Analyse du mouvement par mise en correspondance d'indices visuels

Christophe Discours

THÈSE

présentée par

Christophe Discours

pour obtenir le titre de DOCTEUR
de L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE
(Arrêté ministériel du 23 novembre 1988)
Spécialité INFORMATIQUE

Analyse du Mouvement Par Mise en Correspondance d'Indices Visuels

Date de soutenance : 7 février 1990

Composition du jury :

Jacques Mossière	président
Patrick Bouthemy Alain Chéhikian	rapporteurs
Augustin Lux James L. Crowley	examinateurs

Thèse préparée au sein du laboratoire LIFIA

Résumé

Dans cette thèse nous décrivons un système de vision utilisant la mesure du mouvement pour extraire des informations tri-dimensionnelles. Ce système permet de fournir des indices 3-D à partir d'une séquence d'images. Il s'articule en trois modules qui peuvent fonctionner indépendamment : Un module d'extraction de segments de droite dans une image, un module de suivi de segments de droite dans une séquence et un module d'extraction de segments 3-D à partir de segments 2-D dont le mouvement a été mesuré.

Dans une première partie nous fournissons une étude bibliographique des méthodes utilisant le mouvement dans la vision.

Dans les parties suivantes nous décrivons chacun des trois modules composant notre système :

- Une technique d'extraction de segments de droite à partir d'une image de contours. La caractéristique principale de cette technique est d'utiliser un balayage simple de l'image et qu'elle est particulièrement destinée à être cablée et fonctionner en synchronisation avec le balayage vidéo.

  Pour parvenir à ce résultat, nous découpons le problème en deux : Nous développons d'abord un algorithme de chaînage de points par balayage, puis nous définissons une méthode incrémentale d'extraction de segments de droite. Pour finir nous montrons comment cette méthode d'extraction peut être intégrée à l'intérieur du chaînage.

- Un suivi d'indice performant. Notre système est fondé sur l'utilisation d'un modèle des indices observés et d'un filtre de Kalman.

  Nous montrons, dans le cas où les indices sont des segments de droite, qu'un choix judicieux des paramètres servant à représenter les indices permet d'avoir une formalisation tres simple du filtrage de Kalman. Comme d'autre part l'utilisation de tels outils nous permet de réduire considérablement la combinatoire de la mise en correspondance, le suivi d'indices qui en résulte est très rapide et peut fonctionner sur des images possédant beaucoup d'indices.

  De part sa rapidité et son efficacité, ce système permet d'envisager le suivi d'indices en temps réel. De telles caractéristiques en font un élément fondamental de tous système de vision utilisant la mesure du mouvement.

- Une reconstruction de segments 3-D. Cette reconstruction, utilisant l'étiquetage de segments dans une séquence fourni par le suivi d'indice, est fondée sur l'utilisation des équations de la stéréovision.

  Cette reconstruction permet de vérifier le bon comportement du suivi d'indices, et principalement l'utilisation de l'étiquetage sur de grandes séquences d'images.

Dans chacune de ces parties nous montrons les résultats d'expériences menées avec chacun des modules.

Sommaire

Chapitre 1

Introduction

La robotique utilise par définition le mouvement. Les systèmes de vision, destinés à être appliqués à la robotique, ne peuvent donc pas négliger la connaissance du mouvement. Nous présentons dans cette thèse une technique fondée sur la mesure et l'utilisation du mouvement, permettant à un système de vision de percevoir et de modéliser l'évolution dynamique de son environnement.

Dans ce premier chapitre, nous allons décrire comment se place ce problème à l'intérieur du domaine de la vision. Ensuite nous présentons une esquisse des techniques développées dans cette thèse. Pour finir nous donnons un résumé des différents chapitres.

1.1 Le problème d'un système de vision mobile

On peut définir la vision comme une méthode qui, à partir d'images, fournie une description de l'univers immédiatement observable (la "scène"). Cette approche n'est jamais appliquée telle quelle, de façon globale. Elle est constituée de la composition de plusieurs méthodes qui travaillent sur différents niveaux de représentation de l'information visuelle.

Une approche classique de la vision consiste à utiliser quatre niveaux de représentations (figure 1.1) :

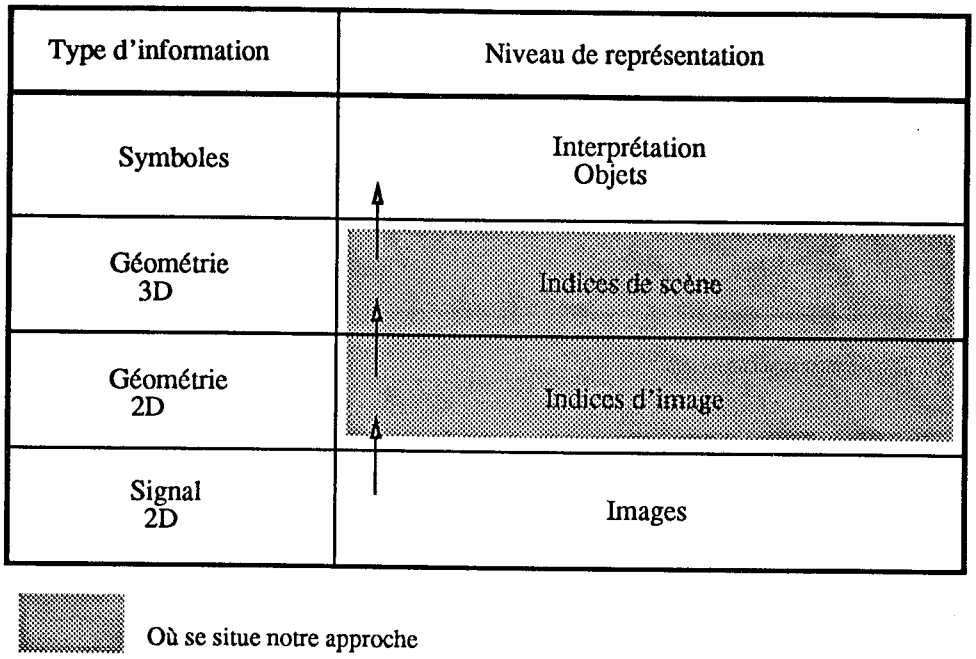


Figure 1.1: La vision et les niveaux de représentations

- Le niveau image : Les données sur lesquelles travaille la vision.
- Le niveau indices d'images : Ce sont les informations pertinentes contenues dans les images.
- Le niveau indices de scène : Ce sont les informations géométriques 3D liées aux indices d'image, comme l'orientation d'une surface.
- Le niveau interprétation de scène : Ce qui est vu dans une scène est interprété de façon symbolique (existence, position, relations entre les objets).

On peut faire ressortir deux types d'approches de l'analyse du mouvement. Chacune de ces deux approches se situe différemment à l'intérieur des différents niveaux de la vision :

- L'approche "optic flow" travaille directement sur les données tridimensionnelles que constitue une séquence d'image, afin d'en extraire des "indices dynamiques".
- L'approche par suivi d'indices ("token tracking") qui fait d'abord subir à chacune des images une extraction d'indices, et travaille donc sur une séquence d'indices.

A ce niveau la seule information que l'on calcule est la valeur du mouvement. Cette seule information peut suffire dans le cas d'un suivi. Mais dans le cadre de la robotique, ce que l'on demande à la vision est généralement de fournir une représentation de l'univers dans lequel se trouve un robot.

L'autre problème de l'analyse du mouvement est donc de retrouver des informations spatiales à partir des informations de mouvement. La meilleure qualité des informations fournies et la plus grande généralité du suivi d'indice par rapport à l'approche "optic flow" en fait la solution de choix pour réaliser cette tâche.

1.2 Notre approche

Dans la figure 1.1 nous avons schématisé la position de nos travaux à l'intérieur de la vision : Un système qui à partir d'une séquence d'images extrait les informations géométriques tri-dimensionnelles de la scène. La figure 1.2 montre comment nous avons donc découpé le problème en trois parties indépendantes, les solutions de chacun de ces sous-problèmes s'imbriquant pour former le système global :

1. **L'obtention d'indices** : C'est une partie fondamentale dans la vision. Même si ce domaine a été l'objet de nombreux travaux, on ne peut pas vraiment affirmer qu'il soit définitivement fermé. Il y a néanmoins dans ce qui existe déjà matière à réaliser des extracteurs d'indices de façon fiable et automatique, facilement transposable en "composant cablés" s'intégrant dans une "machine" de vision. Notre contribution dans ce domaine consiste à pousser cette automatisation jusqu'à l'obtention d'indices d'un niveau suffisamment élevé (en l'occurence des segments de droite) et d'une manière qui soit aisément transformable en version "câblée".

2. **Le suivi d'indices** : Ceci est tout à fait fondamental dans un système de vision mobile. Si notre approche sur les indices nous amène à utiliser des segments de droite, il nous semble néanmoins que le système que nous décrivons est le même quel que soit le type d'indices utilisés. D'autre part notre système étant un suivi dans le plan des images, il est totalement indépendant de la manière dont est effectuée la prise de vue. Nous développons ainsi une base nécessaire pour tout système de vision mobile au même titre que peut l'être le calcul du gradient pour l'extraction d'indices.

3. **La reconstruction tridimensionnelle** : Un des buts principaux de la vision mobile est de retrouver la troisième dimension. En utilisant un environnement suffisamment contraint (caméra montée dans la pince d'un robot manipulateur), nous avons vérifié la validité de notre suivi d'indice en fournissant des informations géométriques dans le repère de la scène observée.

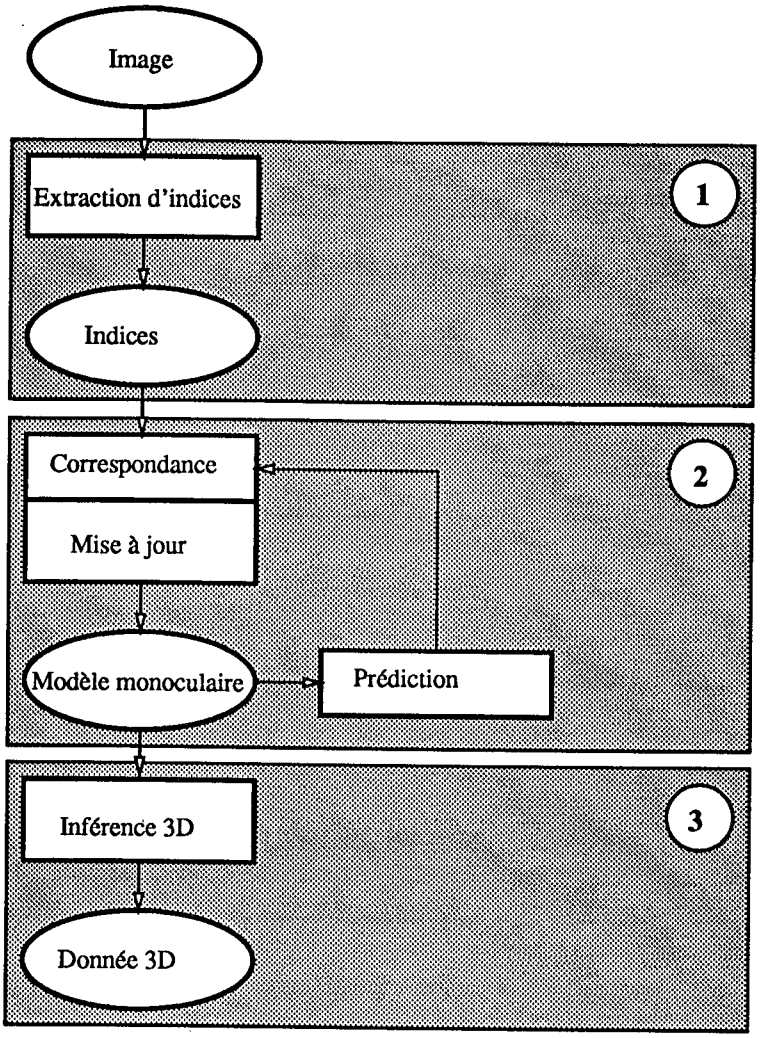


Figure 1.2: Description schématique de notre système

En ce qui concerne l'extraction de segments, nous montrons la faisabilité d'un traitement fonctionnant en un balayage simple de l'image. L'avantage d'une telle méthode est évident lorsqu'il s'agit de réaliser un système cablé.

Avec le type de méthode que nous développons, la complexité du problème de l'extraction de segments de droite est bornée par la valeur $n * m * O$, où n et m sont les dimensions de l'image de départ (en pixels) et O est l'opération la plus complexe qui puisse être effectuée en un pixel donné. Cette opération est elle-même constituée d'un nombre fixe d'opérations élémentaires. Elle peut donc facilement être effectuée pour chaque pixel au cours d'un balayage de l'image à la cadence vidéo.

Notre apport principal se situe sur la méthode de suivi d'indices. Le problème le plus important dans une telle méthode est celui de la mise en correspondance entre indices d'images consécutives. Les solutions classiques d'un tel problème font appel à des correspondances entre graphes et sont très combinatoires.

Nous proposons une solution plus simple qui utilise la connaissance acquise du mouvement pour établir les correspondances. Notre approche s'appuie sur deux idées principales :

- Au lieu d'effectuer une correspondance d'image à image, nous effectuons une mise en correspondance entre un modèle des indices déjà observés et l'image en cours.
- Le mouvement des indices dans le modèle est estimé en utilisant un filtre de Kalman.

Le modèle des indices permet, en donnant une "vie" à ces indices, de s'affranchir d'une partie du bruit présent dans les observations. Le filtre de Kalman permet quant à lui de minimiser le bruit dans l'estimation du mouvement des indices. De plus en fournissant une évaluation de l'incertitude dans leurs positions, il donne les outils nécessaires pour effectuer une mise en correspondance simple et rapide.

1.3 Organisation du rapport

Nous allons dans un premier temps fournir une analyse bibliographique du domaine, qui nous aidera dans le choix des méthodes à appliquer pour arriver à nos fins. Nous effectuons une étude générale de la vision par ordinateur et développons plus particulièrement les différentes méthodes d'analyse du mouvement. Ensuite nous étudions quels sont les différents types d'indices visuels et comment les extraire d'une image. Puis nous analysons les différentes méthodes utilisant le suivi d'indices. Pour finir nous examinons comment retrouver les informations 3D à partir du mouvement extrait dans une séquence d'image.

Dans le chapitre 3 nous décrivons une méthode d'extraction par balayage de segments de droite dans une image binaire. Une analyse du problème nous amène à séparer en deux sa résolution : D'abord un chaînage des points, ensuite une segmentation des chaînes.

En étudiant les différents aspects du voisinage immédiat d'un pixel, nous développons une méthode très simple d'extraction des chaînes de points qui s'effectue par un simple balayage de l'image.

Nous montrons ensuite comment nous pouvons extraire des segments de droite dans une chaîne de points de manière incrémentale, c'est à dire sans connaître la totalité de la chaîne au départ. Nous montrons également comment l'équation d'une droite peut être représentée seulement par des nombres entiers.

Nous proposons alors dans 3.2 la réalisation du chaînage et de l'extraction de droites, mais de façon à ce que la seconde se trouve intégrée dans le premier. Nous décrivons d'abord la réalisation du chaînage seul, puis l'intégration de l'extraction d'une droite dans la recherche d'une chaîne, pour finir par l'intégration de l'extraction d'une droite dans la recherche de plusieurs chaînes.

Puis nous montrons des résultats que nous comparons avec ceux obtenus par une méthode de recherche exhaustive.

Dans le chapitre 4 nous décrivons notre système de suivi. Nous avons dans notre étude bibliographique relevé les points suivants :

- La nécessité de l'utilisation d'un modèle de ce que nous observons.
- L'intéret de l'utilisation d'un filtre de Kalman pour la réalisation d'un suivi d'indices.
- Le problème très ouvert de la mise en correspondance et sa relation avec les points précédents.

Nous décrivons donc en premier lieu le principe du modèle dynamique et comment l'utiliser. Nous décrivons ensuite le principe du filtre de Kalman, mais en étudiant au préalable l'importance du choix des paramètres servant à représenter nos indices. En effet le fonctionnement de ce filtre dépend des relations existant entre les valeurs estimées. Pour finir nous étudions comment les incertitudes estimées par le filtre de Kalman sont utilisées pour aider la mise en correspondance.

Dans une seconde partie nous décrivons plus particulièrement l'algorithme de suivi. Nous résolvons certains problèmes spécifiques à la mise en oeuvre d'un filtre de Kalman dans un suivi d'indices, comme par exemple le problème de l'initialisation du filtre. Nous proposons également des solutions pour certains problèmes apparus lors des expérimentations. Pour finir nous abordons les contraintes demandées par une réalisation câblée d'un tel algorithme.

Nous terminons le chapitre du suivi d'indice en montrant les résultats des expériences que nous avons menées.

Le dernier chapitre est conscré à la description d'une méthode de reconstruction des segments en trois dimensions. Cette méthode suppose le calibrage des caméras

et du robot que nous utilisons. Nous décrivons donc les équations de stéréovision que nous employons. Ensuite nous exposons comment effectuer une reconstruction tri-dimensionnelle de points à partir des données du calibrage et comment utiliser les informations épipolaires pour ramener les segments à des paires de points. Pour finir nous montrons les résultats de la reconstruction sur les séquences d'images que nous avons utilisée avec le suivi.

Chapitre 2

Un système de vision mobile

2.1 Définition du problème

Nous allons dans cette partie rappeler quels sont les buts de la vision par ordinateur, quels sont les moyens existants pour atteindre ces buts et comment nous nous sommes situés dans ce domaine. Par la suite nous développons plus particulièrement nos préoccupations et les applications déjà réalisées.

2.1.1 Vision : Modéliser la géométrie d'une scène

D'un point de vue le plus général possible, la vision par ordinateur consiste, à partir d'images, à obtenir des informations modélisant d'une certaine façon l'univers extérieur, du moins la partie qui en est visible dans l'image, et qui soient facilement exploitables par un système ayant besoin de connaître de cette façon l'univers, en général pour pouvoir agir sur cet univers (cas de la robotique).

Nous venons donc de définir trois notions qui définissent la vision et que nous allons décrire :

- L'"image", qui correspond aux données à traiter.
- Le modèle de représentation de l'univers qui est ce que l'on cherche à obtenir à partir de l'"image".

- Le traitement qui permet d'obtenir le modèle à partir de l'"image".

Une "image" est le plus souvent obtenue au moyen d'une caméra vidéo et peut être constituée d'une seule image, de deux images (stéréo) ou d'une série d'images (vision mobile). Nous savons donc comment ces données, accessibles sous la forme d'un tableau à deux dimensions de valeurs appelées pixels, sont liées à l'univers extérieur (présence de lumière, réflectance des objets, projection perspective).

Le modèle de représentation dépend du type d'informations dont a besoin le système qui utilise la vision. Dans le cas classique de la robotique, ce que l'on a besoin de connaître c'est principalement l'existence et la position d'objets par rapport à un repère propre au robot. Dans notre cas nous nous intéresserons plus particulièrement à la modélisation géométrique des objets visibles. Cette modélisation peut fournir des informations sur le volume des objets, leur surface, leur emplacement ou plus simplement les représenter sous la forme de nuages de points tridimensionnels.

Il apparaît donc que le traitement de vision consiste à obtenir une modélisation géométrique d'une scène en trois dimensions à partir d'un ou de plusieurs tableaux de pixels. tous les systèmes existants utilisent plusieurs niveaux de modélisation de l'univers, permettant ainsi une hiérarchisation du traitement dans le but de faciliter son approche.

2.1.2 Les niveaux de représentation et les différentes sources de données intermédiaires

Au niveau le plus bas, c'est à dire une image représentée sous la forme d'un tableau de pixels, la quantité d'informations est beaucoup trop grande pour être directement exploitée. Le tableau en lui-même prend généralement énormément de place en mémoire. D'autre part les informations ne sont pas distribuées uniformément sur toute l'image. La nécessité d'utiliser des niveaux intermédiaires apparaît donc très tôt dans le traitement.

Yves Demazeau ([Dem 86]) propose une classification en trois approches différentes des systèmes de vision. Chacune de ces approches utilise son propre découpage en niveaux de représentation. Il ressort néanmoins certaines constantes dans ces découpages :

- Comme nous l'avons déjà évoqué, un système de vision ne travaille pas directement sur l'image. Une première étape consiste à extraire des indices visuels dans cette image. Cela peut être des points, des contours, éventuellement sous forme de segments de droites, d'arc de courbes ou d'angles, des régions. Ces indices conservent la structure bi-dimensionnelle de l'image.

