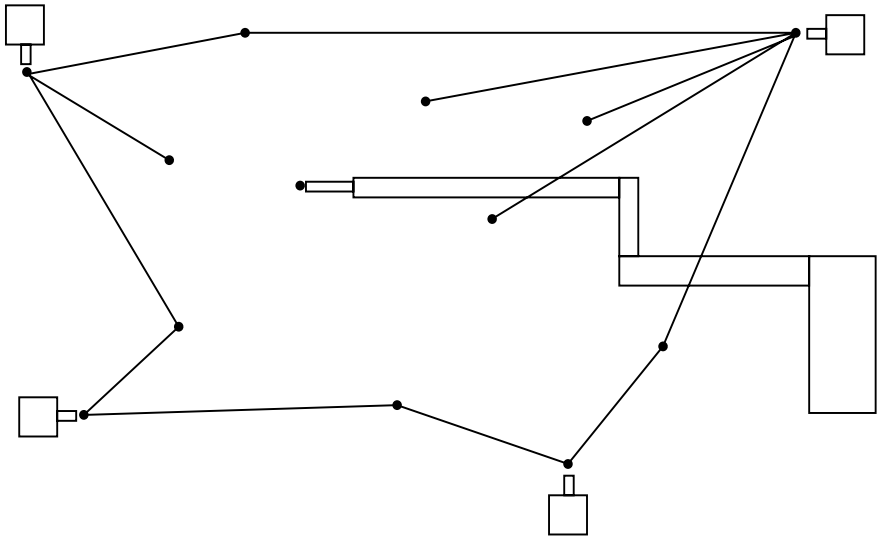


Figure 10.9. *Rayons associés aux points mesurés.*

Lorsque les rayons sont concourants, l'algorithme décrit dans [1] se ramène à un tri des rayons selon leurs orientations (voir figure 10.12.). Dans le cas de segments observés d'un seul point de vue cette méthode peut être utilisée afin de reconstruire la section de la scène dans chaque plan de coupe.

Dans le cas de plusieurs objets, l'information induite par les rayons permet d'obtenir une reconstruction incluant la solution exacte [1]. Un algorithme consistant à suivre les enveloppes externes des rayons [18] permet alors de déterminer les polygones correspondant aux frontières des objets. Cette méthode peut être utilisée lorsque les segments sont observés de différents points de vue. Son principe est d'identifier les plus petits polygones contenant les objets par calcul des points d'intersection des rayons entre eux (voir figure 10.13.).

Si on implante ces deux algorithmes (reconstruction par tri et reconstruction par suivi des enveloppes externes) sur une station de travail SUN-3, on obtient des temps de calcul de l'ordre d'une seconde pour une centaine de points. Les figures 10.14. à 10.18. présentent quelques résultats obtenus sur des données

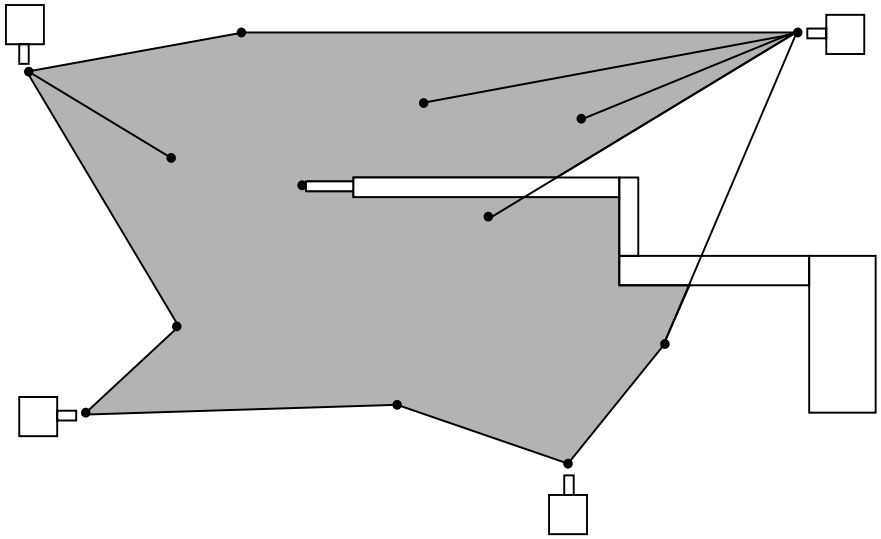


Figure 10.10. *Plus petit polygone dans lequel est inclu l'objet (enveloppe externe des rayons).*

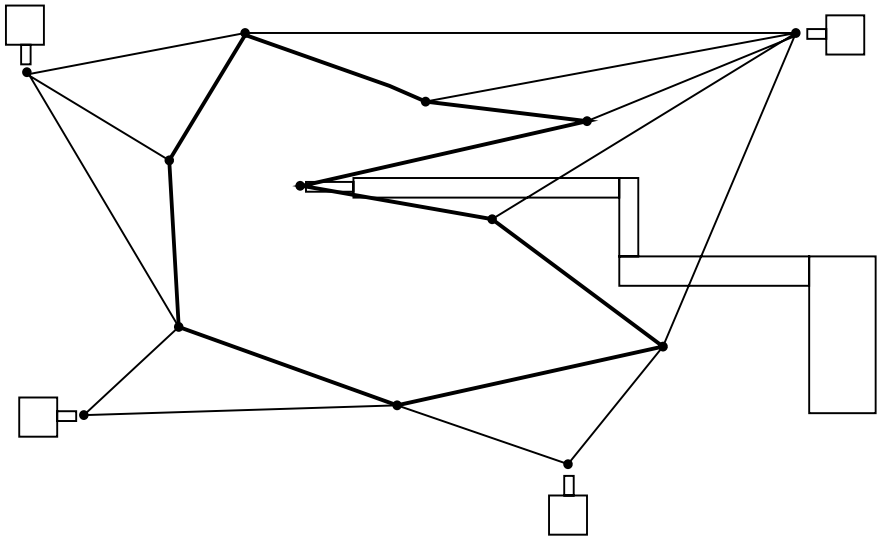


Figure 10.11. *Solution unique pour le contour de l'objet dans le cas d'une forme polygonale pour laquelle on a mesuré tous les sommets.*

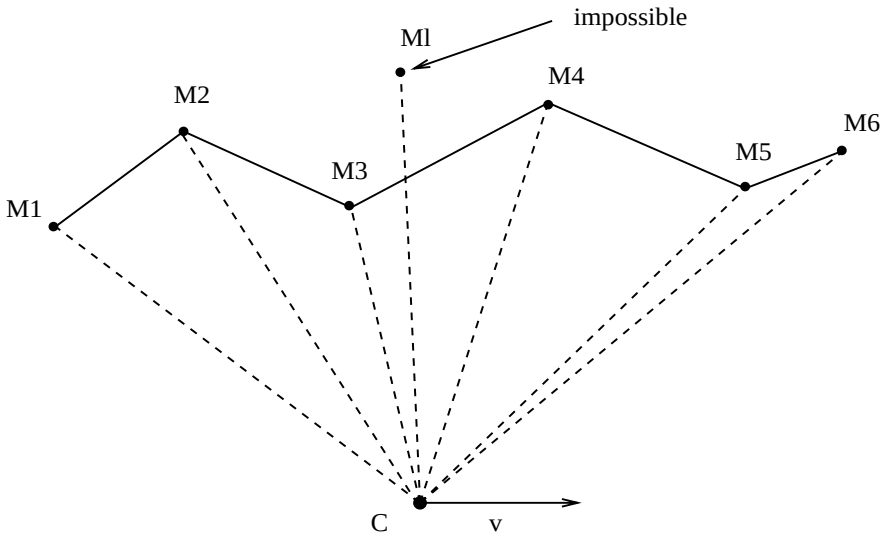


Figure 10.12. *Tri des rayons selon leurs orientations.*

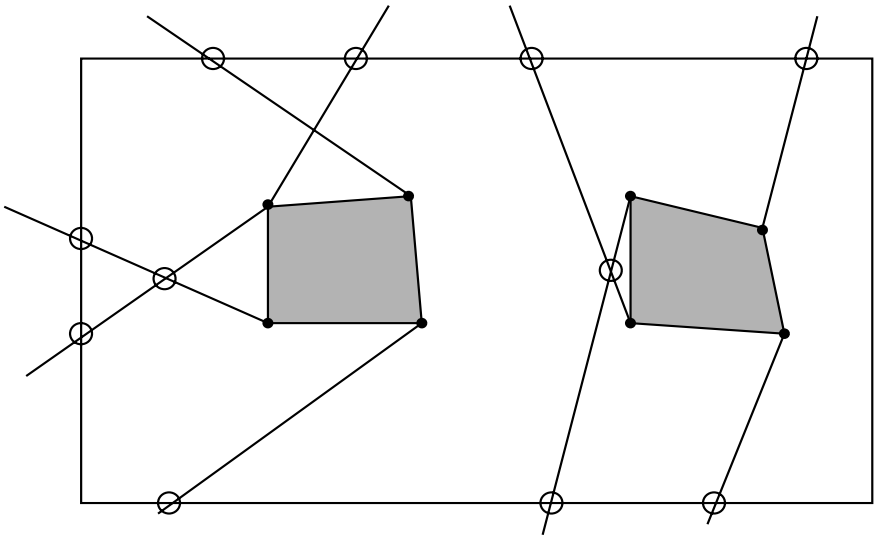


Figure 10.13. *Suivi de l'enveloppe externe des rayons.*

synthétiques.

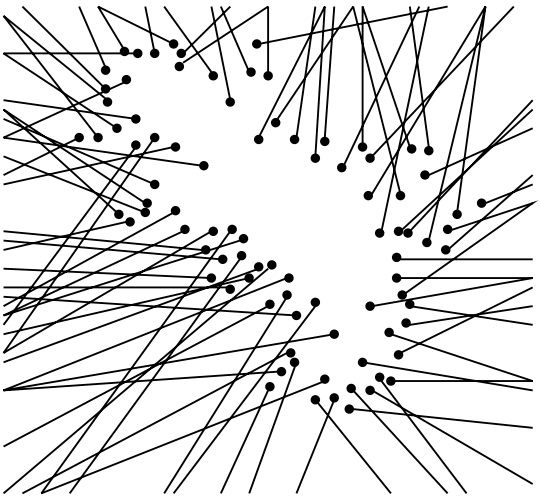


Figure 10.14. *Points et rayons.*

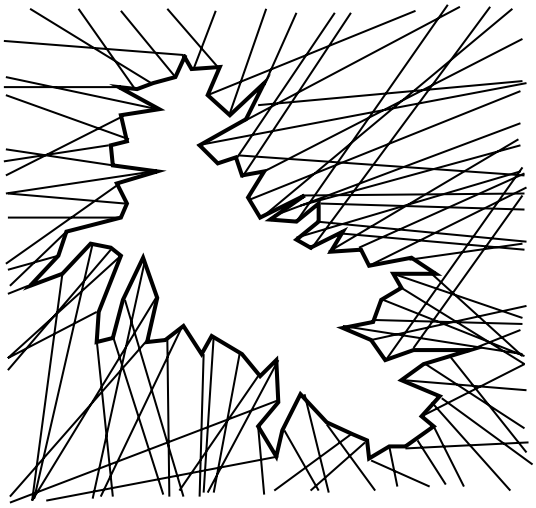


Figure 10.15. *Polygone correspondant à la figure précédente.*

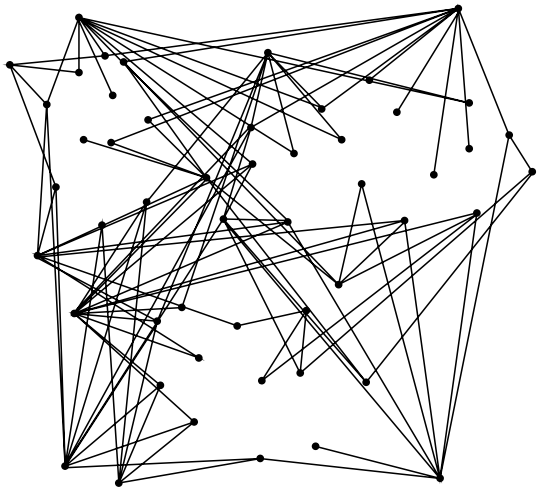


Figure 10.16. *Points et rayons.*

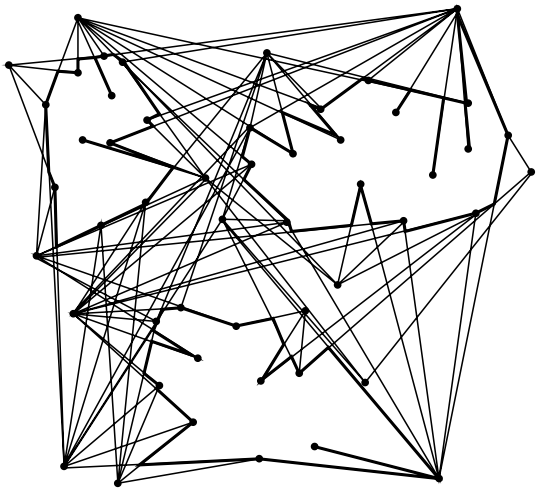


Figure 10.17. *Enveloppes externes des objets correspondant à la figure précédente.*

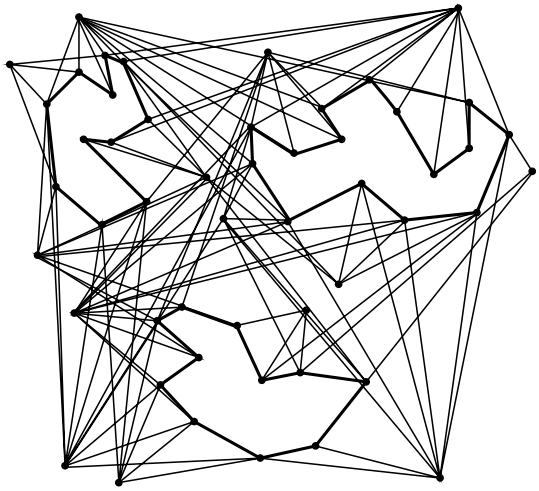


Figure 10.18. *Polygones correspondant aux enveloppes externes de la figure précédente.*

10.3.1.4. Reconstruction de surfaces joignant les polygones

Dans le cas d'un objet simple on peut utiliser une méthode optimale de triangulation introduite par Fuchs [64]. On obtient la triangulation qui minimise la somme de la longueur des arêtes [146] par recherche d'un chemin optimal dans le graphe de triangulation.

Les principes de cet algorithme de triangulation sont les suivants [146] :
Soit $L_1 = (P_1, ..P_m)$ la liste ordonnée des points d'un contour simple situé dans un plan. De même soit $L_2 = (Q_1, \ldots Q_n)$ la liste ordonnée des points d'un autre contour simple situé dans un plan parallèle au premier (voir figure 10.19.).
On peut par exemple supposer que la triangulation recherchée contient les arêtes Q_1P_1 et Q_nP_m. On représente l'ensemble des triangulation possibles à l'aide d'un graphe $G = (C_{ij}, N)$ associé à L_1 et à L_2 (voir figure 10.20.).

Les nœuds de ce graphe notés C_{ij} représentent un segment $[P_iQ_j]$ joignant le point P_i de L_i au point Q_j de L_2. Un arc $A = (C_{ij}, C_{ik})$ de G qui joint C_{ij} à C_{ik} représente le triangle $P_iQ_jQ_k$.

Or la triangulation recherchée vérifie :

1. si le segment $[P_iQ_j]$ appartient à la triangulation alors $[P_{i+1}Q_j]$ ou $[P_iQ_{j+1}]$ appartient aussi à la triangulation. Cela signifie que tout nœud C_{ij}

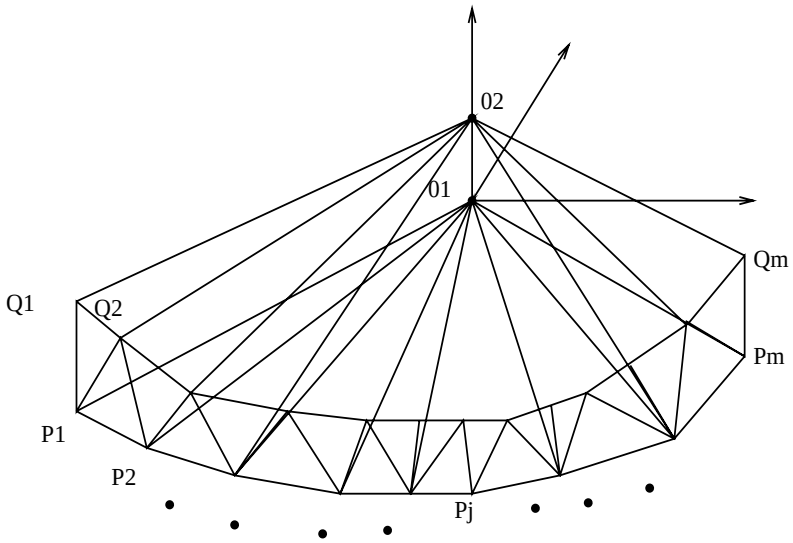


Figure 10.19. *Contours dans des plans parallèles.*

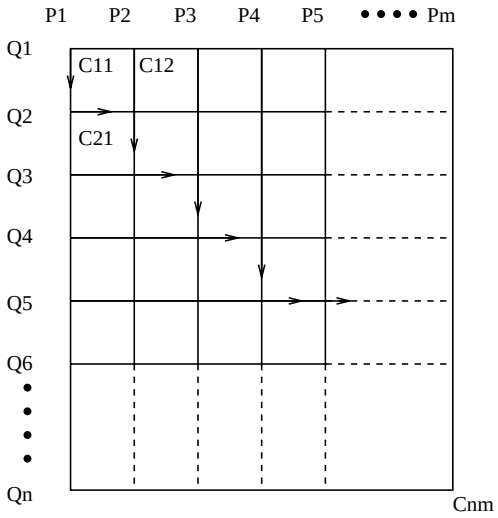


Figure 10.20. *Graphe des triangulations.*

du graphe associé est relié par un arc aux nœuds $C_{i(j+1)}$ et $C_{(i+1)j}$.

2. tout point du contour est relié à au moins un point de l'autre contour. Soit une triangulation correspond à un chemin dans G du nœud C_{11} au nœud C_{mn} passant au moins une fois sur chaque ligne et une fois sur chaque colonne.

3. les arêtes de la triangulation ne se croisent pas : Si un chemin dans G, représentant une triangulation de L_1L_2 passe par C_{ij} et $C_{(i+1)j}$ alors il ne peut pas passer par $C_{i(j+1)}$; d'où l'orientation du graphe G.

Toute triangulation associée aux contours L_1 et L_2 est donc représentée par un chemin de G joignant C_{11} à C_{mn}. On peut associer à un tel chemin un coût caractéristique de la triangulation correspondante : longueur des arêtes [146], volume compris entre les deux plans de coupe ...La recherche de la meilleure triangulation se ramène alors à la recherche d'un chemin optimal dans le graphe des triangulations G. Cet algorithme se met en œuvre aisément en utilisant des relations récursives entre les longueurs des chemins optimaux.

Dans le cas de plusieurs objets on peut utiliser un algorithme de reconstruction basée sur la triangulation de Delaunay [20]. Son principe est de calculer la triangulation de Delaunay des points des deux sections, puis de supprimer les tétraèdres à l'extérieur des objets (chaque section définit des contours fermés) ou provoquant des singularités (s'appuyant sur un point ou une arête).

10.3.2. Reconstruction par sculpture de la triangulation de Delaunay

Une autre alternative développée dans [17] consiste à construire la triangulation de Delaunay [19] des extrémités des segments 3D en contraignant chaque segment à être une arête de Delaunay. La triangulation de Delaunay correspondant au pavage de l'espace le plus isotrope possible (dans le cas 2D on montre que la triangulation de Delaunay est la triangulation la plus équilatérale possible), on obtient ainsi les relations de voisinage les plus vraisemblables. On peut donc supposer que la surface polyédrique contenant les objets de la scène est incluse dans le pavage de Delaunay. La contrainte de visibilité implique que toutes les facettes planes traversées par des rayons (voir partie précédente) n'existent pas (ou sont transparentes). On élimine les facettes des tétraèdres de Delaunay traversés par les rayons définis par les segments 3D et les centres optiques des caméras. Ainsi on obtient un ensemble de tétraèdres définissant un

volume et une surface polyédrique qui constitue une approximation des surfaces de la scène 3D.

L'avantage de cet algorithme par rapport au précédent est qu'il s'adapte automatiquement au cas de plusieurs vues avec des positions quelconques pour les centres optiques. L'inconvénient réside dans la complexité algorithmique de la triangulation contrainte et surtout de l'élimination des tétraèdres qui cachent des segments.

10.3.3. Résultats expérimentaux et conclusion

Ces deux algorithmes ont été implantés et testés sur des données stéréoscopiques réelles. Le résultat de la reconstruction est un ensemble de facettes triangulaires définissant une surface polyédrique. Les résultats obtenus sont satisfaisants et assez similaires pour les deux méthodes. Pour la première méthode (reconstruction à partir de coupes) on obtient pour deux cent segments un temps de calcul de l'ordre d'une dizaine de secondes sur une station SUN-3.

Ces travaux illustrent l'intérêt des méthodes de reconstruction de formes avec des rayons pour l'extraction de surfaces à partir de données stéréoscopiques. En effet ce type d'approche conduit à des algorithmes non heuristiques, de faible complexité, et qui ne requièrent aucune information à priori.

Chapitre 11

Des images volumiques 3D à la géométrie des surfaces

Dans ce chapitre nous décrivons l'extraction et la caractérisation de surfaces à partir d'images volumiques 3D. Nous présentons des résultats expérimentaux obtenus sur des images médicales.

11.1. Détection de contours 3D

De nombreuses applications industrielles ou médicales fournissent des images 3D représentant une information volumique. Les techniques modernes comme la tomographie (CT) produisent des images 3D où le niveau de gris est proportionnel à la densité locale. Par exemple, dans le domaine biomédical des images 3D sont obtenues par résonance magnétique nucléaire (IRM), scanner X, ou échographie. Ainsi pour de telles données les volumes présentant une distribution homogène en niveaux de gris correspondent aux entités de la structure 3D. Ces volumes peuvent être obtenus soit en partitionnant l'image 3D en zones homogènes (approche régions), ou en identifiant les discontinuités des niveaux de gris correspondant à la trace des surfaces (approche contours). La segmentation d'images 2D ou 3D en régions pose le délicat problème de trouver des critères d'homogénéité adéquats. On remarque que les algorithmes de segmentation décrits dans le chapitre 4 s'adaptent naturellement à la segmentation des images 3D. Malgré cette adéquation on choisit le plus souvent une approche contour qui semble, pour ce problème, avoir de meilleures qualités de stabilité et un moindre coût calculatoire. La plupart des approches comprennent deux

étapes successives : détection de contours 3D par filtrage puis suivi de contours 3D en utilisant des propriétés morphologiques.

Ainsi une première étape pour identifier ces surfaces est de rechercher leur trace en utilisant seulement l'information du signal. Les points des surfaces sont localisés où le gradient de la fonction des niveaux de gris est localement maximum. On retrouve le problème classique de la détection de contours largement étudié pour des images 2D [32, 44, 165], mais beaucoup moins dans le cas 3D. A la base, la détection de contours 2D ou 3D posent le même problème, soit la détection des discontinuités d'une fonction discrète et bruitée (2D ou 3D).

La plupart des opérateurs de détection de contours 3D sont issus d'une généralisation en 3D des opérateurs 2D [185, 135]. Leur principe est d'approximer le gradient ou le laplacien de l'image en utilisant des masques de convolution. Un compromis entre la taille des masques de convolution et leurs performances de détection et de localisation est alors à trouver. Particulièrement dans le cas 3D, la taille parfois considérable des données rend la complexité algorithmique et l'espace mémoire essentiels pour la détection de contours. Ainsi la taille des masques de convolution utilisés pour implanter l'opérateur ne peut être trop importante afin d'éviter un coût calculatoire trop élevé. La convolution d'une image 3D $dimx \times dimy \times dimz$ par un masque $n \times n \times n$ coûte $o(n^3 dimx dimy dimz)$. Par exemple cette complexité rend quasiment impossible (sur une machine séquentielle classique) l'utilisation de masques de taille plus importante que $5 \times 5 \times 5$ pour une image $512 \times 512 \times 100$. Ainsi une limitation importante de la plupart des opérateurs (linéaires) de détection de contours 3D réside dans l'utilisation de petits masques de convolution très sensibles au bruit. Une autre difficulté, théorique celle-ci, posée par la détection de contours 3D, et que partage d'ailleurs le cas 2D, est l'estimation des performances des opérateurs. En effet, le plus souvent, on dérive des opérateurs 2D ou 3D d'opérateurs 1D optimaux par rapport à un modèle monodimensionnel. Une question intéressante est alors posée par l'évaluation des performances des opérateurs pour des modèles non plus 1D mais 2D ou 3D. Des réponses partielles à ces deux problèmes (complexité et évaluation théorique des performances) sont amenées par le filtrage séparable récursif et une généralisation en 2D et 3D des critères de Canny [134].

A cause du bruit, cette étape de filtrage n'est parfois pas suffisante pour obtenir des résultats pouvant être utilisés pour la reconstruction et la modélisation des surfaces. De manière à améliorer la carte des contours 3D, il peut être

utile d'introduire des informations morphologiques [120]. On peut par exemple utiliser un algorithme de suivi de contours 3D résultant d'une extension d'une méthode de fermeture de contours 2D [47].

11.1.1. Gradient et laplacien d'une image 3D

Le filtrage s'implante par cascade de filtrages récursifs 1D et est décrit au chapitre 2 pour le cas 2D. Le cas 3D est une simple généralisation.

Une fois le filtre et le type d'approche choisis (voir chapitre 2), nous calculons le gradient ou le laplacien comme suit : Soit $I(x, y, z)$ une image 3D et $G(x, y, z)$ le gradient de I :

$$G = (\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}, \frac{\partial I}{\partial z})^t$$

Le calcul des composantes du gradient est réalisé en calculant des images (D_i) correspondant aux dérivées par rapport à $(x, y, z) = (x_1, x_2, x_3)$ par une cascade de filtrage 1D comme suit :

pour $i = 1, 2, 3$ faire
$\left[\begin{array}{l} D_i = I \\ \text{pour } j \in [1,2,3]\backslash\{i\} \text{ faire} \\ [D_i = D_i \star s(x_j) \\ D_i = D_i \star d(x_i) \end{array}\right.$

où $s(x)$ et $d(x)$ sont respectivement les opérateurs de lissage et de dérivée première ($d(x) = s'(x)$ aux constantes de normalisation près).

Pour une image de dimensions $dimx \times dimy \times dimz$ le calcul des composantes du gradient demande 3 convolutions par point. Ainsi, nous obtenons pour la totalité du calcul $3dimxdimydimz$ convolutions par point. Si on utilise une implantation directe avec un masque de convolution 1D de taille k, nous obtenons une complexité de l'ordre de : $3kdimxdimydimz$. Un filtre récursif d'ordre r permet de la réduire à : $3rdimxdimydimz$.