- Le passage à une information en trois dimensions peut se faire en utilisant plusieurs images différentes d'une même scène, ou en utilisant certaines contraintes connues de l'univers (vision monoculaire). Nous avons à ce niveau là une description tri-dimensionnelle de la scène (indices 3-D) mais sans connaissance de la composition de cette scène (quels sont les objets en présence ?).
- A partir de là, on peut "reconnaître" des objets. "Reconnaître" peut être par exemple reconstituer un volume à partir d'un nuage de points 3-D. Ce peut aussi être la reconnaissance d'objets dont on connait a priori les caractéristiques.

L'obtention d'indices 3-D à partir d'indices 2-D se fait de différentes manières demandant d'utiliser autant de types différents d'indices visuels, ainsi que différentes façons d'interpréter la ou les images et les indices qui en sont extraits. Nous allons maintenant détailler ces diverses techniques.

Stéréovision

C'est l'approche la plus classique : Obtenir des coordonnées 3-D à partir de coordonnées 2-D dans l'image, si on connait la projection perpective qui les relie, peut être vu comme un problème de résolution d'un système d'équations sous-dimensionné. Utiliser deux images de la même scène, prises à deux endroits différents, enlève la sous-détermination (en fait le système est même sur-dimensionné).

Dans cette approche, la phase la plus critique est celle qui consiste à mettre en correspondance deux indices bi-dimensionnels chacun se trouvant dans une image différente. Cette tâche est aidée par les contraintes de la géométrie tri-dimensionelle (contraintes épipolaires). Mais il n'existe pas de solution générale à ce problème.

Comme le système à résoudre est sur-déterminé, il a été proposé d'utiliser plus que deux images (par exemple Ayache dans [Aya 88] utilise trois images). La résolution se fait à l'aide d'une méthode de moindres carrés ou un filtrage similaire. De plus la multiplication des images, pour autant que leur disposition soit judicieusement choisie, permet d'augmenter les contraintes épipolaires, facilitant ainsi le processus de mise en correspondance.

Le principe de la stéréovision implique que l'on connaîsse exactement la position des caméras, ou du moins la position des caméras l'une par rapport à l'autre. Il faut de même connaître les caractéristiques des transformations perspective, qui dépendent donc des objectifs, des photocapteurs et de leurs défauts respectifs. Toutes ces informations sont obtenues par une phase d'initialisation appelée calibration.

Mouvement

Les techniques utilisant le mouvement visent à peu près le même but que les techniques de stéréovision : Obtenir des coordonnées 3-D de points 2-D apparaissant dans plusieurs images. Le problème principal est également à peu près le même, à savoir la mise en correspondance d'indices visuels.

Par rapport à la stéréovision, la vision du mouvement est beaucoup moins contrainte. Cela implique que l'on ne peut plus utiliser les méthodes propres à la stéréovision comme la contrainte épipolaire, mais par contre le champ d'investigation est plus large. Généralement les reconstructions tri-dimensionelles sont effectuées à un facteur d'échelle près, mais des paramètres dynamiques (tels que le mouvement propre de la caméra) sont en même temps évalués.

Cette approche est très riche, pas toujours facilement maîtrisable, et nous en développons les principales orientations actuelles dans la section 2.1.3.

"Shape from X"

Les deux approches précédentes faisaient appel à plusieurs images pour obtenir la troisième dimension. Ayant une connaissance a priori du mécanisme de formation des images, on peut aussi essayer d'en extraire directement les informations 3-D. Ces techniques sont connues sous les appellations en anglais de "shape from shading" et "shape from texture".

L'approche "shape from shading" utilise directement les niveaux de gris dans l'image. Elle consiste à supposer que les différences d'intensité lumineuse observées sont dues à des différences d'orientations des surfaces par rapport à la source lumineuse. Les informations tri-dimensionelles qui en sont extraites sont généralement de la forme $\frac{\partial^2 \vec{n}}{\partial xy}$, où $\vec{n}$ représente la normale aux surfaces. Des discontinuités dans cette fonction permettent eventuellement de repérer les arêtes des objets.

Les différences d'intensités peuvent être aussi produit par des différences d'albédo des matériaux constituant les objets. D'autre part dans un environnement artificiel (atelier), les sources de lumière sont multiples ou diffuses. Tous ces effets vont donc fausser l'interprétation du "shape from shading".

Le "shape from texture" est une approche similaire qui au lieu d'utiliser l'intensité lumineuse se base sur les modifications dans l'aspect de motifs ("pattern") perçus dans l'image. Les indices visuels 2-D utilisés sont donc d'un niveau plus élevé (comme ceux que propose Marr dans [Mar 82]), mais permettent de s'affranchir des problèmes propres au "shape from shading".

Techniques mixtes

Ces différentes méthodes peuvent être combinées entre elles. Le cas le plus classique est l'association de la stéréovision avec le mouvement. La stéréovision permet d'obtenir une information 3-D absolue dans le repère des caméras tandis que le mouvement fourni les informations dynamiques de vitesses et d'accélération.

2.1.3 L'étude du mouvement dans une séquence d'images

L'étude du mouvement en vision est une discipline dont le développement est assez récent et les méthodologies employées sont un peu disparates et pas très bien classifiables Néanmoins de nombreux auteurs s'accordent à séparer en deux grands thèmes ce domaine : L'approche par flot optique ("optic flow") et l'approche par suivi d'indices. On peut trouver une présentation assez exhaustive de ce domaine en utilisant cette classification dans le travail d'Aggarwal et Nandhakumar [Agg 88].

L'approche "optic flow"

L'idée consiste à associer les différences d'intensités lumineuses entre les images successives à des mouvements dans une image. Le problème est généralement formulé de la façon suivante : Soit $g(x, y, t)$ l'intensité lumineuse en un point x, y d'une image à l'instant t. A l'instant $t + \Delta t$, cette intensité lumineuse se sera déplacée au point $(x + \Delta x, y + \Delta y)$, ce qui s'écrit :

$$g(x + \Delta x, y + \Delta y, t + \Delta t) = g(x, y, t) \tag{2.1}$$

Δx, Δy et Δt étant petits. Le membre gauche de cette équation est approximé par un développement en série de Taylor où les termes de degrés supérieurs ou égaux à deux sont ignorés :

$$g(x + \Delta x, y + \Delta y, t + \Delta t) = g(x, y, t) + \frac{\partial g}{\partial x}\Delta x + \frac{\partial g}{\partial y}\Delta y + \frac{\partial g}{\partial t}\Delta t + \epsilon \tag{2.2}$$

On a donc, lorsque Δt tend vers 0 :

$$2.1\&2.2 \Rightarrow \frac{\partial g}{\partial x}u + \frac{\partial g}{\partial y}v + \frac{\partial g}{\partial t} = 0 \tag{2.3}$$

Dans cette dernière équation, $\frac{\partial g}{\partial x}$, $\frac{\partial g}{\partial y}$ et $\frac{\partial g}{\partial t}$ sont mesurés dans la série d'images, et les vitesses $u = \frac{dx}{dt}$ et $v = \frac{dy}{dt}$ sont les deux composantes du flot optique que l'on

cherche.

L'équation 2.3 ne suffit pas évidemment pour trouver les composantes u et v. C'est pourquoi de nombreuses contraintes supplémentaires sont utilisées pour aider à leur évaluation.

Les plus courantes sont que les variations du champ de vitesses dans l'image ne se font pas brusquement et que des points voisins ont des vitesses sembables. Horn et Schunk ([Hor 81]) utilisent cette contrainte en formulant une fonction d'erreur qui combine l'erreur de l'équation 2.2 et les variations spatiales des vitesses u et v. Leur but est donc de calculer u et v minimisant cette fonction d'erreur. Ils proposent une méthode itérative calculant ces valeurs. Ils utilisent cette méthode en la réitérant jusqu'à la convergence entre deux images ou en exécutant une itération par images, éventuellement jusqu'à une convergence.

D'autres approches consistent à prendre en considération les termes de degrés supérieurs ou égaux à deux dans le développement de 2.2. Nagel dans [Nag 87], en comparant différentes études de ces types de contraintes faites par d'autres auteurs, propose des contraintes qui soient les moins fortes possibles mais néanmois suffisantes. Elles reposent sur l'utilisation d'un détecteur de "coin" d'intensité lumineuse, où l'estimation de u et v est la plus aisée. Il ressort de ces études que la recherche du champ de vitesse est basée sur l'utilisation de deux hypothèses :

1. Comme nous l'avons déjà vu on suppose que le champ de vitesse est constant dans l'image, ou du moins que les variations sont faibles ("smoothness constraint").

2. La première hypothèse est indispensable car on peut voir dans l'équation 2.3 (et même si on développe à un degré supérieur ou égal à deux) que la vitesse ne peut pas être connue là où l'intensité lumineuse est constante. En fait u et v ne sont évaluables qu'en des endroits particuliers de l'image comme les "coins" de Nagel.

Cette deuxième hypothèse est importante pour toutes les méthodes d'estimation du mouvement, principalement la partie sur laquelle Hildreth a basé son étude dans [Hil 84]. Hildreth part de ce qu'elle appelle le "problème de l'ouverture" (figure 2.1) :

En regardant le mouvement d'un bord par une petite ouverture, seule la composante de la vitesse perpendiculaire à ce bord peut être évaluée, mais le mouvement du bord reste inconnu. Vu d'une autre façon, si on change de repère dans l'image de telle manière que y soit parallèle au "bord" et x perpendiculaire, la notion de "bord" s'exprime de sous la forme : $\frac{\partial g}{\partial y} = 0$. L'équation 2.3 devient donc :

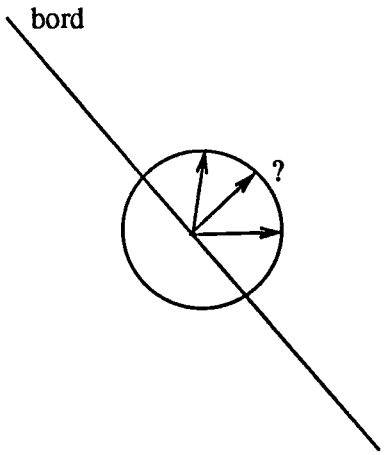


Figure 2.1: Probleme de l'ouverture

$$2.3 \Rightarrow \frac{\partial g}{\partial x}u = -\frac{\partial g}{\partial t} \tag{2.4}$$

Ce qui permet de déterminer u la composante "perpendiculaire" et laisse v totalement indéterminée.

Hildreth utilise un extracteur de contour et calcule le déplacement perpendiculaire aux courbes extraites entre deux images. Pour retrouver le déplacement elle utilise la première hypothèse en minimisant la disparité des vitesses le long des courbes ($\frac{\partial \vec{V}}{\partial s}$ où s est l'abscisse curviligne).

Indépendammant des contraintes dans l'image, d'autres auteurs étudient le mouvement lorsque les déplacements du capteur et/ou de la scène sont contraints ([Law 81], [Pra 83]) : Déplacement rectiligne de la caméra, déplacement dans un plan.

D'autres méthodes, bien que débordant un peu de cette classification, peuvent être assimilées à l'approche "optic flow". Bolles, Baker et Marimont ([Bol 87]) utilisent une séquence d'images temporellement très dense. Ils considèrent ainsi leurs données comme une image à trois dimensions x, y et t. Ils découpent cette image en tranches u, t de plans épipolaires qu'ils étudient comme une image classique, principalement en y recherchant des droites. Ils s'intéressent plus particulièrement aux occlusions et, le mouvement ayant généré leur "bloc" d'images étant connu, à la recherche de l'espace libre entre les objets.

Enfin le flot optique peut être utilisé pour segmenter les images, au même titre que les différences d'intensité lumineuse en vision classique. Par exemple [Bou 87] definissent une méthode de "segmentation/fusion" permettant de découper une im-

age en régions de mouvement homogène.

La méthode d'évaluation du mouvement par "optic flow" demande de calculer des dérivés partielles de l'intensité lumineuse. Cette intensité est généralement bruitée, et les opérations de dérivations augmentent le bruit, les résultats fournis par cette technique ne sont pas directements utilisables. C'est pourquoi il convient d'utiliser les deux hypothèses que nous avons formulées pour améliorer ces résultats. Mais ces hypothèses sont contradictoires et contribuent plus à corriger l'erreur moyenne qu'à supprimer des valeurs fausses. L'hypothèse du champ constant n'est vraie que localement mais fausse globalement. La deuxième hypothèse va à l'encontre de la première : La vitesse ne peut être évaluée que là où l'intensité change, ce qui souvent correspond aux occlusions entre objets, donc à des discontinuités du champ de vitesses.

La recherche du flot optique est donc une méthode délicate à utiliser. Elle ne demande pas une "compréhension" de l'image (sauf partiellement, lorsqu'on veut ajouter des contraintes) et peut être vue comme un moyen d'obtenir des "indices" de mouvement.

Suivi d'indices

Contrairement à l'approche précédente, cette méthode n'utilise pas directement l'intensité lumineuse, mais est basée sur l'utilisation d'indices image : Ces indices sont mis en correspondance entre les images et le déplacement est ainsi mis en évidence.

Il apparaît donc un nouveau problème, qui occupe d'ailleurs une place importante dans cette méthode, celui de la mise en correspondance. Celle-ci est faite à différents niveaux et peut interférer avec l'estimation du mouvement.

La mise en correspondance peut être faite par mise en corrélation d'indices images, ou même de morceaux d'images en niveaux de gris, par le calcul d'une fonction ayant un maximum quand deux motifs équivalents se recouvrent. Burt, Yen et Xu ([Bur 82]) proposent une forme générale de cette fonction en une dimension :

$$C(i,\delta) = \sum_m w(m)(F[f(i+m)] - F[h(i+m+\delta)]) \qquad (2.5)$$

Le but est de chercher δ maximisant C pour le point i. f est la fonction "première image" et h la fonction "deuxième image" qui renvoient les valeurs des points pour chaque image. w est une fonction de fenètrage qui permet de ne s'occuper que d'une certaine partie de l'image. L'étude de Burt, Yen et Xu porte sur la comparaison de

différentes fonctions de filtrage F et leur comportement sur des images bruitées.

Garcia et Doncarli ([Gar]) utilisent une approche similaire, mais cette fois ci en deux dimensions. Dans ce cas, en plus des déplacements en translation selon x et y (δ dans le cas précédent), ils tiennent compte d'une rotation θ de la scène et d'une modification d'échelle dans les deux directions (s_x et s_y).

En utilisant des indices d'image contenant plus d'informations, la structure des images peut aider la mise en correspondance. Par exemple Thorpe et Shafer ([Tho 83]) utilisent comme points d'intérêt les jonctions entre trois segments, qui correspondent dans la scène aux coins entre trois plans. Utilisant les relations entre indices qu'impliquent la structure tridimensionnelle supposée de ces indices, les correspondances qui ne sont pas consistantes sont supprimées au fur et à mesure dans un graphe où au départ toutes les correspondances possibles sont représentées. L'algorithme se termine lorsqu'à un indice d'une image ne correspond plus qu'un seul indice d'une autre image, ou éventuellement aucun. Il peut y avoir différentes façons de réduire un graphe de correspondance, comme par exemple lorsqu'on cherche à apparier deux images d'un cube.

Les méthodes exposées jusqu'à maintenant n'utilisaient pas la notion de mouvement pour la résolution du problème de mise en correspondance. La première idée pour en tenir compte est de supposer que deux images successives ne sont pas très éloignées temporellement et que les différents indices ne se sont pas beaucoup déplacés dans ces images. On privilégie donc les appariements entre indices de coordonnées proches.

Une contrainte qui tient mieux compte des réalités physiques de la scène observée est que les accélérations sont faibles, donc que les vitesses des indices sont faiblement modifiées d'une image à une autre. Au problème de la correspondance vient ainsi s'ajouter la notion de prédiction qui consiste à estimer la position des indices connaissant partiellement leur mouvement.

Généralement, le mouvement est estimé dans la scène, en utilisant les caractéristiques propres aux objets, pour ajouter des contraintes au mouvement supposé (contraintes de rigidité, mouvement décomposé en translation et rotation). Par exemple Gennery dans [Gen 82] cherche à suivre un objet qu'il connaît à l'avance par sa modélisation tridimensionnelle. La mise en correspondance est effectuée entre les indices image et la projection en deux dimensions de l'objet, mais l'estimation du mouvement est faite dans l'espace de la scène.

Harris ([Har 88]) quant à lui n'a aucune connaissance des objets mais effectue néanmoins son suivi sur la projection inverse en trois dimensions des indices image.

Lorsqu'un indice n'est connu que par une seule image, Harris lui suppose simplement une très grande incertitude en profondeur. il introduit donc une phase de mise en route, constituée d'au moins deux images, grâce à laquelle une estimation de la distance en profondeur d'indice peut être faite.

Ces deux études introduisent la notion de filtrage : Les valeurs observées étant bruitées et constituées d'une séquence d'images (plutôt que seulement deux images), on tient compte à la fois du mouvement précédent et des données actuelles pour estimer le mouvement présent et futur. Un filtre souvent utilisé est le filtre de Kalman : Il permet dans le cas de Harris d'estimer plus de valeurs (mouvement 3D) qu'il n'y en a d'observés (coordonnées 2D des indices).

Une utilisation typique de ce filtre est le travail effectué par Pocztar, Loopuyt et Maître pour la description cinématique d'une gerbe de particule ([Loo 86]). Les données sont fournies par deux caméras et la première consiste donc en une reconstruction 3D de type stéréovision pour obtenir des indices tridimensionnels à partir desquels est effectué le filtrage. Néanmoins la mise en correspondance temporelle est faite dans chacune des deux images par reprojection en deux dimensions des indices 3D.

Par rapport à la méthode "optic flow", le suivi d'indices utilise des informations de niveaux plus élevés, comme des contraintes provenant des caractéristiques de la scène observée. Cette méthode est donc moins sensible aux bruits dans les images. La principale difficulté est la mise en correspondance : Si il est logique d'utiliser le mouvement observé pour faciliter cette opération, il n'existe cependant pas de solution générale, comme nous l'avons déjà fait remarquer dans le cas de la stéréovision.

2.2 Indices visuels

Comme nous l'avons dit, un système de vision utilise plusieurs niveaux de représentations. Le niveau le plus bas après le niveau pixels est celui des indices d'images. Nous allons maintenant discuter de l'obtention de ce type d'indices et de quel genre d'informations il est souhaitable d'y trouver, sachant que tous les traitements de niveaux supérieurs en dépendent.

2.2.1 Quels types d'indices ?

Certains types d'opérations, effectuées sur l'image de pixel, ont pour effet de filtrer le bruit, augmenter le contraste ou modifier l'histogramme des valeurs d'intensités ([Bal 82]). Elles visent à améliorer l'aspect de ces images afin d'en faciliter l'interprétation par un humain, par exemple en médecine sur des radiographies, en météorologie sur

des images satellite. Elles fournissent néanmoins une idée de ce que l'on cherche dans un indice visuel : L'image étant le résultat de la projection d'une scène tridimensionnelle, on va chercher dans l'image la projection de ce qui permet de comprendre et modéliser cette scène : arêtes, surfaces des objets.

L'approche la plus classique est celle qui consiste à rechercher des contours dans l'image : Il est facile de faire le parallèle entre des discontinuités dans l'orientation des surfaces des objets et des discontinuités de l'intensité lumineuse. En fait comme le font remarquer Ballard et Brown, lorsque les surfaces des objets sont complexes, qu'il y a des ombres projetées, il est très difficile d'extraire les arêtes des objets directement à partir de l'image et il vaut mieux en extraire une image intermédiaire de discontinuités locales d'intensités. Même si cette image de "contours" ne correspond pas exactement à la projection des contours 3-D, elle en est très proche. De plus, par rapport à d'autres types d'indices, les contours sont assez stables et contiennent un grand nombre d'invariants : Points alignés en segments de droites, en segments de courbes, angles entre deux alignements, etc.

Nous allons donc développer ici les méthodes d'extraction de ce type d'indices, ainsi que les types d'informations pertinentes qu'il convient d'y rechercher.

2.2.2 Extraction des contours

De par la richesse des informations que l'on peut en tirer, l'extraction de contour a déjà une longue histoire et les méthodes en sont nombreuses. Nous en rappelons ici les grandes lignes ([Bal 82], [Aub 89]).

Le gradient étant une notion continue, Roberts a été le premier à en fournir une approximation discrète, sous la forme de deux quantités : La norme $s(x)$ et l'orientation $\phi(x)$ ([Bal 82]) :

$$s(x) = \sqrt{\Delta_1^2 + \Delta_2^2} \tag{2.6}$$

$$\phi(x) = \tan^{-1}(\frac{\Delta_2}{\Delta_1}) \tag{2.7}$$

Les fonctions Δ_1 et Δ_2 sont des opérateurs de différence de gradient dans deux directions orthogonales. Elles sont généralement obtenues en convoluant une partie de l'image autour du point x par une (des) matrice(s). De cette méthode découlent, en fonction du choix des matrices, les opérateurs de Roberts, Prewitt et Sobel.

Ces différents opérateurs sont assez sensibles au bruit qui affecte toute image. En cherchant à optimiser l'extraction de contours en présence de bruit, Canny introduit une méthode utilisant des convolutions entre l'intensité lumineuse et des gaussiennes et dérivés de gaussiennes ([Can 86]). Deriche améliore l'opérateur de

Canny en le rendant plus facilement exploitable par ordinateur et plus rapide à calculer ([Der 87a], [Der 87b] et [Der 87c]). On peut trouver une bonne présentation de ces différents opérateurs chez [Aub 89].

D'autres méthodes visent à calculer le laplacien de l'intensité lumineuse (mais alors il n'y a pas d'information de direction), ou utilisent des convolutions avec des laplaciens de gaussiens (qui au lieu de fournir le gradient, qui est donc maximum là où il y a un changement d'intensité, donne une fonction qui s'annule sur les discontinuités).

On voit que les transformations effectuées par les opérateurs que nous venons de décrire ne modifie pas la structure de l'image : Le gradient peut être vu comme une (ou deux) image(s) de pixels, et si les crêtes de gradient maximum sont extraites, ou si le gradient est seuillé, on obtient toujours une image de pixels, mais cette fois-ci binaire. Il reste donc à assembler et segmenter ces pixels pour obtenir de nouveaux types de structures : Chaînes, segments, courbes, etc.

2.2.3 Chaînage et segmentation

Le calcul du gradient, ou du Laplacien, n'étant pas réellement la finalité de l'extraction de contours, il a été proposé des méthodes de sélection d'ensembles de points selon certains critères. Ainsi la méthode de Hough appliquée à la détection de courbes : Pour chaque point de l'image vérifiant certaines contraintes sur le gradient (et éventuellement en utilisant la valeur de son orientation), on augmente dans une image duale représentant les courbes tous les "points-courbes" (points supportants la représentation d'une courbe. Ce peut être par exemple le centre d'un cercle de rayon donné) qui peuvent supporter ce point. A la fin du processus, des valeurs élevées dans l'image duale reflètent l'existence de tel ou tel type de courbe dans l'image originale ([Bal 82]).

Ballard et Brown présentent d'autres méthodes comme le suivi de crête vu comme une recherche dans un graphe, basée sur l'utilisation de l'orientation du gradient, ou en utilisant la programmation dynamique.

Les méthodes les plus classiques sont celles qui consistent à rechercher dans l'image un point appartenant à une crête assez sûre du gradient puis de suivre ensuite cette ligne de crête dans un sens puis dans l'autre. Pour une recherche exhaustive des chaînes de points, les crêtes déjà étudiées sont marquées pour éviter de repasser dessus. Certains auteurs se sont intéressés à l'extraction d'indices dans une image considérée comme un flot de données : L'image n'est vue qu'une fois par le processus qui fourni en sortie les indices recherchés.

Agin ([Agi 80], [Agi 82]) travaille sur une image binarisée. Il décrit un algorithme qui permet en lisant l'image ligne par ligne et pixel par pixel d'étiqueter les régions (blobs) de "1" et les trous de "0" à l'intérieurs de ces régions. En ajoutant des registres qui accumulent en parallèle certaines quantités, il décrit comment il peut calculer, le balayage étant terminé, les centres de gravité des régions, leurs surfaces, ou encore une ellipse englobante. Veillon ([Vei 78]) propose une approche similaire et étudie quels types de caractéristiques peuvent être extraites en parallèle : Périmètre, surface, axe principal, etc.

Ces deux méthodes font une reconstitution de région. Elles ne nécessitent pas de conserver la valeur des pixels pour le traitement d'un pixel donné. Pour effectuer un suivi de ligne, une partie des informations de l'image doit être conservée par le processus. Giraudon ([Gir 87a], [Gir 87b]) propose une solution mixte : Son but étant d'obtenir des chaînes complètes, à la limite des boucles, avec le moins de bruit possible, une partie du traitement se fera après le balayage de l'image. Le traitement en cours de balayage nécessite pour chaque pixel d'avoir conservé les huit pixels voisins, ce qui peut être fait en conservant une bande d'épaisseur trois pixels de l'image. Ce premier pas directement sur l'image fourni des morceaux de chaînes. Giraudon élimine ensuite les chaînes trop petites et essaye de lier les chaînes restantes entre elles jusqu'à obtenir les plus grandes chaînes possibles.

Indépendamment de ces travaux, Lux ([Lux 85]) propose des méthodes pour extraire des caractéristiques de type segment de manière incrémentale : Le système CAIMAN peut effectuer simultanément une évaluation du gradient, un chaînage de points et un calcul de droites ou de courbes. On peut voir cela comme un empilement de processus indépendant mais dont les plus hauts se "nourissent" des résultats fournis par ceux qui les précédent, ces résultats/données étant considérés comme un flot continu (pipeline).

2.3 Mesure du mouvement par suivi d'indices

Nous avons vu dans 2.1.3 que l'optique principale de l'approche "optic flow" était de mesurer le mouvement bi-dimensionnel dans l'image, et ce en partant directement des pixels de l'image, tandis que les travaux basés sur le suivi d'indices avaient pour but de modéliser le mouvement tri-dimensionnel dans la scène, et ce généralement à partir d'indices d'images de niveau assez élevé. Il n'y a donc pas de recoupement entre les deux méthodes, même si elles doivent aboutir au même résultat dans le cas idéal, à savoir une estimation du mouvement dans l'image qui soit bien la projection du mouvement 3-D existant dans la scène.

Il nous a donc semblé intéressant d'utiliser le suivi d'indices pour reconstruire une sorte de flot optique : Dans ce cas les vitesses seraient liées à la présence

d'indices, et on ne s'occuperait pas de son estimation dans les "trous" entre ces indices (là où justement les méthodes "optic flow" ne donnent de résultat qu'en utilisant des solutions pas toujours très satisfaisantes). D'autre part la recherche de mouvement dans l'image est une approche plus générale que celle de la recherche du mouvement dans l'espace : Il n'est plus nécessaire de contraindre la manière dont les objets observés sont sensés se déplacer. Par contre il faut trouver de nouvelles contraintes, 2-D cette fois-ci, qui permettent de mener à bien un suivi dans une image.

Nous allons donc maintenant étudier quelles sont les étapes du fonctionnement du suivi d'indices et les solutions généralement utilisées.

2.3.1 Nécessité d'une modélisation de l'image

La plupart des applications de suivi d'indices que nous avons étudiées utilisent une représentation interne des indices que nous appellerons modèle. En effet, lorsqu'on fait un suivi dans une séquence d'images, on ne veut pas se contenter de faire une mise en correspondance image par image entre les indices de l'image courante et ceux de la précédente. En particulier il est souhaitable que toutes les images déjà traitées influencent la manière d'effectuer les correspondances à un instant donné.

En utilisant un modèle, la mise en correspondance se fait entre les indices du modèle et ceux de l'image courante. Plus qu'une simple copie des indices des images, les indices du modèle possèdent des attributs dynamiques (vitesse, accélération, etc.) qui sont le reflet de leur comportement observé tout au long de la séquence.

Dans notre réalisation nous nous sommes largement inspiré des travaux de Crowley ([Cro 85]) pour une application de robotique mobile. Crowley utilise un modèle des obstacles (sous forme de segments de droites) observés par le capteur à ultrasons du robot. A chacun de ces déplacements, le robot compare ce modèle, qui a été modifié en fonction du déplacement, avec les obstacles qu'il observe à ce moment là.

Crowley introduit pour chacun des indices du modèle la notion de facteur de confiance. Cette technique, héritée du domaine de l'intelligence artificielle (technique de MYCIN) et des statistiques (technique de Bayes), associe à chaque indice une valeur qui représente la "confiance" que l'on a dans l'existence de cet indice. Chaque fois qu'il est mis en correspondance avec un indice d'une image (chaque fois qu'il est "vu") son facteur augmente, sinon il diminue.

De cette façon un indice modèle possède une existence plus large temporellement que son observation dans la séquence d'image, ce qui permet de s'affranchir

de certains problèmes de bruit dépendant de la méthode d'observation et assure une certaine robustesse vis à vis des problèmes de mise en correspondance.

2.3.2 Modélisation du mouvement

Comme nous venons de le dire, les indices du modèle possèdent en plus, par rapport aux indices d'images, des attributs dynamiques. Ces attributs permettent entre autre d'estimer à l'avance la position des indices dans les images qui vont suivre, facilitant ainsi la mise en correspondance. De nombreux auteurs ([Gen 82], [Har 88], [Loo 86]) utilisent pour cette gestion de la dynamique un filtre de Kalman.

Cette méthode est très polyvalente car elle permet d'estimer automatiquement des paramètres que l'on ne mesure pas dans les images. Nous rappelons Harris qui estime des valeurs 3-D directement à partir de mesures 2-D, mais les autres auteurs estiment par exemple les vitesses des indices directement à partir de la mesure de leurs positions.

La méthode du filtre de Kalman permet d'effectuer dans le cadre d'un suivi d'indices un certain nombre d'opérations et d'estimer de nombreuses valeurs utilisées par le processus ou qui fournissent les résultats :

- Les valeurs d'un indices du modèle ne sont pas simplement remplacées par celles de l'indice image lui correspondant. Un filtrage est effectué qui tient compte des précisions respectives de chacun des deux types d'indices, ce qui dans la plupart des cas permet d'éliminer une partie du bruit du capteur.

- Les valeurs qui ne sont pas mesurables directement sur une seule image et qui expriment la dynamique du modèle sont estimées automatiquement.

- Les paramètres dynamiques étant connus, les valeurs sont automatiquement mises à jour pour que les indices du modèle s'approchent le plus possible de ce que seront les indices des images suivantes.

- A chaque valeur est associée une incertitude qui permet la réalisation du premier point. Mais ces incertitudes peuvent également servir à calculer une tolérance utilisée lors du processus de mise en correspondance.

2.3.3 La mise en correspondance

La partie la plus délicate du suivi d'indices va donc être la mise en correspondance entre le modèle interne et l'image.

Nous avons vu que certaines méthodes de suivi utilisaient la corrélation entre une image observée et une "image" modèle ([Bur 82], [Gar]). Une bonne mise en correspondance est alors une relation qui maximise (ou minimise) une fonction de qualité (ou de coût). Mais quand on utilise une représentation des images hiérarchisée en indices, on souhaiterait que la recherche des correspondances soit aidée par cette hiérarchisation.

La façon la plus générale de considérer ce problème c'est de ramener la mise en correspondance à la recherche d'isomorphismes entre graphes. Cette façon de procéder est utilisée plus ou moins explicitement dans les technique de vision statique où le but est de reconnaître dans une image des objets déjà modélisés ([Aya 86], [Bol 86]).

Ballard et Brown ([Bal 82]) décrivent quatre types de correspondances entre graphes :

- Isomorphisme de graphes : Trouver une fonction qui relie totalement deux graphes entre eux.
- Isomorphisme de sous-graphes : Trouver une fonction qui relie un des graphes à un sous-graphe de l'autre graphe.
- "Double" isomorphisme de sous-graphes : Trouver une fonction qui ne relie "que" deux sous-graphes de graphes.
- Dans certains cas, si les noeuds des graphes n'ont pas tous la même importance, la fonction n'est pas vraiment un isomorphisme.

Dans la pratique on ne peut pas espérer trouver un isomorphisme de graphe entre le modèle et l'observation, c'est pourquoi on recherche plutôt un "double" isomorphisme de sous-graphe : Une partie seulement des indices du modèle correspondent effectivement à une partie seulement de ceux de l'image.

Ce "double" isomorphisme de sous-graphe est trouvé par la méthode des "cliques maximales". Certains auteurs comme Horaud et Skordas ont totalement exploité cette méthode ([Bol 86], [Hor 88], [Hor 89], [Sko 88]) et on peut résumer ainsi leur façon de procéder (figure 2.2) :

1. Possédant deux ensembles d'indices, on crée un ensemble d'hypothèses, une hypothèse étant constituée d'une paire d'indices. L'appariement ne se fait que si les paramètres des deux indices vérifient une fonction de ressemblance. On élimine ainsi un certain nombre d'appariement entre des indices manifestement différents.

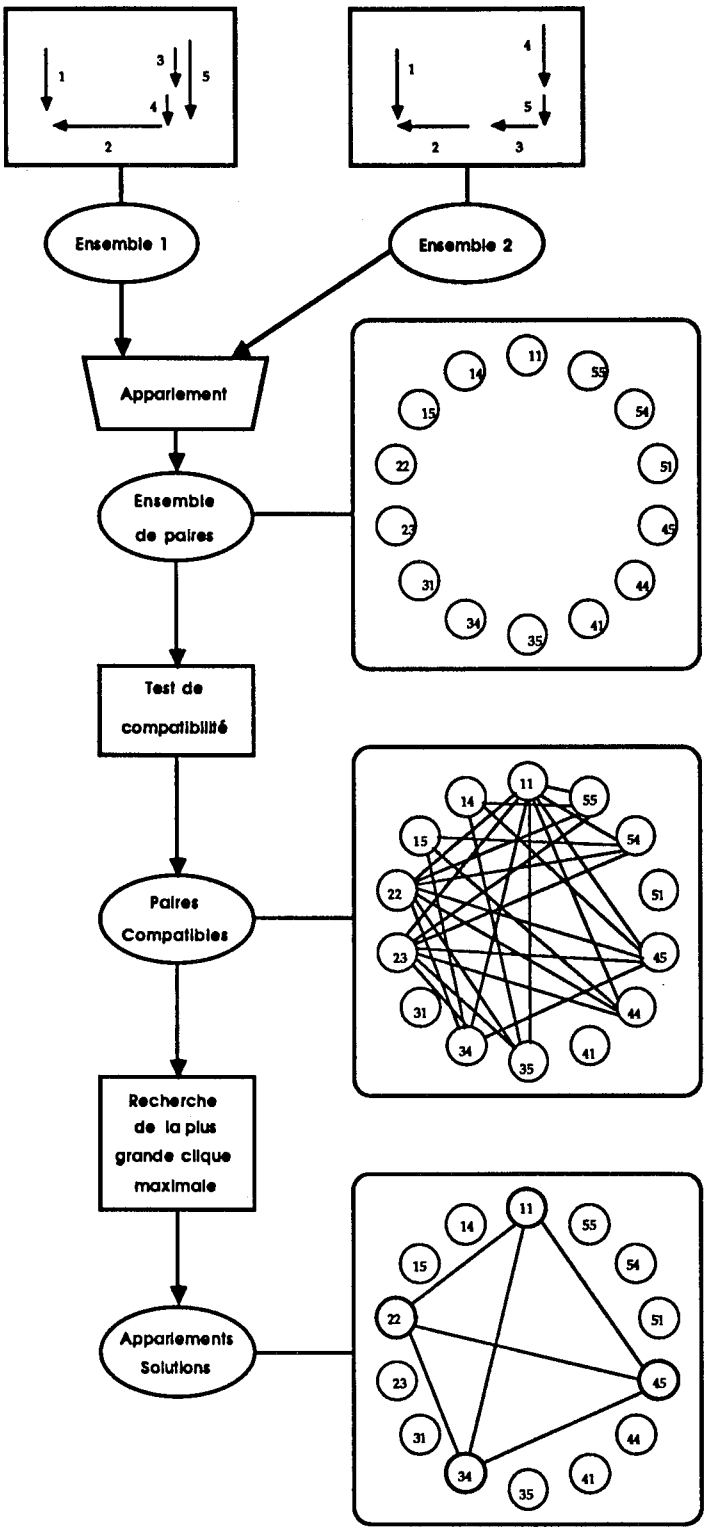


Figure 2.2: Processus de mise en correspondance par recherche de cliques maximale

2. Ensuite on construit des sous-ensembles de paires compatibles entre elles (on construit un graphe de compatibilité sur l'ensemble d'hypothèses).

3. Puis on ne garde que le plus grand sous-ensemble d'hypothèses (on recherche la plus grande clique maximale dans le graphe).

On voit que les parties **2** et **3** possédent une très grande combinatoire (si il y a beaucoup d'indices) et que l'on a intérêt à bien restreindre les conditions de correspondance entre deux indices dans la partie **1** pour limiter cette combinatoire. Selon Ballard et Brown la recherche de cliques maximales est exponentielle en fonction de la taille des graphes dans le pire cas. Cela peut rester satisfaisant dans le cas de la stéréovision mais peut être génant lorsqu'on fait du suivi d'indices où le temps est un des paramètres du système.

Pour le suivi, la position des indices est plus souvent exploitée que la structure des images. De ce fait la mise en correspondance est basée sur l'hypothèse que le processus d'évaluation des paramètres des indices du modèle fournit les meilleures estimations possibles.

On va ainsi dans un premier temps éliminer toutes les correspondances qui dépassent une certaine tolérance. Par exemple [Cro 85] compare les segments du modèle et ceux observés, estime leur différence en orientation et leur distance perpendiculaire puis vérifie que le segment image se trouve à l'intérieur d'une "boite" de tolérance autour du segment modèle.

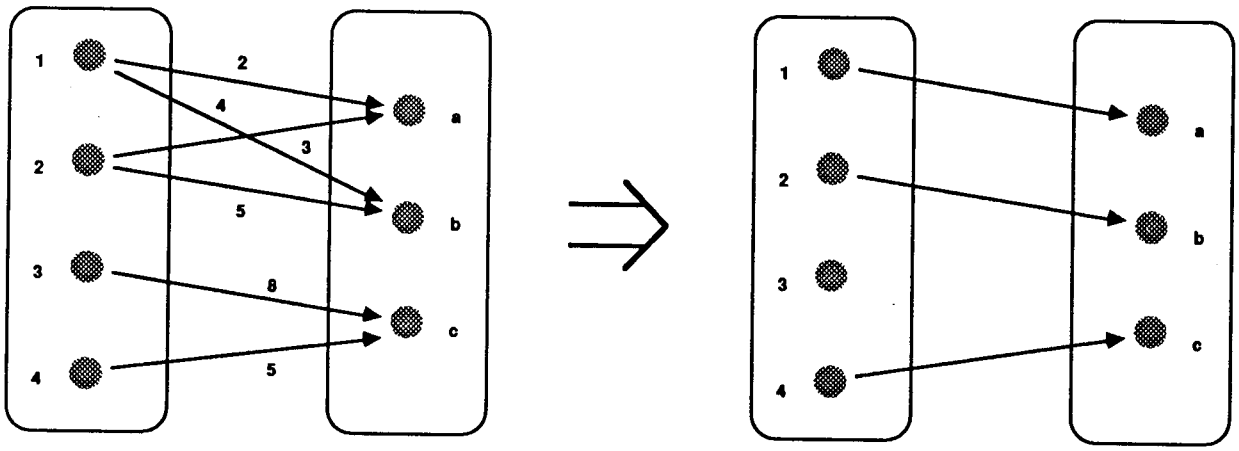


Figure 2.3: Exemple d'appariements après avoir calculé un coût en utilisant le "Greedy Algorithm"

Ensuite si il existe plusieurs correspondances possibles pour un même indice, on

va calculer une fonction de coût pour chacun des appariements et ne garder que le meilleur. Si il y a des appariements multiples aussi bien du coté du modèle que de celui de l'image, une simple élimination risque de faire perdre certains appariements. Il est alors pratique d'utiliser une méthode de "Greedy Algorithm" qui tout en favorisant les coûts faibles, permet de conserver ceux de coût élevé qui n'affaiblissent pas la mise en correspondance globale (voir la figure 2.3). Cette méthode fonctionne en deux passes :

- Chaque correspondance potentielle est rangée dans un tableau par ordre de coût croissant.
- Ensuite en partant du premier élément du tableau, on regarde si un des membres du couple à déjà été choisi, auquel cas cette hypothèse est rejetée, sinon ce couple est choisi.

Pour leur part, [Loo 86] utilisent une discrimination hors ligne : Chaque appariement multivoque est conservé et ils construisent tout au long du processus un arbre de suivi temporel. A la fin d'un suivi seule la branche la plus profonde d'un arbre est conservée (N'ont d'intérêt en fait que les branches qui permettent de fournir un chemin du début à la fin du suivi).

2.4 Reconstruction 3-D

Le mouvement, une fois extrait d'une séquence d'images, peut servir pour reconstruire une scène tri-dimensionnelle, dans le cas d'un suivi 2-D. Nous allons maintenant étudier quels sont les méthodes de modélisation 3-D et quelles informations peuvent être effectivement fournies par la connaissance du mouvement.

2.4.1 Quelles informations en fonction de quelles contraintes ?

La façon d'extraire certaines informations dépend de la façon dont on considère le mouvement. On peut classer en trois grands types les différentes méthodes :

- Caméra mobile, objets fixes : On trouve ici les mêmes contraintes que celles de la stéréovision.
- Caméra fixe, objets mobiles : C'est l'approche en général de la méthode "optic flow". Pour faciliter le passage en 3-D l'hypothèse de rigidité des objets est souvent employée.
- Caméra et objets mobiles.