Le laplacien peut être calculé en additionnant les dérivées secondes. Le calcul des dérivées secondes peut être réalisé en utilisant le même algorithme que pour les dérivées premières où on remplace l'opérateur de dérivée première d par l'opérateur de dérivée seconde. On peut aussi calculer directement le laplacien par filtrage récursif, ce qui permet de diminuer le coût du filtrage [44].

11.1.2. Performances des filtres 3D

La prise en compte de modèles de contours 3D même aussi élémentaires qu'un contour orienté nécessite un effort théorique pour évaluer les performances des opérateurs. Le calcul présenté au chapitre 2 section 2.7. montre que par exemple dans le cas du filtre Deriche, les valeurs des critères de détection et de localisation dépendent de l'orientation du contour. L'utilisation de modèles plus complexes incluant des angles et des courbures est certainement intéressante à condition de pouvoir aboutir à des algorithmes de complexité raisonnable...

11.1.3. Traitements en aval : maxima locaux, seuillage

Bien que le calcul du gradient ou du laplacien 3D constitue la partie essentielle de la détection de contours, cette étape ne fournit pas directement les points de contours. Ces étapes complémentaires diffèrent selon l'approche (gradient ou laplacien).

11.1.3.1. Approche gradient

On suppose que les contours soient localisés où la norme du gradient est extremum le long de la direction du gradient :

$$\nabla(\sqrt{(\frac{\partial I}{\partial x})^2 + (\frac{\partial I}{\partial y})^2 + (\frac{\partial I}{\partial z})^2}).\nabla(I) = 0$$

où ∇ est l'opérateur gradient et I l'image.
Dans la pratique, on détermine les points de contours en dérivant localement la norme du gradient dans la direction du gradient. Par rapport au calcul direct du critère cette méthode a l'avantage de ne pas calculer explicitement les dérivées secondes.
Dans les images réelles le bruit introduit des extremums locaux du gradient qui ne correspondent pas à des points de contours. Pour les éliminer on utilise un seuillage par hystéresis 3D [135]. Ce type de seuillage prend en compte la connexité des points et permet d'améliorer la continuité des contours. On peut

améliorer cette méthode en rajoutant une contrainte de suivi dans la direction du plan tangent [134].

La figure 11.1. présente le résultat d'une détection de contour 3D par filtrage séparable récursif (filtre Deriche) avec une approche gradient sur une image de crâne obtenue par scanner X.

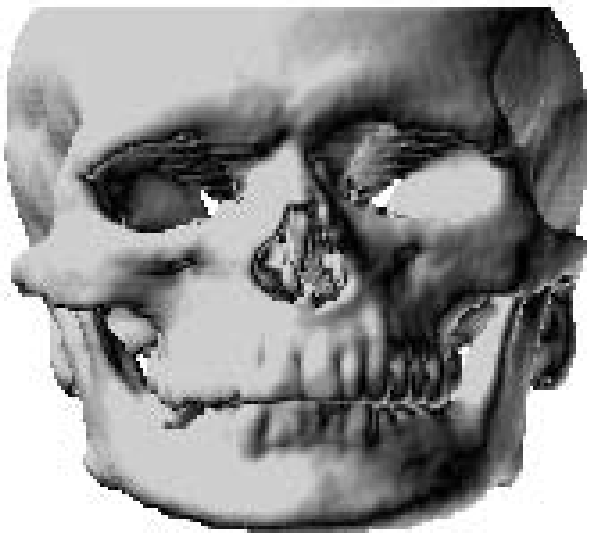


Figure 11.1. *Vue en perspective du résultat d'une détection de contours 3D (filtre de Deriche 3D) pour une image de crâne obtenue par scanner X.*

11.1.3.2. Approche laplacien

Dans cette approche (voir chapitre 2 section 2.4.2) on suppose que les points de contours correspondent aux passages par zéro du laplacien soit :

$$\Delta I = 0$$

De manière à déterminer les points qui annulent le laplacien on crée une image de polarité où les points de laplacien positif sont à 1 et les autres à 0. Les

transitions 1-0 (théorème des valeurs intermédiaires) définissent les passages par zéro. Comme pour les extréma locaux du gradient, on obtient des passages par zéro ne correspondant pas à des contours. De manière à les supprimer, on rajoute, comme pour l'approche gradient, une dernière étape de seuillage nécessitant le calcul du gradient aux points de passage par zéro.

11.1.4. Suivi de contour 3D

Dans les parties d'image fortement dégradées, les informations locales ne suffisent pas à détecter les contours. Une idée classique est de décider si un point appartient à un contour en fonction d'une conjonction d'informations locales. Cette conjonction se définira, par exemple, par un chemin optimal dans un graphe valué prenant en compte des informations de gradient et de variation de la courbure.

On peut étendre au cas 3D une méthode de fermeture de contours 2D proposée par R. Deriche et J.P. Cocquerez [47] (voir chapitre 4, section 4.5). Le principe de cet algorithme est de déterminer des points de contours à priori fiables (en fonction de critères pré-définis) et d'utiliser ensuite une méthode de suivi/fermeture permettant de compléter les contours.

11.1.5. Conclusion

Ces algorithmes ont été implantés et testés sur des images IRM, scanner X, et échographiques. Les résultats obtenus se comparent favorablement aux autres méthodes existantes tant du point de vue de la qualité des contours que du coût calculatoire. Les bonnes propriétés des filtres exponentiels et l'implantation séparable récursive expliquent les performances de ces filtres 3D par rapport aux opérateurs classiques. De plus, le suivi de contour permet encore une petite amélioration des résultats.

11.2. Des contours 3D aux courbures et aux lignes de crête

Dans la partie précédente on a décrit des méthodes fournissant des points formant la trace discrète de surfaces. Cette nouvelle représentation de l'information image est plus compacte et manipulable que les données brutes. Néanmoins, si on veut résoudre automatiquement (ou presque) des problèmes de

recalage ou de mise en correspondance, elle reste peu satisfaisante. D'où l'intérêt suscité par les travaux sur la caractérisation des surfaces à partir d'une trace discrète obtenue par détection de contours 3D [161, 162] ou capteur laser [27, 154]. Dans le cas de surfaces polyédriques, des algorithmes d'approximation par facettes planes peuvent être utilisés [60]. Lorsque les surfaces sont gauches, des algorithmes d'approximation par morceaux de surfaces quadriques existent [60]. Mais ces méthodes soulèvent le même problème que les algorithmes de segmentation d'images : la définition de propriétés d'homogénéité permettant l'obtention d'un partitionnement stable. Ainsi la plupart des approches existantes [161, 162, 27, 154] repose sur l'extraction de caractéristiques différentielles locales du second ou du troisième ordre de la surface tracée par les contours : courbures, lignes de crête (lignes où la courbure est localement maximum)...Le plus souvent, les méthodes existantes déconnectent l'étape de filtrage-détection de contours de la modélisation des surfaces. On obtient ainsi des algorithmes pas toujours très robustes au bruit car ne prenant en compte le plus souvent que la position des points. Une idée développée récemment consiste à utiliser l'étape de filtrage de l'image 3D afin d'assurer une meilleure cohérence des résultats [133, 132, 168]. Ce principe est similaire à celui de l'algorithme présenté par J. Ponce et M. Brady pour des images 2D de profondeur [154].

11.2.1. Rappels de géométrie différentielle

11.2.1.1. Notion de courbure

Avant d'introduire les deux formes fondamentales, il nous a semblé utile de rappeler de façon très précise la notion géométrique de courbure (voir figure 11.2.). On considère pour cela un point M_0 situé sur un arc γ. Alors, [109] :

"Pour tout $M \in \gamma$ et suffisamment voisin de M_0 , il existe un cercle unique γ_M tangent à γ en M_0 et passant par M ; et le *cercle osculateur* à γ en M_0 est la limite de γ_M quand M tend vers M_0 sur γ."

La courbure en M_0 est égale à l'inverse du rayon du cercle osculateur. La courbure en un point M_0 d'une surface le long d'une direction $\vec{d}$ du plan tangent correspondra à la courbure de la courbe plane obtenue par intersection

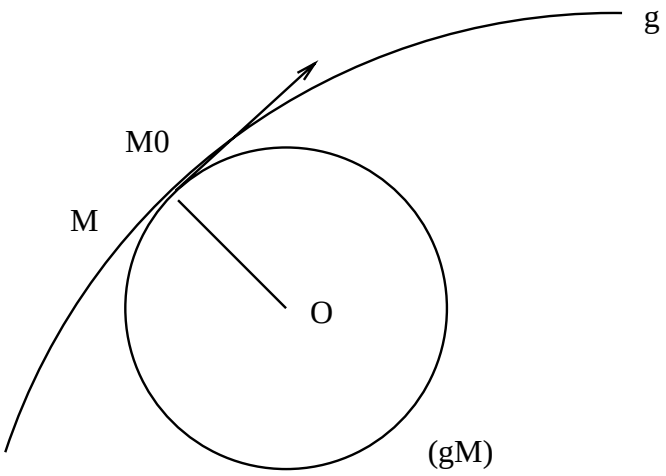


Figure 11.2. *Courbure sur une courbe régulière dans un plan.*

de la surface et du plan normal P_{M_0} (voir figure 11.3.), d'où le terme spécifique de *courbure normale*. On verra plus loin que la courbure normale selon une direction quelconque s'exprime en fonction de deux courbures (courbures principales) correspondant à deux directions orthogonales.
Nous nous placerons dans la suite de ce paragraphe dans le cadre d'une surface paramétrée, notée S, par $U \times V$ dont le couple générique sera noté (u, v). On notera X l'application définie de $U \times V$ dans S qui fait correspondre à tout couple (u, v) le point $X(u, v)$ [49]. On supposera cette surface régulière, c'est-à-dire que l'application X est continue et dérivable à tout ordre.

11.2.1.2. Première forme fondamentale

Soit P un point de S, et T_P son plan tangent engendré par la base $(\vec{X}_u = \frac{\partial \vec{X}}{\partial u}, \vec{X}_v = \frac{\partial \vec{X}}{\partial v})$, la *première forme fondamentale* I permet de définir une métrique liée au plan tangent. Soit $\vec{w} \in T_P$,

$$I(\vec{w}) = \vec{w}^t M_1 \vec{w}$$

$$\text{où } M_1 = \begin{pmatrix} E & F \\ F & G \end{pmatrix} \quad E = < \vec{X}_u, \vec{X}_u > \quad F = < \vec{X}_u, \vec{X}_v > \quad G = < \vec{X}_v, \vec{X}_v >$$

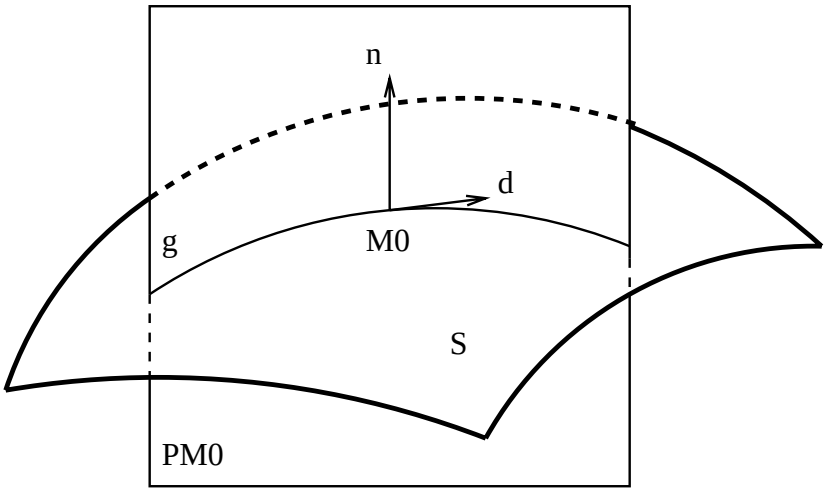


Figure 11.3. *Courbure normale sur une surface.*

E, F, G sont appelés *cœfficients de la première forme fondamentale.* Concernant le déterminant de cette première forme fondamentale, il est à noter la relation intéressante suivante puisqu'elle fait le lien entre les deux formes fondamentales. En effet, le calcul de la courbure implique de dériver des vecteurs normés :

$$\| \vec{X}_u \wedge \vec{X}_v \| = \sqrt{EG - F^2}$$

La première forme fondamentale définit la variation de la position d'un point dans l'espace du plan tangent pour une variation des paramètres ; ce qui revient à exprimer le vecteur reliant $X(u,v)$ et $X(u+du, v+dv)$ dans le repère du plan tangent ayant pour d'origine le point considéré et de vecteurs directeurs $(\frac{\partial X}{\partial u}, \frac{\partial X}{\partial v})$; cela correspond à la relation existant entre les distances dans l'espace des paramètres et dans l'espace du plan tangent.

11.2.1.3. Deuxième forme fondamentale

On définit d'abord l'*application de Gauss*, notée N, ainsi : elle associe à tout point de la surface son vecteur normal unitaire :

$$N : U \times V \longrightarrow S(1) \text{ sphère unitaire}$$

$$(u , v) \longmapsto \vec{N}_{(u,v)}$$

On introduit alors l'*endomorphisme de Weingarten*, noté dN, définissant une application (linéaire) du plan tangent sur lui-même qui associe à tout vecteur tangent le vecteur représentant la variation de la normale le long de cette direction. Ce vecteur appartient bien au plan tangent : soit en effet $\alpha(t) = X(u(t), v(t))$ une courbe sur S passant par $P = \alpha(0)$ et de tangente en P de direction $\vec{w} = \vec{\alpha'(0)}$,

$$dN(\vec{w}) = \vec{N}_u u' + \vec{N}_v v'$$

$$\text{Or} \quad < \vec{N}, \vec{N} >= 1 \Rightarrow \vec{N}_u \perp \vec{N}, \vec{N}_v \perp \vec{N} \Rightarrow \vec{N}_u, \vec{N}_v \in T_P \Rightarrow dN(\vec{w}) \in T_P$$

$dN(\vec{w})$ correspond en fait à la dérivée directionnelle de l'application de Gauss N selon $\vec{w}$.

La *deuxième forme fondamentale* est la forme quadratique associée à l'endomorphisme de Weingarten. Elle permettra en fait de calculer la courbure normale en un point de la surface le long d'une direction donnée, c'est-à-dire l'inverse du rayon du cercle *osculateur*. Sa matrice associée, notée M_2 , s'écrit :

$$M_2 = \begin{pmatrix} e & f \\ f & g \end{pmatrix}$$

avec $e =< \vec{N}, \vec{X}_{uu} > \quad f =< \vec{N}, \vec{X}_{uv} > \quad g =< \vec{N}, \vec{X}_{vv} >$
e, f, g sont appelés *cœfficients de la deuxième forme fondamentale*. De plus, on a la relation suivante :

$$dN(\vec{w}) = W\vec{w}$$

$$\text{avec} \quad W = \ -M_1^{-1} \ M_2$$

La deuxième forme fondamentale définit la variation de la normale en un point pour une variation des paramètres ; elle caractérise donc les relations entre $N(u, v)$ et $N(u + du, v + dv)$ dans le repère lié au plan tangent $(\frac{\partial X}{\partial u}, \frac{\partial X}{\partial v})$; l'endomorphisme de Weingarten caractérise la variation de la position de la normale pour un déplacement du point dans le plan tangent et a logiquement comme matrice associée $M_1^{-1} M_2$.

11.2.1.4. Courbures principales et directions principales de courbure

On démontre aisément que l'endomorphisme de Weingarten est une application linéaire auto-adjointe. Ainsi, il existe une base orthonormale $(\vec{e}_1, \vec{e}_2)$ du plan tangent, telle que :

$$dN(\vec{e}_1) = \lambda_1.\vec{e}_1 \quad \text{et} \quad dN(\vec{e}_2) = \lambda_2.\vec{e}_2$$

et λ_1 et λ_2 sont respectivement le maximum et le minimum de la forme quadratique associée, c'est-à-dire de la deuxième forme fondamentale (pour une démonstration complète, voir [49, 103]).

λ_1 et λ_2 sont appelées *courbures principales*, $\vec{e}_1$ et $\vec{e}_2$ étant les *directions principales de courbure* associées. Le produit $\lambda_1\lambda_2$ se nomme *courbure gaussienne*, la demi-somme $(\lambda_1 + \lambda_2)/2$ *courbure moyenne*. Il est à noter que l'orientation de la surface détermine le signe des courbures, la courbure gaussienne étant elle invariante dans $\mathcal{R}^3$. On remarquera que la connaissance des deux courbures principales permet de déterminer la courbure dans une direction quelconque [49].
D'autre part, on appelle *ligne de courbure* une courbe portée par la surface telle qu'en chaque point, la tangente à la courbe soit une direction principale de courbure. On sait ainsi qu'en chaque point régulier de S, il passe deux lignes de courbure orthogonales (voir [109], page 527).

11.2.1.5. Courbure et direction maxima de courbure

La courbure maximum est la courbure principale de plus grand module et la direction maximum de courbure la direction correspondante.

11.2.1.6. Points ombilics

Lorsque les deux courbures principales sont égales, la courbure est identique dans toutes les directions (voir figure 11.4.). On dit alors que le point est un ombilic (la surface est localement sphérique au voisinage).

11.2.1.7. Points hyperboliques, elliptiques et lignes paraboliques

Un point dont la courbure gaussienne est négative est dit hyperbolique. Cela correspond à la notion de point selle : la courbure est de sens opposé le long

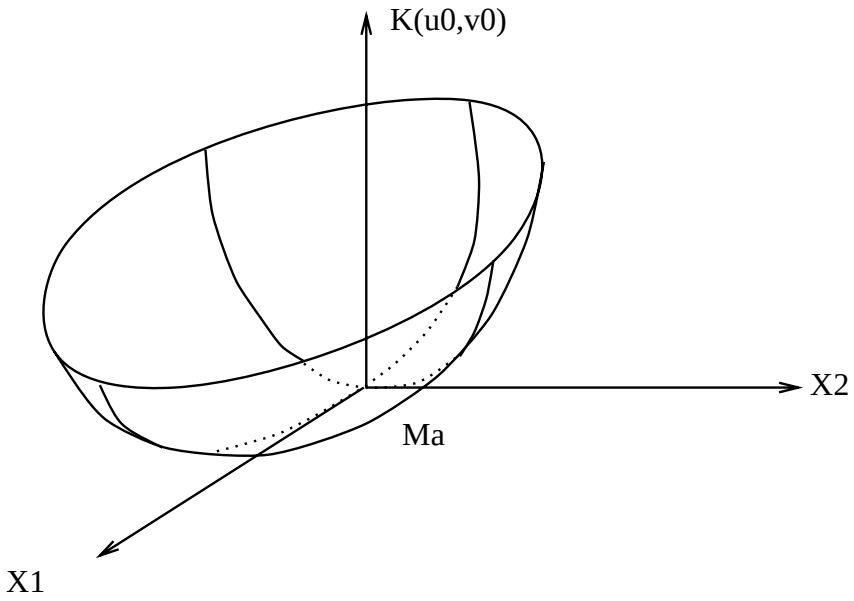


Figure 11.4. *Forme de la surface au voisinage d'un point ombilic.*

des deux directions principales (voir figure 11.5.).

Un point dont la courbure gaussienne est positive est dit elliptique. La surface est alors localement un ellipsoïde (voir figure 11.6.).

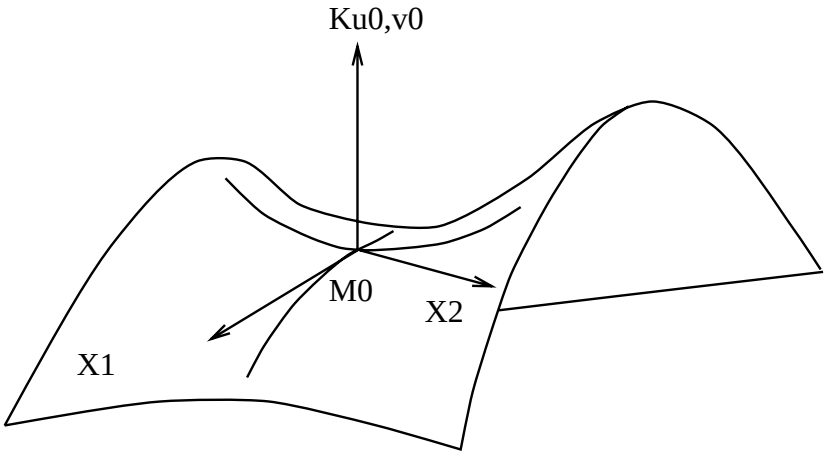
Un point dont la courbure gaussienne est nulle est dit parabolique. La surface admet alors une courbure non nulle selon une seule direction principale.

On montre que les points paraboliques forment des lignes séparant les zones de points hyperboliques des zones de points elliptiques. Ces lignes sont appelées lignes paraboliques.

11.2.1.8. Définition d'un extremum de courbure

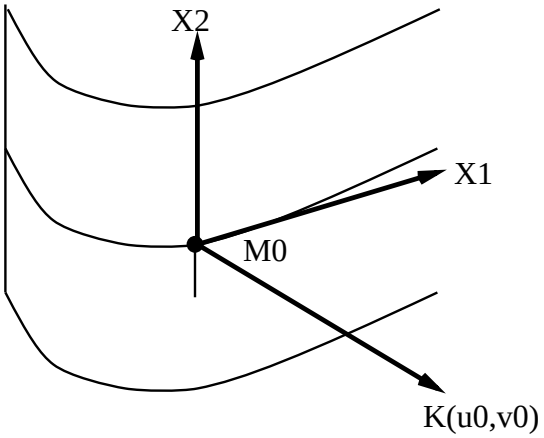
On dira qu 'un point m de S est un *extremum de courbure* s'il est un maximum local de la courbure principale maximum (c'est à dire de plus forte valeur absolue) dans la direction principale de courbure associée. Cette définition est à rapprocher de celle d'un point de contour qui est défini comme un extremum de la norme du gradient dans la direction du gradient [134].

Une définition proche est fournie par Yuille et Leyton [181] qui présentent



Forme de la surface au voisinage de M0

Figure 11.5. *Forme de la surface au voisinage d'un point hyperbolique.*



Forme de la surface au voisinage de M0

Figure 11.6. *Forme de la surface au voisinage d'un point parabolique.*

les extremums de courbure comme les extremums de la courbure principale le long des lignes de courbure. Ils démontrent avec cette définition qu'on peut associer à chaque point extremum de courbure des axes de symétrie de la surface, et que cette propriété, de plus, les caractérisent. Kœnderink passe par l'intermédiaire des "centro-surfaces", qui sont les surfaces formées par les centres de courbures principaux. Cet auteur énonce que à chaque point singulier de celles-ci correspond un point de crête [106].

11.2.2. Approximation locale de surface

On dispose donc d'un ensemble de points 3D formant la trace discrète d'une surface dont on recherche les courbures et les lignes de crête. Ces points définissent explicitement un graphe d'adjacence. Une approche naturelle pour calculer ces courbures est d'approximer localement la trace au voisinage de chaque point par une surface analytique (au moins du second ordre). Une méthode de ce type se caractérise essentiellement par :

1. les données utilisées : points, normales, incertitudes...;
2. le modèle local : quadrique, spline...
3. la méthode d'ajustement : moindres carrés, moindres carrés médian, filtre de Kalman...
4. le voisinage pris en compte : fixe, adaptatif...

11.2.2.1. Données prises en compte

L'étape de détection de contours 3D fournit un ensemble de points $P = \{P_1, P_2, \ldots P_n\}$ et les gradients 3D correspondants $G = \{\vec{G_1}, \vec{G_2}, \ldots . \vec{G_n}\}$ où $\vec{G_i}$ est le gradient au point P_i (voir chapitre 2, section 2.2.2).

Si on suppose que les points de P sont la trace d'une surface et que chaque point P_i séparant localement deux zones homogènes alors les vecteurs $\vec{G_i}$ approximent les normales à la surface. On peut donc prendre en compte ces normales pour réaliser l'approximation locale. De plus ces points et ces gradients provenant d'une opération de filtrage on peut déterminer les matrices de covariance associées à leurs incertitudes comme suit :

Soit $I(i,j,k)$ la fonction des niveaux de gris de l'image 3D.
Le résultat de la détection de contours s'exprime comme quatre images $C(i,j,k)$, $G_x(i,j,k)$, $G_y(i,j,k)$ et $G_z(i,j,k)$: C prend pour valeur 1 pour tout point de contour et 0 sinon ; G_x, G_y, et G_z sont les trois composantes du gradient.

Soit un point de contour M_0 de coordonnées $(x_0, y_0, z_0)^t$ et de vecteur gradient $G_0 = (g_{x_0}, g_{y_0}, g_{z_0})^t$. On considère M_0 et G_0 comme les réalisations de deux variables aléatoires de moyennes la position réelle du point de contour (pour M_0) et le gradient réel (pour G_0). Soient W_{M_0} et W_{G_0} les matrices de covariance associées, si on suppose les trois composantes de M_0 et G_0 non corrélées (ce qui est une hypothèse simplificatrice mais raisonnable), on a dans le repère de l'image :

$$W_{M_0} = \begin{pmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_z^2 \end{pmatrix},$$

$$W_{G_0} = \begin{pmatrix} \sigma_{g_x}^2 & 0 & 0 \\ 0 & \sigma_{g_y}^2 & 0 \\ 0 & 0 & \sigma_{g_z}^2 \end{pmatrix},$$

où les diagonales des matrices correspondent aux variances des variables aléatoires pour les coordonnées des points et du gradient.
La généralisation du modèle de Canny au cas 3D présentée dans le chapitre 2, section 2.7 permet un calcul théorique de ces incertitudes réalisé dans le cas particulier de l'opérateur Deriche-3D [135]. Le critère de localisation correspond à l'écart type de la distance du point de contour détecté au vrai contour (un plan 3D), soit à l'écart type de la coordonnée parallèle à la normale du contour. En effet pour une détection de contours par filtrage linéaire l'incertitude se situe le long de l'orthogonale au contour . Dans la pratique on rajoute un bruit d'échantillonnage (Δ) pour les autres composantes [132]. Pour M_0 on obtient donc dans un repère orthonormé dont le premier axe est parallèle à la normale au contour, la matrice de covariance suivante :

$$\Sigma_{M_0} = \begin{pmatrix} \Delta & 0 & 0 \\ 0 & \Delta & 0 \\ 0 & 0 & {\sigma^2}_{\perp} + \Delta \end{pmatrix}.$$

avec

$$\sigma_{\perp} = \sqrt{\frac{3}{2\alpha}} \frac{\eta_0}{\|G_{M_0}\|} q(a,b)$$

où

$$q(a,b) = g(a,b)\, h(a,b).$$

$g(a,b)$ et $h(a,b)$ sont des fonctions de l'orientation du contour définies au chapitre 2, section 2.7 pour un cas particulier, α est la largeur de l'opérateur, et η_0 l'écart type du bruit blanc. Pratiquement on pose :

$$(a,b) = (\frac{G_x}{G_z}, \frac{G_y}{G_z})$$

Dans le repère de l'image la matrice de covariance s'écrit :

$$W_{M_0} = R \Sigma_{M_0} R^t,$$

Pour W_{G_0} on obtient dans le repère de l'image :

$$W_{G_0} = \frac{3\alpha\eta_0^2}{2g^2(a,b)}\, I_3.$$

où I_3 est la matrice identité dans R^3.

11.2.2.2. Modèle local

Soit un ensemble de points P et les gradients G (approximant les normales à la surface).