Il n'est pas possible de connaître à la fois les coordonnées spatiales des objets et leur mouvement dans l'espace en observant leur mouvement en deux dimensions. De ce fait la plupart des méthodes de reconstruction font implicitement l'hypothèse de se trouver dans un des deux premiers cas de figure.

2.4.2 Suivi 3-D

Comme nous l'avons déjà vu, de nombreux auteurs font en réalité le suivi directement dans l'espace. Par exemple [Loo 86] utilisent une séquence de paires stéréo. Leur but est uniquement de connaître la trajectoire dans l'espace de particules assimilés à des points, ce qui permet de simplifier le modèle du mouvement des objets.

De leurs cotés [Har 88] et [Gen 82] utilisent une séquence monoculaire. Leur estimation 3-D est évaluée avec une très grande incertitude, qu'au cours du suivi il font diminuer jusqu'à ce que l'estimation possède une précision suffisante.

Un exemple de ce type de méthode est fourni par [Ram 89] : La projection inverse d'un point d'une image est sur une droite. Mais en faisant l'hypothèse que le point 3-D ne se trouve pas derrière la caméra ni plus loin qu'une distance arbitrairement choisie, on peut ramener l'espace de projection à un segment de droite. En représentant ce segment sous la forme d'un point associé à une distribution Gaussienne, on peut ainsi fusionner plusieurs projections du même point, diminuant ainsi la covariance de la distribution, et donc retrouver la position réelle.

2.4.3 Modélisation du mouvement 3-D

caméra mobile

Dans ce cas le problème est assez semblable à celui de la stéréovision : Le déplacement de la caméra est modélisé par une transformation composée d'une rotation et d'une translation.

Lorsque cette transformation est parfaitement connue, les coordonnées des points observés sont retrouvées en résolvant un système d'équations. Quand on possède plusieurs observations des mêmes points, ce qui est le cas dans une séquence d'images, ce type de système d'équations est résolu par une méthode au moindres carrés, ou un filtrage comme Kalman.

On peut remarquer que tout point suivi fournit deux équations (équations de la projection perspective) dans une image [Agg 88]. En multipliant les images on multiplie d'autant le nombre d'équations, tandis qu'un point n'est déterminé que

par trois inconnues qui sont ses coordonnées spatiales. En utilisant l'observation de suffisamment de points, on peut ainsi extraire des informations à la fois sur les coordonnées spatiales de ces points et sur les transformations géométriques qui ont amené la caméra d'une position à une autre.

Par exemple [Tos 87] proposent une méthode de calcul de la transformation entre deux images tenant compte du bruit dans la position des points observés. Ils calculent dans un premier temps la translation T et la rotation R par minimisation d'une fonction de coût, à partir desquels ils reconstruisent les points en trois dimensions. Puis en reprojetant les points dans les images et en les comparant avec leurs valeurs d'origine, ils affinent le calcul de T et R de manière à minimiser ces erreurs.

En modélisant la caméra par un torseur cinématique, [Riv 87] estiment récursivement (au moyen d'un filtre de Kalman) les coordonnées des indices observés (points et segments). De plus au cours du processus, ils commandent à la caméra des modifications de la trajectoire (commande en boucle fermée) de façon à optimiser la précision de cette estimation.

En marge de ces travaux, certains auteurs se sont penchés sur l'utilisation de l'accélération pour résoudre les équations du mouvement, en tablant sur le fait qu'il est plus facile de mesurer l'accélération d'un mouvement que sa vitesse [Arn].

Objets mobiles

Pour contraindre ce problème il est nécessaire de faire appel explicitement à la notion de rigidité des objets observés. [You 88] modélisent le mouvement d'un objet à l'aide d'une translation, d'une rotation et d'une précession de l'axe de rotation.

Chapitre 3

Un processus d'extraction de segments de droite

Nous avons développé dans le cadre de notre système une méthode d'extraction de segments de droite dans une image binaire, que nous allons décrire dans ce chapitre.

3.1 Analyse du problème

Nous avons vu dans 2.2 que les méthodes d'extraction de contours sont nombreuses et variées, et constituent en fait la base de la vision par ordinateur.

Si les méthodes d'extractions de points de contraste font souvent l'objet de réalisations en processeurs cablés, il n'en est pas de même des traitements visant à structurer cette information, comme le chaînage et la construction de segments de droites ou de courbes. Les études qui ont été poussées le plus loin dans cette direction nous semblent être les travaux de Giraudon [Gir 87a] [Gir 87b].

Nous avons donc pour notre part, en nous basant sur ces travaux et ceux de Lux [Lux 85], essayé de construire un processus de chaînage de points et d'extraction de segments de droites qui soit simple, rapide et si possible efficace.

3.1.1 Les segments de droite en tant qu'indices visuels

Les segments de droite héritent des caractéristiques des points les constituant. Les segments constitués d'alignements de points de contraste possèdent donc entre autre celles des points de contraste. Ils représentent souvent des arêtes des objets observés, et ont généralement une position stable sur les objets. Le seul problème est qu'ils peuvent également être générés par des ombres et dans ce cas ils ne sont pas fixes sur les objets.

Par rapport aux points les constituant, les segments de droite possèdent un plus grand nombre de caractéristiques permettant leur identification. Lors de leur utilisation dans un suivi d'indices, ces caractéristiques peuvent restreindre fortement les possiblités d'appariement d'une image à une autre et donc faciliter la mise en correspondance. Outre les caractéristiques d'identification propres aux points le constituant (comme par exemple la luminance ou le gradient de l'intensité lumineuse), le segment possède deux caractéristiques permettant de le décrire géométriquement : L'angle θ qu'il fait avec un des axes de l'image (en général avec l'horizontale) et sa longueur L (ou encore sa demi-longueur h_L).

L'angle est généralement stable lorsque l'on fait l'hypothèse d'une fréquence d'échantillonnage élevée des images par rapport au mouvement des objets observés. Elle est dans ce cas de toutes les caractéristiques attachées aux segments celle qui nous permettra le mieux de les différencier.

La longueur ne possède pas la même stabilité que l'angle. Les points de contraste situés aux extrémités, ayant un faible contraste, peuvent passer d'une image à une autre, en fonction de l'éclairage, en dessus ou en dessous du seuil au delà duquel ils sont pris en compte. Le processus d'extraction de points de contraste peut donc faire varier la longueur d'un segment. D'autre part le découpage d'une chaîne de points en segments peut fortement varier avec de faibles déplacements des points de la chaîne. On peut donc avoir dans deux images successives un segment qui en devient plusieurs petits ou le contraire. Le découpage polygonal peut donc créer d'encore plus grandes variations de longueur.

L'utilisation de la longueur comme aide à l'identification des segments est donc plus délicate que celle de l'angle. Mais en supposant une faible fréquence des incidents que nous venons d'exposer, elle permet d'améliorer le suivi de segments.

L'information de mouvement contenue dans le déplacement d'un segment varie en fonction de la direction. Comme nous venons de le remarquer, la longueur et la position des extrémités ne sont pas très fiables. On découpe donc ce mouvement en

deux parties : Une partie parallèle à la droite porteuse et une perpendiculaire. La partie parallèle étant la plus susceptible de ne pas être fiable, elle n'est généralement pas prise en compte dans l'estimation du mouvement, ou alors avec une très grande incertitude.

En se référant au problème de l'ouverture [Hil 84], on voit que seule la partie perpendiculaire de ce mouvement fournie une information pleinement exploitable. Comme cette information est incomplète, elle ne sera utilisable qu'en combinant plusieurs segments connus pour avoir le même . Par exemple en prenant deux segments de direction différente mais de déplacement similaire, on reconstitue la valeur du mouvement à partir des deux composantes perpendiculaires, avec une précision maximale lorsque les deux segments sont perpendiculaires. Plus généralement on utilise une combinaison filtrée du mouvement de plusieurs segments (comme par exemple une approximation aux moindres carrés). On peut aussi faire implicitement cette combinaison en étudiant le mouvement du point situé à l'intersection de deux segments consécutifs.

3.1.2 Etapes de construction des segments

Nous voulons extraire les segments de droite inclus dans des chaînes de points de contraste. Ce problème peut se décomposer de la façon suivante :

1. extraire des points de contraste parmi les pixels de l'image.
2. chaîner ces points lorsqu'ils sont adjacents.
3. extraire des segments de droite dans ces chaînes.

Nous n'allons pas développer ici l'extraction de points de contraste. Nous allons donc décrire un algorithme d'extraction de segments de droite, à partir d'une image binaire, qui correspondra aux deux autres parties.

Le chaînage sera réalisé en suivant un balayage simulant un balayage vidéo, les chaînes étant fournies en une seule scrutation de l'écran. Ceci permet éventuellement de faire un chaînage utilisant comme données ce que fournit le balayage, et donc de créer les chaînes à la vitesse d'apparition des images.

L'extracteur de segments quant à lui devra trouver ces segments parmi un flot de points chaînés. Ceci permettra d'utiliser cet extracteur comme un processus dépendant du processus de chaînage, et donc de fournir également les segments à la vitesse d'acquisition des images.

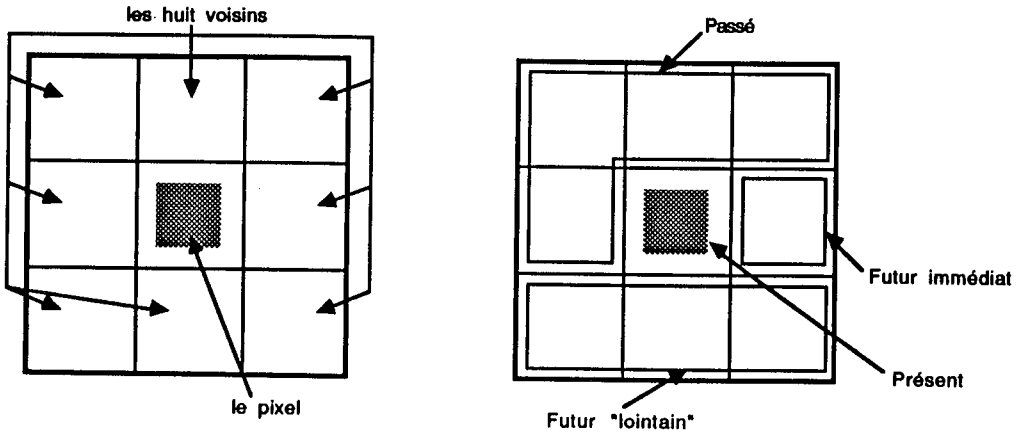


Figure 3.1: Voisinage d'un pixel

Nous allons maintenant développer les principes utilisés permettant de réaliser ces deux algorithmes en respectant les contraintes que nous venons d'établir. Puis nous décrirons plus précisément leur réalisation et leur fonctionnement.

3.1.3 Chainage de points de contraste

L'idée de base de ce chaînage est de ne considérer, pour un pixel donné, qu'un voisinage restreint. En balayant ligne par ligne, le voisinage situé sur les lignes adjacentes pourra être fourni par une sauvegarde de ces lignes effectuée lors de leur scrutation. Dans ce cas le traitement d'un pixel ne se fera pas simultanément au balayage mais avec un peu de retard. En limitant le voisinage on limite donc le nombre de lignes à conserver et le retard sur le balayage.

Nous basant sur les travaux de Giraudon [Gir 87a] [Gir 87b], nous avons choisi de faire fonctionner notre algorithme sur un voisinage de taille 3*3 : C'est la taille minimale, qui demande donc de conserver le moins de lignes en mémoire (trois lignes) et assure le retard minimal (retard d'une ligne). La figure 3.1 montre le découpage du voisinage d'un pixel :

- Le **passé** constitué des pixels qui ont déjà été traités.
- Le **présent**, c'est à dire le pixel en cours de traitement.
- Le **futur** : Les pixels qui n'ont pas encore été traités. Le futur est lui-même composé de deux parties :
  - Le **futur immédiat** qui est le pixel suivant immédiatement.
  - Le **futur "lointain"** regroupant les pixels de la ligne suivante et qui ne seront donc traités qu'une ligne plus tard.

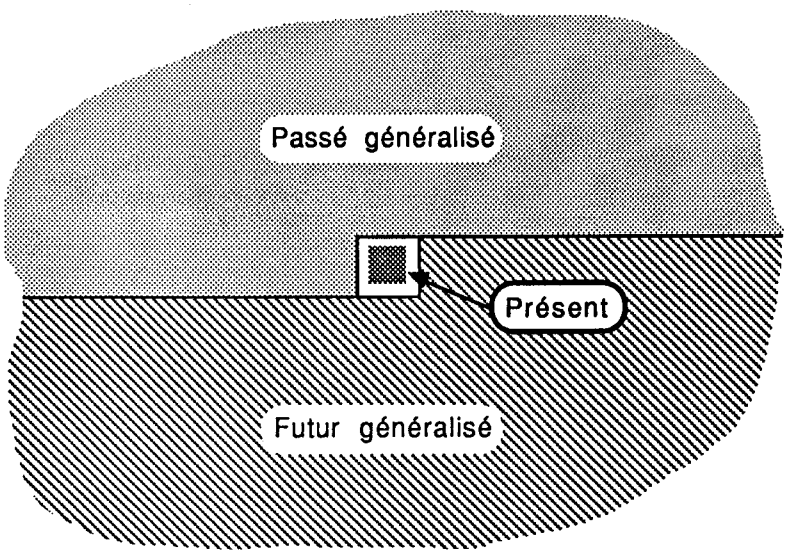


Figure 3.2: Voisinage généralisé d'un pixel

Notion de "processus chaîne"

En suivant les lignes de l'image pour chercher les chaînes, on peut voir que ces chaînes ne pourront pas être traitées une par une. Il va donc falloir mettre en oeuvre plusieurs traitements de chaînes simultanément à un instant donné dans l'algorithme. L'ensemble du processus sera constitué de plusieurs sous processus : un principal effectuant le balayage et gérant les autres qui sont ceux associés aux chaînes.

Avant de décrire plus précisément les processus chaîne, il convient de faire quelques remarques :

- Le processus principal ne pourra spécifier à un processus chaîne qu'il peut ajouter un point que si ce processus chaîne existe déjà. Si l'on considère le voisinage généralisé d'un pixel (voir la figure 3.2), à l'instant du traitement de ce pixel toutes les chaînes existantes sont constituées de points situés dans le passé généralisé. Il en résulte qu'une chaîne ne peut être constituée que de points rangés de haut en bas ou de gauche à droite.

- Une chaîne constituée de points ne vérifiant pas l'alignement de haut en bas ou de gauche à droite sera en fait constituée de plusieurs sous-chaînes (voir figure 3.3). La chaîne ne sera reconstituée qu'après la fin de l'algorithme. Si elle était reconstituée au fur et à mesure, les opérations à effectuer dépendraient du nombre de points dans les sous-chaînes, ce qui ne permet pas de garantir le suivi à la cadence du balayage.

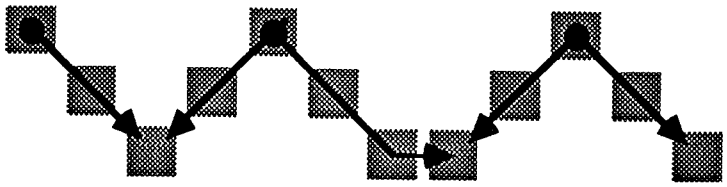


Figure 3.3: Découpage en sous chaînes d'une chaîne tordue

- Le but final étant de retrouver des segments de droite parmi des chaînes, il ne sera pas nécessaire de reconstituer une chaîne si l'on peut extraire un segment de plusieurs sous-chaînes.

Nous allons donc chercher à ne construire que des chaînes constituées de points vérifiant l'alignement précité, ce qui va permettre d'effectuer des manipulations de processus chaîne assez simples.

Manipulation des processus

Au cours du déroulement de l'algorithme, le processus principal va créer des processus chaîne et les ranger dans différentes listes : Toute nouvelle chaîne est rangée dans une liste de chaînes actives, et lorsqu'une chaîne existante se termine elle passe dans une deuxième liste de chaînes "fermées". A la terminaison de l'algorithme on ne doit plus avoir que des chaînes fermées.

Dans la liste active les chaînes sont rangées par ordre d'apparition au cours d'un balayage horizontal. De ce fait on peut accéder à la prochaine chaîne que l'on va renconter, revenant à la première à la fin de ce balayage. Lorsqu'une nouvelle chaîne apparait elle est directement placée dans la liste devant celle qui est attendue et lorsqu'une chaîne passe dans la liste fermée elle est simplement retirée de la liste active. En imposant que lors d'une intersection de chaînes elles soient toutes les deux fermées (avec éventuellement la création de nouvelles chaînes), les manipulations dans la liste active se limite à ces deux primitives : L'insertion à un endroit donné et l'extraction de l'objet pointé.

Chaque chaîne contient un certain nombre d'information, dont le moyen d'accéder aux points la constituant et des liens permettant de retrouver d'autres chaînes associées lorsqu'il y a une jonction entre plusieurs chaînes. D'autres informations peuvent être ajoutées dans la structure, mais elles ne seront pas utilisées lors de la manipulation des processus. Cette manipulation fera uniquement appel à l'étude du voisinage.

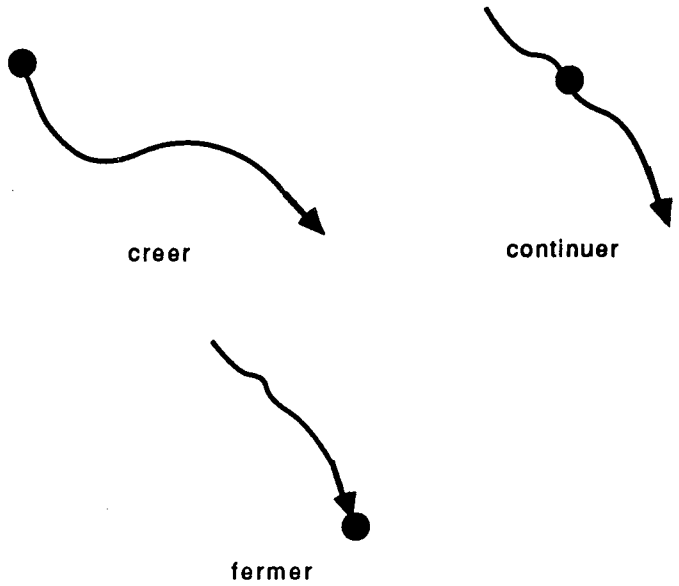


Figure 3.4: Opérations de base

Opérations à effectuer

Les opérations qui assurent la gestion des processus chaîne dépendent du voisinage du pixel. Chaque opération est la combinaison de cinq opérations de base (figure 3.4) :

1. Ne rien faire, ce qui arrive dans la plupart des cas. On passe alors directement au pixel suivant.
2. Création d'une chaîne : Une nouvelle chaîne est insérée dans la liste des chaînes actives. Nous appellerons cette opération *créer*.
3. Continuation d'une chaîne : Un nouveau point est ajouté à la chaîne. Nous l'appellerons *continuer*.
4. Fermeture d'une chaîne : La chaîne est retirée de la liste active et mise dans la liste fermée. Nous l'appellerons *fermer*.
5. Connexion entre deux chaînes : Cette opération n'a pas d'influence sur la liste active. Elle permet de créer des liens entre les chaînes que l'on peut ensuite utiliser au sortir de cet algorithme pour réaliser la fermeture de chaînes circulaires ou reconstruire celles ne vérifiant pas les conditions d'alignement des points.

La dernière opération n'est pas nécessaire au déroulement du processus et sert uniquement à fournir une information complémentaire dans le résultat. On peut donc concevoir un chaînage basé uniquement sur les quatre premières opérations.

Choix des opérations en fonction du voisinage

Remarquons d'abord que lorsqu'il n'y a pas de pixel dans le présent, il n'y a aucune action à envisager. En effet nous avons basé notre traitement d'opération en fonction du voisinage sur l'idée que le pixel présent est ajouté dans une ou plusieurs chaînes. Ayant posé cette condition étudions les cas les plus simples :

1. Lorsqu'il n'y a de pixel ni dans le passé ni dans le futur on ne fait rien : Un pixel isolé n'est pas considéré comme une chaîne et est donc ignoré.

2. Si il y a un pixel dans le futur, une nouvelle chaîne est créée et insérée dans la liste active. Cette chaîne est constituée à ce moment du pixel présent.

3. Quand il y a un pixel dans le passé et dans le futur, l'opération *continuer* est utilisée. Le pixel présent est ajouté à la chaîne courante.

4. Lorsqu'il n'y a qu'un pixel dans le passé c'est l'opération *fermer* qui est utilisée. Le processus chaîne courant est retiré de la liste active et mis dans la liste fermée, après avoir ajouté le pixel présent.

Ces quatre cas rassemblent quasiment tous ceux que l'on peut rencontrer dans une image de points de contraste. Il existe cependant des cas où deux, voire davantage de pixels se trouvent dans un des champ futur ou passé. Lorsque l'extracteur de points de contraste ne fonctionne pas correctement, ou lorsqu'on fournit une image binaire créée aléatoirement, le système peut se trouver confronté à des configurations totalement insolubles.

Comme nous avons voulu que notre algorithme effectue son chaînage dans tous les cas, il a fallu trouver des solutions simples permettant de traiter en partie ces configurations que l'on appelle des "pâtés". Néanmoins un certain nombre de cas ne relèvent pas du pâté et nous pouvons exprimer logiquement les opérations qu'il convient d'effectuer, en tenant compte des remarques que nous avons formulées sur l'ordonnancement des points d'une chaîne.

Lorsqu'il n'y a pas plus de deux pixels dans un des champs passé ou futur, et qu'aucun de ces pixels ne sont adjacents, nous sommes en présence d'une jonction de chaînes au pixel présent. Il va donc y avoir selon la configuration fermeture ou création de chaînes, le pixel présent appartenant à toutes les chaînes, et connexion entre toutes ces chaînes. Il existe cinq cas différents :

1. Deux pixels dans le passé, aucun dans le futur : Deux fermetures de chaînes.

2. Deux pixels dans le passé, un dans le futur : Deux fermetures et une création de chaînes.

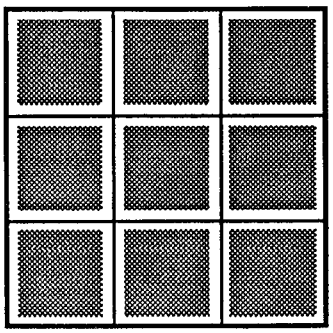


Figure 3.5: Le pâté

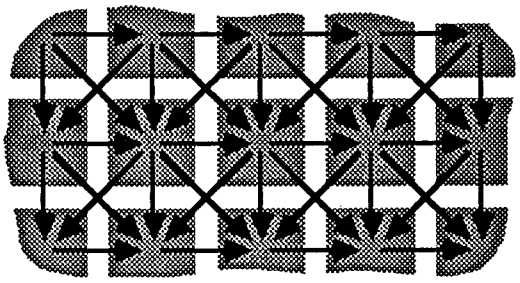


Figure 3.6: Effet théorique d'un pâté dans le chaînage

3. Deux pixels dans le passé et deux dans le futur : Deux fermetures et deux créations de chaînes.

4. Un pixel dans le passé et deux dans le futur : Une fermeture et deux créations de chaînes.

5. Deux pixels dans le futur, aucun dans le passé : Deux créations de chaînes.

On peut remarquer qu'il n'y a jamais de continuation de chaîne.

Il reste maintenant le problème constitué par la présence de plus de deux pixels dans un des champs. Pour cela étudions la configuration du "paté total" dans laquelle il y a des points partout (figure 3.5). Une solution serait de considérer chaque point indépendamment : Chaque pixel du passé indique la présence d'une chaîne qu'il faut fermer et chacun de ceux du futur une nouvelle chaîne à créer. Cette solution donnerait dans une zone de l'image où tous les pixels sont adjacents un ensemble de petites chaînes de deux points liant tous les pixels entre eux (voir figure 3.6).

Il faut quand même remarquer que :

- Certaines de ces petites chaînes se croisent.

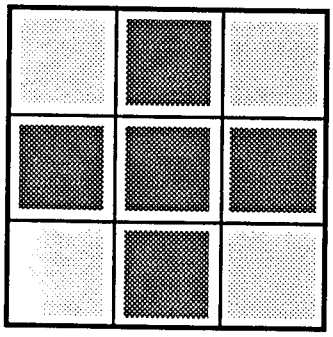


Figure 3.7: Comment nous considérons un pâté

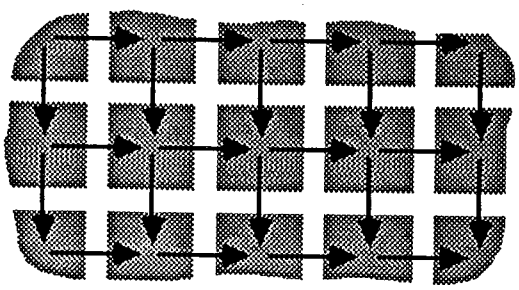


Figure 3.8: Effet du pâté tel que nous le considérons

- Des chaînes de deux pixels n'ont pas beaucoup d'intérêt; autant limiter leur nombre.
- Si dans une telle configuration il existe une chaîne virtuelle, elle ne peut être trouvée qu'en fonction d'un plus grand environnement que le voisinage 3*3. Dans notre voisinage il est impossible de retrouver cette chaîne et la configuration des petites chaînes, leur ordonnancement, n'a pas d'importance.

Ceci nous amène à privilégier les pixels dans certaines directions : Nous ne prendrons en considération que les pixels situés verticalement ou horizontalement par rapport au pixel présent, ignorant ceux situés sur les diagonales. Avec cette solution le paté devient équivalent à la configuration décrite dans la figure 3.7. Dans une zone d'image où tous les pixels sont adjacents, nous avons toujours les points liés par de petites chaînes mais en moins grand nombre (voir figure 3.8)

Nous avons généralisé cette solution à tous les types de voisinages. Lorsqu'il y a plus de deux pixels dans un champ, il y a en au moins un situé en diagonale et adjacent à un pixel situé horizontalement ou verticalement : Ce pixel est donc ignoré. Ceci reste vrai lorsqu'il n'y a pas plus de deux pixels par champs mais que deux pixels de champs différents sont adjacents : on élimine celui situé sur la diagonale (figure 3.9).

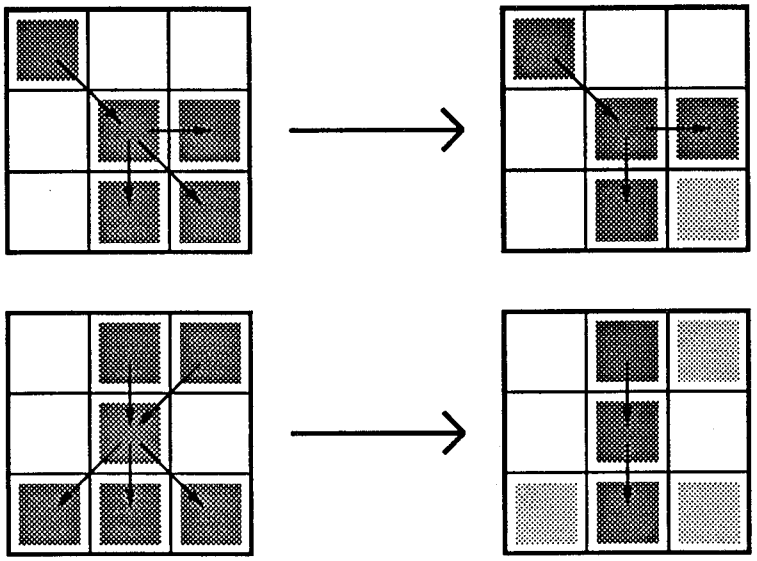


Figure 3.9: Deux exemples de modification du traitement dans les cas difficiles

Avec cette façon de procéder, certains cas qui aurait pu casser des chaînes ou en créer d'inutiles (figure 3.10) ne posent plus de problème et la continuité de l'alignement des pixels est assuré de manière élégante (figure 3.11).

Limites de cette méthode

Nous avons formulé tout un ensemble de méthodes pour résoudre notre problème initial visant à extraire des chaînes de points simultanément à un balayage de l'image. Pour chaque pixel nous avons un nombre restreint de choix qui dépendent uniquement de l'étude du voisinage. Le choix étant fixé, le nombre d'opérations à effectuer devient borné. Envisager de suivre la cadence du balayage vidéo semble donc raisonnable.

Cependant les résultats fournis ne seront pas directement exploitables. Les chaînes sont constituées de points respectant un certain alignement et peuvent engendrer le découpage d'une chaîne en plusieurs sous-chaînes comme dans la figure 3.3. Ce problème pourrait être résolu en rangeant les points dans une liste doublement chaînée : Le processus qui exploiterait le résultat du chaînage saurait, en fonction du type de connexion liant deux chaînes, dans quel sens lire les points. Notre but étant d'extraire des segments de droite, nous verrons plus tard comment ce problème peut être résolu.

En outre, l'algorithme proposé fonctionne sur une image binaire, faisant totale-

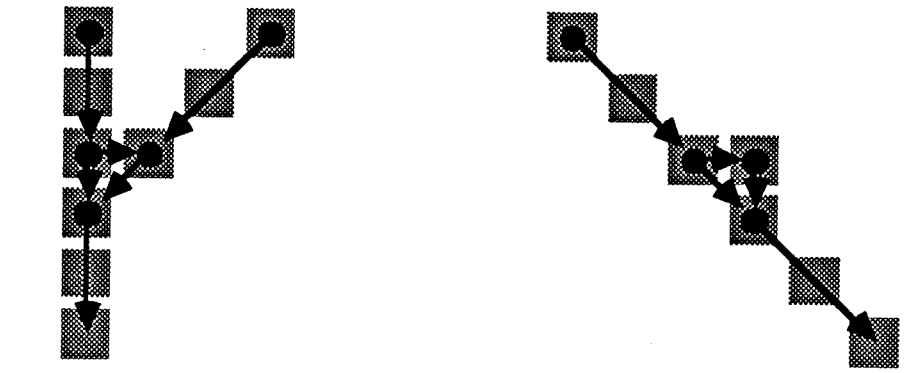


Figure 3.10: Mauvais découpage de chaînes

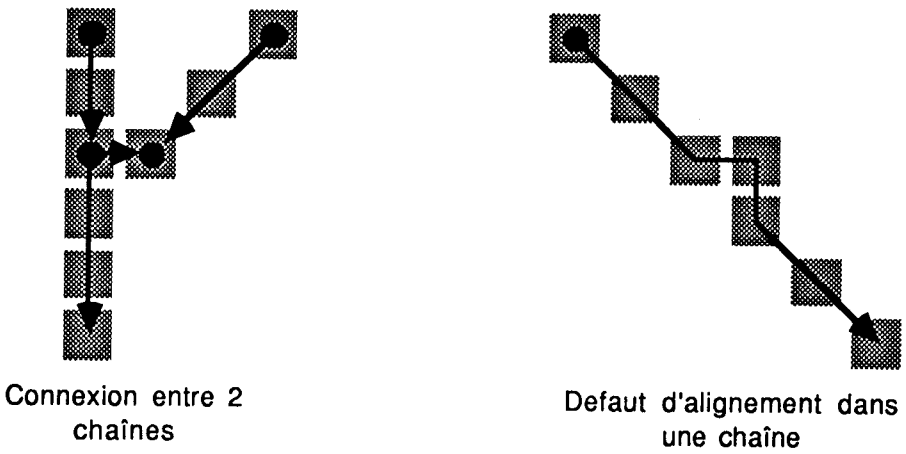


Figure 3.11: Découpage plus satisfaisant des chaînes

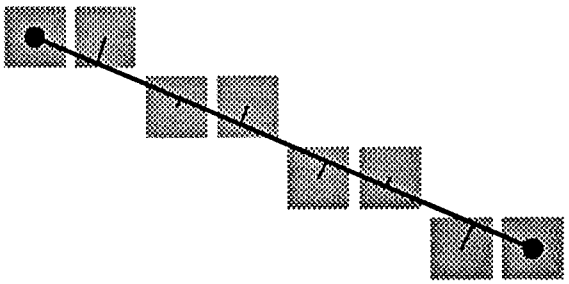


Figure 3.12: Effet de la discrétisation sur une ligne : Bien qu'alignés, les points ne sont pas situés sur la droite

ment abstraction de l'image numérisée dont peut provenir cette image binaire. Or il est souvent intéressant d'avoir des chaînes orientées en fonction de la direction du gradient lumineux, ce qu'il est impossible d'obtenir ici.

On voit donc que notre algorithme ne pourra pas résoudre tous les types de contraintes que l'on peut imposer à ce genre d'algorithme. Il est rapide mais n'est pas complet.

3.1.4 Recherche de segments de droites dans une chaîne de points

L'approximation polygonale d'une chaîne de points consiste à la représenter sous la forme d'un ensemble de segments de droite. Chaque segment de droite doit être défini de manière à ce que la distance perpendiculaire entre un point qu'il est censé recouvrir et sa droite porteuse vérifie la relation :

$$dist^{\perp}(P, \Delta) \leq \epsilon \tag{3.8}$$

Où ϵ est une tolérance d'erreur sur cette droite.

Cette tolérance est nécessaire ne serait-ce qu'à cause de la discrétisation des pi-xels qui empêche une suite de points d'être parfaitement alignée (figure 3.12). D'autre part une image est toujours bruitée, par le capteur d'abord (image vidéo), par la discrétisation de l'intensité lumineuse et par la technique d' extraction des points de contraste. L'utilisation de cette tolérance nous permet donc d'extraire des droites dans presque tous les cas, même à la limite sur une chaîne de points courbe.

Méthode récursive

La méthode d'approximation polygonale qui donne les meilleurs résultats est probablement le découpage récursif. Nous décrivons ci-dessous cet algorithme dans le

cas le plus général [Lux 85], sous la forme d'une procédure dont les paramètres sont les suivants :

- ϵ : La tolérance d'erreur pour une droite.
- *minpt* : Le nombre minimal de points constituant un segment (cela peut être aussi *minlo* la longueur minimale d'un segment et qui ne se calcule pas de la même façon).
- *chaîne*$[1, n]$: une chaîne de points à découper.

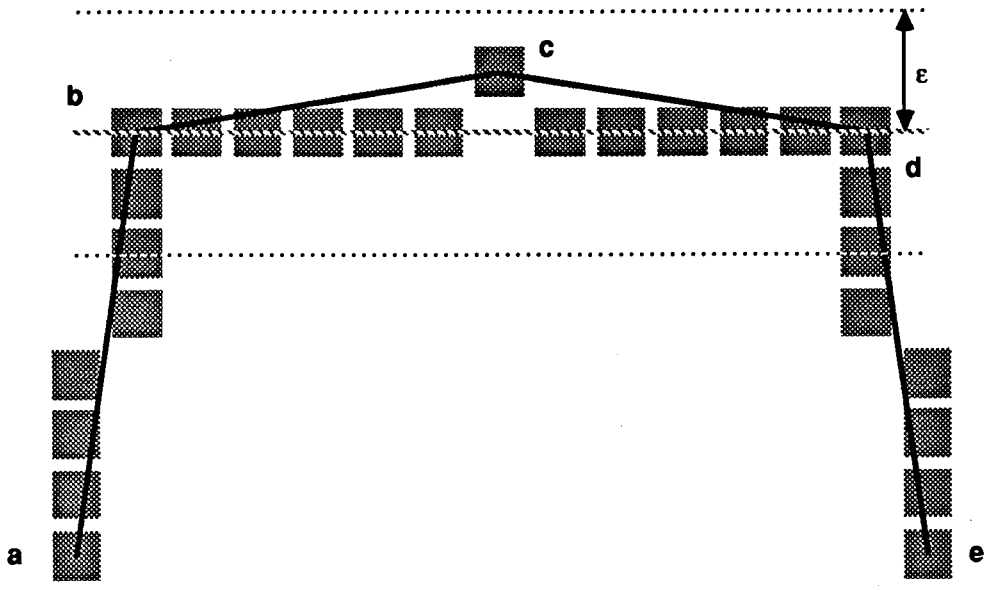
La description algorithmique de la procédure est la suivante :

procédure découpage_récursif $(\epsilon, minpt, chaîne[1, n])$
    Δ := la droite décrite par les deux points extrémités de la chaîne
    si $(\max(dist^{\perp}(\Delta, P_i)) \leq \epsilon)$ ou $(n \leq minpt)$
    alors
        la chaîne est un segment de droite porteuse Δ
    sinon
        j := l'indice vérifiant : $dist^{\perp}(\Delta, P_j) = \max(dist^{\perp}(\Delta, P_i))$
        découpage_récursif $(\epsilon, minpt, chaîne[1, j])$
        découpage_récursif $(\epsilon, minpt, chaîne[j, n])$

Le résultat fourni par la procédure peut être des droites définies par les points extrémités des segments ou obtenues approximation aux moindres carrés à partir des points des segments.

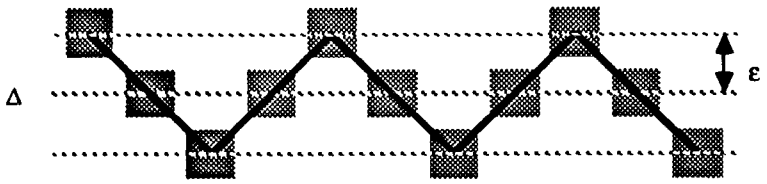
Cette procédure fournie le meilleur découpage possible sauf dans certains cas comme celui de la figure 3.13 où elle découpe inutilement un segment en deux, ou encore celui de la figure 3.14 où elle n'arrive pas à trouver la droite qui recouvre toute la chaîne. Ces problèmes ne sont pas trop génants si il importe peu qu'un long segment soit découpé en plusieurs petits. Ils sont d'autre part suffisamment peu fréquents et ne remettent pas en cause le bon comportement global de l'algorithme.

Le seul véritable problème vient de la récursivité : Il est impossible de borner le nombre d'opérations effectuées et la place mémoire utilisée. Ces deux paramètres dépendent du nombre de points dans la chaîne. Il n'est donc pas envisageable d'intégrer directement cette méthode au chaînage que nous avons développé dans la section précédente. Elle est par contre utilisable en sortie, tout en tenant compte de l'incomplétude du résultat du chaînage.



Le premier appel de découpage récursif coupe la chaîne au point c, alors qu'il y a une droite entre b et d.

Figure 3.13: Premier cas de mauvais découpage



Tous les points vérifient l'alignement sur la droite Δ mais l'initialisation du découpage récursif conduit à totalement découper le zig-zag

Figure 3.14: Deuxième cas de mauvais découpage (zig-zag)

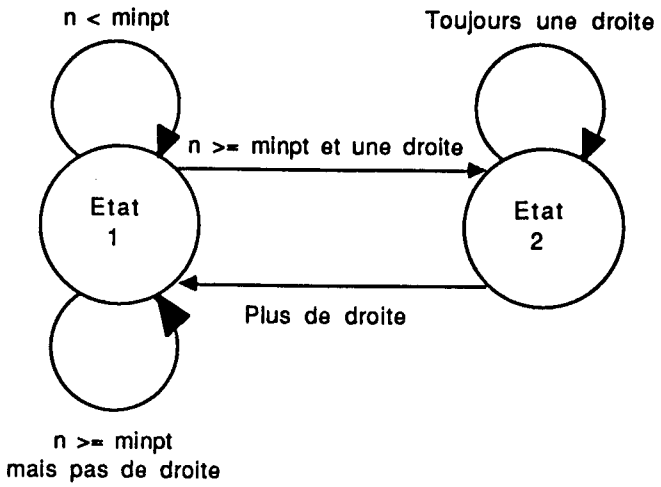


Figure 3.15: Automates d'extraction de segments de droite

Méthode itérative

Cette méthode fonctionne de la manière suivante [Lux 85] :

- Les paramètres sont les mêmes que pour le découpage récursif.
- On met en mémoire dans un petit vecteur les *minpt* premiers points de la chaîne. On vérifie que ce petit vecteur vérifie les conditions en faisant un segment de droite, à savoir :

$$\max(dist^{\perp}(\Delta, P_i)) \leq \epsilon$$

  Où Δ est la droite définie par les deux points extrémités ou par une approximation aux moindres carrés. Deux cas peuvent se produire :

  - L'inéquation n'est pas vérifiée, on réitère en partant du point de distance maximale.
  - L'inéquation est vérifiée, on essaye donc de prolonger la droite en ajoutant incrémentalement chacun des points de la chaîne. Lorsqu'un point sort de la droite, on recommence la recherche d'une droite en partant de ce point (ou du précédent si un point peut appartenir à deux segments à la fois).

On peut voir cet algorithme sous la forme d'un automate à deux états (figure 3.15) où chaque pas est constitué par l'acquisition d'un nouveau point de la chaîne par l'automate. De ce fait on peut imaginer d'intégrer cette méthode au chaînage dans lequel chaque processus chaîne reçoit ces points un par un.

Par rapport au découpage récursif cette méthode souffre d'un problème de manque de précision de la position des points de cassure. En effet lorsqu'on suit une droite, on ne détecte pas immédiatement une cassure mais seulement quand arrive un point éloigné de plus de ϵ de la droite. Ceci a pour conséquence que l'algorithme trouve des points de cassure "après" les points de cassure réels, "après" dépendant du sens de parcours de la chaîne.