Soit $M \in P$ un point de la trace de la surface et $\vec{N} \in G$ la normale associée. En utilisant $\vec{N}$ on peut définir un repère lié au plan tangent à M que nous noterons (M, Q, R, N). On notera que la base (M, Q, R) du plan tangent à M est arbitraire (la seule contrainte est que le repère soit direct et orthonormé).
Dans la suite, M est un point où le modèle local est ajusté, et les points M_i définissent un voisinage de M avec des normales associées $\vec{N_i}$. On supposera les coordonnées de M_i et $\vec{N_i}$ données dans le repère du plan tangent (le développement est plus simple dans ce repère).
On suppose donc que les données fournies par la détection de contour représentent une estimation bruitée des points et des normales d'une surface S.

On considère la surface comme une variété différentiable de dimension 2 et on construit une paramétrisation locale en tout point. Ainsi, pour un point $M \in P$ on suppose que la paramétrisation locale (ψ, U)

$$\psi : U \subset \mathcal{S} \to I\!R^2$$

(ψ étant un difféomorphisme, $M \in$ ensemble ouvert U) est telle que $\psi(P) = (0,0)$ et que sa réciproque restreinte à $\psi(U)$

$$\phi = i \circ \psi^{-1} : \psi(U) \subset I\!R^2 \to I\!R^3$$

dans $I\!R^3$ (exprimée dans le repère lié au plan tangent à M), est une fonction $h : \psi(U) \to I\!R$ avec

$$\begin{aligned} h(0,0) &= 0, \\ h_q(0,0) = \left.\frac{\partial h}{\partial q}\right|_{(0,0)} &= 0 \\ h_r(0,0) = \left.\frac{\partial h}{\partial r}\right|_{(0,0)} &= 0 \end{aligned}$$

(cela est toujours vrai pour au moins une paramétrisation locale). la fonction $h(q,r)$ définit l'allure de la surface au voisinage du point M et correspond au modèle local. On peut choisir des fonctions différentes pour ce modèle local. On considérera dans la suite le modèle proposé dans [161] (une quadrique parabolique) justifié par les considérations qui suivent. Le développement de Taylor de h au voisinage de l'origine est :

$$h(q,r) = \frac{1}{2}\left(h_{qq}q^2 + 2h_{qr}qr + h_{rr}r^2\right) + R.$$

Si notre objectif est le calcul de propriétés différentielles d'ordre 2 comme les courbures le modèle de paramétrisation locale le plus simple est :

$$h = \frac{1}{2}eq^2 + fqr + \frac{1}{2}gr^2, \qquad [11.1]$$

où

$$e = h_{qq}(0,0), \quad f = h_{qr}(0,0), \quad g = h_{rr}(0,0)$$

(connue comme une quadrique parabolique). Dans ce cas particulier la matrice hessienne de la surface définie par $h(q,r)$ au point M (G_M) correspond à l'endomorphisme de Weingarten soit :

$$G_M = \begin{pmatrix} e & f \\ f & g \end{pmatrix}$$

Les valeurs propres de G_M sont les courbures principales :

$$K_1 = \frac{(e+g+\sqrt{(e-g)^2+4f^2})}{2}$$

$$K_2 = \frac{(e+g-\sqrt{(e-g)^2+4f^2})}{2}$$

La courbure gaussienne (K : déterminant de G_M) et la courbure moyenne (H : trace de G_M) sont donc :

$$K = eg - f^2$$

$$H = \frac{(e+g)}{2}$$

On calcule ainsi les vecteurs propres associés qui correspondent aux directions principales de courbure [161].

11.2.2.3. Méthode d'ajustement

Une fois qu'on a choisi les données prises en compte et le modèle local, il s'agit d'ajuster ce modèle sur les données.

11.2.2.3.1. Instanciation du modèle

On cherche donc la surface quadrique locale passant par le point M qui approxime le mieux (au sens d'un certain critère) les points voisins : $M_i = (q_i, r_i, n_i)^t$ et leurs normales $\vec{n_i} = (\alpha_i', \beta_i', \gamma_i')^t$. On obtient dans le repère lié au plan tangent au point M une première équation de mesure pour chaque point M_i :

$$E_1(e,f,g) = q_i^2 e + 2q_i r_i f + r_i^2 g - 2n_i = 0 \qquad [11.2]$$

Cette équation mesure la différence entre la position de M_i et la quadrique définie par (e,f,g) passant par le point M. On notera qu'on pourrait aussi ne pas contraindre le modèle local à passer par le point M.
La prise en compte (éventuelle) du gradient qui approxime la normale à la surface permet de rajouter des équations de mesure. En effet, dans le repère du plan tangent la normale à la quadrique au point M est :

$$n_i = \begin{pmatrix} -q_i e - r_i f \\ -q_i f - r_i g \\ 1 \end{pmatrix}.$$

Si on note la normale "normalisée" au point M_i par

$$(\alpha_i, \beta_i, 1)^t = (\alpha'_i/\gamma'_i, \beta'_i\gamma'_i, \gamma'_i/\gamma'_i)^t$$

on obtient deux équations de mesure supplémentaires :

$$E_2(e, f, g) = q_i e + r_i f + \alpha_i = 0, \quad [11.3]$$

$$E_3(e, f, g) = q_i f + r_i g + \beta_i = 0. \quad [11.4]$$

Les équations $[E_1$–$E_3]$ sont les trois équations de mesure qui contraignent la détermination des paramètres e, f, g de la quadrique au point P. Ces équations sont comparables aux quatre équations de [163]. La seule restriction de ces équations est que l'on suppose que les normales N_i des points M_i, mesurées dans le repère lié au plan tangent, ont une troisième composante non nulle (ce qui est raisonnable si on suppose les M_i dans le voisinage de M). Si on note :

$$A_i = \begin{pmatrix} q_i^2 & 2q_i r_i & r_i^2 \\ q_i & r_i & 0 \\ 0 & q_i & r_i \end{pmatrix}, \qquad b_i = \begin{pmatrix} 2n_i \\ -\alpha_i \\ -\beta_i \end{pmatrix}, \qquad x = \begin{pmatrix} e \\ f \\ g \end{pmatrix},$$

Les équations de mesure [11.3] et [11.4] en M peuvent être mises sous forme matricielle :

$$A_i x = b_i.$$

11.2.2.3.2. Solution classique au sens des moindres carrés

De manière à traiter le cas le plus général on va supposer que l'on pondère les équations de mesure avec les incertitudes des mesures (les coordonnées des points M_i et les gradients $\vec{G_i} = \vec{N_i}$). Dans le cas contraire on pose les matrices W_i, définies ci dessous, comme égale à l'identité. On détermine ainsi une matrice W_i qui est la covariance de $A_i x - b_i$. L'introduction de cette matrice permet de pondérer les équations du moindre carré en fonction de la confiance qu'on leur accorde (fonction des incertitudes de leurs termes).

$$W_i = E\left[(A_i x - b_i)(A_i x - b_i)^t\right].$$

Ainsi, une solution au sens des moindres carrés pondérée par les matrices de covariance minimise :

$$C = \sum_i (A_i x - b_i)^t W_i^{-1} (A_i x - b_i),$$

et est donnée par :

$$x = \left(A^t W^{-1} A\right)^{-1} A^t W^{-1} b,$$

où

$$A = \begin{pmatrix} A_1 \\ \vdots \\ A_n \end{pmatrix}, \qquad b = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}, \qquad \text{et} \quad W = \begin{pmatrix} W_1 & & \\ & \ddots & \\ & & W_n \end{pmatrix}.$$

11.2.2.3.3. Solution récursive

On peut implanter une solution récursive à ce type de problème de minimisation connue sous le nom de *filtre de Kalman* [116, 9]. Par cette méthode, chaque fois qu'une nouvelle mesure M_i est donnée, il est seulement nécessaire de calculer A_i et b_i *pour ce point* et de mettre à jour la solution courante (x_i, S_i) en utilisant les équations récursives :

$$\begin{cases} x_i &= x_{i-1} + K_i(b_i - A_i x_{i-1}), \\ K_i &= S_{i-1} A_i^t (W_i + A_i S_{i-1} A_i^t)^{-1}, \\ S_i &= (I - K_i A_i) S_{i-1}. \end{cases}$$

S_i est la matrice de covariance de l'estimée courante des paramètres :

$$S_i = E\left[(x_i - x)(x_i - x)^t\right]$$

exprimant l'adéquation de l'estimée courante x_i et de la valeur idéale $\mathbf{x}$ du vecteur paramètre. C'est une mesure de la qualité de l'estimation : une petite covariance signifie que l'estimée calculée x_i est proche des véritables paramètres $\mathbf{x}$. Il est nécessaire d'initialiser le filtre avec (x_0, S_0), qui peut prendre les valeurs $x_0 = 0, S_0 = \infty$ si aucune information *a priori* n'est disponible pour aucun des paramêtres.

11.2.2.3.4. Détails pratiques

Par simplicité le développement précédent a été présenté en considérant des coordonnées exprimées dans le repère du plan tangent. Nous montrons maintenant comment transformer les coordonnées réelles des points M_i et des vecteurs normaux N_i ainsi que les matrices de covariance correspondantes W_{M_i} et W_{N_i}, du repère de l'image, en des variables $v = (q_i, r_i, n_i, \alpha_i, \beta_i)^t$ et des matrices de covariance W_i dans le repère lié au plan tangent à P. Ainsi on peut appliquer le développement précédent directement aux données de l'image.
Supposons que $M_i = (x_i, y_i, z_i)^t$ et $n_i = (n_{x_i}, n_{y_i}, n_{z_i})^t$ soient données dans un système de coordonnées lié à l'image (X, Y, Z). Pour les exprimer dans le repère du plan tangent lié à $M = (x, y, z)^t$ (Q, R, N) on calcule :

$$\begin{pmatrix} q_i \\ r_i \\ n_i \end{pmatrix} = R \begin{pmatrix} x_i - x \\ y_i - y \\ z_i - z \end{pmatrix}, \qquad \begin{pmatrix} \alpha'_i \\ \beta'_i \\ \gamma'_i \end{pmatrix} = R \begin{pmatrix} n_{x_i} \\ n_{y_i} \\ n_{z_i} \end{pmatrix},$$

et

$$\alpha_i = \frac{\alpha'_i}{\gamma'_i}, \qquad \beta_i = \frac{\beta'_i}{\gamma'_i},$$

pour

$$R = \begin{pmatrix} Q_x & Q_y & Q_z \\ R_x & R_y & R_z \\ N_x & N_y & N_z \end{pmatrix}.$$

où les coordonnées des vecteurs du repère lié au plan tangent en M sont dans le repère global (lié à l'image) (X, Y, Z) :

$$P = \begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix}, \qquad Q = \begin{pmatrix} Q_x \\ Q_y \\ Q_z \end{pmatrix}, \qquad N = \begin{pmatrix} N_x \\ N_y \\ N_z \end{pmatrix}.$$

On suppose que les matrices de covariance du point M_i et de la normale associée N_i soient données dans le *repère global* (X, Y, Z) respectivement par W'_{Q_i} et W'_{n_i}. Pour une transformation affine quelconque d'une variable aléatoire $v_i \to w_i = M\,(v_i - v)$ on a :

$$E\left[(w_i - w)(w_i - w)^t\right] = M\,E\left[(v_i - v)(v_i - v)^t\right]\,M^t,$$

Les matrices de covariance exprimées dans le repère lié au plan tangent à M (M, Q, R, N) sont

$$W_{Q_i} = RW'_{Q_i}R^t, \qquad W_{n_i} = RW'_{n_i}R^t,$$

qui sont des matrices 3×3.

En fait si on exprime $N_i = (\alpha'_i, \beta'_i, \gamma'_i)^t$ dans le repère lié au plan tangent, on doit calculer $\alpha_i = \alpha'_i/\gamma'_i$ and $\beta_i = \beta'_i/\gamma'_i$. De manière à obtenir une approximation du premier ordre on calcule la matrice 2×2 $\widetilde{W}_{n_i}$:

$$\widetilde{W}_{n_i} = J_1 W_{n_i} J_1^t$$

où J_1 est la matrice jacobienne du changement de variables :

$$J_1 = \begin{pmatrix} \frac{1}{\gamma'_i} & 0 & \frac{-\alpha'_i}{\gamma'^2_i} \\ 0 & \frac{1}{\gamma'_i} & \frac{-\beta'_i}{\gamma'^2_i} \end{pmatrix}$$

Ainsi la matrice 5×5

$$W_{M,n} = \begin{pmatrix} W_M & 0_{3\times 2} \\ 0_{2\times 3} & \widetilde{W}_{n_i} \end{pmatrix}$$

est la covariance du vecteur de mesure $(q_i, r_i, n_i, \alpha_i, \beta_i)$.

On détermine une approximation du premier ordre de la matrice 3×3 W_i :

$$W_i = J_2 W_{M,n_i} J_2^t,$$

où J_2 est la matrice jacobienne :

$$J_2 = \begin{pmatrix} 2q_i e + 2r_i f & 2q_i f + 2r_i g & -2 & 0 & 0 \\ e & f & 0 & 1 & 0 \\ 0 & f & g & 0 & 1 \end{pmatrix}.$$

La matrice de covariance 2×2 des courbures est déterminée de manière similaire :

$$W_c = J_3 W_{M_i} J_3^t,$$

avec la jacobienne :

$$J_3 = \begin{pmatrix} g & -2f & e \\ \frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix}.$$

11.2.2.4. Voisinage pris en compte

La détermination du voisinage pris en compte pour ajuster le modèle local conditionne les résultats de ce type d'algorithme. Dans le développement précédent ce voisinage est défini implicitement par les points M_i.

Une première étape, commune à la plupart des méthodes, consiste à construire le graphe d'adjacence des points de contour. Ce graphe définit les relations de connexité entre les points de la trace de la surface. Si les données sont suffisamment denses, la 26-connexité (chaque voxel est voisin avec les voxels ayant une face, une arête ou un point en commun) suffit pour construire ce graphe. Si les données ne sont pas suffisamment denses, des méthodes de géométrie algorithmique basées sur la triangulation de Delaunay permettent de définir des relations d'adjacence [20, 19].

Une deuxième étape consiste à définir le voisinage pris en compte pour l'approximation locale en chaque point. Ce voisinage est une composante connexe du graphe d'adjacence contenant le point. A cet effet plusieurs méthodes peuvent être utilisées :

1. L'algorithme le plus élémentaire consiste à déterminer l'intersection d'une sphère dont le centre sera le point considéré (M) et de rayon r (r est fixé et définira l'amplitude du voisinage) avec l'ensemble des points de contour. On sélectionne ensuite les points appartenant à la composante connexe du point M. L'inconvénient de cet algorithme est qu'il revient à regarder la surface avec une même résolution en tous les points.

2. On peut améliorer la méthode précédente en rajoutant des critères de sélection des points (dans ce cas la valeur du paramètre r peut être plus importante) :

(a) Seuls les points dont la normale (gradient 3D issu de la détection de contour) a une orientation proche de la normale au point considéré (M) sont sélectionnés. Dans les expérimentations décrites dans [132] l'angle maximum entre les normales est fixé à $\frac{\pi}{4}$.

(b) Si on utilise une méthode récursive de prise en compte des points voisins, l'accroissement de l'erreur d'approximation peut être aussi un critère de sélection. L'inconvénient de ce schéma est que le résultat dépend alors de l'ordre de prise en compte des points. On peut amoindrir ce problème en ordonnant les points en fonction de leur proximité du point central (M).

(c) Une autre méthode introduite dans [161] consiste à calculer le conditionnement de la matrice des moindres carrés normalisée (rapport entre les deux valeurs propres extrémales). Ce conditionnement caractérise la stabilité de la solution aux petites perturbations des données. On peut ainsi augmenter la taille du voisinage jusqu'à ce que le conditionnement ne prenne une valeur trop importante.

11.2.2.5. Amélioration de la continuité des champs de courbure

Les algorithmes décrits précédemment fournissent les courbures et les directions principales de courbure en tout point de la trace (point de contour 3D) par approximation locale. Lorsque les données sont trop bruitées, on peut obtenir des courbures et des directions principales de courbure manquant de continuité. Pour remédier à ce problème on met en œuvre des méthode itératives permettant d'améliorer la cohérence des champs de valeurs obtenues. Un exemple de ce type d'approche est décrit dans [161]. Nous en décrivons les points essentiels : Soit un point M et ses voisins M_i.
On peut associer à M le repère de Darboux $(M, \vec{X_1}, \vec{X_2}, \vec{N})$ défini comme suit :

- $\vec{X}1$ est un vecteur normé de même direction que la direction principale de courbure associée à la courbure de module maximum.
- $\vec{X_2}$ est un vecteur normé de même direction que la direction principale de courbure de module minimum.
- $\vec{N_i}$ est un vecteur normé de même direction que la normale à la surface.
- $(M, \vec{X_1}, \vec{X_2}, \vec{N_i})$ forme un repère orthonormé direct de l'espace.

Soit $(M_i, \vec{X_{1_i}}, \vec{X_{2_i}}, \vec{N_i})$ le repère de Darboux associé à un point voisin M_i. Si on suppose la surface relativement lisse les repères de Darboux associés à chaque point doivent varier de façon continue. De manière à améliorer cette continuité, on peut modifier de manière itérative le repère de Darboux $(M, \vec{X_1}, \vec{X_2}, \vec{N})$ en minimisant :

$$E^2 = \sum_{i=1}^{n} ((\vec{X_1} - \vec{X_{1_i}})^2 + (\vec{N} - \vec{N_i})^2)$$

où le carré dénote le produit scalaire avec la contrainte d'orthogonalité :

$$F = < \vec{X_1}, \vec{N} > = 0$$

En utilisant le multiplicateur de Lagrange λ la minimisation sous contrainte précédente peut s'écrire comme la minimisation non linéaire de :

$$J = E^2 + \lambda F$$

Les détails de la solution se trouvent dans [161]. L'algorithme consiste donc à remplacer chaque repère de Darboux associé à un point par un nouveau repère obtenu par minimisation d'une contrainte non linéaire exprimant la continuité des repères dans l'espace.
Ce type de méthode permet d'éliminer des erreurs lorsque le bruit perturbe trop l'étape d'approximation locale. Néanmoins ce type d'algorithme doit être utilisé avec précaution car :

- la non-linéarité du critère de continuité peut induire des temps de calcul importants,
- bien que les directions maximums de courbure et les normales obtenues finalement soient davantage continues, leur adéquation avec les données réelles dépend du bien-fondé du critère (E) pour chaque cas particulier.

11.2.2.6. Des courbures aux attributs caractéristiques des surfaces

Les algorithmes précédents déterminent pour chaque point de la trace de la surface (point de contour 3D) les courbures moyennes et gaussiennes ainsi que les directions principales de courbure. L'étape suivante consiste à utiliser ces courbures pour extraire des lignes ou des points caractéristiques tracés sur ces surfaces. On détaillera plus spécialement dans cette partie le cas des lignes de crête utilisées avec succès pour la fusion de données [77].

11.2.2.6.1. Cartes locales de courbure

Un moyen pratique de caractériser le comportement de la courbure au voisinage d'un point est de définir des cartes locales de courbure. Soit un point M et son plan tangent défini par (M, Q, R), soit V l'intersection d'une sphère de centre M et de rayon r avec les points de contour (ceci définit un voisinage de M), et soit W la projection orthogonale des points de V sur le plan tangent. A tout point de W nous associons les courbures des points correspondants de V. La taille de V peut être déterminée en utilisant la distance entre les points

et le plan tangent et l'angle entre le gradient en un point et le gradient en M. On définit ainsi une carte locale caractérisant le comportement des courbures au voisinage de P.

11.2.2.6.2. Extraction de lignes de crête

A partir des cartes locales de courbure, on peut extraire, par exemple, les maximums de la courbure maximum dans la direction maximum de courbure. Pour cela, on peut utiliser des principes similaires à ceux de l'extraction des maxima locaux de la norme du gradient pour la détection de contours (voir chapitre 2, section 2.4).

Soit $C(M)$ la carte locale des courbures maximums du point M et $G(M)$ la projection de la direction maximum de courbure sur le plan tangent. On compare les valeurs de la courbure maximum le long de la droite définie par M et $G(M)$ de part et d'autre de M. On retient M si sa courbure maximum est localement extremum le long de cette direction.

Ainsi, on obtient des points susceptibles d'être des extremums de courbure. Pour éliminer les faux extremums, on peut réaliser un seuillage par hystéresis 3D en utilisant la courbure maximum. La méthode est similaire au seuillage des extremums locaux du gradient en utilisant la norme du gradient décrite au chapitre 2, section 2.2.3. Les figures 11.7. et 11.8. présentent un exemple d'extraction de lignes de crête fournie par cette méthode. Les courbures ont été déterminées avec l'algorithme d'approximation locale décrit dans [132].

11.2.3. Des dérivées partielles aux courbures

Dans l'hypothèse, réaliste dans la plupart des cas, où le gradient en un point de contour correspond à la normale à la surface, il semble naturel de pouvoir exprimer les courbures en fonction des dérivées secondes de l'image. En effet les courbures sont définies par les variations locales de l'orientation de la normale. Les normales correspondant aux vecteurs gradient, leurs variations s'expriment donc en fonction des dérivées du gradient soit des dérivées partielles secondes de l'image. Nous décrirons deux méthodes illustrant cette remarque : un algorithme de calcul direct utilisant l'hypothèse que le gradient définit un champ différentiable sur toute l'image, une autre approche considérant l'image comme une hypersurface dans R^4.

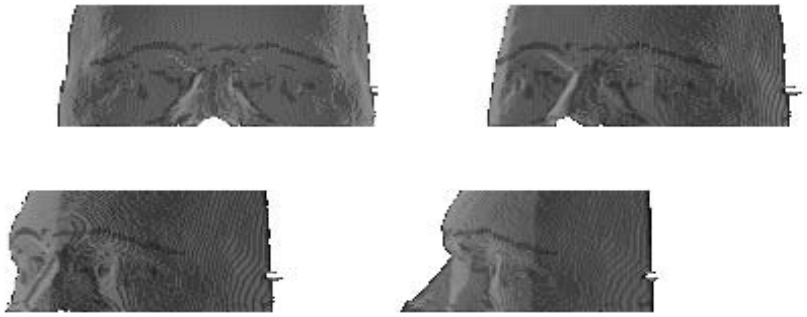


Figure 11.7. *Vues en perspective des contours 3D sur lesquelles on a tracé en noir les lignes de crête, l'image originale est une image RMN 3D d'un visage.*

Figure 11.8. *Projection des lignes de crête de la figure précédente.*

11.2.3.1. Relations entre les dérivées des images et les courbures des contours : application au calcul des courbures et à la caractérisation des points de crête à partir des dérivées partielles

On a vu au chapitre "détection de contours" que pour un modèle d'iso-contour (i.e. le contour sépare localement deux zones de niveaux de gris constants) le gradient correspond à la normale au contour. Dans le cas 3D le gradient est parallèle à la normale à la surface. Intuitivement, cela signifie que les propriétés différentielles des contours des objets (2D ou 3D) représentés par les images s'expriment en fonction des dérivées partielles des images. Nous présentons les calculs correspondant au cas 2D puis au cas 3D et nous montrons comment les appliquer à l'extraction des points de crête.

11.2.3.1.1. Cas 2D

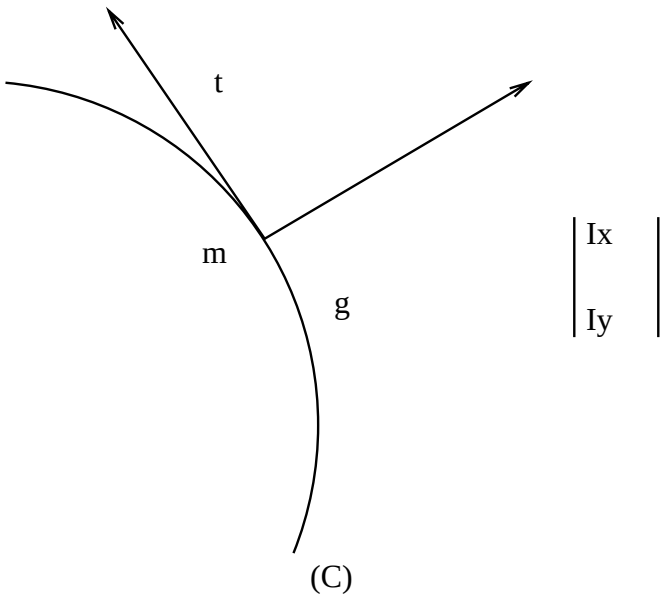


Figure 11.9. *Gradient et normale : cas 2D.*

Soit (C) un contour dans une image $I(x, y, z)$. On suppose que I est différentiable et que (C) est un iso-contour (i.e. le gradient est parallèle à la normale).

On note $\vec{g}$ le gradient de I supposé normal à la courbe (C) (voir figure 11.9.). On veut calculer la courbure k de (C) au point m.

Soit $\vec{t}$ le vecteur tangent unitaire à la courbe (C) au point m

$$\vec{g} \cdot \vec{t} = 0$$

Soit s une abscisse curviligne le long de C, nous avons :

$$\frac{d(\vec{g} \cdot \vec{t})}{ds} = \frac{d\vec{g}}{ds} \cdot \vec{t} + \vec{g} \cdot \frac{d\vec{t}}{ds} = 0 \qquad [11.5]$$

La courbure k est défini comme suit :

$$\frac{d\vec{t}}{ds} = k\,\vec{n} = k\frac{\vec{g}}{\|\vec{g}\|} \qquad [11.6]$$

où $\vec{n}$ est un vecteur normal unitaire (égal à $\frac{\vec{g}}{\|\vec{g}\|}$). *On notera que l'équation précédente revient à signer la courbure selon l'orientation de* $\vec{g}$.