Dans le découpage récursif la manière de calculer les droites porteuses, par les points extrémités ou par approximation aux moindres carrés, n'a pas d'influence sur la segmentation. Ici ce n'est plus le cas : L'estimation des points de cassure dépend de la droite en cours, donc de la façon de la calculer.

On voit donc que cette méthode ne donnera pas tout à fait d'aussi bons résultats que le découpage récursif. En choisissant bien les valeurs de ϵ et *minpt*, on peut néanmoins trouver une assez bonne segmentation. On peut également remarquer que dans le cas de la figure 3.13, la méthode itérative effectue le bon découpage, aux problèmes de localisation des points de cassure près.

Malgré ces quelques défauts, la méthode itérative sera celle que nous allons employer. Il est assez facile d'intégrer l'automate de la figure 3.15 dans un processus chaîne de notre chaînage : La taille *minpt* du vecteur de points étant fixée, nous pouvons borner le nombre d'opérations à effectuer lors de l'ajout d'un point. Il suffira d'autre part d'ajouter à une chaîne, en plus des données propres au chaînage, un vecteur de points de taille fixe et quelques variables servant à représenter l'équation d'une droite et l'état d'un automate.

Représentation des droites

Nous avons voulu dans tous ces algorithmes (le chaînage et l'extraction de segments) effectuer des calculs uniquement sur des entiers. Il a donc fallu pour l'extraction de segments de droite chercher une représentation de ces droites ne faisant appel qu'à des nombres entiers. Nous allons donc étudier les moyens d'y parvenir dans le cas où l'on définit une droite par les points extrémités d'un segment.

Une droite est généralement représentée par l'équation suivante :

$$P(x,y) \in \Delta \Leftrightarrow ax + by + c = 0 \tag{3.9}$$

Cette représentation avec trois paramètres permet d'éviter tout problème de valeurs limites en fonction de l'orientation de la droite. Une droite est définie par deux points et on peut entre autre appliquer cette équation aux points extrémités P_a et P_b :

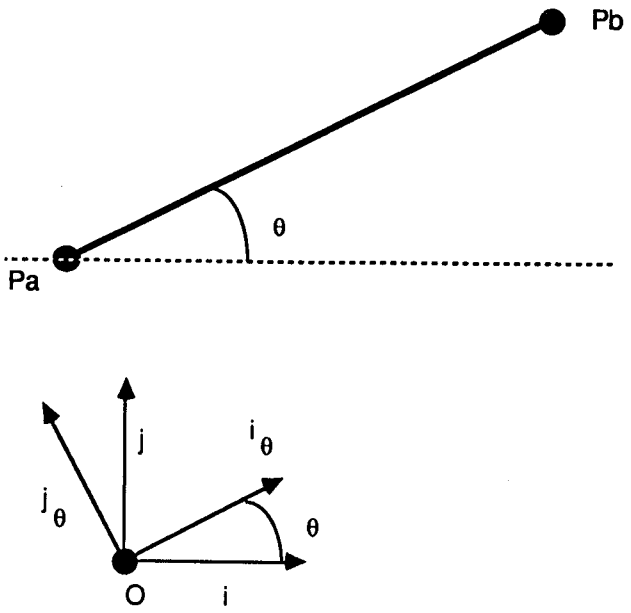


Figure 3.16: Repère de la droite

$$\begin{cases} ax_a + by_a + c = 0 \\ ax_b + by_b + c = 0 \end{cases} \tag{3.10}$$

On remarque que si les points P_a et P_b sont connus par leurs coordonnées en pixels dans l'image, les quatre nombres x_a, y_a, x_b et y_b sont entiers. Le système 3.10 est sous déterminé. Il peut donc se résoudre en choisissant une valeur arbitraire pour a, b ou c, les deux autres variables se déduisent de cette valeur. Nous avons :

$$3.10 \Rightarrow ax_a + by_a = ax_b + by_b$$

$$\Leftrightarrow b(y_a - y_b) = a(x_b - x_a) \tag{3.11}$$

Si l'on choisi pour a la valeur $y_a - y_b$, on en déduit donc les valeurs de b et c :

$$\begin{cases} a = y_a - y_b \\ b = x_b - x_a \\ c = x_a y_b - x_b y_a \end{cases} \tag{3.12}$$

Et les trois valeurs a, b et c sont entières.

Connaissant l'équation de la droite, il faut aussi pouvoir calculer la distance perpendiculaire d'un point à cette droite. Si on considère un repère $(O, \vec{i_\Delta}, \vec{j_\Delta})$ lié à la droite Δ (voir figure 3.16), les coordonnées x_Δ, y_Δ d'un point P de coordonnées x, y dans le repère d'origine sont données par les équations :

$$\begin{cases} x_\Delta = x\cos\theta_\Delta + y\sin\theta_\Delta \\ y_\Delta = -x\sin\theta_\Delta + y\cos\theta_\Delta \end{cases} \tag{3.13}$$

Et on a également les relations suivantes :

$$d = \| \vec{P_a P_b} \| \tag{3.14}$$

$$\cos\theta_\Delta = \frac{x_b - x_a}{d} = \frac{b}{d} \tag{3.15}$$

$$\sin\theta_\Delta = \frac{y_b - y_a}{d} = -\frac{a}{d} \tag{3.16}$$

La distance perpendiculaire $dist^\perp(P, \Delta)$ entre un point et la droite Δ est donnée par la différence des ordonnées dans le repère de la droite. L'ordonnée des points de la droite peut être obtenue à partir de P_a ou de P_b :

$$y_{a_\Delta} = y_{b_\Delta} = \frac{x_b y_a - x_a y_b}{d} = -\frac{c}{d} \tag{3.17}$$

On en déduit donc la distance perpendiculaire :

$$dist^\perp(P, \Delta) = \sqrt{(y_\Delta + \frac{c}{d})^2} = \frac{\sqrt{(ax + by + c)^2}}{d} \tag{3.18}$$

Lors du découpage, le calcul de la distance perpendiculaire n'est utilisé qu'à l'intérieur d'une inéquation pour vérifier si cette distance est bien inférieure à la valeur ϵ, comme nous l'avons formulé dans 3.8. On peut réécrire cette inéquation d'une autre manière :

$$3.8 \Leftrightarrow (dist^\perp(P, \Delta))^2 \leq \epsilon^2$$

$$\Leftrightarrow (ax + by + c)^2 \leq (d\epsilon)^2 ou \mid ax + by + c \mid \leq \mid d\epsilon \mid \tag{3.19}$$

Nous allons donc à l'intérieur de la structure d'une droite conserver une valeur $deps2 = (d\epsilon)^2$ et pour savoir si un point est suffisamment proche de la droite nous regarderons si il vérifie l'inéquation 3.19. Dans ce cas encore on s'aperçoit que tous les calculs peuvent être effectués sur des entiers.

Principal défaut du découpage itératif

Le problème du découpage itératif est qu'il a une vue limitée à la taille de son vecteur de la chaîne de points. Ce n'est pas le cas avec le découpage récursif qui lui part avec une vue globale de cette chaîne.

Par exemple dans le cas (figure 3.17) où la droite est presque horizontale (lorsque la pente θ de la droite est telle que $\mid \tan\theta \mid < \frac{1}{minpt}$) ou qu'elle est presque verticale

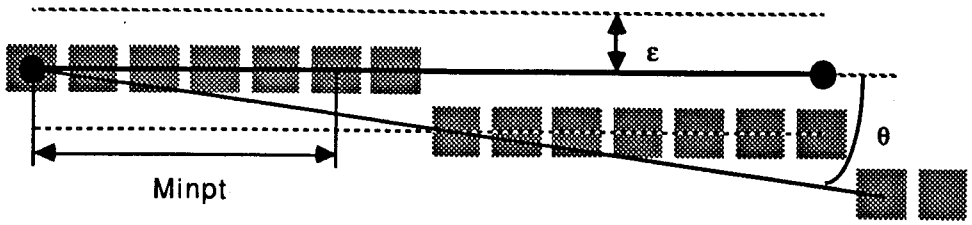


Figure 3.17: Ligne alignée sur une droite de pente trop faible

(lorsque $\mid \tan\theta \mid > minpt$), la droite sera supposée être parfaitement horizontale (ou verticale selon le cas). Lors de la poursuite de l'algorithme, un point finira par dépasser en distance perpendiculaire la valeur ϵ et le segment de droite sera arbitrairement coupé.

On voit donc que pour éviter ce problème la solution est d'agrandir au maximum la taille du vecteur. Mais dans ce cas il est impossible de trouver un segment de longueur inférieure à *minpt*. La valeur de *minpt* ne peut donc être qu'un compromis entre la taille minimale des segments de droite et le besoin d'obtenir des segments de la plus grande taille possible.

Sortie fournie par l'approximation polygonale

Comme nous voulons faire des calculs uniquement sur des entiers, les résultats fournis seront aussi de type entier. Lorsqu'un segment de droite se termine, soit par cassure, soit par terminaison de la chaîne de points, l'information que nous possédons sur lui est la droite porteuse et les deux points extrémités. Si le premier point est effectivement sur la droite porteuse (par définition de cette droite) ce n'est pas le cas du dernier. Plutôt que de calculer la projection de ce dernier point sur la droite, nous avons préféré simplement sortir les deux points extrémités pour représenter le segment.

Lorsqu'une chaîne de points est constituée de plusieurs segments concaténés, cette solution permet à chaque segment adjacent de partager un point extrémité. Par contre, il faut remarquer que si la droite porteuse d'un segment à été calculée avec un précision de ϵ, la droite qui en résulte n'a plus qu'une précision de 2ϵ (figure 3.18).

Problèmes liés au chaînage

Nous avons vu dans la partie 3.1.3 traitant du chaînage de points que l'algorithme que nous proposons ne fournit pas toujours des résultats totalement satisfaisants.

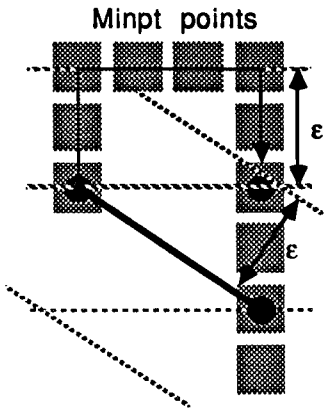


Figure 3.18: Précision réelle d'une droite en sortie

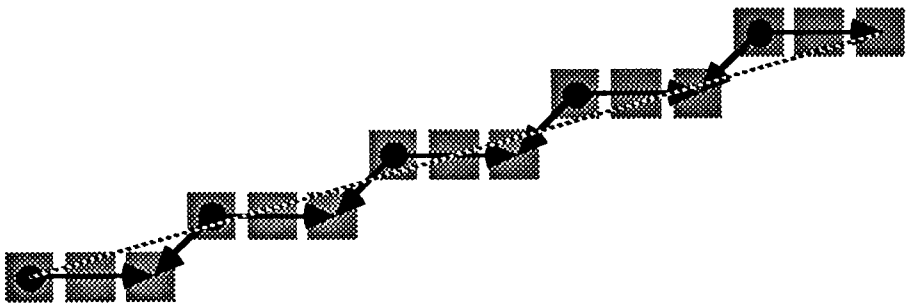


Figure 3.19: Chaîne de points alignés dans le mauvais sens

Ceci est vrai dans un cas comme celui de la figure 3.3 mais plus encore dans celui de la figure 3.19 où l'on reconnait parfaitement une droite.

Dans des configurations comme celles-ci, la liaison entre un processus chaîne et un processus droite ne sera pas suffisante. On voit clairement sur la figure 3.19 que le processus droite est attaché à plusieurs processus chaîne.

Nous verrons dans la partie réalisation algorithmique comment ces problèmes peuvent être résolus en approfondissant la fonction des opérations de jonction entre chaînes.

3.2 Réalisation algorithmique

Nous venons de présenter comment nous envisageons de réaliser le chaînage de points de contraste et l'extraction de segments de droite sur ces chaînes de points. Nous allons maintenant décrire les algorithmes tels qu'ils ont été implémentés. Nous décrivons d'abord plus en détail le chaînage. Puis cet algorithme étant conçu, comment l'extraction de segments de droite s'inscrit à l'intérieur du chaînage pour ne faire qu'un seul algorithme combinant les deux traitements.

3.2.1 Réalisation du chaînage

Représentation des chaînes

Lors de leur utilisation, les processus chaînes se trouveront dans une liste simplement chaînée. En plus de l'information nécessaire à la réalisation de ce chaînage (un pointeur) nous conserverons dans chaque chaîne les paramètres suivants :

- Les coordonnées x_1 et y_1 du premier point de la chaîne.
- Les coordonnées x_n et y_n du dernier point courant de la chaîne.
- Deux pointeurs permettant d'accéder au premier et au dernier point courant de la chaîne. Tous les points d'une chaîne sont rangés en liste simplement chaînée. Le premier pointeur permet d'accéder à la liste, le second permet d'ajouter facilement un point en queue de liste.
- Deux ensembles de trois pointeurs permettant d'accéder aux chaînes qui sont jointes. On a vu qu'à un point de jonction il pouvait y avoir au maximum quatre chaînes. Donc pour chaque chaîne nous devons prévoir qu'elle peut être jointe à trois autres chaînes au début, et trois à la fin.

Les pointeurs d'accès aux chaînes de points ne servent que lorsqu'on veut explicitement extraire ces chaînes. Lorsque l'extracteur de segments de droite sera intégré, les chaînes n'auront plus besoin d'être conservées et ces paramètres pourront être supprimés.

Description des opérations

Au cours du balayage de l'image, nous allons balayer parallèlement la liste de chaînes actives : Nous avons un pointeur courant qui pointe sur la prochaîne chaîne à rencontrer ou en cours de traitement, et un autre pointant sur la précédente, ceci afin de faciliter la manipulation des éléments de la liste. L'évolution de ces deux pointeurs n'est controlée que par le voisinage de chacun des pixels rencontrés. En fonction de ce voisinage la chaîne courante sera retirée, une chaîne sera insérée entre les deux actuellement pointées ou le processus chaîne courant évoluera et les pointeurs avanceront dans la liste.

Ceci nous amène à faire deux remarques sur la manipulation des pointeurs au cours du traitement d'une chaîne :

- Si le voisinage indique que la chaîne va se prolonger dans le futur "lointain", les pointeurs peuvent évoluer afin d'aller chercher la chaîne prochaînement rencontrée.
- Si par contre le voisinage indique que la chaîne va se poursuivre dans le futur immédiat, se sera toujours la même chaîne qui va être traitée au pas suivant, et les pointeurs doivent donc rester figés.

Dans notre réalisation nous avons donc utilisé deux types d'opérations *créer* (*créer1* et *créer2*) et deux types d'opérations *continuer* (*continuer1* et *continuer2*) : *créer1* et *continuer1* dans le cas où la chaîne se prolonge dans le futur immédiat et *créer2* et *continuer2* dans celui où la chaîne se prolonge dans le futur "lointain".

Ces opérations étant définies, nous pouvons détailler comment elles seront réalisées en fonction du voisinage. Nous introduisons pour cela une fonction *ajout* qui en utilisant les pointeurs de liste de points ajoute un point à une chaîne. Cette fonction servira ultérieurement à alimenter les automates d'extraction des segments de droite. Au départ d'un balayage, le pointeur courant se trouve sur la première chaîne de la liste (le pointeur précédent ne pointe lui sur rien).

La liste des actions se résume à :

1. **Créer1** : Une nouvelle chaîne est créée à laquelle est ajouté le point courant. Elle est placée entre la chaîne précédente et la chaîne courante, le pointeur courant continue à pointer sur elle.

2. **Créer2** : Comme *créer1*, mais la nouvelle chaîne devient la chaîne précédente tandis que le pointeur courant va sur la chaîne suivante (anciennement courante).

3. **Continuer1** : Le point courant est ajouté à la chaîne courante, les pointeurs ne sont pas modifiés.

4. **Continuer2** : Dans ce cas les pointeurs avancent jusqu'à la chaîne suivante.

5. **Fermer** : Le point courant est ajouté à la chaîne courante qui est retirée de la liste. Le pointeur précédent n'est pas modifié mais le courant va pointer sur la chaîne suivante.

Lors d'opérations plus complexes, ces six premières opérations servent de bases. Dans les autres cas nous sommes en présence de connexions entre chaînes, d'où leur appellations de *jonctions*. Nous avons dénombré neuf *jonctions* (appelées *jonction1* à *jonction9*) dont les configurations sont données dans la figure 3.20 et que nous décrivons sous la forme de combinaisons entre les opérations de base. L'ordre dans lequel elles s'effectuent nous garantit une manipulation correcte des processus chaînes dans la liste :

1. **jonction1** : *fermer*
   fermer
   connexion entre les deux chaînes

2. **jonction2** : *fermer*
   fermer
   creer1
   connexion entre les trois chaînes

3. **jonction3** : *fermer*
   fermer
   creer2
   connexion entre les trois chaînes

4. **jonction4** : *fermer*
   creer2
   creer1
   connexion entre les trois chaînes

5. **jonction5** : *fermer*
   creer2
   creer2
   connexion entre les trois chaînes

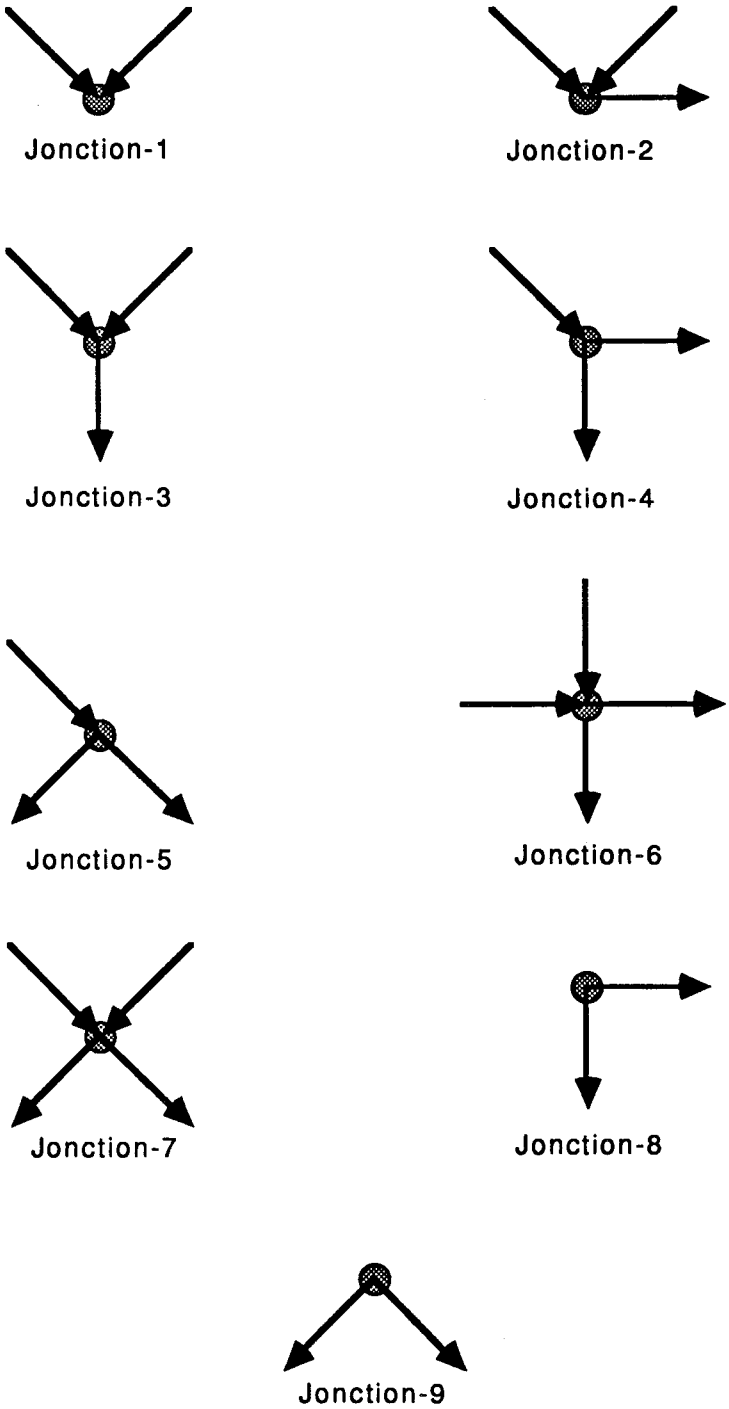


Figure 3.20: Jonctions complexes

6. **jonction6 :** *fermer*
   fermer
   creer2
   creer1
   connexion entre les quatre chaînes

7. **jonction7 :** *fermer*
   fermer
   creer2
   creer2
   connexion entre les quatre chaînes

8. **jonction8 :** *creer2*
   creer1
   connexion entre les deux chaînes

9. **jonction9 :** *creer2*
   creer2
   connexion entre les deux chaînes

Voisinages et opérations

A chaque pixel traité, l'étude du voisinage permet de décider le type d'opération qui doit être effectuée, comme nous l'avons vu dans 3.1.3.