Le problème est donc de dériver le vecteur tangent (orthogonal au gradient) par rapport à une abscisse curviligne de C. Or on ne connaît pas d'abscisse curviligne. Par contre on connaît les variations du gradient (orthogonal au vecteur tangent) dans les deux directions définies par le repère de l'image OXY. On suppose que le gradient définit sur toute l'image une forme différentielle et correspond à la normale à (C) en tous les points de (C). Le gradient étant calculé sur toute l'image par filtrage linéaire cette hypothèse est raisonnable (voir chapitre 2, section 2.2.3). Ainsi l'application de la règle de composition des applications et le fait que la tangente est par définition $\vec{t} = (\frac{dx}{ds}, \frac{dy}{ds})^t$ permet d'exprimer les variations du gradient en fonction de s sans utiliser explicitement l'abscisse curviligne. On obtient :

$$\frac{d\vec{g}}{ds} = \frac{\partial \vec{g}}{\partial x}\frac{dx}{ds} + \frac{\partial \vec{g}}{\partial y}\frac{dy}{ds} = H\vec{t} \qquad [11.7]$$

où H est le hessien de la fonction des niveaux de gris $I(x, y)$:

$$H = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix}$$

En combinant les équations [11.5], [11.6] et [11.7] on obtient

$$k = \frac{-t^T H t}{\|\vec{g}\|}$$

Le vecteur tangent $\vec{t}$ s'exprime simplement comme :

$$\|\vec{g}\|\vec{t} = \begin{pmatrix} -I_y \\ I_x \end{pmatrix}$$

Pour caractériser les points où la courbure est localement extremum (points de crête pour les maxima et points de vallée pour les minima (voir figure 11.10.) on annule la dérivée de la courbure par rapport à l'abscisse curviligne. :

$$k(s) = \frac{-t^T H t}{\|\vec{g}\|} \text{ avec } \|\vec{g}\|\vec{t} = \begin{pmatrix} -I_y \\ I_x \end{pmatrix} \text{ et } \vec{t} = \begin{pmatrix} x'(s) \\ y'(s) \end{pmatrix}$$

Les extremums de courbure sont localisés où la dérivée $k'(s)$ est nulle i.e. :

$$\frac{\frac{d(t^T H t)}{ds} \|\vec{g}\| - (t^T H t) \frac{d\|\vec{g}\|}{ds}}{\|\vec{g}\|^2} = 0$$

comme :

$$\frac{d\|\vec{g}\|}{ds} = \frac{g^T H t}{\|\vec{g}\|}$$

$$\frac{d(t^T H t)}{ds} = \frac{dt}{ds}^T H t + t^T \frac{d(Ht)}{ds}$$

$$\frac{d(Ht)}{ds} = \begin{pmatrix} t^T H_x t \\ t^T H_y t \end{pmatrix} + H\frac{dt}{ds}$$

avec :

$$H_x = \begin{pmatrix} I_{xxx} & I_{xyx} \\ I_{yxx} & I_{yyx} \end{pmatrix}, \; H_y = \begin{pmatrix} I_{xxy} & I_{xyy} \\ I_{yxy} & I_{yyy} \end{pmatrix}$$

On obtient finalement l'équation des extremums de courbure :

$$\frac{3\,(t^T H t)(g^T H t) - \|\vec{g}\|^2\, t^T \begin{pmatrix} t^T H_x t \\ t^T H_y t \end{pmatrix}}{\|\vec{g}\|^3} = 0$$

La figure 11.10. présente un exemple de résultat obtenu sur une image synthétique. Les dérivées partielles de l'image ont été calculées avec les filtres récursifs décrits dans la section 11.2.3.3.

Figure 11.10. *Résultats sur une ellipse synthétique 2D : A gauche l'image originale de l'ellipse, au milieu les contours extraits et à droite les extremums de courbure. Les deux faux extremums peuvent être supprimés par un simple seuillage utilisant la courbure maximum.*

11.2.3.1.2. Cas 3D

On considère une surface $\sum$ définie comme une iso-surface de $I(x, y, z)$.

Soit $\vec{t}$ un vecteur unitaire du plan tangent T_m en un point m et $\vec{g}$ la normale à la surface au point m (voir figure 11.11.).

On veut calculer la courbure de $\sum$ dans la direction $\vec{t}$, que nous noterons $k_{\vec{t}}$. Par définition [49, 103] la courbure d'une surface en un point dans une direction donnée est la courbure d'une courbe tracée sur la surface dont la normale principale est la normale à la surface en tous ses points et dont la tangente au point considéré est la direction choisie (on montre que le résultat est unique quelle que soit la courbe considérée [49]).

En utilisant le fait que $\vec{g} \cdot \vec{t} = 0$ sur $\sum$, et en considérant la dérivée directionnelle selon la direction définie par $\vec{t}$, on obtient comme dans le cas 2D :

$$L_{\vec{t}}(\vec{g} \cdot \vec{t}) = t^T H t + k_{\vec{t}} \vec{g} \cdot \vec{n}$$

où $L_{\vec{t}}$ est l'opérateur de dérivation de Lie et H le Hessien de la fonction des niveaux de gris $I(x, y, z)$:

$$H = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix}$$

Comme :

$$\vec{n} = \frac{\vec{g}}{||\vec{g}||}$$

en utilisant :

$$\frac{d\vec{g}}{ds} = H\vec{t} \qquad [11.8]$$

et

$$\frac{d\vec{t}}{ds} = k_{\vec{t}}\,\vec{n} = k_{\vec{t}}\frac{\vec{g}}{||\vec{g}||} \qquad [11.9]$$

il vient :

$$k_{\vec{t}} = -\frac{t^T H t}{||\vec{g}||}$$

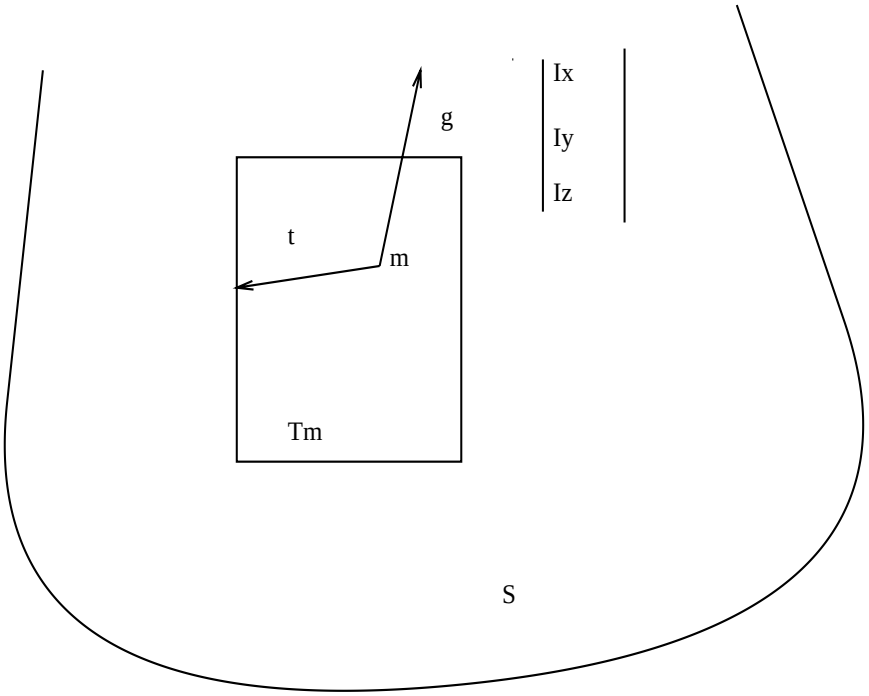


Figure 11.11. *Contour 3D, gradient et normale.*

Les courbures principales et les directions principales de courbure peuvent être obtenues en recherchant les directions $\vec{t}$ pour lesquelles $k_{\vec{t}}$ est un extremum. On peut utiliser le calcul suivant :

$$\vec{g} = \begin{bmatrix} I_x \\ I_y \\ I_z \end{bmatrix}$$

Soit $(\vec{a},\vec{b})$ une base orthonormée du plan tangent :

$$\|\vec{a}\| = \|\vec{b}\| = 1$$
$$\vec{a} \cdot \vec{g} = \vec{b} \cdot \vec{g} = 0;$$
$$\vec{t} = \cos\theta\ \vec{a} + \sin\theta\ \vec{b}$$

Si on dérive $k_{\vec{t}}$ par rapport à θ il vient :

$$-\|\vec{g}\|\frac{dk_{\vec{t}}}{d\theta} = \frac{d(t^T H t)}{d\theta} = \sin 2\theta(\vec{b}^T H\vec{b} - \vec{a}^T H\vec{a}) + 2\cos 2\theta\vec{a}^T H\vec{b}$$

Ainsi, si θ est une direction principale de courbure, cette dérivée s'annule, c'est-à-dire :

$$\frac{dk_{\vec{t}}}{d\theta} = 0 \iff \tan 2\theta = \frac{2\vec{a}^T H\vec{b}}{\vec{a}^T H\vec{a} - \vec{b}^T H\vec{b}} = E$$
$$\iff \left(\theta = \theta_1 = \frac{\arctan E}{2}\right) \vee \left(\theta = \theta_2 = \theta_1 + \frac{\pi}{2}\right)$$

On notera que la première équivalence implique que le dénominateur soit non nul. En fait le dénominateur s'annule si et seulement si le point est un ombilic [49]. Les deux directions principales de courbure $\vec{t}_1$ et $\vec{t}_2$ sont :

$$\vec{t}_1 = \cos\theta_1\ \vec{a} + \sin\theta_1\ \vec{b}$$
$$\vec{t}_2 = \cos\theta_2\ \vec{a} + \sin\theta_2\ \vec{b}$$

Les deux courbures principales sont :

$$K_1 = k_{\vec{t}_1} = \frac{t_1^T H t_1}{\|\vec{g}\|}$$
$$K_2 = k_{\vec{t}_2} = \frac{t_2^T H t_2}{\|\vec{g}\|}$$

Dans la suite nous supposerons que $\mid k_{\vec{t}_1} \mid > \mid k_{\vec{t}_2} \mid$ i.e. $\vec{t}_1$ est la direction maximum de courbure et $k_{\vec{t}_1}$ la courbure maximum.

De manière à éviter le calcul de l'arc-tangente la méthode suivante peut aussi être utilisée pour déterminer les courbures principales et les directions principales de courbure :

Les courbures principales sont la solution du problème d'optimisation sous contrainte suivant :

$$Min \ < \ H \ d \ , \ d \ >$$
$$s.c. \begin{cases} < \ d \ , \ d \ > = \ 1 \\ < \ d \ , \ g \ > = \ 0 \end{cases}$$

On définit une matrice de rotation P de manière à obtenir un nouveau repère dont le premier vecteur est parallèle à $\vec{g}$:

$$P = \begin{pmatrix} \frac{I_x}{\delta} & \frac{I_y}{\gamma} & \frac{I_z \ I_x}{\gamma \delta} \\ \frac{I_y}{\delta} & \frac{-\ I_x}{\gamma} & \frac{I_y \ I_z}{\gamma \delta} \\ \frac{I_z}{\delta} & 0 & -\ \frac{\gamma}{\delta} \end{pmatrix} = \begin{pmatrix} \vec{g} & \vec{h} & \vec{f} \end{pmatrix}$$

avec

$$\gamma = \sqrt{I_x^2 + I_y^2} \text{ and } \delta = \sqrt{I_x^2 + I_y^2 + I_z^2}$$

Après le changement de variables $P \ w \ = \ d$ le problème de minimisation précédent devient :

$$Min \ < \ H \ P \ w \ , \ P \ w \ >$$
$$s.c. \begin{cases} < \ w \ , \ w \ > = \ 1 \\ \quad w_1 \ = \ 0 \end{cases}$$

avec

$$w = (w_1, w_2, w_3)^T$$

En introduisant les notations suivantes :

$$H^{'} \;=\; P^T \; H \; P \;=\; \begin{pmatrix} \cdot & \cdot & \cdot \\ \cdot & & H_e^{'} \\ \cdot & & \end{pmatrix}$$

$$w \;=\; \begin{pmatrix} \cdot \\ w_e \end{pmatrix}$$

On obtient le problème équivalent :

$$Min \;<\; H_e^{'} \; w_e \;,\; w_e \;>$$

$$s.c. \;<\; w_e \;,\; w_e \;> =\; 1$$

En utilisant *la technique des multiplicateurs de Lagrange*, on réduit ce problème à la diagonalisation d'une matrice $H_e^{'}$. Les valeurs propres de la matrice $H_e^{'}$ correspondent aux courbures principales. Les directions principales de courbure sont obtenues en appliquant la matrice P aux vecteurs propres de $H_e^{'}$. Cette méthode permet d'exprimer les courbures principales K_i et les directions principales de courbure (d_i) en utilisant seulement les dérivées premières et secondes des images :

$$K_i \;= \frac{h^T H h + f^T H f \;\pm\; \sqrt{(h^T H h - f^T H f)^2 \;+\; 4\;(h^T H f)^2}}{2}$$

$$d_i = \begin{pmatrix} h_1 + f_1 \frac{K_i - h^T H h}{f^T H h} \\ h_2 + f_2 \frac{K_i - h^T H h}{f^T H h} \\ h_3 + f_3 \frac{K_i - h^T H h}{f^T H h} \end{pmatrix} \text{ avec } i = 1, 2$$

Les figures 11.12. à 11.15. présentent des exemples de résultats obtenus en appliquant ces formules. Les dérivées partielles de l'image sont obtenues avec les filtres récursifs décrits dans la section 11.2.3.

Figure 11.12. *Extremums de courbure obtenus par simple seuillage du critère d'extrémalité sur une image synthétique d'un ellipsoïde. On obtient à la fois les extremums maximums de courbure (points de crête) mais aussi les extremums minimums de courbure (points de creux) qui sont cependant plus instables.*

Pour la caractérisation des points de crête, le cas 3D n'est pas exactement l'extension du cas 2D car il existe une courbure pour chaque direction, i.e. pour chaque courbe de la surface incluant un point donné m, et telle que la normale à la courbe correspond à la normale à la surface. Un extremum local de courbure est alors un extremum local de la courbure maximum dans la direction maximum de courbure. Il est important de remarquer que cette définition est légale à cause de la continuité et de la différentiabilité du champ des courbures principales sur une surface régulière (voir [49] page 156). Ainsi

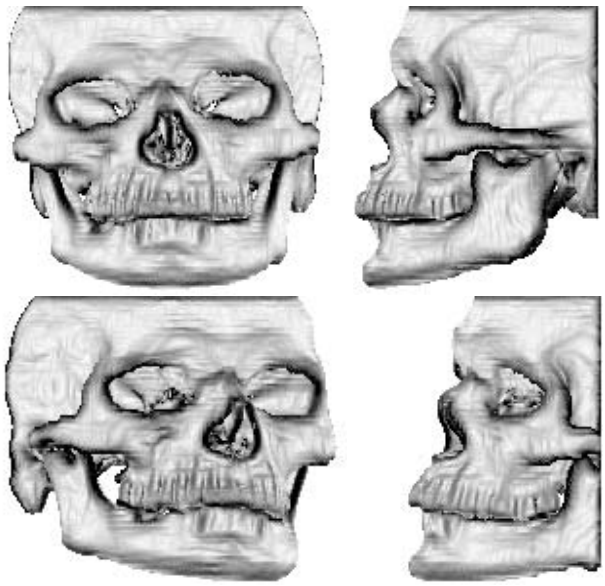


Figure 11.13. *Vues en perspective des courbures maximums pour les deux positions, aucun rendu n'est réalisé : seul l'amplitude de la courbure maximum est affichée (en haut à gauche : vue de face pour la position A , en haut à droite : vue de profil pour la position A, en bas à gauche : vue de face pour la position B, en bas à doite : vue de profil pour la position B).*

Figure 11.14. *Images obtenues par seuillage de la courbure maximum. En haut : vue de face pour la position A (les points de courbure maximum élevée sont en noir) ; en bas : vue de face pour la position B.*

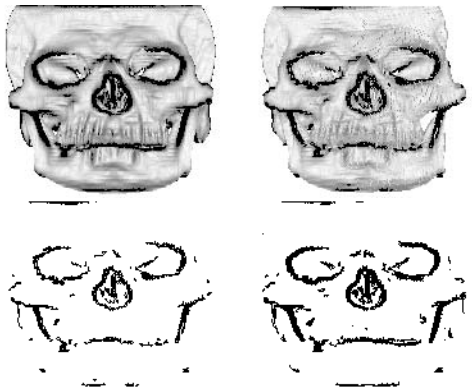


Figure 11.15. *En haut à gauche : seuillage des courbures maximums pour la position A, en haut à droite : même chose pour la position B après l'application de la transformation rigide (rotation et translation) de B vers A, en bas à gauche même chose qu'en haut à droite mais en ne visualisant que les points de forte courbure maximum, en bas à droite : superposition des deux résultats.*

on calcule la dérivée de la courbure $k_{\vec{d}}$ dans la direction $\vec{d}$, les extremums locaux sont les points annulant cette dérivée dans la direction maximum de courbure. On notera que l'on peut aussi définir un point de crête comme un extremum de la courbure principale le long d'une ligne de courbure ce qui est très similaire [181].
Soit $\vec{t_1}$ la direction maximum de courbure, la dérivée de la courbure dans la direction $\vec{t_1}$ est :

$$\nabla_{\vec{t_1}} k_{\vec{t_1}}(m) = \lim_{\lambda \to 0} \frac{k_{\vec{t_1}}(m + \lambda \vec{t_1}) - k_{\vec{t_1}}(m)}{\lambda} =$$

$$\nabla k_{\vec{t_1}}(m) \cdot \vec{t_1} = \begin{bmatrix} \frac{\partial k_{\vec{t_1}}}{\partial x}(m) \\ \frac{\partial k_{\vec{t_1}}}{\partial y}(m) \\ \frac{\partial k_{\vec{t_1}}}{\partial z}(m) \end{bmatrix} \cdot \begin{bmatrix} t_{1x} \\ t_{1y} \\ t_{1z} \end{bmatrix}$$

Nous avons :

$$k_{t_1}(m) = \frac{I_{xx}t_{1x}^2 + I_{yy}t_{1y}^2 + I_{zz}t_{1z}^2 + 2I_{xy}t_{1x}t_{1y} + 2I_{xz}t_{1x}t_{1z} + 2I_{yz}t_{1y}t_{1z}}{\|\vec{g}\|^2}$$

Il vient :

$$\nabla_{\vec{t_1}} k_{t_1}(m) = \frac{\|\vec{g}\|^2 t_1^T \cdot \begin{bmatrix} t_1^T H_x t_1 \\ t_1^T H_y t_1 \\ t_1^T H_z t_1 \end{bmatrix} - \left(t_1^T H t_1\right)\left(t_1^T H g\right)}{\|\vec{g}\|^3}$$

H étant le hessien de I et H_x, H_y, H_z ses dérivées partielles.

$$H = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}$$

$$H_x = \frac{\partial H}{\partial x} = \begin{bmatrix} I_{xxx} & I_{xyx} & I_{xzx} \\ I_{yxx} & I_{yyx} & I_{yzx} \\ I_{zxx} & I_{zyx} & I_{zzx} \end{bmatrix}$$

L'expression $\nabla_{\vec{t_1}} k_{\vec{t_1}}(m)$ est appelée critère d'extrémalité. Il est important de remarquer que ce critère d'extrêmalité est un "invariant euclidien **au signe près**" car le sens des directions pricipales de courbures n'est pas défini. Par contre les passages par zéro de ce critère sont invariants par transformation rigide. Cette remarque a des conséquences lorsque l'on veut déterminer les zéros du critère en détectant ses changements de signe. En effet, même si dans la pratique on peut définir une cohérence du sens des directions maximums de courbure, il subsiste des changements de signe du critère "normalisé" ne correspondant pas à un passage par zéro. Une manière de tourner cette difficulté est de calculer non pas simplement la dérivée de la courbure maximum le long de la direction maximum de courbure mais *le produit des dérivées des deux courbures principales le long des directions principales correspondantes.* Moyennant quelques précautions pour définir le sens des directions principales de courbures on définit ainsi un invariant euclidien introduit par JP. Thirion et appellé "extrêmalité gaussienne" [169]. Il est évident que les zéros du critère d'extrêmalité sont aussi des zéros de l'extrêmalité gaussienne. Cependant, en général, l'interprétation géométrique des passages par zéro de l'extrêmalité gaussienne ("maillage extrêma [169]) est difficile.

La figure 11.16. illustre le critère d'extrêmalité sur une image 3D (scanner X) d'un crâne .

Figure 11.16. *Vue en perspective pour la position A des points où le critère d'extrémalité est faible.*

11.2.3.2. Géométrie différentielle dans R^4 et images 3D

De même qu'une image bidimensionnelle $I(x, y)$ définit une surface dans R^3 [140] de trace $(x, y, I(x, y))$, une image 3D définit une hypersurface dans R^4 de trace $(x, y, z, I(x, yz))$ [133]. Les notions de base de géométrie différentielle dans R^3 (première et deuxième formes fondamentales, endomorphisme de Weingarten, courbures et directions principales...) se généralisent, quasiment directement à R^4 (et d'ailleurs à R^n). Une idée naturelle est de s'intéresser aux caractéristiques différentielles de l'hypersurface [105] définie par l'image. Intuitivement, ces caractéristiques différentielles dépendent à la fois de la géométrie des surfaces et de la fonction des niveaux de gris. Un avantage de cette approche est qu'elle s'applique même dans le cas où les surfaces ne correspondent pas localement à des isocontours. Une difficulté est de déterminer parmi les 3 courbures 4D, celles caractérisant les surfaces et celles associées à la fonction des niveaux de gris. On verra que pour un modèle réaliste, en un point de

contour, les deux courbures les plus fortes correspondent aux courbures de la surface, et la courbure la plus faible à la fonction des niveaux de gris.

11.2.3.2.1. Courbures d'une image 3D

Soit E l'espace affine euclidien orienté de dimension 4 et V l'espace vectoriel associé.

Définition 1 (*hypersurface paramétrée*) *Une hypersurface paramétrée de E est un couple (Σ, M) où Σ est un sous-ensemble de E et M une application continue d'un ouvert U de R^3 vers E dont l'image $M(U)$ est égale à Σ; M est appelée représentation paramétrique de l'hypersurface Σ.*

Si M est C^k l'hypersurface Σ est dite C^k.

Application aux images 3D :

Soit I(x,y,z) la fonction des niveaux de gris d'une image 3D et soit :

$$\Sigma = \left\{ \vec{v} \in R^4 / \vec{v} = (x, y, z, I(x, y, z)) \right\}$$

$$M : R^3 \longrightarrow E$$

donné par

$$(x, y, z) \longrightarrow (x, y, z, I(x, y, z))$$

Ainsi on peut considérer une image 3D comme une hypersurface dans R^4

Définition 2 (*hyperplan tangent*) *Soit (Σ, M) une hypersurface paramétrée de classe C^k $(k \geq 1)$. Si $M_0 = M(u_0, v_0, w_0)$ est tel que les vecteurs $\frac{\partial \vec{M}}{\partial v}, \frac{\partial \vec{M}}{\partial v}$ et $\frac{\partial \vec{M}}{\partial w}$ calculés au point M_0 sont linéairement indépendants, alors l'hypersurface (Σ, M) admet un hyperplan tangent au point M_0 et l'hyperplan tangent Σ en M_0 est donné par :*

$$T_{M_0}\Sigma = R\frac{\partial \vec{M}}{\partial u} \oplus R\frac{\partial \vec{M}}{\partial v} \oplus R\frac{\partial \vec{M}}{\partial w}$$

La normale à l'hypersurface en M_0 est donnée par :

$$\vec{K_0} = \frac{\partial \vec{M}}{\partial u} \wedge \frac{\partial \vec{M}}{\partial v} \wedge \frac{\partial \vec{M}}{\partial w}$$

Application aux images 3D :

Soit $\vec{G} = (I_x, I_y, I_z)^t$ le gradient en un point $P_0(x_0, y_0, z_0)$; (ici nous utilisons les indices pour noter les dérivées partielles). Le point $P_0 : (x_0, y_0, z_0)$ dans R^3 correspond au point $\vec{M}(x_0, y_0, z_0) = M_0 : (x_0, y_0, z_0, I(x_0, y_0, z_0))$ dans R^4 ainsi :

$$\frac{\partial \vec{M}}{\partial u} = (1, 0, 0, \ I_x)^t$$

$$\frac{\partial \vec{M}}{\partial v} = (0, 1, 0 \ I_y)^t$$

$$\frac{\partial \vec{M}}{\partial w} = (0, 0, 1, \ I_z)^t$$

La normale à l'hypersurface au point M_0 est :

$$\vec{K_0} = \frac{\partial \vec{M}}{\partial u} \wedge \frac{\partial \vec{M}}{\partial v} \wedge \frac{\partial \vec{M}}{\partial w}$$

$$= \frac{1}{\sqrt{D}}(I_x, I_y, I_z, -1)^t$$

avec :

$$D = 1 + I_x^2 + I_y^2 + I_z^2 = \|\nabla M\|^2$$

Définition 3 (*première forme fondamentale*) *La restriction du produit scalaire à l'espace tangent $T_{M_0}\Sigma$ en Σ au point M_0 est un produit scalaire euclidien sur cet espace vectoriel qu nous noterons $\phi_1(M_0) = \phi_1(u_0, v_0, w_0)$. Etant donné que $(\frac{\partial \vec{M}}{\partial u}\frac{\partial \vec{M}}{\partial v}, \frac{\partial \vec{M}}{\partial w})$ est une base de $T_{M_0}\Sigma$ la matrice de ϕ_1 pour cette base est :*

$$F_1 = \begin{pmatrix} \frac{\partial \vec{M}}{\partial u} \cdot \frac{\partial \vec{M}}{\partial u} & \frac{\partial \vec{M}}{\partial u} \cdot \frac{\partial \vec{M}}{\partial v} & \frac{\partial \vec{M}}{\partial u} \cdot \frac{\partial \vec{M}}{\partial w} \\ \frac{\partial \vec{M}}{\partial v} \cdot \frac{\partial \vec{M}}{\partial u} & \frac{\partial \vec{M}}{\partial v} \cdot \frac{\partial \vec{M}}{\partial v} & \frac{\partial \vec{M}}{\partial v} \cdot \frac{\partial \vec{M}}{\partial w} \\ \frac{\partial \vec{M}}{\partial w} \cdot \frac{\partial \vec{M}}{\partial u} & \frac{\partial \vec{M}}{\partial w} \cdot \frac{\partial \vec{M}}{\partial v} & \frac{\partial \vec{M}}{\partial w} \cdot \frac{\partial \vec{M}}{\partial w} \end{pmatrix} = \begin{pmatrix} e & f & h \\ f & g & i \\ h & i & j \end{pmatrix}$$

ϕ_1 est appelée la première forme fondamentale de l'hypersurface Σ au point M_0.

Une application directe de la définition précédente donne la première forme fondamentale de l'hypersurface définie par la fonction des niveaux de gris en un point P_0 :

$$F_1 = \begin{pmatrix} 1+I_x & I_xI_y & I_xI_z \\ I_xI_y & 1+I_y & I_yI_z \\ I_xI_z & I_yI_z & 1+I_z \end{pmatrix}$$

Définition 4 ***(seconde forme fondamentale)*** *La seconde forme fondamentale de l'hypersurface en un point M_0 est définie par sa matrice dans la base :*

$$(\frac{\partial \vec{M}}{\partial u}, \frac{\partial \vec{M}}{\partial v}(u_0, v_0, w_0) \frac{\partial \vec{M}}{\partial w}(u_0, v_0, w_0))$$

En utilisant la normale à la surface $K = (u_0, v_0, w_0)$, on obtient :

$$F_2 = \begin{pmatrix} l & m & o \\ m & n & p \\ o & p & q \end{pmatrix}$$

avec :

$$l = -\frac{\partial \vec{M}}{\partial u} \cdot \frac{\partial \vec{K}}{\partial u} = \vec{K} \cdot \frac{\partial \vec{M}}{\partial u^2}$$

$$m = -\frac{\partial \vec{M}}{\partial u} \cdot \frac{\partial \vec{K}}{\partial v} = \vec{K} \cdot \frac{\partial^2 \vec{M}}{\partial u \partial v}$$

$$o = -\frac{\partial \vec{M}}{\partial u} \cdot \frac{\partial \vec{K}}{\partial w} = \vec{K} \cdot \frac{\partial^2 \vec{M}}{\partial u \partial w}$$

$$p = -\frac{\partial \vec{M}}{\partial v} \cdot \frac{\partial \vec{K}}{\partial w} = \vec{K} \cdot \frac{\partial^2 \vec{M}}{\partial v \partial w}$$

$$q = -\frac{\partial \vec{M}}{\partial w} \cdot \frac{\partial \vec{K}}{\partial w} = \vec{K} \cdot \frac{\partial^2 \vec{M}}{\partial w^2}$$

$$n = -\frac{\partial \vec{M}}{\partial v} \cdot \frac{\partial \vec{K}}{\partial v} = \vec{K} \cdot \frac{\partial^2 \vec{M}}{\partial v^2}$$

Une application directe de la définition 4 donne la seconde forme fondamentale en un point $P_0 = (x_0, y_0, z_0)$

$$F_2 = \frac{1}{\sqrt{D}} \begin{pmatrix} -I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & -I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & -I_{zz} \end{pmatrix}$$

Définition 5 *L'endomorphisme de Weingarten de l'hypersurface au point M_0 est l'application L_0 telle que :*

$$L_0 : T_{M_0}\Sigma \longrightarrow T_{M_0}\Sigma$$

$$L_0(X_u \frac{\partial \vec{M}}{\partial u} + X_v \frac{\partial \vec{M}}{\partial v} + X_w \frac{\partial \vec{M}}{\partial w})$$

$$= -X_u \frac{\partial \vec{K}}{\partial u} - X_v \frac{\partial \vec{K}}{\partial v} - X_w \frac{\partial \vec{K}}{\partial w}$$

On peut aisément montrer que la matrice de L_0 dans la base $(\frac{\partial \vec{M}}{\partial u}, \frac{\partial \vec{M}}{\partial v}, \frac{\partial \vec{M}}{\partial w})$ est donnée par :

$$W = F_1^{-1} F_2$$

où F_1 et F_2 sont respectivement les matrices des première et deuxième formes fondamentales dans la base $(\frac{\partial \vec{M}}{\partial u} \frac{\partial \vec{M}}{\partial v}, \frac{\partial \vec{M}}{\partial w})$.