On pourrait pour cela utiliser un algorithme qui ayant le voisinage comme donnée trouverait la configuration. Mais il faut remarquer que le voisinage d'un pixel est constitué de 8 points de valeurs binaires. Le voisinage est donc un vecteur de 8 chiffres binaires pouvant prendre 256 valeurs différentes. Nous avons donc eu l'idée d'utiliser un tableau de 256 éléments : Chacun des éléments du tableau contient un numéro codant une des opérations que nous venons de décrire, et un élément est adressé par la valeur du vecteur de voisinage. De cette façon l'accès aux opérations est instantané pour un voisinage donné.

Les valeurs du tableau sont rentrées une fois pour toute à la main ou par un programme d'analyse du voisinage.

Déroulement du processus

Nous pouvons maintenant décrire le déroulement du processus de chaînage : L'image étant découpée en lignes, le balayage de ces lignes se fait simultanément avec un balayage d'une liste de chaîne en cours de construction.

Au départ de ce balayage il n'y a aucune chaîne. Par la suite chaque nouvelle chaîne créée est insérée dans la liste des chaînes à une place reflétant son emplacement dans l'image. Chaque chaîne qui se termine est retirée de cette liste puis sauvegardée. A la fin du balayage de l'image, toutes les chaînes auront été fermées, et le résultat du chaînage sera donc constitué de toutes les chaînes sauvegardées au cours du traitement.

3.2.2 Réalisation de l'extraction de segments et intégration dans le chaînage. Première approche simple : Un processus chaîne = un automate d'extraction de segments.

Comme nous l'avons évoqué dans 3.1.4 nous allons intégrer l'automate d'extraction de segments dans chaque processus chaîne.

Dans ce cas nous ne conservons plus les points de la chaîne, mais la liste des segments inclus dans cette chaîne. Il sera néanmoins nécessaire de conserver un certain nombre de points pour obtenir le vecteur de recherche de droite. En s'imposant de prendre une petite taille *minpt* pour ce vecteur, nous l'intégrons dans la structure du processus chaîne. Pour faciliter sa manipulation ce vecteur est organisé en *pile FIFO*, ce qui permet d'ajouter continuellement de nouveaux points.

Les informations contenues dans un processus chaîne sont maintenant les suivantes :

- Les coordonnées x_1 et y_1 du premier point du segment en cours de recherche ou de continuation d'extraction.
- Les coordonnées x_n et y_n du dernier point courant.
- Deux pointeurs permettant d'accéder à la liste de segments de droite.
- Les deux ensembles de trois pointeurs permettant d'accéder aux chaînes jointes.
- *etat* l'état de l'automate d'extraction de segments.
- n le nombre de points dans le segment en cours.
- a, b, c, *deps*2 les paramètres de représentation de la droite courante.

- Un vecteur d'au moins *minpt* points avec les pointeurs permettant de le gérer en *pile FIFO.*

L'évolution de l'automate se fera à chaque *ajout* de point dans le processus chaîne. Le fonctionnement de cet automate est détaillé dans la figure 3.21.

A l'initialisation d'une chaîne, c'est à dire au cours de l'opération *créer*, l'automate démarre dans l'état 1. A la fermeture d'une chaîne, au cours de l'opération *fermer*, si un segment de droite est en cours de prolongement, il est terminé et sauvé. Dans ce cas ce n'est pas l'opération *ajout* qui effectue la fermeture mais l'opération *fermer.*

3.2.3 Intégration de l'automate dans plusieurs chaînes

Dans la partie 3.1.4 nous avions déjà signalé que dans certains cas (par exemple celui de la figure 3.19), un automate de recherche de segments intégré à un seul processus chaîne n'était pas en mesure de retrouver complètement une droite.

Dans de telles configurations il faudrait pouvoir faire correspondre un automate à plusieurs processus chaîne différents. Nous avons préféré considérer cette liaison entre un automate et une chaîne, mais approfondir les relations que peuvent avoir les chaînes entre elles, et donc créer des relations entre les automates.

Cas de deux chaînes partant du même point

Nous allons dans un premier temps nous intéresser au cas correspondant aux opérations *jonction8* et *jonction9* dans lesquelles sont créées deux chaînes ayant leur premier point en commun.

Dans l'algorithme d'extraction de segments de droite tel que nous l'avons décrit jusqu'ici, chacune des deux chaînes aurait cherché son propre segment indépendamment l'une de l'autre et ce malgré la présence d'un point commun. Nous allons donc dans pareille configuration lier les deux chaînes entre elles en leur donnant un état "lié" et en leur assignant un pointeur permettant à chacune d'elle d'adresser l'autre. On permet par la même occasion à l'automate d'une chaîne de faire évoluer l'automate de l'autre chaîne.

Les nouvelles informations contenues dans un processus chaîne sont donc :

- *etatll* qui permet de savoir si une chaîne est liée à une autre ou pas.
- *lié* un pointeur sur l'autre chaîne dans le cas d'une liaison.

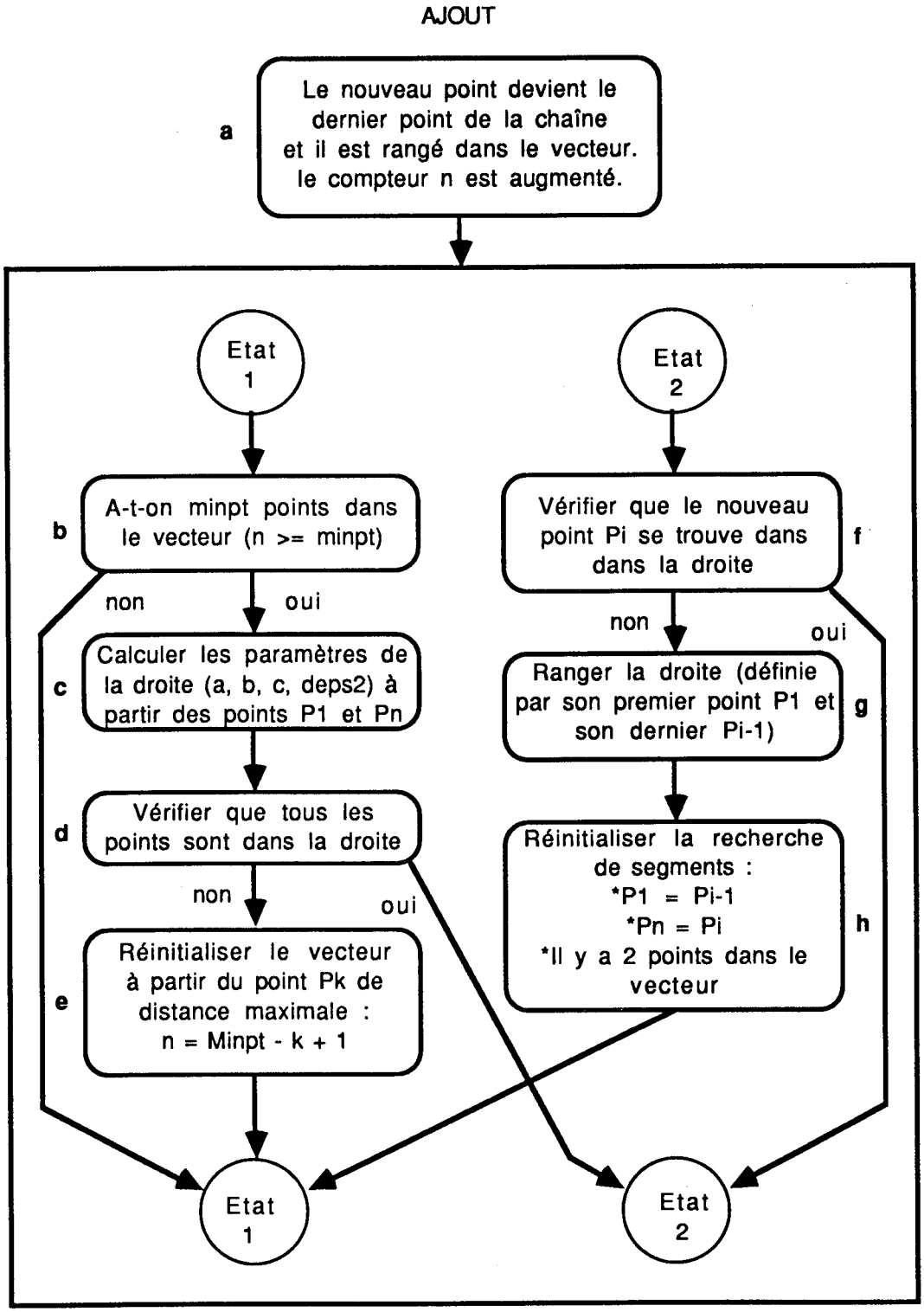


Figure 3.21: Description de l'automate à l'intérieur de l'opération *ajout*

- n le compteur du nombre de points dans le segment en cours est séparé en deux :
  - t_n le compteur du nombre de points total des deux chaînes.
  - l_n le compteur du nombre de points appartenant seulement à cette chaîne.

Lorsqu'un point est ajouté à une chaîne, deux cas peuvent se produire : Soit cette chaîne n'est liée à aucune autre et nous avons toujours l'évolution décrite dans la figure 3.21, soit elle est liée à une autre chaîne et nous avons alors l'évolution décrite dans la figure 3.22. Dans ce dernier cas les deux automates fonctionnent ensemble, c'est à dire que la modification de l'état de l'un implique la modification immédiate dans le même sens de l'état de l'autre. Mais à partir du moment où les deux chaînes sont dissociées, les automates redeviennent indépendants.

Dans la partie **f** de la figure 3.22, nous avons simplifié les opérations en dissociant les vecteurs. D'autre part, et nous ne l'avons pas explicité dans la figure, lorsque la chaîne liée ne possède pas d'autre point que le point commun, tous les points actuellement extraits se trouvent dans la chaîne courante. Nous nous retrouvons alors dans le même cas que dans la partie **e** de la figure 3.21.

Lors de la fermeture d'une chaîne liée à une autre, deux cas peuvent se produire :

- Les automates sont dans l'état 2 : On dissocie simplement les deux chaînes, la chaîne précédemment liée qui est encore active devenant seule propriétaire de la droite.
- Les automates sont dans l'état 1 : On dissocie également les deux chaînes, mais on transfère tous les points de la chaîne courante dans l'autre chaîne, ce qui impose d'ailleurs d'entrer ces points avant le premier point dans le vecteur (les derniers points d'une chaîne étant les premiers de la chaîne liée).

Cas général

On s'aperçoit que jusqu'à maintenant nous n'avons toujours pas résolu le problème de la figure 3.19. Dans ce cas en effet les morceaux de chaînes constituant la droite sont reliés par des opérations *jonction8* et des opérations *jonction1*. Si nous avons bien résolu le problème créé par *jonction8*, il n'en est pas de même pour *jonction1*.

En résolvant le problème de *jonction1*, nous aurons résolu toute une classe de problèmes. En effet, à part *jonction1*, *jonction8* et *jonction9*, tous les autres types de jonctions se font entre au moins trois chaînes. Ce type de jonction n'est pas soluble simplement par l'étude du voisinage d'un pixel, car si un segment de droite peut se prolonger sur une autre chaîne, il faut pouvoir décider de laquelle. Et celle

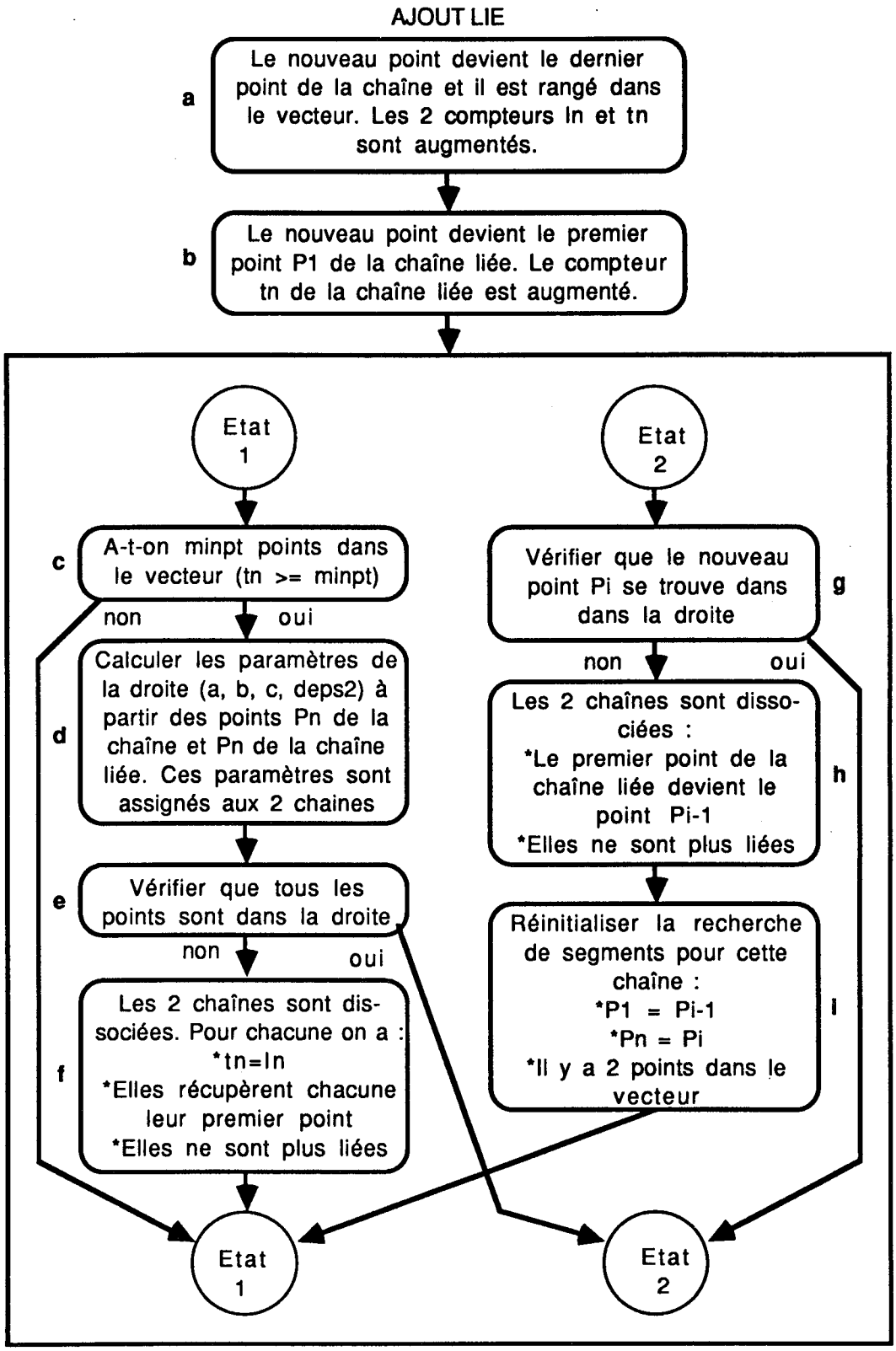


Figure 3.22: Description de l'automate dans le cas où une chaîne est lié à une autre

qui localement prolonge le mieux la droite peut ne pas être le bon choix dans le cas où l'image est bruitée.

Les jonctions entre deux chaînes quant à elles sont en fait des liaisons entre des sous-chaînes d'une même chaîne qui a été cassée arbitrairement en vertu des contraintes de fonctionnement de l'algorithme de chaînage. Il est donc nécessaire dans ces cas-là de prolonger des segments de droite sur plusieurs chaînes.

Modification de *jonction1*

Si nous reprenons la liste d'opérations de bases effectuées lors de *jonction1* :

jonction1 : *fermer*
fermer
connexion entre les deux chaînes

On s'aperçoit que les opérations *fermer* rangent les droites qui étaient en cours de prolongement. Nous modifions donc dans ce cas l'opération *fermer* en une opération *sfermer* de telle façon qu'elle ne change ni l'état ni le contenu de l'automate d'extraction de segment. Mais si une chaîne était liée à une autre, nous transférons toujours ces caractéristiques sur l'autre chaîne, ce qui permet dans pareil cas de ne traiter qu'avec la chaîne liée.

Après avoir accompli les opérations *sfermer* nous pouvons donc regarder si il est possible de fusionner des automates ou de modifier des liaisons entre chaînes déjà liées. Nous nous trouvons en présence d'un grand nombre de cas que nous exposons dans le tableau de la figure 3.23.

Nous dénombrons 16 opérations différentes car nous les faisons correspondre aux états des automates et des chaînes : Il y a deux chaînes avec chacune un état binaire (*etat*) de l'automate d'extraction de segments et un indicateur binaire (*etatll*) de liaison éventuelle. Nous avons donc bien 4 chiffres binaires qui donnent 16 valeurs différentes. Ces valeurs peuvent servir pour adresser les opérations comme nous l'avons fait pour le chaînage dans 3.2.1.

Les différentes opérations se détaillent de la manière suivante :

- *Opérations* 1 à 4 : Selon qu'une chaîne est liée ou n'est pas liée à une autre chaîne, on ne range pas ou on range la droite. Comme il nous importe peu qu'une droite soit coupée en plusieurs petits segments, nous n'essayons pas de fusionner deux droites entre elles.

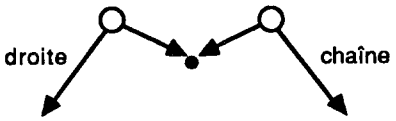
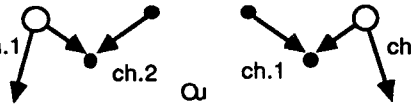
Cas	Sous-cas	Visualisation	Opérations
Cas 1 : 2 droites (état 2)	Les chaînes peuvent être liées ou pas à d'autres chaînes toujours actives	Voir dessins dans les cases ci-dessous	Opérations 1 à 4
Cas 2 : une droite et une chaîne (état 1 et état 2). (En fait 2 cas symétriques)	Cas I : Les 2 chaînes sont liées		Opérations 5 et 6
	Cas II : La droite est liée		Opérations 7 et 8
	Cas III : La chaîne est liée		Opérations 9 et 10
	Cas IV : Ni la chaîne ni la droite n'est liée		Opérations 11 et 12
Cas 3 : aucune droite (état 1)	Cas I : Les 2 chaînes sont liées		Opération 13
	Cas II et III : Une des 2 chaînes liée		Opérations 14 et 15
	Cas IV : aucune chaîne liée		Opération 16

Figure 3.23: Tableau de cas

- *Opérations* 5 à 12 : Dans tous les cas nous essayons d'ajouter les points de la chaîne à la droite. Si nous n'y parvenons pas la droite n'est pas modifiée, sinon la principale modification concerne la valeur d'un des deux points extrémités (la droite est rallongée par la chaîne de points) :
  - *Opérations* 5 et 6 : Le premier point de la droite devient le dernier point de la chaîne. L'automate de la chaîne est modifié pour être le même que celui de la droite et les deux chaînes sont liées.
  - *Opérations* 7 et 8 : Le premier point de la droite prend la valeur du premier point de la chaîne.
  - *Opérations* 9 et 10 : Le dernier point de la droite prend la valeur du dernier de la chaîne et la chaîne (c'est à dire son automate) devient la droite.
  - *Opérations* 11 et 12 : Le dernier point de la droite prend la valeur du premier de la chaîne et la droite est rangée.
- *Opérations* 13 à 16 : Nous fusionnons les deux chaînes. deux cas peuvent se produire : Soit il n'y a pas *minpt* points en tout, soit il y en a au moins *minpt* (peut-être plus) et nous pouvons alors tester l'alignement de ces points pour créer une nouvelle droite :
  - *Opération* 13 :
    * $n_1 + n_2 - 1 < minpt$: Les deux chaînes sont liées et possédent le même automate (le premier point de l'une est le dernier de l'autre).
    * $n_1 + n_2 - 1 \geq minpt$: On teste l'alignement de tous les points sur la droite définie par les derniers points des deux chaînes. Si c'est le cas on crée une droite commune aux deux chaînes et on les lie. Sinon elles ne sont pas liées et restent dans leur état initial.
  - *Opération* 14 et 15 :
    * $n_1 + n_2 - 1 < minpt$: Les points de la chaîne qui se ferme sont transférés dans le vecteur de la chaîne active.
    * $n_1 + n_2 - 1 \geq minpt$: On teste l'alignement de tous les points sur la droite définie par le premier point de la chaîne qui se ferme et le dernier de celle encore active. Si l'alignement est respecté, on crée une droite dans la chaîne active. Sinon la chaîne active reste dans son état initial.
  - *Opération* 16 :
    * $n_1 + n_2 - 1 < minpt$: Il n'y a rien à faire.
    * $n_1 + n_2 - 1 \geq minpt$: On teste l'alignement de tous les points sur la droite définie par les premiers points des deux chaînes. Si l'alignement est respecté on range la droite, sinon il n'y a rien à faire.