Définition 6 *Les vecteurs propres de L_0 que nous noterons $\vec{X}_1, \vec{X}_2, \vec{X}_3$ sont appelés les directions principales de courbure, et les valeurs propres correspondantes : $\lambda_1, \lambda_2, \lambda_3$ sont les courbures principales de (Σ, M) à M_0. Le produit $\lambda_1\lambda_2\lambda_3 (= det\, W)$ est la courbure gaussienne. La moyenne des valeurs des courbures : $\frac{\lambda_1 + \lambda_2 + \lambda_3}{3} (= \frac{trace\, W}{3})$ est la courbure moyenne.*

Remarque

Dans ce cas le signe de la courbure gaussienne n'est pas invariant [167] par changement d'orientation de la surface.

Définition 7 *Le vecteur propre de L_0 correspondant à la valeur propre de module maximum est la direction maximum de courbure $\vec{X}_m$ et la valeur propre associée λ_m est la courbure maximum.*

L'application des définitions 5,6,7 aux images 3D est directe. Ces caractéristiques différentielles peuvent être utilisées afin de caractériser les surfaces [133]. On peut montrer que les deux courbures de modules les plus importants sont reliées à la géométrie de la surface dans les cas réalistes [133]. La figure 11.17. illustre les relations fortes existant entre la courbure maximum de l'hypersurface et la courbure maximum de la surface.

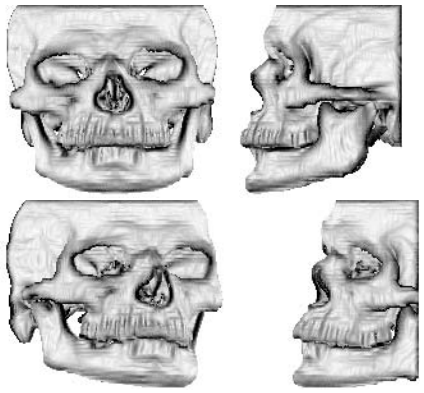


Figure 11.17. *Vues en perspective des courbures maximums de l'hypersurface pour les deux positions, aucun rendu n'est effectué : la valeur absolue de la courbure maximum est visualisée (en haut à gauche : vue de face pour la position A, en haut à droite : vue de profil pour la position A, en bas à gauche : vue de face pour la position B, en bas à droite : vue de profil pour la position B).*

11.2.3.3. Calcul des dérivées partielles d'une image 3D

On peut calculer ces dérivées avec le même type d'opérateurs de filtrage linéaire que ceux décrits au chapitre 2 pour la détection de contours. Dans cette partie, on décrit l'exemple particulier de filtres basés sur le filtre de Deriche [134]. Cependant, d'autres filtres comme le filtre gaussien peuvent être utilisé [168] (voir chapitre : "Détection de contours"). Dans les exemples présentés précédemment, on calcule les dérivées partielles de l'image avec les filtres :

$f_0(x) = c_0(1 + \alpha \mid x \mid)e^{-\alpha|x|}$, (composante de lissage);
$f_1(x) = -c_1 x\alpha^2 e^{-\alpha|x|}$, (composante de première dérivée);
$f_2(x) = c_2(1 - c_3\alpha \mid x \mid)e^{-\alpha|x|}$, (composante de seconde dérivée)
$f_3(x) = c_4\frac{\mid x \mid}{x}(c_5 + 1 - \alpha c_5 \mid x \mid)e^{-\alpha|x|}$, (composante de troisième dérivée).

Ces fonctions sont utilisées comme des filtres de convolution pour le lissage, la dérivation première et la dérivation seconde d'une fonction à une variable. Les cœfficients de normalisation $c_0, c_1, c_2, c_3, c_4, c_5$ sont choisis de manière à assurer une réponse correcte pour une fonction polynomiale (séparable en X,Y,Z) :

$$\int_{-\infty}^{+\infty} f_0(x)dx = 1;$$

$$\int_{-\infty}^{+\infty} x f_1(x)dx = 1;$$

$$\int_{-\infty}^{+\infty} f_2(x)dx = 0 \text{ et } \int_{-\infty}^{+\infty} \frac{x^2}{2} f_2(x)dx = 1;$$

$$\int_{-\infty}^{+\infty} \frac{x^3}{6} f_3(x)dx = 1 \text{ et } \int_{-\infty}^{+\infty} x f_3(x)dx = 0.$$

Ces conditions assurent donc que les filtres f_0, f_1, f_2, f_3 produisent des dérivées correctes si on les applique à des polynomes. Quand les filtres sont appliqués à des données discrètes, les formules discrètes équivalentes sont utilisées pour trouver les cœfficients de normalisation. Par exemple pour un échantillonnage entier on obtient pour c_0 :

$$\sum_{-\infty}^{+\infty} f_0(n) = 1.$$

En utilisant des versions discrètes des formules ci dessus on obtient les valeurs suivantes pour les cœfficients de normalisation dans le cas des convolutions discrètes :

$$c_0 = \frac{(1 - e^{-\alpha})^2}{1 + 2e^{-\alpha}\alpha - e^{-2\alpha}};$$

$$c_1 = \frac{-(1 - e^{-\alpha})^3}{2\alpha^2 e^{-\alpha}(1 + e^{-\alpha})};$$

$$c_2 = \frac{-2(1-e^{-\alpha})^4}{1+2e^{-\alpha}-2e^{-3\alpha}-e^{-4\alpha}};$$

$$c_3 = \frac{(1-e^{-2\alpha})}{2\alpha e^{-\alpha}};$$

$$c_4 = -\frac{(1-e^{-\alpha})^4((\alpha+1)e^{-\alpha}+\alpha-1)}{2\alpha^2 e^{-2\alpha}(1+e^{-\alpha})};$$

$$c_5 = \frac{1-e^{-\alpha}}{(\alpha+1)e^{-\alpha}+\alpha-1}.$$

La valeur de α contrôle le degré de lissage, et doit être spécifiée pour compléter la définition de : f_0, f_1, f_2, f_3.

Chaque fonction $f_{i+1}(x)$ est (par construction) proportionnelle à la dérivée de la fonction précédente $f_i(x)$, excepté en $x = 0$, si les fonctions ne sont pas différentiables.

Soit une fonction image $I = I(x, y, z)$, des filtres séparables sont utilisés pour calculer les dérivées partielles (lissées). En utilisant des indices pour noter la dérivation partielle, on utilise les approximations suivantes pour calculer les dérivées partielles :

$$I_x = (f_1(x)f_0(y)f_0(z)) * I,$$

$$I_{xx} = (f_2(x)f_0(y)f_0(z)) * I,$$

$$I_{xy} = (f_1(x)f_1(y)f_0(z)) * I,$$

$$I_{xyz} = (f_1(x)f_1(y)f_1(z)) * I,$$

$$I_{xxz} = (f_2(x)f_0(y)f_1(z)) * I,$$

$$I_{xxx} = (f_3(x)f_0(y)f_0(z)) * I.$$

Ces formules donnent des résultats exacts quand I est un polynome séparable en (x, y, z) mais dans les autres cas des résultats approchés. Ces filtres s'implantent de manière séparable et récursive ou par masque de convolution (voir chapitre 2, section 2.5.4).

11.2.3.4. Une approche multi-échelle

11.2.3.4.1. Introduction

Nous décrivons dans cette partie un exemple d'introduction de la notion d'échelle pour le problème particulier d'extraction des lignes de crête dans des images volumiques 3D [136]. Ce cas particulier illustre bien la notion d'échelle toujours présente de manière implicite ou explicite en Vision par Ordinateur et plus généralement en perception automatique. Le nombre important de travaux consacrés à la multi-résolution montre la motivation des chercheurs et aussi la difficulté de trouver des solutions générales. Ce constat est principalement dû au fait que la notion d'échelle est intuitive pour toute tâche de perception mais que la plupart des modèles mathématiques ne restitue que partiellement la notion perceptuelle.

11.2.3.4.2. Pourquoi une approche multi-échelle

Les résultats des algorithmes d'extraction de caractéristiques différentielles décrits précédemment dépendent du calcul des dérivées partielles de l'image (voir section précédente). Ainsi les caractéristiques des opérateurs mis en œuvre (α pour les filtres Deriche et σ pour les filtres gaussiens) conditionnent fortement les résultats. Pour l'extraction des lignes de crête qui correspondent aux passages par zéro du critère d'extrémalité on constate généralement les faits suivants :

- Pour des données simples, on obtient de bons résultats avec une valeur de α mais on ne sait comment trouver automatiquement cette valeur (ce qui est d'ailleurs par nature un problème de poule et d'œuf!).
- Pour des données plus complexes, la "bonne" (en fonction de critères visuels humains) valeur de α varie selon les zones de l'image.
- Pour des données bruitées, seules les lignes de crête qui sont détectées à plusieurs résolutions sont intéressantes.

Ainsi de la même manière que pour la détection de contours, une approche multi-échelle comporte un intérêt important. Le problème est : quelle approche multi-échelle? Nous donnons dans la section suivante, un exemple d'approche multi-échelle qui a le mérite de poser clairement les problèmes.

11.2.3.4.3. Exemple d'algorithme

Les principales étapes de l'algorithme d'extraction des lignes de crêtes des iso-surfaces dans une image volumique décrit précédemment sont :

1. Calcul des dérivées partielles d'ordre 1,2,3 de l'image $I(x,y,z)$ ($\frac{\partial^n f}{\partial x^m \partial y^p \partial z^q}$, $m+p+q=3$ en utilisant des filtres récursifs par exemple approximant les filtres gaussiens (voir chapitre : "Détection de contours") pour une valeur donnée de σ.

2. Détermination de la carte des contours $3D$ en utilisant les dérivées d'ordre 1 de I (voir partie : "Détection de contours 3d") ;

3. Pour chaque point de contour $3D$ calculer :

- Les deux courbures principales et les directions principales correspondantes en utilisant les formules de la partie : "relations entre les dérivées des images et les courbures des contours" ;
- Le critère d'extrêmalité (dérivée de la courbure maximum le long de la direction principale associée ;

4. Construction d'une image du critère d'extrémalité $C_\sigma(x,y,z)$ telle que pour chaque point de contour (x,y,z), $C_\sigma(x,y,z)$ prend la valeur du critère d'extrémalité et 0 sinon ;

5. Construction d'une image $Z_\sigma(x,y,z)$ mise à 1 en chaque point de contour qui est un passage par zéro du critère d'extrémalité et à 0 sinon.

La dernière étape de cet algorithme consiste à déterminer les passages par zéro d'une fonction définie sur la trace discrète d'une surface (tracée par les points de contour $3D$) ce qui est un problème difficile en soit même. On peut mettre en œuvre des méthodes simples pour extraire ces passages par zéro, mais une solution "propre" demande plus d'attention. Une solution intéressante sera trouvée dans [168].

Ainsi, la sortie finale de cet algorithme est une image Z_σ représentant l'ensemble des points de contour qui sont les passages par zéro du critère d'extrêmalité. Chaque valeur de σ définit une image Z_σ représentant les lignes de crête pour l'échelle définie par σ.

De manière à représenter et à fusionner les résultats obtenus à différentes échelles $\sigma_i, i = 1,n$, on peut par exemple utiliser une structure de données

que nous appelerons "Graphe d'adjacence multi-échelle" : $G_{\sigma_1,\sigma_2,\ldots\sigma_n}$. définie comme suit :

1. Chaque nœud de $G_{\sigma_1,\sigma_2,\ldots\sigma_n}$ correspond à un point (i,j,k) tel que pour au moins une échelle σ_m nous avons $Z_{\sigma_m}(i,j,k) = 1$;

2. Les attributs associés à chaque nœud sont :

(a) Les coordonnées du point $3D$ correspondant : $((i,j,k))$;

(b) Les valeurs des échelles pour lesquelles ce point est un point de crête (tous les σ_p tels que $Z_{\sigma_p}(i,j,k) = 1$) ;

(c) Les caractéristiques différentielles extraites pour toutes les échelles : courbures principales et directions principales de courbure, valeur du critère d'extrêmalité.

Ainsi $G_{\sigma_1,\sigma_2,\ldots\sigma_n}$ représente les résultats de l'extraction des points de crête pour les différentes échelles et leurs relations spatiales. Cette structure de données est particulièrement efficace quand la stabilité des points de crête pour différentes échelles est un bon critère de sélection. Par exemple on peut mettre en œuvre la stratégie suivante :

1. Sélectionner les nœuds du graphe correspondant aux points qui sont points de crête pour un nombre "suffisant" d'échelles.

2. Extraire les composantes connexes qui ont un nombre suffisant de nœuds (ce seuil correspond à la longueur minimale d'une ligne de crête).

Les figures suivantes illustrent les résultats de ce type d'algorithme sur des données réelles (deux images scanner 3D du même crane pour deux positions différentes).

11.2.3.5. Conclusion

Des calculs établissent les relations mathémathiques existant entre les caractéristiques différentielles des surfaces et les caractéristiques différentielles des images 3D. L'hypothèse clé est que le gradient d'une image 3D définit un champ différentiable qui correspond en tout point de contour (trace de la surface) à la normale à la surface. Dans le cas où cette hypothèse n'est pas vérifiée, on peut rechercher des invariants de l'hypersurface définie par l'image 3D. En effet par définition les caractéristiques différentielles de l'hypersurface sont des propriétés intrinsèques (invariantes par le groupe euclidien) de l'image 3D.

On illustre ces calculs en déterminant les dérivées partielles de l'image avec des filtres récursifs du même type que ceux utilisés pour la détection des contours. De plus la largeur des filtres mis en œuvre définit de manière naturelle une notion d'échelle pour le calcul des caractéristiques différentielles. Sur des données réelles, les résultats expérimentaux sont comparables à ceux obtenus par les méthodes d'approximation locale décrites précédemment mais impliquent un coût calculatoire beaucoup moins important.

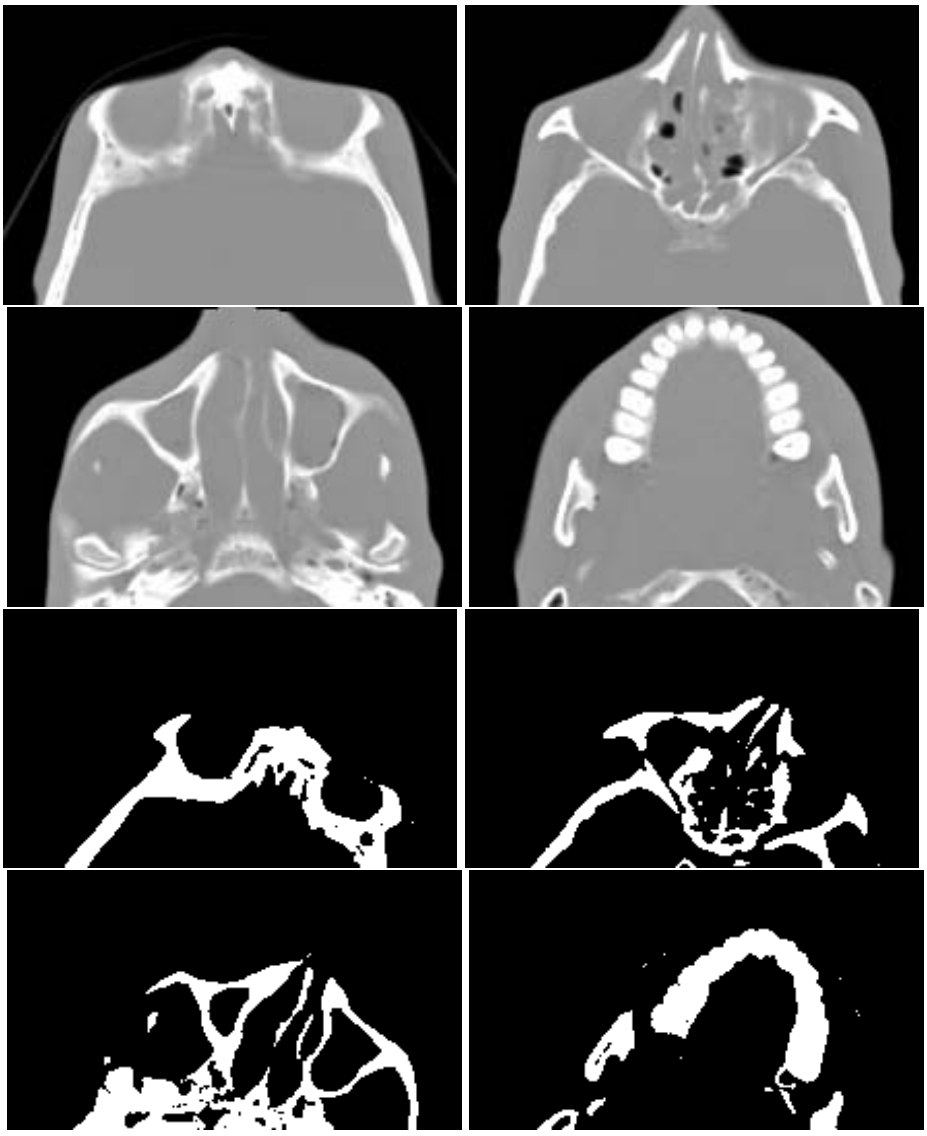


Figure 11.18. *Coupes de deux images obtenues par scanner X du même crane dans deux positions différentes A and B (dimensions des images : 192.128.151 pour la position A and 220.128.148 pour la position B). en haut : coupes correspondant à la position A, en bas : coupes correspondant à la position B.*

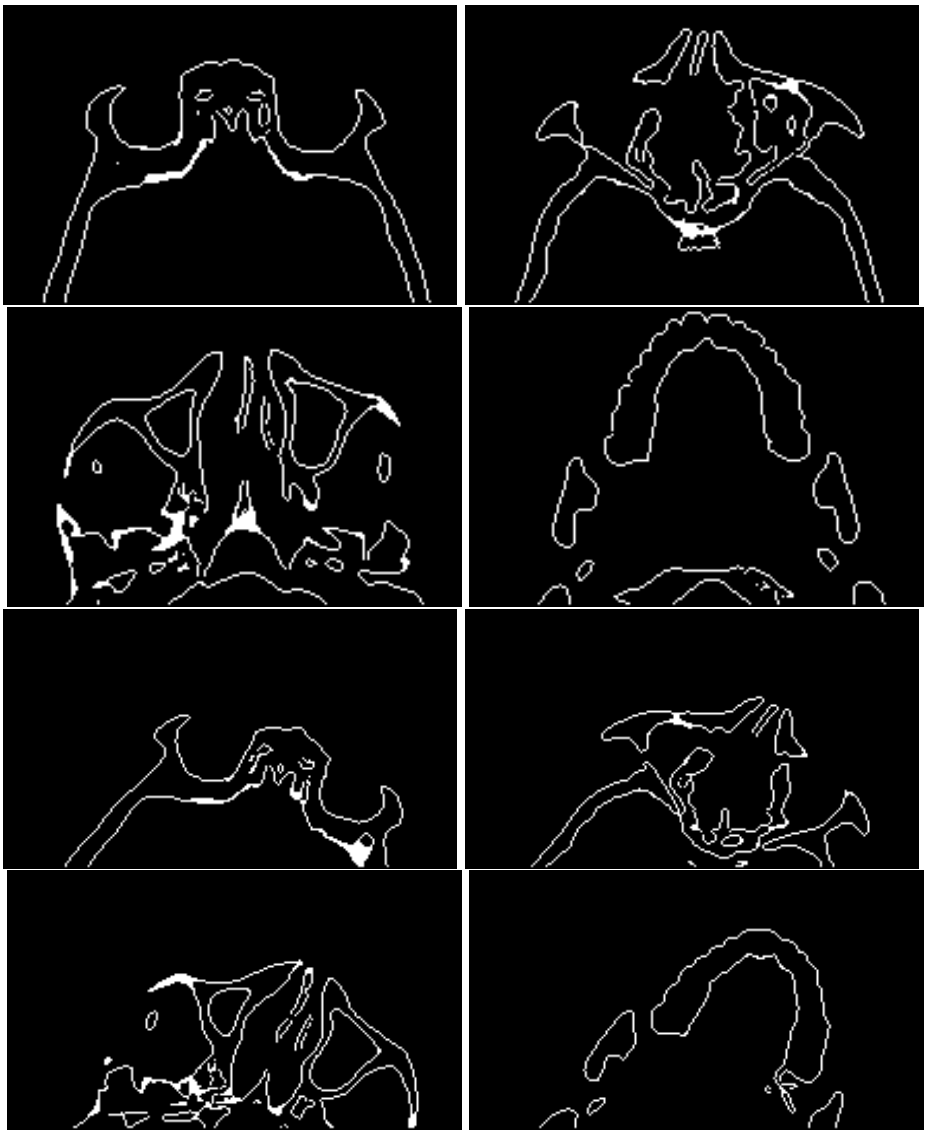


Figure 11.19. *Coupes de la carte des contours 3D correspondant à la figure précédente). En haut : coupes correspondant à la position A, En bas : coupes correspondant à la position B.*

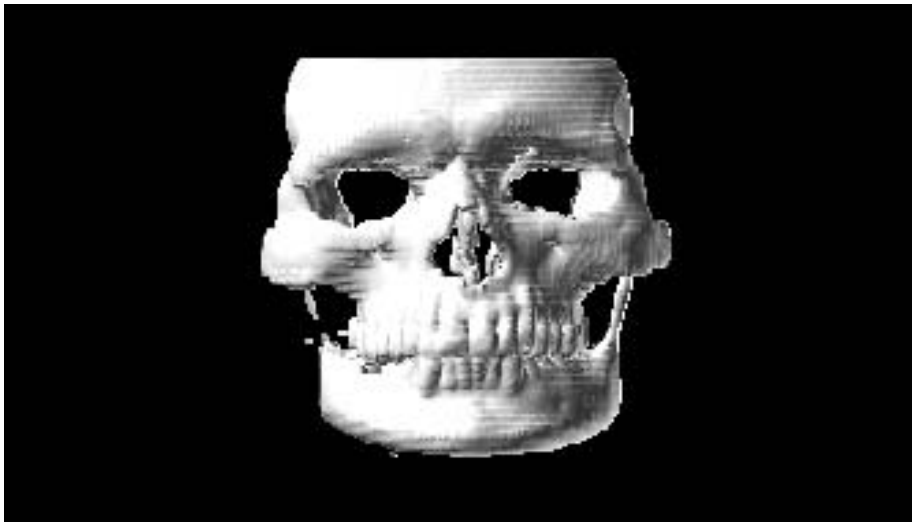


Figure 11.20. *Vue en perspective de la carte des contours 3D pour la position A*

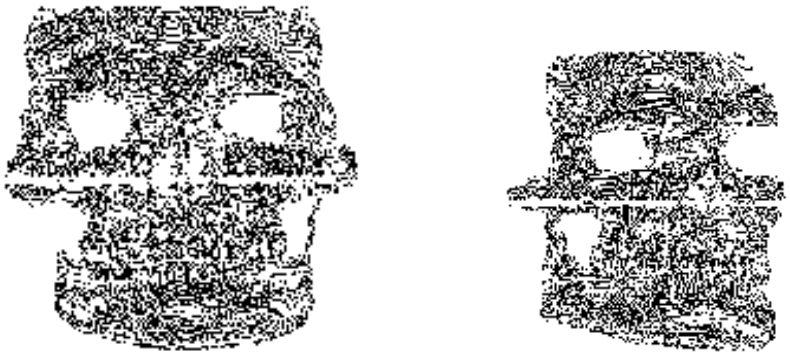


Figure 11.21. *Vue en perspective des lignes de crête pour les positions A et B (σ est égal à 1)*

Figure 11.22. *σ est égal à 3*

Figure 11.23. *σ est égal à 5*

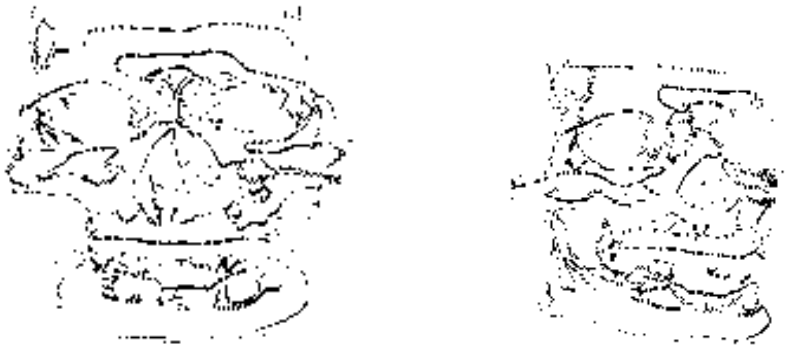


Figure 11.24. *σ est égal à 7*

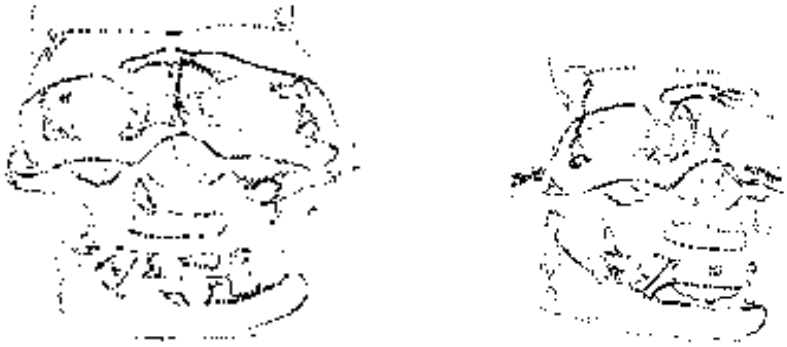


Figure 11.25. *σ est égal à 9*

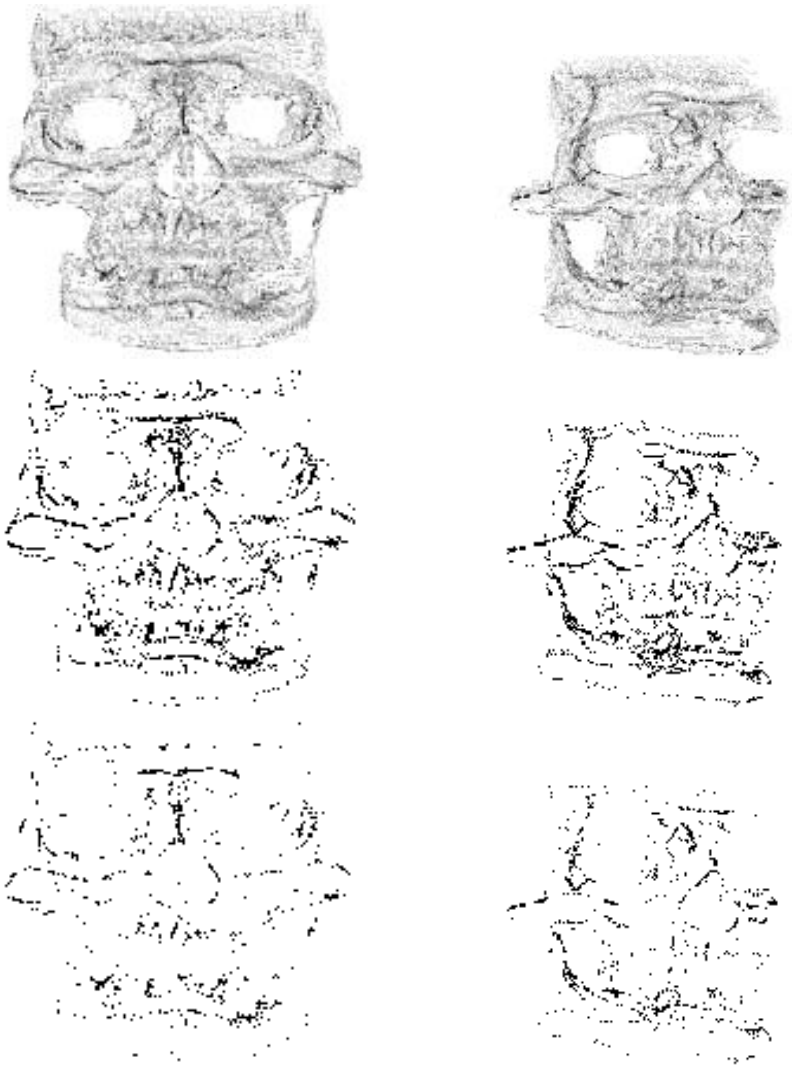


Figure 11.26. *En haut : vues en perspective correspondant aux positions A (gauche) et B (droite), le niveau de gris correspond au nombre d'échelles pour lesquelles le point est un point de crête ; au milieu : vues en perspective correspondant aux positions A (gauche) et B (droite) où seuls les points qui sont points de crête pour au moins* 4 *échelles sont marqués ; en bas : vues en perspective correspondant aux positions A (gauche) et B (droite) où seuls les points qui sont points de crête pour au moins* 5 *échelles sont marqués*

Chapitre 12

Des cartes de profondeur à la géométrie des surfaces

Un autre type de données 3D est constitué par les cartes de profondeur obtenues par télémètre laser [154]. Dans ce cas on obtient une image de profondeur $I(x, y)$ représentant la trace discrète d'une surface 3D. En effet ici l'information 3D est accessible directement sous la forme des points de coordonnées $(x, y, I(x, y))$ dans l'espace cartésien 3D. I définit alors un paramétrage naturel de la surface que l'on peut mettre à profit pour calculer ses caractéristiques [154].

Les méthodes développées dans ce cadre peuvent aussi être utilisées pour caractériser la surface définie implicitement par une image de niveaux de gris $((x, y, I(x, y)))$. Un exemple d'application à l'extraction de réseaux fins dans des images de niveaux de gris sera présenté [141].

12.1. Courbures des surfaces à partir d'images de profondeur

Soit une image de profondeur $I(x, y)$ représentant la trace discrète d'une surface S (ensemble des points de coordonnées $(x, y, I(x, y))$ dans l'espace cartésien 3D). On considère que I définit un paramétrage régulier de la surface. S est donc définie par l'équation :

$$\vec{S}(x, y) = (x, y, I(x, y))^t \qquad [12.1]$$

On définit le plan tangent $S_T(x, y)$ de la surface $\vec{S}(x, y)$ en chaque point P par :

$$\begin{array}{c} \vec{S_x} = (\frac{\partial \vec{S}}{\partial x}) = (1, 0, I_x)^t \\ \vec{S_y} = (\frac{\partial \vec{S}}{\partial y}) = (0, 1, I_y)^t \\ S_T(x, y) = \{\vec{S_x}, \vec{S_y}\} \end{array} \quad [12.2]$$

où I_x et I_y représentent respectivement les dérivées partielles de I selon x et y.

On peut utiliser la première et la seconde forme fondamentale pour calculer les courbures principales et les directions principales de courbure en chaque point P de la surface $\vec{S}(x, y)$ (voir chapitre : "Des images volumiques à la géométrie des surfaces").

Les cœfficients de la première forme fondamentale dans la base $\{\vec{S_x}, \vec{S_y}\}$ forment une matrice F_1 définie par :

$$F_1 = \begin{pmatrix} e & f \\ f & g \end{pmatrix}$$

où

$$e = \|\vec{S_x}\|^2 \quad f = \vec{S_x}.\vec{S_y} \quad g = \|\vec{S_y}\|^2 \quad [12.3]$$

En utilisant les dérivées partielles de l'image $I(x, y)$ pour calculer les dérivées de $\vec{S}(x, y)$, on obtient :

$$e = 1 + I_x^2 \quad f = I_x I_y \quad g = 1 + I_y^2 \quad [12.4]$$

De la même manière, on obtient pour la deuxième forme fondamentale dans la même base, une matrice notée F_2 :

$$F_2 = \begin{pmatrix} l & m \\ m & n \end{pmatrix}$$

Les cœfficients l, m, et n peuvent s'écrire :

$$l = \vec{N}.\vec{S_{xx}} \quad m = \vec{N}.\vec{S_{xy}} \quad n = \vec{N}.\vec{S_{yy}} \quad [12.5]$$

où $\vec{S_{xx}}, \vec{S_{xy}}, \vec{S_{yy}}$ sont les dérivées secondes de $\vec{S}(x, y)$ le long des axes correspondants et $\vec{N}$ la normale à la surface donnée par :

$$\vec{N} = \frac{\vec{S_x} \wedge \vec{S_y}}{\|\vec{S_x} \wedge \vec{S_y}\|} \quad [12.6]$$

En utilisant les dérivées partielles on obtient :

$$\begin{array}{l} l = \frac{I_{xx}}{\sqrt{(1+I_x^2+I_y^2)}} \\ m = \frac{I_{xy}}{\sqrt{(1+I_x^2+I_y^2)}} \\ n = \frac{I_{yy}}{\sqrt{(1+I_x^2+I_y^2)}} \end{array} \qquad [12.7]$$

Les courbures principales et les directions principales correspondent respectivement aux valeurs propres (k_1 et k_2) et aux vecteurs propres ($\vec{t_1}$ et $\vec{t_2}$) de l'endomorphisme de Weingarten dont la matrice est :

$$W = F_1^{-1} F_2 = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix}$$

Les cœfficients de W se calculent comme suit :

$$\begin{array}{l} w_{11} = \frac{1}{H}(gl - fm) \\ w_{12} = \frac{1}{H}(gm - fn) \\ w_{21} = \frac{1}{H}(em - fl) \\ w_{22} = \frac{1}{H}(en - fm) \qquad (\text{avec} \quad H = eg - f^2) \end{array} \qquad [12.8]$$

Par définition K=$k_1 k_2$ est la courbure gausienne et S=$\frac{1}{2}(k_1 + k_2)$ est la courbure moyenne. Ainsi les expressions de K et S sont :

$$\begin{array}{l} K = w_{11}w_{22} - w_{12}w_{21} = \frac{1}{H}(ln - m^2) \\ S = \frac{1}{2}(w_{11}w_{22}) = \frac{1}{2H}(en - 2fm + gl) \end{array} \qquad [12.9]$$

En utilisant les équations [12.4] et [12.7], K et S sont données par :

$$\begin{array}{l} K = w_{11}w_{22} - w_{12}w_{21} = \frac{1}{H}(ln - m^2) \\ S = \frac{1}{2}(w_{11}w_{22}) = \frac{1}{2H}(en - 2fm + gl) \end{array} \qquad [12.10]$$

Finalement, les courbures principales k_1 et k_2 sont les solutions d'une équation du second degré donnée par :

$$\begin{array}{l} k_1 = S + \sqrt{S^2 - K} \\ k_2 = S - \sqrt{S^2 - K} \end{array} \qquad [12.11]$$

dont les solutions sont :

$$
\begin{aligned}
k_{1,2} = \tfrac{1}{2d^{3/2}} \quad & [I_{xx} - 2I_x I_{xy} I_y + I_{xx} I_y^2 + I_{yy} + I_x^2 I_{yy} \\
& \pm [\ I_{xx}^2 + 4I_{xy}^2 - 2I_{xx} I_{yy} + 4I_x^2 I_{xy}^2 + I_x^4 I_{yy}^2 - \\
& \quad 2I_{xx} I_y^2 I_{yy} - 4I_x I_{xx} I_{xy} I_y^3 - 2I_x^2 I_{xx} I_{yy} - \\
& \quad 4I_x I_{xy} I_y I_{yy} + 2I_{xx}^2 I_y^2 + 4I_{xy}^2 I_y^2 + I_{xx}^2 I_y^4 + \\
& \quad I_{yy}^2 + 2I_x^2 I_{yy}^2 - 4I_x^3 I_{xy} I_y I_{yy} - 4I_x I_{xx} I_{xy} I_y + \\
& \quad 2I_x^2 I_{xx} I_y^2 I_{yy} + 4I_x^2 I_{xy}^2 I_y^2 \\
& \ ]^{1/2} \\
&]. \quad (\text{avec} \ \ d = 1 + I_x^2 + I_y^2)
\end{aligned}
\qquad [12.12]
$$

Une fois que k_1 et k_2 sont calculées, les directions principales $\vec{t_1}$ and $\vec{t_2}$ sont les deux vecteurs associés. Si on note $\vec{t_i} = (u_i \ v_i)^t$ dans la base $\{\vec{S_x}, \vec{S_y}\}$ on obtient le système d'équations suivant :

$$
\begin{array}{rcrcl}
(w_{11} - k_i) u_i & + & w_{12} v_i & = 0 & \\
w_{21} u_i & + & (w_{22} - k_i) v_i & = 0 & i \in 1, 2.
\end{array}
\qquad [12.13]
$$

Soit :

$$
\vec{t_1} = \begin{pmatrix} w_{12} \\ k_1 - w_{11} \end{pmatrix} \qquad \vec{t_2} = \begin{pmatrix} w_{12} \\ k_2 - w_{11} \end{pmatrix} \qquad [12.14]
$$

On définit aussi la courbure maximum $k_{max} = k_1$ comme la courbure principale de plus grande valeur absolue ($|k_1| \geq |k_2|$) et aussi la direction maximum de courbure $\vec{t_{max}} = \vec{t_1}$..

On notera que ces formules n'ont rien à voir avec celles définies précédemment dans le cas des images volumiques car ici l'image I définit directement un paramétrage de l'image.

12.2. Calcul des dérivées partielles de l'image

Les dérivées partielles de l'image peuvent être calculées avec des versions $2D$ des filtres séparables récursifs décrits pour le cas des images volumiques $3D$ (voir chapitre :"Détection de contours").

On peut aussi lisser les données avec un filtre de Gauss et ensuite dériver localement avec de petits masques [154, 168].

On notera que contrairement au cas des images volumiques 3D où le lissage de l'image est invariant par similitude donc intrinsèque à la surface, ici le lissage de la surface dépend de son orientation par rapport au capteur.

12.3. Le travail de de Ponce-Brady

L'algorithme décrit dans [154] se décompose en trois étapes :

• filtrage de l'image par une suite de Gaussiennes d'écarts types σ_i croissants ; on obtient ainsi un ensemble de surfaces S_i et on calcule pour chacune d'elles les courbures et les directions principales ;

• pour chaque surface S_i, on marque les passages par zéro de la courbure de Gauss et les extrémums de la courbure maximum dans la direction associée (points de crête) ;

• les descriptions des surfaces S_i sont mises en correspondance afin de détecter, localiser et décrire les points qui sont des pas, des toits ou des jonctions (voir figures 12.1., 12.2., 12.3.) ;

On trouvera dans [154] une étude du comportement des toits, des jonctions et des pas selon la résolution. Cette étude permet de réaliser de manière efficace l'étape de mise en correspondance.

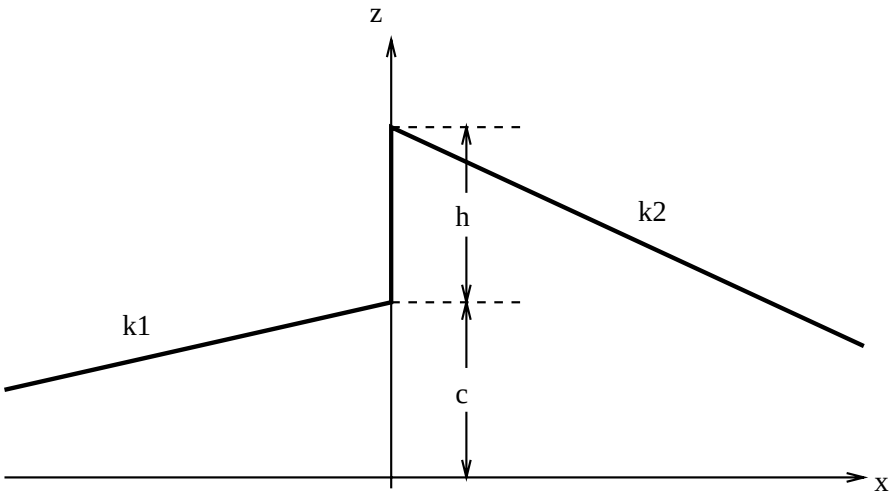


Figure 12.1. *Le modèle du pas consiste en deux demi-plans séparés par une distance h à l'origine.*

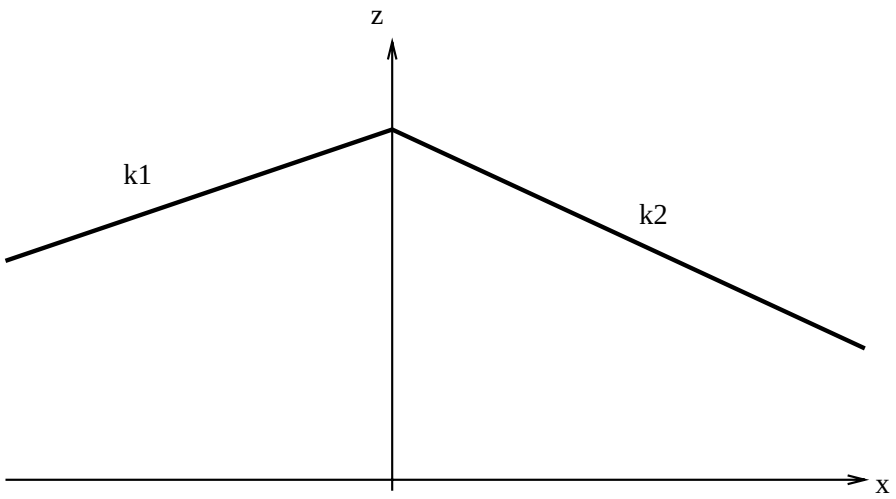


Figure 12.2. *Le modèle du toit consiste en deux demi-plans inclinés joints à l'origine mais ayant des pentes différentes.*

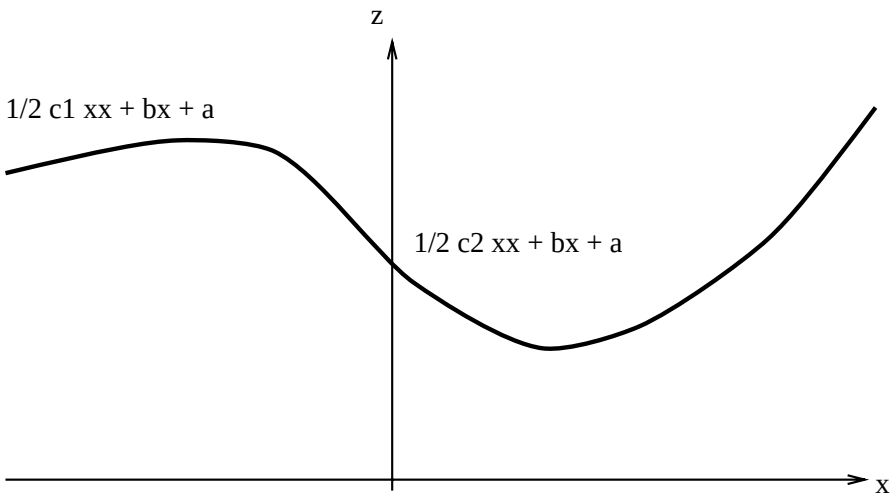


Figure 12.3. *Le modèle de la jonction consiste en deux demi-paraboles ayant des courbures différentes.*

12.4. Extraction de lignes de crête et applications

Si on reprend les notations de la section "Caractéristiques différentielles...", on peut exprimer le critère d'extrémalité défini précedemment pour des iso-surfaces d'images 3D (voir chapitre : "Des images volumiques à la géométrie des surfaces") dans le cas des cartes de profondeur. Soit k_{max} la direction maximum de courbure et $\vec{t_{max}}$ la direction principale associée. Le critère d'extrémalité que nous noterons DMC est par définition la dérivée de la courbure maximum dans la direction principale correspondante soit :

$$DMC = \nabla \vec{k_{max}} \; . \; \vec{t_{max}} \qquad [12.15]$$

où

$$\nabla \vec{k_{max}} = \begin{pmatrix} \frac{\partial k_{max}}{\partial x} \\ \frac{\partial k_{max}}{\partial y} \end{pmatrix} \qquad \vec{t_{max}} = \begin{pmatrix} w_{12} \\ k_{max} - w_{11} \end{pmatrix} \qquad [12.16]$$

l'expression de DMC peut aussi s'écrire :

$$DMC = (\frac{\partial k_{max}}{\partial x})(w_{12}) + (\frac{\partial k_{max}}{\partial y})(k_{max} - w_{11}) \qquad [12.17]$$

L'expression explicite de DMC est présentée en détail dans **??**.

Une fois DMC calculé, on extrait les points tels que $DMC = 0$ (les passages par zéro de DMC).

12.4.1. Extraction des lignes de crête

L'extraction des passages par zéro de DMC pose le même problème que dans le cas $3D$ (voir chapitre : "Des images volumiques à la géométrie des surfaces") du au fait que DMC n'est pas "complètement" un invariant euclidien. On sélectionne ensuite les passages par zéro de DMC telles que la valeur absolue de la courbure maximum k_{max} est plus grande qu'un seuil. Cette étape de seuillage permet de retenir seulement les points où la courbure maximum est localement maximum (les passages par zéro de DMC correspondent à la fois aux maximums et aux minimums de la courbure maximum).

Les principales étapes de l'algorithme d'extraction des lignes de crête sont donc :

1. Calcul des dérivées partielles d'ordre 1,2,3 de l'image : I_x, I_{xx}, I_{xxx}, I_y, I_{yy}, I_{yyy}, I_{xy}, I_{xyy}, I_{xxy}.

2. Calcul des deux courbures principales k_1 et k_2 ainsi que des deux directions principales de courbures $\vec{t_1}$, $\vec{t_2}$.

3. En utilisant les dérivées partielles de l'image, calcul de la dérivée de la courbure maximum selon la direction principale associée (DMC) :

(a) Détermination des points méplats et des ombillics pour les traiter à part.

(b) Calcul de DMC sur les autres points :

$$\begin{cases} \text{si } |k_1| > |k_2| \text{ faire} & k_{max} = k_1 \quad \vec{t_{max}} = \vec{t_1}, \\ \text{sinon} & k_{max} = k_2 \quad \vec{t_{max}} = \vec{t_2}, \\ \text{fin si} & \\ \text{faire} \quad DMC = \nabla \vec{k_{max}} \ . \ \vec{t_{max}} & \end{cases}$$

4. Extraction des passages par zéro de DMC : en utilisant l'image de signe (S_{dmc}) de DMC, on extrait les transitions $0 \nearrow 1$ ou $1 \searrow 0$.

5. Sélection des passages par zéro de DMC telles que valeur absolue de la courbure maximum k_{max} est plus grande qu'un certain seuil.

Les étapes 3,4,5 peuvent être remplacées par la détermination des points où la valeur absolue de la courbure maximum est maximum dans la direction principale correspondante. On peut par exemple utiliser un algorithme similaire à celui de l'extraction des maxima locaux de la norme du gradient dans la direction du gradient. On évite ainsi le calcul des dérivées partielles d'ordre 3 mais l'algorithme est "mathématiquement moins propre".

12.4.2. Résultats expérimentaux

Dans cette section, nous présentons des résultats expérimentaux obtenus sur des données synthétiques et réelles. Les images réelles sont des cartes de profondeur et des images de niveaux de gris. Dans les images de niveaux de gris les lignes de crête de la surface image correspondent aux réseaux fins : par exemple les routes dans les images satellites et les vaisseaux sanguins dans les images médicales. Les dérivées partielles sont calculées avec les filtres récursifs approximant le filtre gaussien et ses dérivées (voir chapitre : "Détection de contours").

Les figures 12.4. à 12.8. illustrent la signification des lignes de crête grâce à une image représentant un cercle et un losange avec un bruit blanc gaussien

additif. Les figures 12.5., 12.7., et 12.8. montrent en particulier la différence entre la détection de contours et l'extraction de lignes de crête.

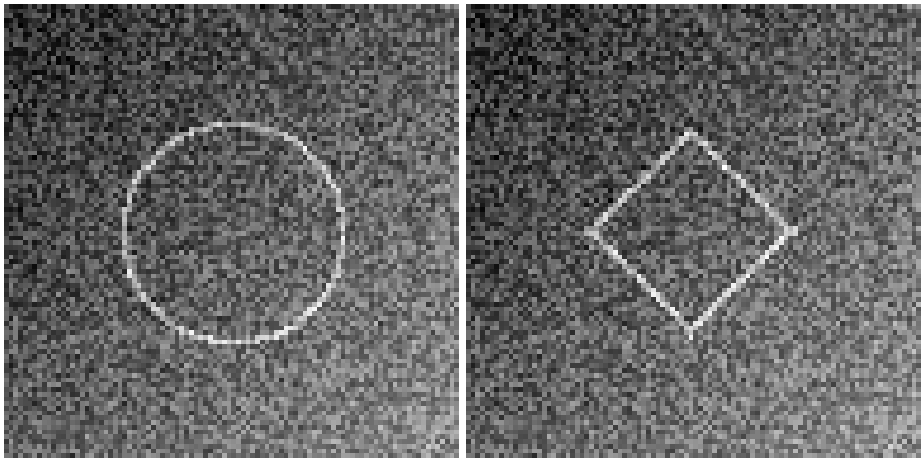


Figure 12.4. *Images originales : Cercle et losange avec un bruit blanc gaussien additif.*

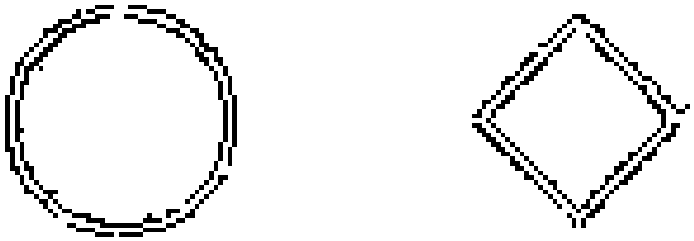


Figure 12.5. *Détection de contours par les filtres gaussiens ($\sigma = 1$).*

Les figures 12.9. à 12.12. montrent les résultats obtenus sur des cartes de profondeur décrivant des surfaces [51]. Les figures 12.9. à 12.10. illustrent l'algorithme sur deux surfaces quadriques. Les figures 12.11. à 12.18. montrent les

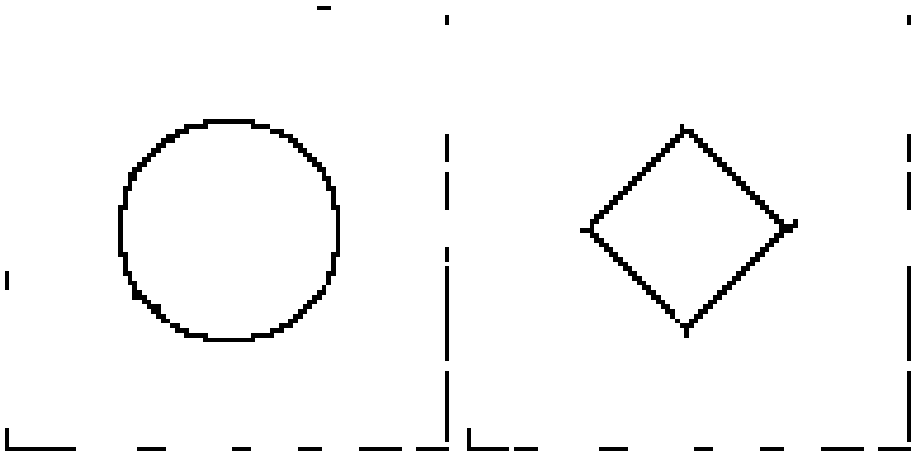


Figure 12.6. *Extraction des lignes de crête, en utlisant les passages par zéro de DMC puis un seuillage de* k_{max}

lignes de crête obtenues sur des cartes de profondeur de visages fournies par un algorithme de vision stéréoscopique [63].

Les figures 12.13. à 12.18. présentent les résultats obtenus sur des images satellites.

Les figures 12.19. et 12.20. présentent des résultats obtenus sur des images médicales obtenues par DSA (Digital Substraction Angiography).

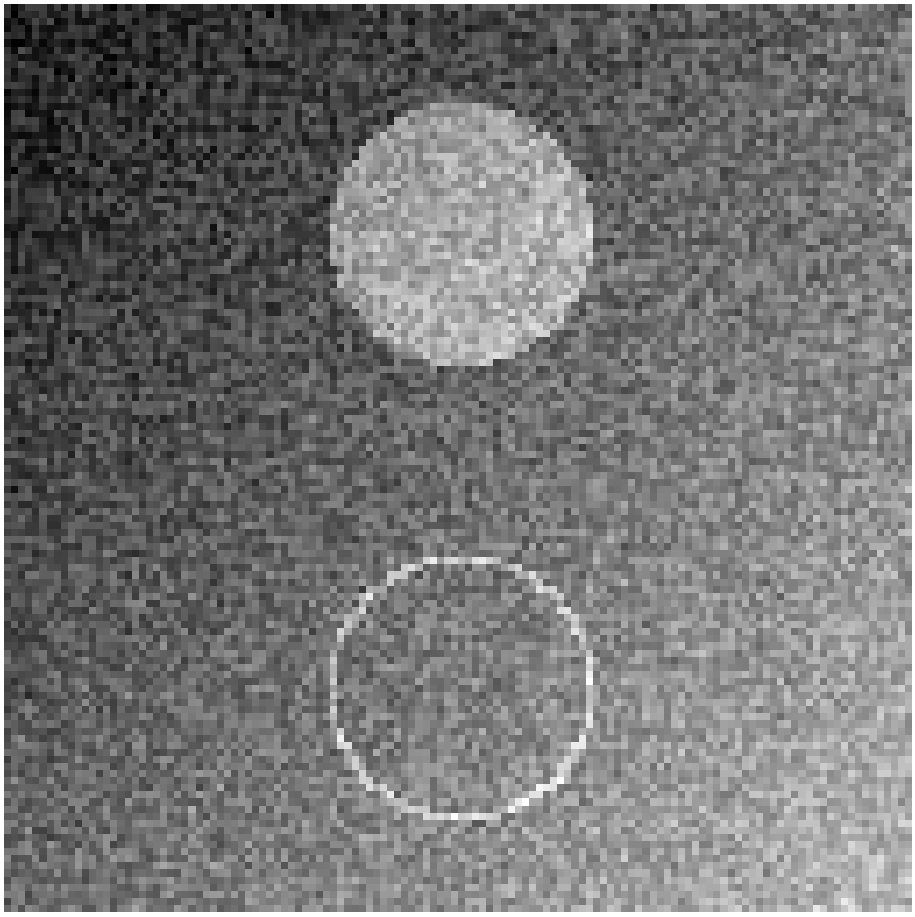


Figure 12.7. *Images originales avec un bruit blanc Gaussien additif*

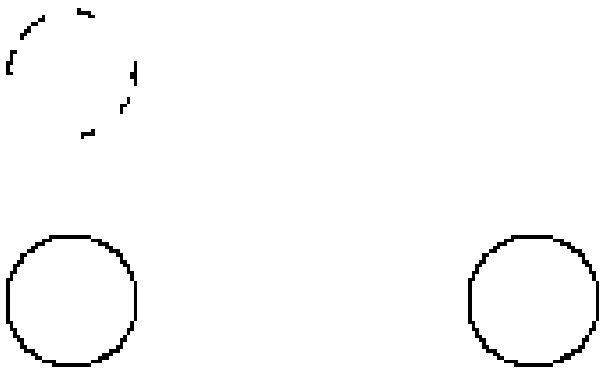


Figure 12.8. *Extraction des lignes de crête . Gauche : seuillage de k_max à 0.5. Droite : seuillage de k_max à 0.8.*

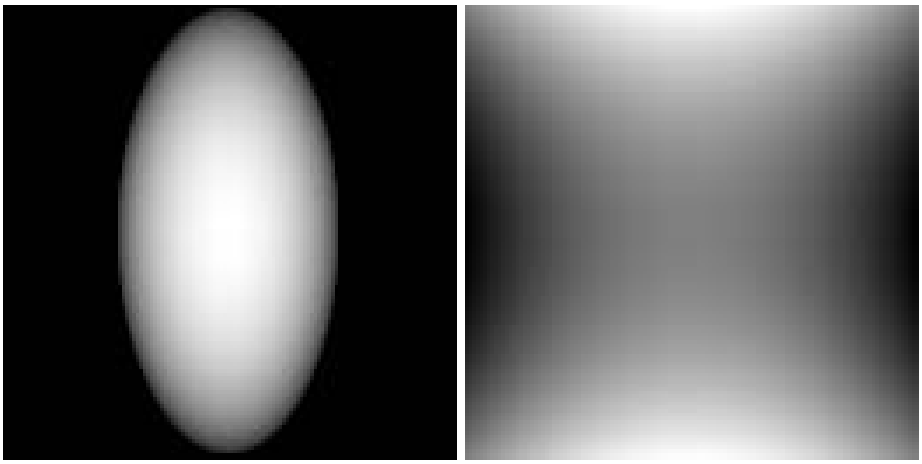


Figure 12.9. *Modèles de quadriques. Gauche : ellipsoide. Droite : hyperboloide parabolique.*

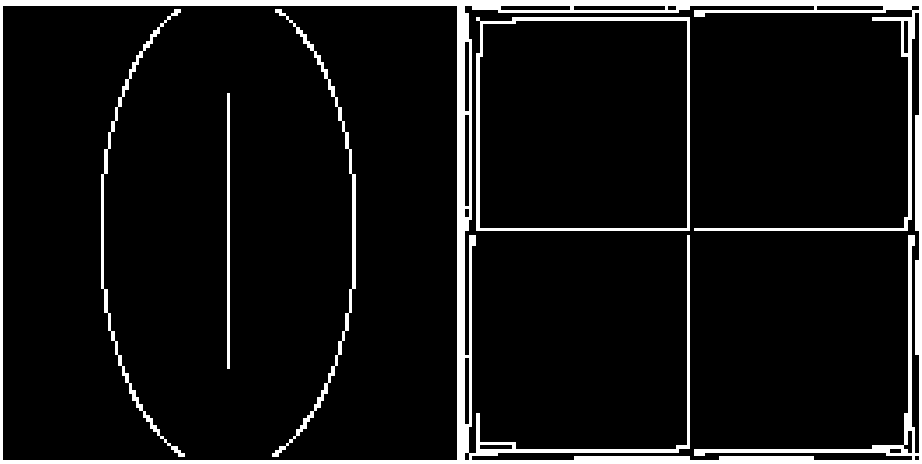


Figure 12.10. *Maxima de la courbure maximun le long de la direction principale associée. Gauche : ellipsoide. Droite : hyperboloide parabolique.*

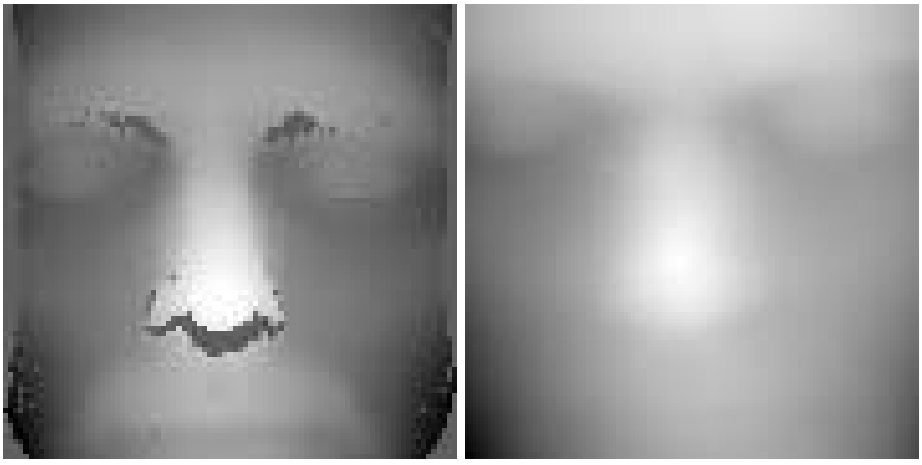


Figure 12.11. *Cartes de profondeur. Droite : Benoît; Gauche : Bernard*

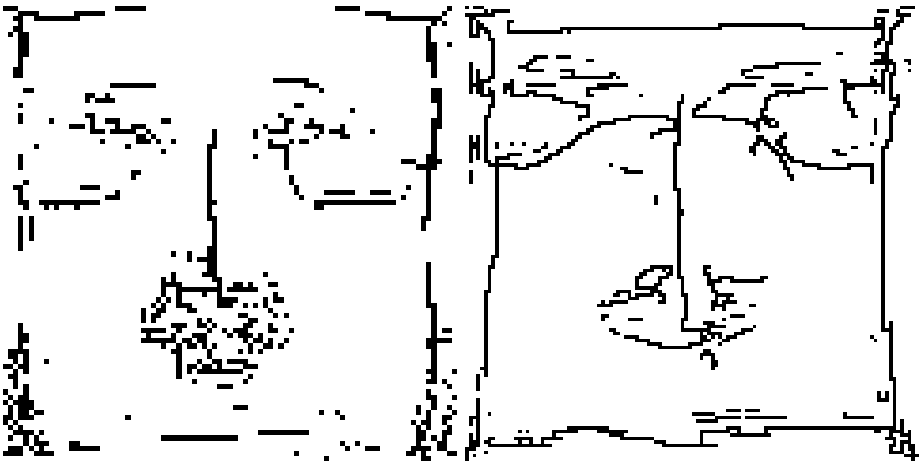


Figure 12.12. *Lignes de crête. Droite : Benoît; Gauche : Bernard*

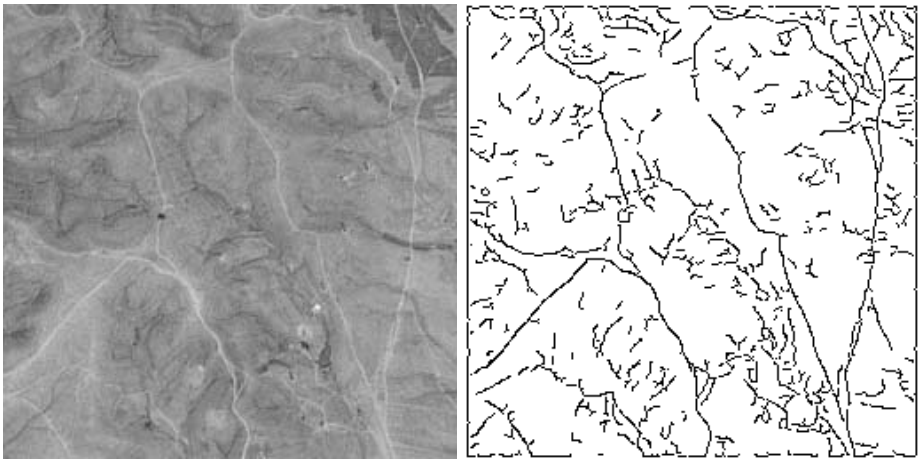


Figure 12.13. *Gauche : Image originale. Droite : Lignes de crêtes .*

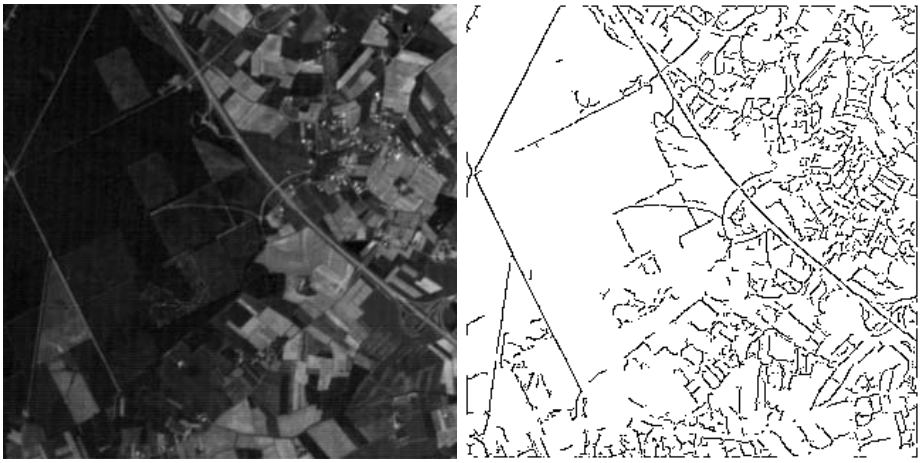


Figure 12.14. *Gauche : Image originale. Droite : Lignes de crêtes .*

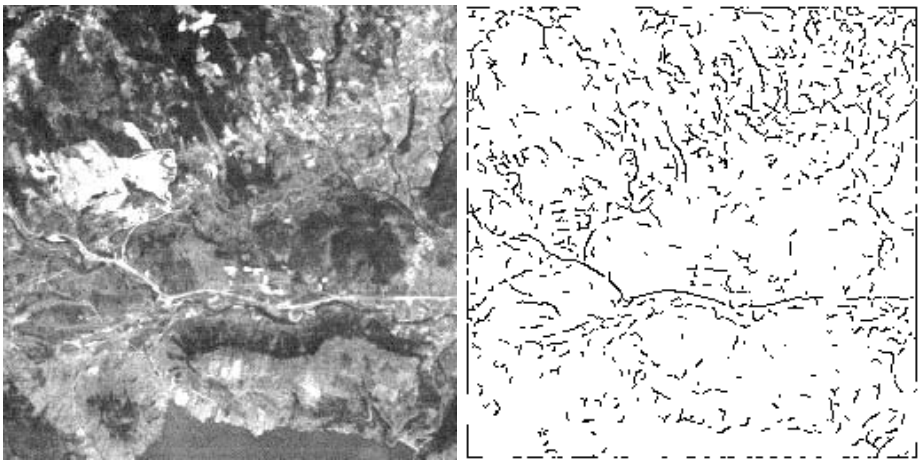


Figure 12.15. *Gauche : Image originale. Droite : Lignes de crêtes .*

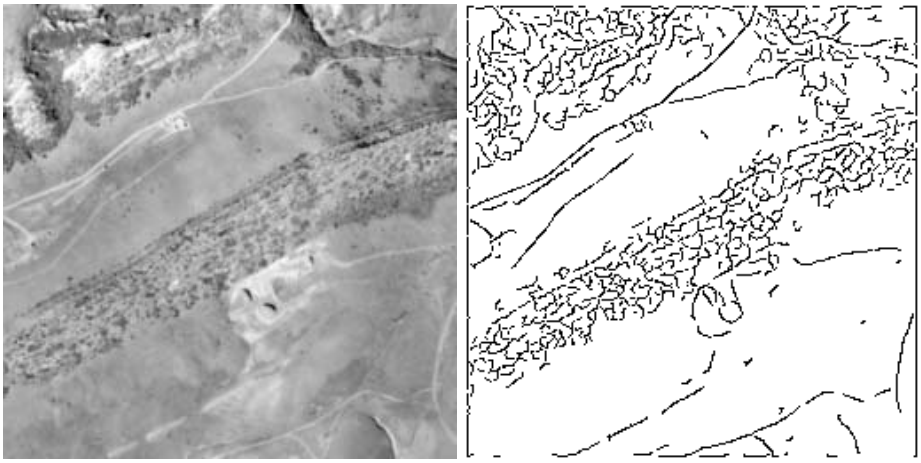


Figure 12.16. *Gauche : Image originale. Droite : Lignes de crêtes .*

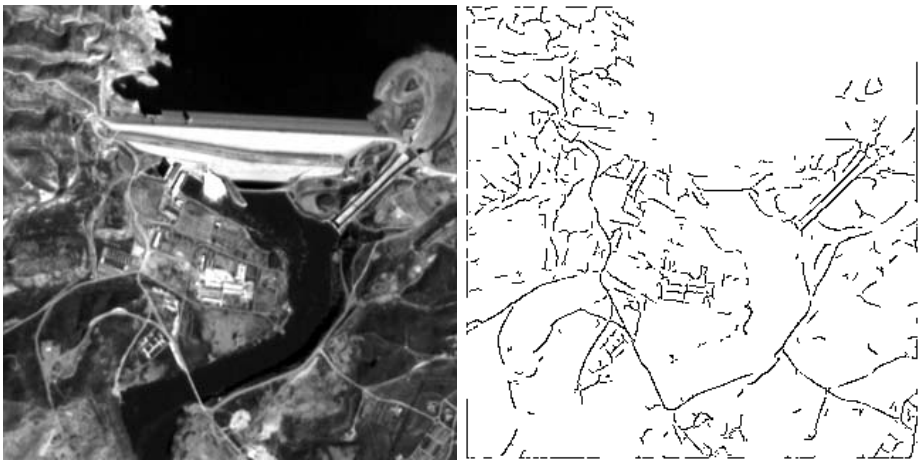


Figure 12.17. *Gauche : Image originale. Droite : Lignes de crêtes .*

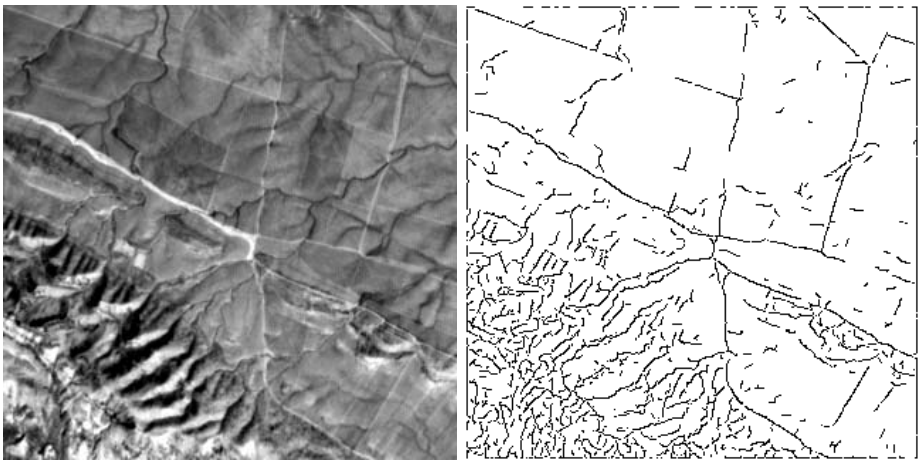


Figure 12.18. *Gauche : Image originale. Droite : Lignes de crêtes .*

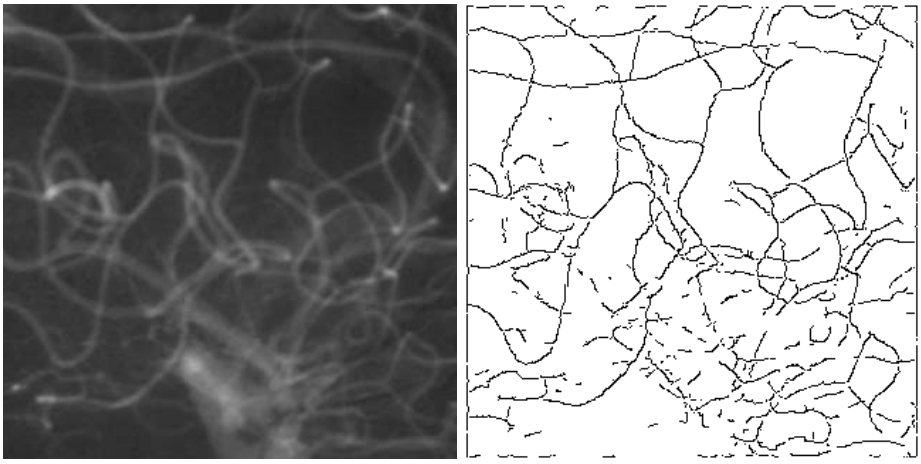


Figure 12.19. *Gauche : Image originale de vaisseaux sanguins obtenue par DSA. Droite : Extraction des lignes de crête ($\sigma = 1$)*

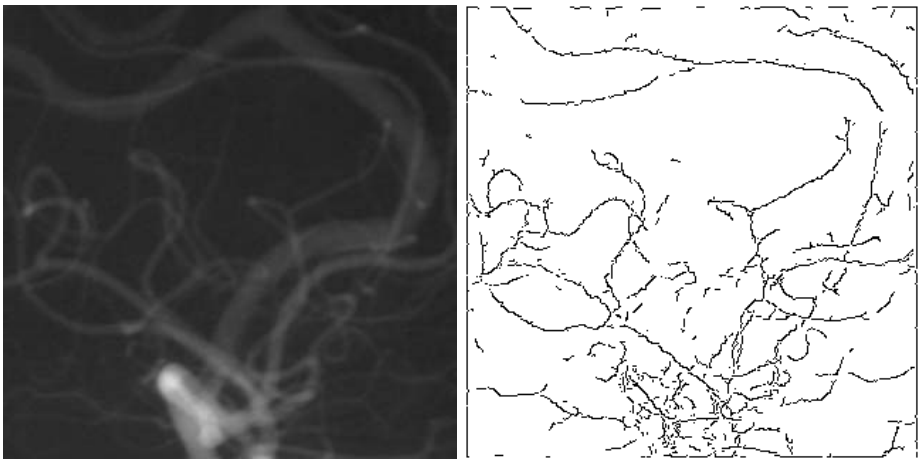


Figure 12.20. *Gauche : Image originale de vaisseaux sanguins obtenue par DSA. Droite : Extraction des lignes de crête* ($\sigma = 1$)

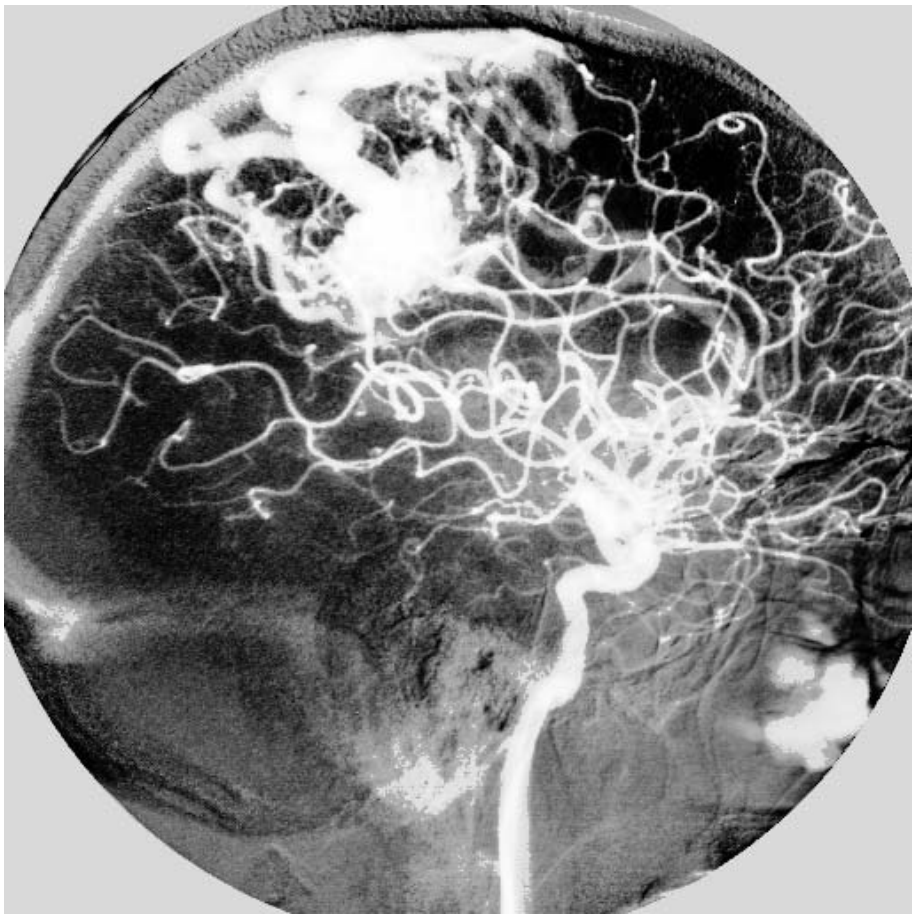


Figure 12.21. *Arbre vasculaire cérébral obtenu par DSA*

12.5. Conclusion

Nous avons vu dans ce chapitre que les algorithmes décrits dans le chapitre précédent s'adaptent aux cartes de profondeur. Cette adaptation demande néanmoins des développements mathématiques sensiblement différents.

L'extraction de singularités différentielles à partir non plus de cartes de profondeur mais de la surface image définie par une image de niveaux de gris quelconque, ouvre des horizons intéressants. En effet, ce type de méthode

Figure 12.22. *Lignes de crêtes obtenues sur l'arbre vasculaire cérébral*

permet de détecter des indices visuels plus sophistiqués que des contours en marche d'escalier et ouvre ainsi des espaces nouveaux. On a vu un exemple pour l'extraction de réseaux fins.

Bibliographie

[1] P. Alevizos, J.D. Boissonnat, and M. Yvinec. An optimal o(nlog(n)) algorithm for contour reconstruction from rays. *ACM Symposium on Computational Geometry, Waterloo*, June 1987.

[2] P. Alevizos, J.D. Boissonnat, and M. Yvinec. On the order induced by a set of rays : Application to the probing of non convex polygons. *INRIA Report research*, November 1988.

[3] A.P. Ambler and H.G. Barrow. A versatile computer assembly system. *Artificial Intelligence*, 6:129–156, 1975.

[4] R. D. Arnold and T. O. Binford. Geometric constraints in stereo vision. In *SPIE Vol. 238 - Image Processing for Missile Guidance*, pages 281–292, 1980.

[5] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-9(5):698–700, September 1987.

[6] T. Asano and N. Yokoya. Image segmentation schema for low level computer vision. In *International Conference on Pattern Recognition*, number 14, pages 267–273, 1981.

[7] P. Aschwanden and W. Guggenbuhl. Experimental results from a comparative study on correlation type registration algorithms. In Forstner and Ruwiedel, editors, *Robust Computer Vision*, pages 268–282. Wichmann, 1992.

[8] N. Ayache. *Vision Stéréoscopique et Perception Multisensorielle.* InterEditions, 1989.

[9] N. Ayache. *Artificial Vision for Mobile Robots – Stereo-Vision and Multisensory Perception.* MIT Press, Boston, 1991.

[10] N. Ayache and O. D. Faugeras. HYPER: A new approach for the recognition and positioning of two-dimensional objects. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-8(1):44–54, January 1986.

[11] N. Ayache and B. Faverjon. Efficient registration of stereo images by matching graph descriptions of edge segments. *International Journal of Computer Vision*, 1(2):107–131, 1987.

[12] N. Ayache and F. Lustman. Trinocular stereo vision for robotics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(1):73–85, January 1991.

[13] H. H. Baker and T. O. Binford. Depth from edge and intensity based stereo. In *Proc. 7-th International Joint Conference on Artificial Intelligence*, pages 631–636, Vancouver, Canada, August 1981.

[14] D. H. Ballard. Generalizing the Hough Transform to Detect Arbitrary Shapes. *Pattern Recogniton*, 13(2):111–122, 1981.

[15] D. H. Ballard and C. M. Brown. *Computer Vision*. Prentice Hall Inc., Englewood Cliffs, New Jersey, 1982.

[16] H. A. Beyer. *Geometric and Radiometric Analysis of a CCD-Camera Based Photogrammetric Closed-Range System*. PhD thesis, Institut fur Geodasie und Photogrammetrie, Zurich, 1992.

[17] J.D. Boissonnat, Olivier D. Faugeras, and E. Lebras. Representing stereo data with the delaunay triangulation. In *IEEE Conference on Robotics and Automation*, Philadelphie, April 1988.

[18] J.D. Boissonnat and Olivier Monga. Scene reconstruction from rays : Application to stereo data. In *IEEE Conference on Robotics and Automation*, Philadelphie, April 1988.

[19] Jean-Daniel Boissonnat. Geometric structures for three-dimensional shape representation. *ACM Transactions on Graphics*, 3:266–286, October 1984.

[20] Jean Daniel Boissonnat. Shape reconstruction from planar cross section. *INRIA Report research*, June 1986.

[21] R. C. Bolles and R. Horaud. 3DPO: A three-dimensional part orientation system. *International Journal of Robotics Research*, 5(3):3–26, Fall 1986.

[22] R. C. Bolles, R. Horaud, and M. J. Hannah. 3DPO: A Three-Dimensional Part Orientation System. In Fischler and Firschein, editors, *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms*, pages 355–359. Morgan Kaufmann, Inc., 1987.

[23] R. C. Bolles, J. H. Kremers, and R. A. Cain. A Simple Sensor to Gather Three-Dimensional Data. Technical Report 249, SRI International, July 1981.

[24] R.C. Bolles and R. Horaud. Configuration understanding in range data. In Hanafusa and Inoue, editors, *The Second International Symposium on Robotics Research*, pages 43–50. MIT Press, 1985.

[25] B. Boufama and R. Mohr. Epipole and fundamental matrix estimation using virtual parallax. In *Proceedings Fifth International Conference on Computer Vision*, Cambridge, Mass., June 1995. IEEE Computer Society Press, Los Alamitos, Ca.

[26] P. Bouthémy and E. François. Motion segmentation and qualitative dynamic scene analysis. *International Journal of Computer Vision*, 10(2):157–182, April 1993.

[27] Michael Brady, Jean Ponce, Alan Yuille, and Haruo Asada. Describing surfaces. In Hideo Hanafusa and Hirochika Inoue, editors, *Proceedings of the Second International Symposium on Robotics Research*, pages 5–16, Cambridge, Mass., 1985. MIT Press.

[28] P. Brand, R. Mohr, and Ph. Bobet. Distorsions optiques : correction dans un modèle projectif. Technical Report 1933, INRIA Rhône-Alpes, Grenoble, France, Juin 1993.

[29] C. Brice and C. Fenema. Scene analysis using regions. In *Artificial Intelligence*, number 1, pages 205–226, 1970.

[30] J. Bryant. On the clustering of multidimensional pictorial data. In *Pattern Recognition*, number 11, pages 115–125, 1979.

[31] P. Burt and B. Julesz. A disparity gradient limit for binocular fusion. *Science*, 208:615–617, May 1980.

[32] J. Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, November 1986.

[33] H. Chen. Pose determination from line-to-plane correspondences: existence solutions and closed-form solutions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):530–541, June 1991.

[34] C.K. Chow and T. Kaneko. Boundary detection of radiographic images by a thresholding method. In S. Watanebe, editor, *Frontiers of pattern recognition*, pages 61–82. Academic Press, 1972. New York.

[35] Ph. Cinquin. Une nouvelle technique de modélisation d'images tridimensionnelles. In *COGNITIVA*, Paris, 1987.

[36] Jean-Pierre Cocquerez and Olivier Monga. Matching regions in stereovision. In *Proceedings of the Fourth Scandinavian Conference on Image Analysis*, Stockholm, 1987.

[37] Laurent Cohen, Laurent Vinet, Peter T. Sander, and Andre Gagalowicz. Hierarchical region based stereo matching. In *IEEE Conference on Vision and Patern Recognition*, San Diego, June 1989.

[38] L.D. Cohen and I. Cohen. A finite element method applied to new active contour models and 3d reconstruction from cross sections. In *Proceedings of the third International Conference on Computer Vision*, pages 587–591, Osaka, December 1990.

[39] J. L. Crowley and A. C. Parker. A representation for shape based on peaks and ridges in the difference of low-pass transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(2):156–170, March 1984.

[40] L. S. Davis and A. Rosenfeld. Cooperating processes for low-level vision: A survey. *Artificial Intelligence*, 17(1-3):245–263, August 1981.

[41] D. F. DeMenthon and L. S. Davis. Model-based object pose in 25 lines of code. In G. Sandini, editor, *Computer Vision – ECCV 92, Proceedings Second European Conference on Computer Vision, Santa Margherita Ligure, May 1992*, pages 335–343. Springer Verlag, May 1992.

[42] R. Deriche. Separable recursive filtering for efficient multi-scale edge detection. In *Proc. International Workshop on Machine Vision and Machine Intelligence*, Tokyo, December 1987. IEEE.

[43] R. Deriche. Using Canny's criteria to derive a recursively implemented optimal edge detector. *International Journal of Computer Vision*, 1(2):167–187, 1987.

[44] R. Deriche. Fast algorithms for low level vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1989.

[45] R. Deriche. Recursively Implementing the Gaussian and Its Derivatives. In *Proc. Second International Conference On Image Processing*, pages 263–267, Singapore, September 7-11 1992. A longer version is INRIA Research Report RR-1893.

[46] R. Deriche and J.-P. Cocquerez. Extraction de composants connexes basée sur une détection optimale des contours. In *CESTA*, Paris, 1987.

[47] Rachid Deriche and Jean-Pierre Cocquerez. Extraction de composantes connexes basée sur une détection optimale des contours. In *Actes, Machines et Reseaux Intelligents, Cognitiva 87, Image Electronique*, volume Tome 2, Paris, Mai 1987.

[48] M. Dhome, M. Richetin, J.T. Lapreste, and G. Rives. Determination of the Attitude of 3D Objects from a Single Perspective View. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(12):1265–1278, December 1989.

[49] Manfredo P. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, Englewood Cliffs, 1976.

[50] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973.

[51] V. Prinet et O. Monga. Adaptative filtering and geometrical invariants in face's depth map. In *Europe-China Workshop on Geometrical modelling and Invariants for Computer Vision*, Xian (Chine), April 1995.

[52] Ivan Bricault et Olivier Monga. From volume medical images to quadratic surface patches. *INRIA Report research No 2380*, 1994.

[53] O. D. Faugeras. Quelques pas vers la vision artificielle en trois dimensions. *Technique et Science Informatiques*, 7(6):547–590, 1988.

[54] O. D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig. In G. Sandini, editor, *Computer Vision – ECCV 92, Proceedings Second European Conference on Computer Vision, Santa Margherita Ligure, May 1992*, pages 563–578. Springer Verlag, May 1992.

[55] O. D. Faugeras. *Three Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, Boston, 1993.

[56] O. D. Faugeras, Q. T. Luong, and S. J. Maybank. Camera self-calibration: Theory and experiments. In G. Sandini, editor, *Computer Vision – ECCV 92, Proceedings Second European Conference on Computer Vision, Santa Margherita Ligure, May 1992*, pages 321–334. Springer Verlag, May 1992.

[57] O. D. Faugeras and G. Toscani. The calibration problem for stereo. In *Proc. Computer Vision and Pattern Recognition*, pages 15–20, Miami Beach, Florida, USA, June 1986.

[58] O. D. Faugeras and G. Toscani. Camera Calibration for 3-D Computer Vision. In *Proc. International Workshop on Machine Vision and Machine Intelligence*, Tokyo, Japan, February 1987.

[59] O.D. Faugeras and M. Hebert. The representation, recognition, and locating of 3-d objects. *International Journal of Robotics Research*, 5(3):27–52, Fall 1986.

[60] O.D. Faugeras, M. Hebert, and E. Pauchon. Segmentation of range data into planar and quadric patches. In *Proc. PRIP 83*, pages 8–13, 1983.

[61] Olivier D. Faugeras. A few steps toward artificial 3d vision. *INRIA Report research*, 1988.

[62] M.A. Fischler and R.C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

[63] P. Fua and Y.G. Leclerc. Representation without correspondences. In *IEEE Conference on Vision and Pattern Recognition*, Seattle, June 1994.

[64] Henry Fuchs, Zvi M. Kedem, and Samuel P. Uselton. Optimal surface reconstruction from planar contours. *Communications of the ACM*, 20(10):693–702, October 1977.

[65] J.P. Gambotto and O. Monga. A parallel and hierarchical algorithm for region growing. In *IEEE Conference on Vision and Pattern Recognition*, San Francisco, June 1985.

[66] S. Ganapathy. Decomposition of transformation matrices for robot vision. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 130–139, Atlanta, Georgia, USA, March 1984.

[67] M. R. Garey and D. S Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completness*. W. H. Freeman, San Francisco, 1979.

[68] P. Garnesson and G. Giraudon. An efficient edge chaining algorithm. In *5th Scandinavian Conference on Image Analysis*, June 1987. Stockholm.

[69] D. M. Gay. Computing optimal constrained steps. *SIAM Journal on Scientific and Statistical Computation*, 1981.

[70] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, London, 1989.

[71] E. Gmur and H. Bunke. 3-D Object Recognition Based on Subgraph Matching in Polynomial Time. In Sanfeliu Mohr, Pavlidis, editor, *Structural Pattern Analysis*, pages 131–147. World Scientific Publ. Co., 1990.

[72] C. Goad. Fast 3D model-based vision. In Alex B. Pentland, editor, *From Pixels to Predicates*, chapter 16, pages 371–391. Ablex Publishing Corporation, Norwood, New Jersey, 1986.

[73] R. C. Gonzales and P. Wintz. *Digital Image Processing*. Addison-Wesley, Reading, Mass., 1977.

[74] W. E. L. Grimson. *From Images to Surfaces*. MIT Press, Cambridge, Massachusetts, 1981.

[75] W.E.L. Grimson and T. Lozano-Perez. Model-based Recognition and Localization from Sparse Range or Tactile Data. *International Journal of Robotics Research*, 3(3):3–35, Fall 1984.

[76] W.E.L. Grimson and T. Lozano-Perez. Localizing overlapping parts by searching the interpretation tree. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(4):469–482, July 1987.

[77] A. Gueziec and N. Ayache. Lissage et reconnaissance de courbes gauches bruitées. In *Congres AFCET*, Lyon, 1991.

[78] M. J. Hannah. Description of SRI's baseline stereo system. Technical Report 342, SRI International, October 1984.

[79] M. J. Hannah. A system for digital stereo image matching. *Photogrammetric Engineering and Remote Sensing*, 55(12):1765–1770, December 1989.

[80] R.M. Haralick and I. Dinstein. A spatial clustering procedure for multi-image data. IEEE Trans. on Circuits and System.

[81] R.M. Haralick and S.G. Shapiro. Survey: image segmentation techniques. In *Computer Vision Graphics and Image Processing*, number 29, pages 100–132, 1985.

[82] R.M. Haralick and L.T. Watson. A facet model for image data. In *Computer Graphics and Image Processing*, number 15, pages 113–129, 1981.

[83] R. Hartley. In defence of the 8-point algorithm. In *Proceedings Fifth International Conference on Computer Vision*, Cambridge, Mass., June 1995. IEEE Computer Society Press, Los Alamitos, Ca.

[84] M. D. Hebden. An algorithm for minimization using exact second derivatives. Technical Report TP 515, Atomic Energy Research Establishment, Harwell, England, 1973.

[85] L. Hérault. *Réseaux de Neurones Récursifs pour l'Optimisation Combinatoire*. PhD thesis, Institut National Polytechnique de Grenoble, February 1991.

[86] R. J. Holt and A. N. Netravali. Camera calibration problem: some new results. *CGVIP-Image Understanding*, 54(3):368–383, November 1991.

[87] R. Horaud. New Methods for Matching 3-D Objects with Single Perspective Views. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-9(3):401–412, May 1987.

[88] R. Horaud and R.C. Bolles. 3DPO: a system for matching 3D objects in range data. In Alex B. Pentland, editor, *From Pixels to Predicates*, chapter 15, pages 359–370. Ablex Publishing Corporation, Norwood, New Jersey, 1986.

[89] R. Horaud, S. Christy, F. Dornaika, and B. Lamiroy. Object pose: Links between paraperspective and perspective. In *Proceedings Fifth International Conference on Computer Vision*, pages 426–433, Cambridge, Mass., June 1995. IEEE Computer Society Press, Los Alamitos, Ca.

[90] R. Horaud, B. Conio, O. Leboulleux, and B. Lacolle. An analytic solution for the perspective 4-point problem. *Computer Vision, Graphics, and Image Processing*, 47(1):33–44, July 1989.

[91] R. Horaud and F. Dornaika. Hand-eye calibration. *International Journal of Robotics Research*, 14(3):195–210, June 1995.

[92] R. Horaud, F. Dornaika, B. Lamiroy, and S. Christy. Object pose: The link between weak perspective, para perspective, and full perspective. Technical Report RR-2356, INRIA, September 1994. Available by anonymous ftp on imag.fr, directory pub/MOVI, file RR-2356.ps.gz.

[93] R. Horaud, R. Mohr, and B. Lorecki. Linear-camera calibration. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 1539–1544, Nice, France, May 1992.

[94] R. Horaud, R. Mohr, and B. Lorecki. On single-scanline camera calibration. *IEEE Transactions on Robotics and Automation*, 9(1):71–75, February 1993.

[95] R. Horaud and Th. Skordas. Model-based strategy planning for recognizing partially occluded parts. *IEEE Computer Magazine*, pages 58–65, August 1987. Special issue on *CAD-Based Robot Vision*.

[96] R. Horaud and Th. Skordas. Stereo Matching through Feature Grouping and Maximal Cliques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-11(11):1168–1180, November 1989.

[97] B. K. P. Horn. *Robot Vision*. MIT Press, Cambridge, Massachusetts, 1986.

[98] S.L. Horowitz and T. Pavlidis. Picture segmentation by a directed split-and-merge procedure. In *Proceedings of the Second International Joint Conference on Pattern Recognition*, pages 424–433, 1974.

[99] D. Hubel. The Brain. *Scientific American*, 241(3):39–47, September 1979.

[100] M.T. Hueckel. An operator which locates edges in digitized pictures. *J. Ass. Comput. Mach.*, 18:113–125, 1971.

[101] Y. Hung, P.-S. Yeh, and D. Harwood. Passive ranging to known planar point sets. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 80–85, Saint-Louis, Missouri, USA, March 1985.

[102] M. Kass, A. Witkin, and D. Terzopoulos. Snakes : Active contours models. *ICCV*, 1987.

[103] Y. Kerbrat and J. M. Braemer. *Géométrie des courbes et des surfaces*. Hermann, Paris, collection méthodes, 1976.

[104] N. Keskes. *Application des techniques d'analyse d'images aux signaux sismiques*. PhD thesis, Université Paris VI, 1984.

[105] Nomizu Kobayashi. *Fundations of Differential Geometry*. John Wiley.

[106] Jan J. Koenderink. *Solid Shape*. MIT Press, Boston, 1990.

[107] J. Krumm and S. Shafer. Local spatial frequency analysis of image texture. *International Conference in Computer Vision, Osaka,Japan*, 1990.

[108] R. Kumar and A. R. Hanson. Robust estimation of camera location and orientation from noisy data having outliers. In *Proc. Workshop on Interpretation of 3-D Scenes*, pages 52–60, Austin, Texas, USA, November 1989.

[109] J. Lelong-ferrand and J.M. Arnaudies. *Géométrie et Cinématique*. Dunod Université, Paris, 1977.

[110] Z.-C. Lin, T. S. Huang, S. D. Blostein, H. Lee, and E.A. Margerum. Motion estimation from 3-d point sets with and without correspondences. In *Proceedings Computer Vision and Pattern Recognition*, pages 194–201, Miami-Beach, Florida, USA, June 1986.

[111] Y. Liu, T. S. Huang, and O. D. Faugeras. Determination of camera location from 2-d to 3-d line and point correspondences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):28–37, January 1990.

[112] P. Long-Limozin. *Vision Stéréoscopique Appliquée à la Robotique*. PhD thesis, Université de Nice, France, October 1986.

[113] D. Lowe. *Perceptual Organization and Visual Recognition*. Kluwer Academic Publisher, 1985.

[114] D. Lowe. Three-dimensional Object Recognition from Single Two-dimensional Images. *Artificial Intelligence*, 31:355–395, 1987.

[115] D. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5):441–450, May 1991.

[116] David G. Luenberger. *Optimization by Vector Space Methods*. Wiley, New York, 1969.

[117] Q-T. Luong. *Matrice Fondamentale et Autocalibration en vision par ordinateur*. PhD thesis, Université de Paris Sud, Orsay, December 1992.

[118] F. Lustman. *Vision Stéréoscopique et Perception du Mouvement en Vision Artificielle*. PhD thesis, Université de Paris-Sud Centre d'Orsay, France, December 1987.

[119] A. Lux. *Algorithmique et Contrôle en Vision par Ordinateur*. PhD thesis, Institut National Polytechnique de Grenoble, September 1985.

[120] G. Malandain, G. Bertrand, and N. Ayache. Topological segmentation of discrete surfaces. In *IEEE Conference on Vision and Pattern Recognition*, Hawaii, June 1991.

[121] D. Marr. *Vision*. W. H. Freeman, San Francisco, 1982.

[122] D. Marr and E. Hildreth. Theory of edge detection. *Proc. Royal Soc. Lond.*, B 207:187–217, 1980.

[123] A. Martelli. Edge detection using heuristic search methods. *Computer Graphics and Image Processing*, 1:169–182, 1972.

[124] B. Mazier. *Mise en Correspondance d'Images Multimodales Appliquée à la Visée Pédiculaire Assistée par Ordinateur*. PhD thesis, Université Joseph Fourier, Grenoble, December 1992.

[125] R. Mohan, G. Medioni, and R. Nevatia. Stereo error detection, correction, and evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(2):113–120, February 1989.

[126] R. Mohr and L. Morin. Relative positioning from geometric invariants. In *Proceedings Computer Vision and Pattern Recognition Conference*, pages 139–144, Lahaina, Maui, Hawaii, June 1991.

[127] O. Monga. An optimal region growing algorithm for image segmentation. *International Journal on Pattern Recognition and Artificial Intelligence*, 1(3):351–376, December 1987.

[128] O. Monga and R. Deriche. 3d edge detection using recursive filtering. In *Computer Vision and Pattern Recognition*, San Diego, June 1989. IEEE.

[129] O. Monga, R. Deriche, and G. Malandain. Recursive filtering: a primary tool for 3d edge detection. In *First European Conference on Computer Vision*. ECCV 90, 1990.

[130] O. Monga and N. Keskes. A hierarchical algorithm for the segmentation of 3-d images. In *Proc. of eighth International Conference on Pattern Recognition*, Paris, October 1986.

[131] O. Monga and B. Wrobel. Une méthode simple de contrôle d'une croissance de régions avec les contours. In *AFCET*, Antibes, 1987.

[132] Olivier Monga, Nicholas Ayache, and Peter Sander. From voxel to curvature. In *IEEE Conference on Vision and Pattern Recognition*, Hawaii, June 1991.

[133] Olivier Monga and Serge Benayoun. Using partial derivatives of 3d images to extract typical surface features. *INRIA Report research*, 1992.

[134] Olivier Monga, Rachid Deriche, and Gregoire Malandain. Recursive filtering and edge closing: two primary tools for 3d edge detection. *Image and Vision Computing, Vol. 9, Number 4*, August 1991. A shortened version is in proc. of ECCV'90, Lecture notes in Computer Science 427.

[135] Olivier Monga, Rachid Deriche, and Jean-Marie Rocchisani. 3d edge detection using recursive filtering: Application to scanner images. *Computer Vision Graphic and Image Processing, Vol. 53, No 1, pp. 76-87*, January 1991.

[136] Olivier Monga, Richard Lengagne, and Rachid Deriche. Extraction of the zero-crossing of the curvature derivative in volumic 3d medical images : a multi-scale approach. *INRIA Report research 2338, a shortened version is in IEEE conference on Computer Vision and Pattern Recognition, Seattle, June 1994*, 1994.

[137] J. J. Moré and D. C. Sorensen. On the use of directions of negative curvature in a modified Newton method. *Mathematical Programming*, pages 1–20, 1979.

[138] J. Muerle and D. Allen. Experimental evaluation of techniques for automatic segmentation of objects in a complex scenes. In G. Cheng

et al., editor, *Pictorial Pattern Recognition*, pages 3–13. Thompson, 1968. Washington, D.C.

[139] M. Nagao and M. Matsuyama. Structural image analysis, 1983.

[140] J.A. Noble. Finding corners. *Image and Vision Computing*, 6(9):121–128, May 1988.

[141] P. Montesinos O. Monga, N. Armande. Using crest lines to extract thin networks in images : application to roads and blood vessels. *INRIA Report research 1995, des versions réduites sont dans Lecture Notes in Computer Science, Springer Verlag, IPMI'95, et SCIA'95*, 1995.

[142] D. Oberkampf, D. F. DeMenthon, and L. S. Davis. Iterative pose estimation using coplanar feature points. In *Proceedings Computer Vision and Pattern Recognition*, pages 626–627, New York, June 1993. IEEE Computer Society Press, Los Alamitos, Ca.

[143] Y. Ohta and T. Kanade. Stereo by intra- and inter-scanline search using dynamic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-7(2):139–154, March 1985.

[144] Y. Ohta, T. Kanade, and T. Sakai. Color information for region segmentation. In *Computer Graphics and Image Processing*, pages 222–241, 1980. No. 13.

[145] R. Olhander, K. Price, and D.R. Reddy. Picture segmentation using a recursive region splitting method. In *Computer Graphics and Image Processing*, number 8, pages 313–333, 1978.

[146] Eric Pauchon. *Numérisation et modélisation automatique d'objet 3D*. PhD thesis, Université d'Orsay (Paris Sud), 1983.

[147] T. Pavlidis and S. L. Horowitz. Segmentation of plane curves. *IEEE Trans. on Computers*, C-23(8):860–870, August 1974.

[148] Theodosios Pavlidis. A critical survey of image analysis methods. In *Proceedings of the Eighth International Conference on Pattern Recognition*, pages 502–511, October 1986.

[149] T. Q. Phong, R. Horaud, A. Yassine, and D. T. Pham. Object pose from 2-D to 3-D point and line correspondences. Technical Report RT 95, LIFIA–IMAG, February 1993.

[150] T. Q. Phong, R. Horaud, A. Yassine, and D. T. Pham. Object pose from 2-D to 3-D point and line correspondences. *International Journal of Computer Vision*, 15(3):225–243, July 1995.

[151] S. B. Pollard. *Identifying Correspondences in Binocular Stereo.* PhD thesis, University of Sheffield, November 1985.

[152] S. B. Pollard, J. E. W. Mayhew, and J. P. Frisby. PMF: A stereo correspondence algorithm using a disparity gradient limit. *Perception*, 14:449–470, 1985.

[153] S. B. Pollard, J. Porril, J. E. W. Mayhew, and J. P. Frisby. Disparity gradient, lipschitz continuity, and computing binocular correspondence. In *The Third International Symposium on Robotics Research*, France, 1986.

[154] Jean Ponce. *Représentation des objets tridimensionnels.* PhD thesis, Université d'Orsay (Paris Sud), 1988. Chapitre 4.

[155] T.C. Pong, L.G. Shapiro, L.T. Watson, and R.M. Haralick. Experiments in segmentation using a facet model region grower. In *Computer Vision Graphics and Image Processing*, number 25, 1980.

[156] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Wetterling. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, 1988.

[157] P. Puget and Th. Skordas. An Optimal Solution for Mobile Camera Calibration. In O. Faugeras, editor, *Computer Vision – ECCV 90, Proceedings First European Conference on Computer Vision, Antibes, France*, pages 187–198. Springer Verlag, April 1990.

[158] P. Remagnino, P. Brand, and R. Mohr. Correlation techniques in adaptive template matching with uncalibrated cameras. In *SPIE Proceedings Vision Geometry III*, pages 252–263, Boston Mass, November 1994.

[159] J.M. Rocchisani, O. Monga, J. Bittoun, and R. Deriche. Automatic spatio-temporal edge detection of the beating heart in mri. In *17ème Congrès International de Radiologie*, Paris, July 1989.

[160] A. Rosenfeld and A. C. Kak. *Digital Picture Processing.* Academic Press, Orlando, Fl., 1982.

[161] Peter T. Sander. *On Reliably Inferring Differential Structure from Three-Dimensional Images*. PhD thesis, McGill University, 1988. Available as Technical Report CIM-88-1, McGill Research Center for Intelligent Machines, McGill University, Montréal.

[162] Peter T. Sander and Steven W. Zucker. Tracing surfaces for surfacing traces. In *Proceedings of the First International Conference on Computer Vision*, pages 241–249, London, June 1987.

[163] Peter T. Sander and Steven W. Zucker. Inferring surface trace and differential structure from 3-D images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(9), September 1990.

[164] L. G. Shapiro and R. M. Haralick. Structural descriptions and inexact matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-3(5):504–519, September 1981.

[165] J. Shen and S. Castan. An optimal linear operator for edge detection. In *Conference on Vision and Pattern Recognition*, USA, Juin 1986. IEEE.

[166] Th. Skordas. *Mise en correspondance et reconstruction stéréo utilisant une description structurelle des images*. PhD thesis, Institut National Polytechnique de Grenoble, October 1988.

[167] M. Spivak. *A comprehensive introduction to differential geometry*, volume 1 to 5. Berkley, 1971.

[168] J-P. Thirion and Gourdon A. The 3d marching lines algorithm and its application to crest lines extraction. Technical Report 1672, INRIA, May 1992.

[169] Jean Philippe Thirion. The extremal mesh and the understanding of 3d surfaces. *IEEE workshop on Biomedical Image Analysis*, June 1994.

[170] R. Y. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal of Robotics and Automation*, RA-3(4):323–344, August 1987.

[171] R.Y. Tsai and R.K. Lenz. Real Time Versatile Robotics Hand/Eye Calibration using 3D Machine Vision. In *IEEE International Conference on Robotics and Automation*, pages 554–561, Philadelphia, Penn, USA, April 1988.

[172] S. Umeyama. Least-squares estimation of transformation parameters between two point pattern. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, April 1991.

[173] M. W. Walker, L. Shao, and R. A. Volz. Estimating 3-d location parameters using dual number quaternions. *CGVIP-Image Understanding*, 54(3):358–367, November 1991.

[174] S. Watanabe and CYBEST Group. An automated apparatus for cancer prescreening: Cybest. In *Computer Graphics and Image Processing*, number 3, pages 350–358, 1974.

[175] Tapani Westman. A combined region and contour based stereo vision system. In *Proc. 6th Scandinavien Conference on Image Analysis, Oulu*, June 1989.

[176] J.S. Weszka, R.N. Nagel, and A. Rosenfeld. A threshold selection technique. In *IEEE Trans. Comput.*, number C-23, pages 1322–1326, 1974.

[177] M. Williams. Algorithm 232 heapsort. In *Graphic and Image Processing Communication of the ACM*, volume 7, pages 347–348, June 1984.

[178] X.Tu and B. Dubuisson. A safe image segmentation algorithm and its data structure adaptable to parallel implementation. In *Proc. 6th Scandinavien Conference on Image Analysis, Oulu*, June 1989.

[179] A. Yassine. *Etudes Adaptatives et Comparatives de Certains Algorithmes en Optimisation. Implémentation Effectives et Applications.* PhD thesis, Université Joseph Fourier, Grenoble, 1989.

[180] J. S.-C. Yuan. A general photogrammetric method for determining object position and orientation. *IEEE Transactions on Robotics and Automation*, 5(2):129–142, April 1989.

[181] A. Yuille and M. Leyton. 3d symmetry-curvature duality theorems. *Computer Vision, Graphics, and Image Processing*, 52:124–140, 1990.

[182] A. L. Yuille and T. Poggio. A generalized ordering constraint for stereo correspondence. Technical Report AI Memo 777, MIT Artificial Intelligence Laboratory, May 1984.

[183] Z. Zhang, R. Deriche, O. D. Faugeras, and Q-T. Luong. A robust technique for matching two uncalibrated images through the recovery of

the unknown epipolar geometry. *Artificial Intelligence*, 1995. Special volume on computer vision.

[184] Steven W. Zucker. Region growing: Childhood and adolescence. *Computer Graphics and Image Processing*, 5:382–399, 1976.

[185] S.W. Zucker and R.M. Hummel. A three-dimensional edge operator. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-3(3):324–331, May 1981.

Index