Nous venons donc de résoudre le problème de l'opération *jonction1*. Notre algorithme est donc actuellement capable de retrouver tous les types de segments recouvrant plusieurs sous-chaînes pour autant que ces sous-chaînes soient liées deux à deux.

Autres améliorations envisageables

Parmi tous les cas de liaisons entre sous-chaînes que nous avons recensés, il y en a pour lesquels nous avons simplifié au maximum le traitement à effectuer. Voici ce que nous aurions pu envisager de faire :

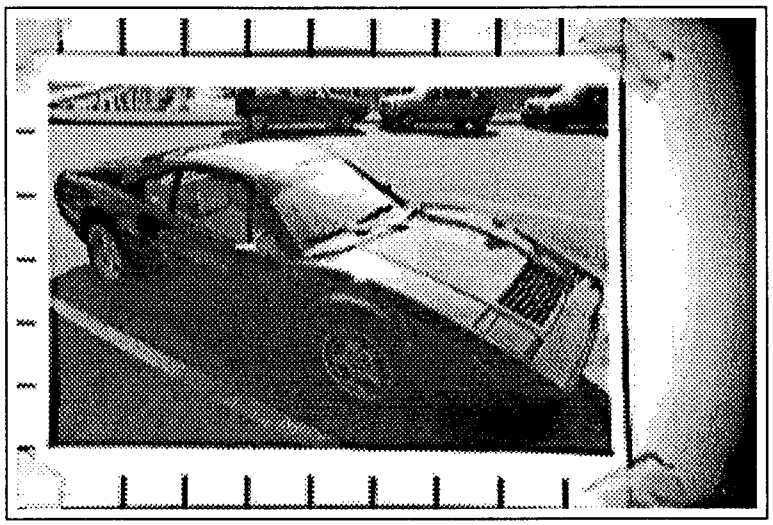


Figure 3.24: Image 1 : Image de scène extérieure

- Dans le cas où la tentative de création de droite entre deux chaînes liées échoue, nous laissons à chacune des deux chaînes leur vecteur de points initial. Nous aurions pu rechercher dans le vecteur commun quel est le point le plus éloigné de la droite pour faire partir les nouvelles recherches de segments. Mais on peut remarquer que dans la plupart des cas (en particulier pour *jonction8* et *jonction9*) le point de jonction est le point de cassure.

- Nous aurions pu dans *jonction1* fusionner deux droites. Cela suppose de créer une nouvelle droite à partir de deux droites ayant vérifié une fonction de similarité, et de modifier les automates en conséquence. Ce type d'opération a tendance à faire perdre de la précision sur les droites porteuses des segments : Jusqu'à maintenant en cherchant un segment avec une précision de ϵ nous étions assurés d'obtenir en sortie un segment dont les points sont alignés sur la droite porteuse à 2ϵ près. En multipliant les fusions de droites, il devient difficile d'assurer une erreur maximale.

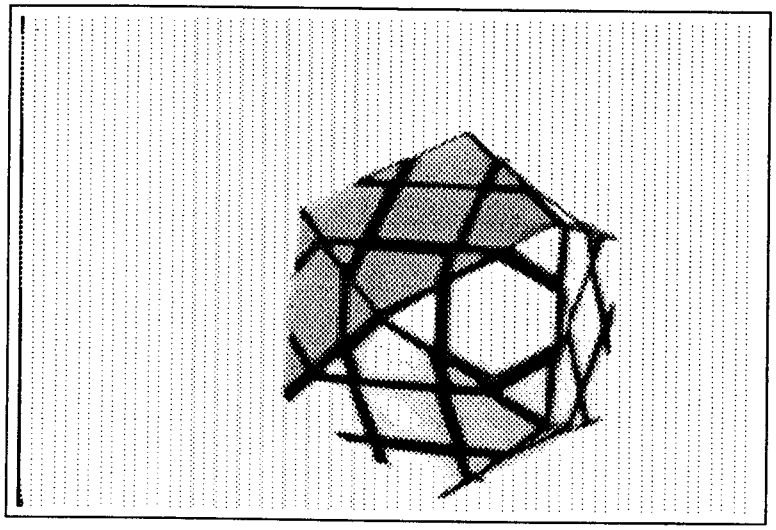


Figure 3.25: Image 2 : Image d'objet manufacturé

3.3 Fonctionnement

3.3.1 Quelques résultats

Nous présentons ici les résultats obtenus avec le chaînage que nous avons réalisé, sur deux types d'images, que nous appellerons l'*image 1* (figure 3.24) et l'*image 2* (figure 3.25). L'image 2 correspond au type d'images que nous serons amenés à traiter souvent puisqu'elle est extraite d'une séquence qui servira aux expérimentations des chapitres suivants.

Pour faire fonctionner notre algorithme, nous avons donc extrait des points de contours à l'aide de l'opérateur développé par Deriche [Der 87c]. Ce sont donc les images binaires des figures 3.26 et 3.27 qui ont été fournies en entrée du processus. Les résultats sont présentés sous forme graphique sur les figures 3.28 et 3.29.

Comme nous n'envisageons pas l'utilisation des segments de trop petite taille dans les étapes suivantes de notre étude, les figures 3.30 et 3.31 représentent les mêmes résultats où les segments de longueur inférieure à 10 pixels ont été supprimés.

En référence nous présentons les résultats obtenus sur les mêmes images par la méthode d'extraction de segments développée à l'INRIA (figures 3.32 et 3.33). Dans ce cas, après une recherche de points de contours basé sur l'algorithme de Deriche, les points sont chaînés par la méthode de Giraudon [Gir 87a] [Gir 87b] puis des segments de droite sont extraits de ces chaînes par la méthode récursive (voir partie 3.1.4). Ici aussi nous présentons les résultats après l'élimination des petits segments (figures 3.34 et 3.35).

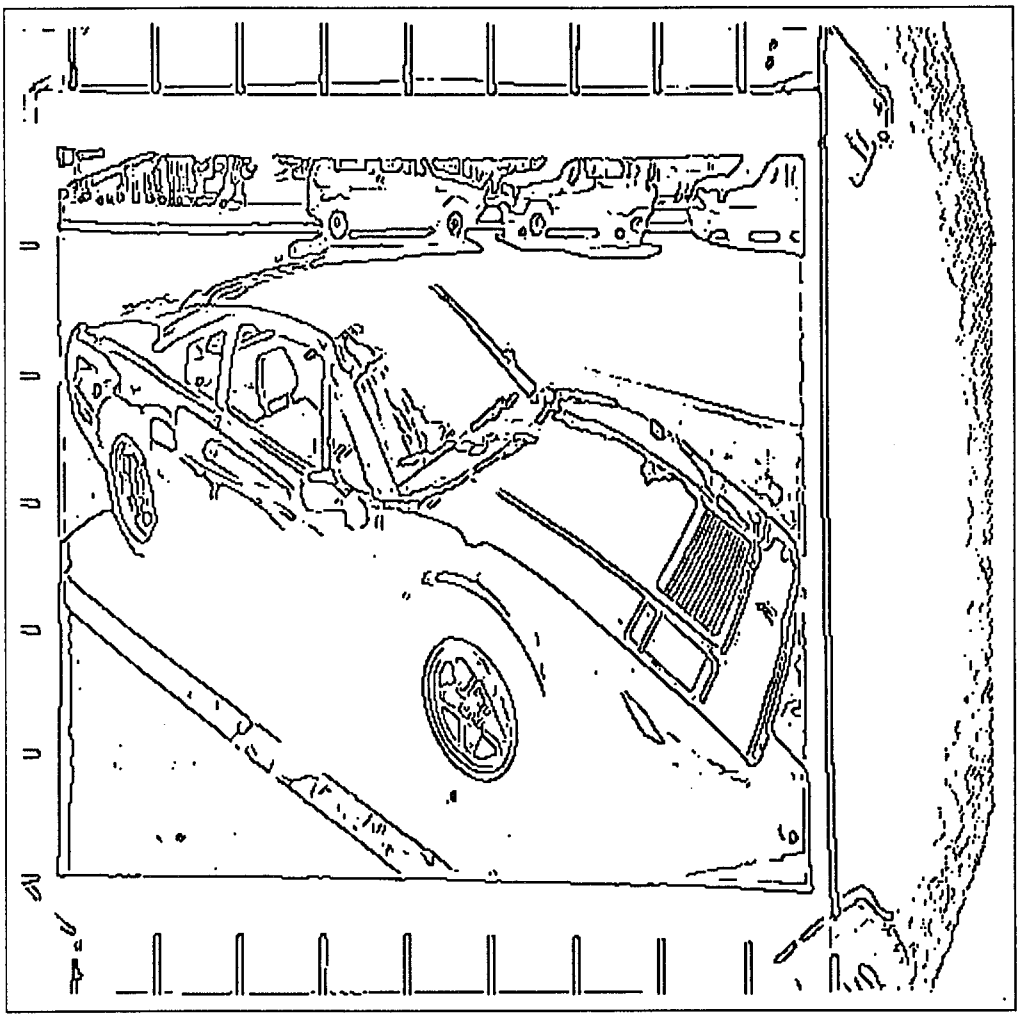


Figure 3.26: points de contours de l'image 1 obtenue avec la méthode de R. Deriche

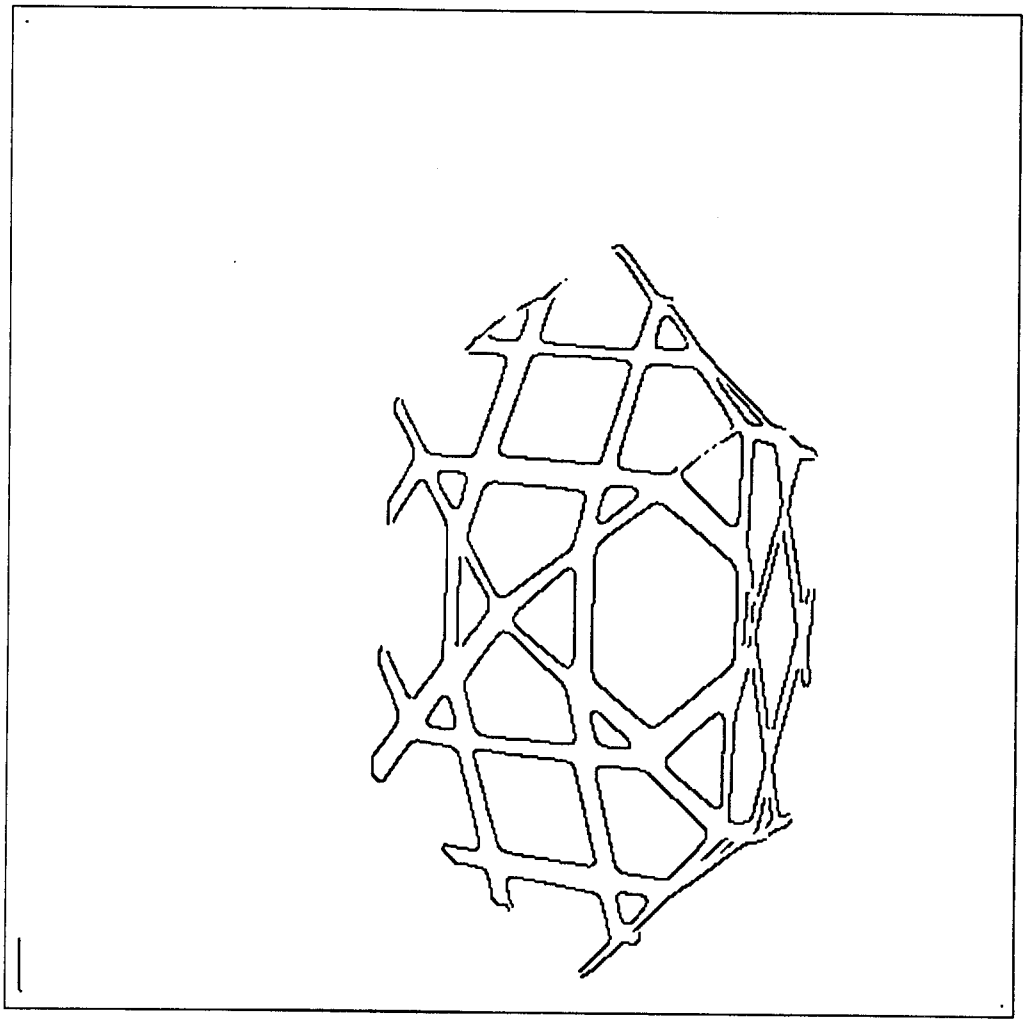


Figure 3.27: points de contours de l'image 2 obtenue avec la méthode de R. Deriche

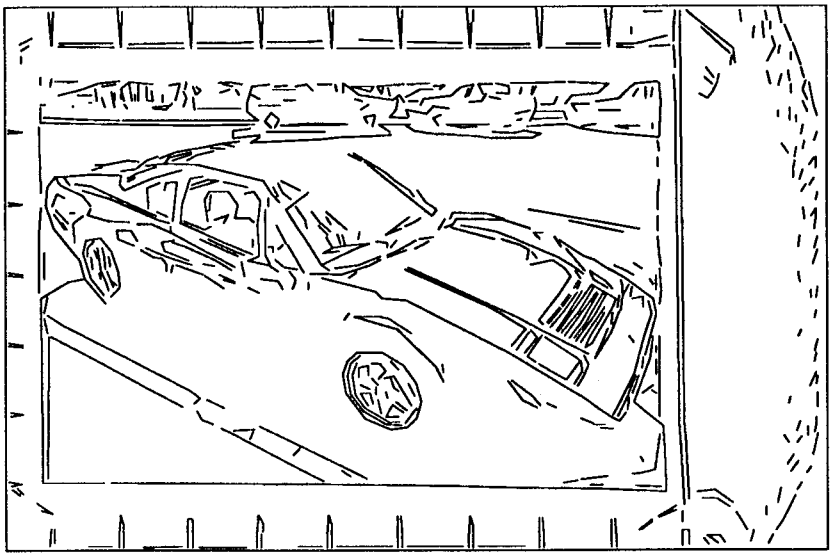


Figure 3.28: Segments de droites extraits par notre méthode sur la première image

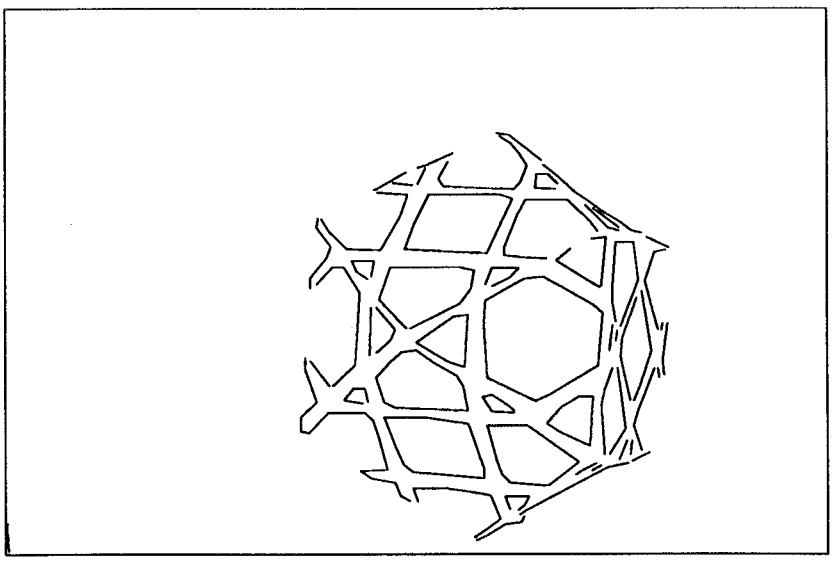


Figure 3.29: Segments de droites extraits par notre méthode sur la deuxième image

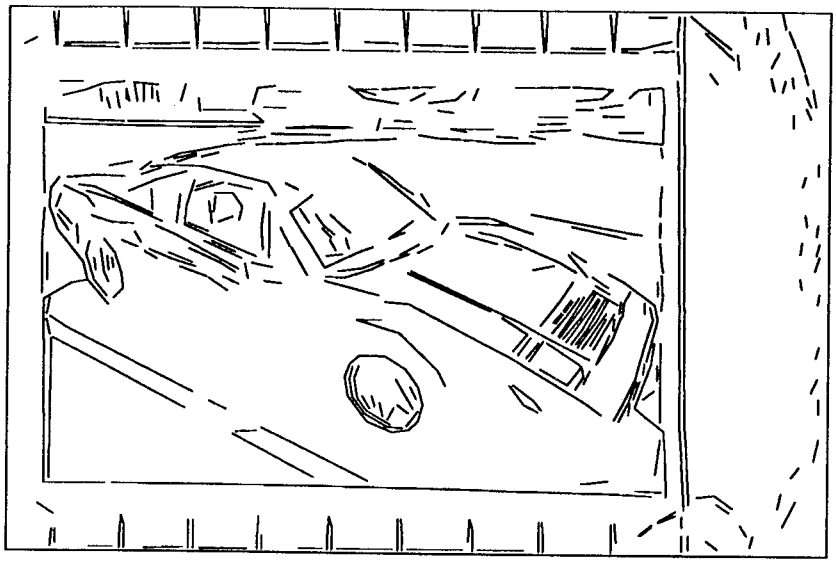


Figure 3.30: Segments pris en compte dans la figure 3.28

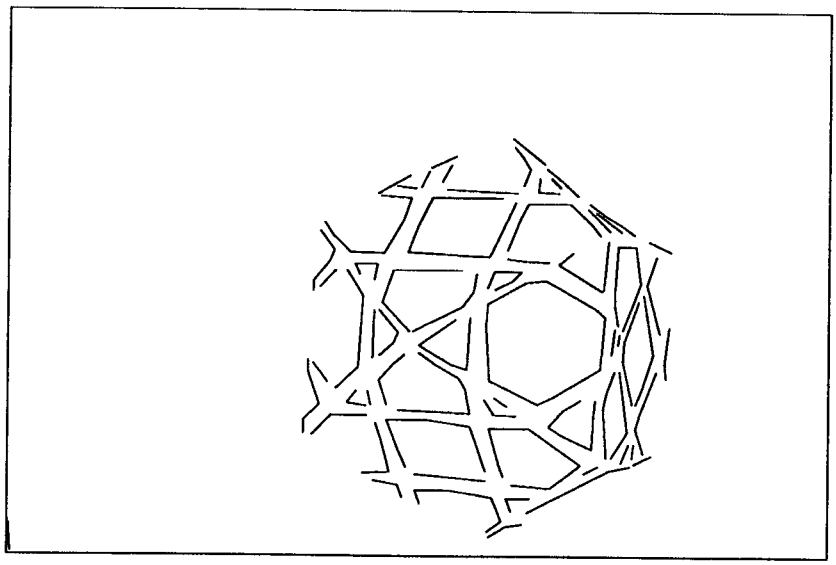


Figure 3.31: Segments pris en compte dans la figure 3.29

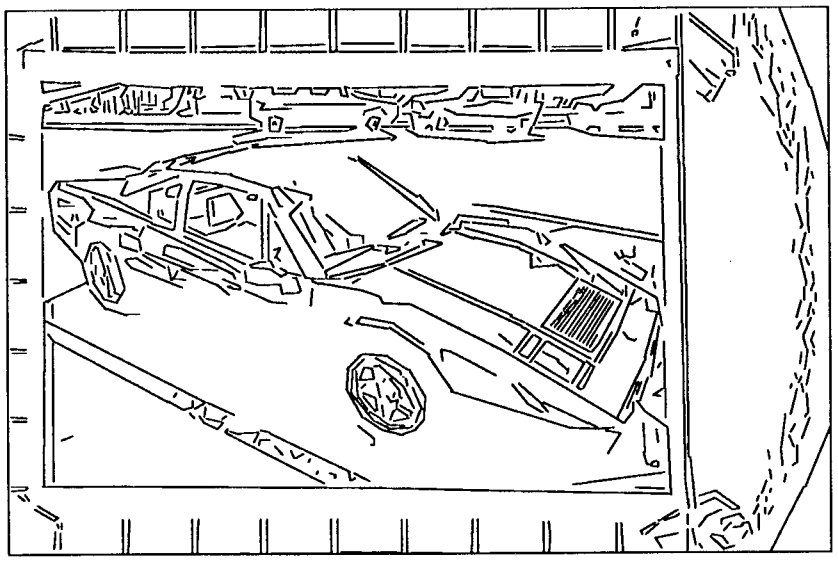


Figure 3.32: Segments de droites extraits par la méthode de l'INRIA sur la première image

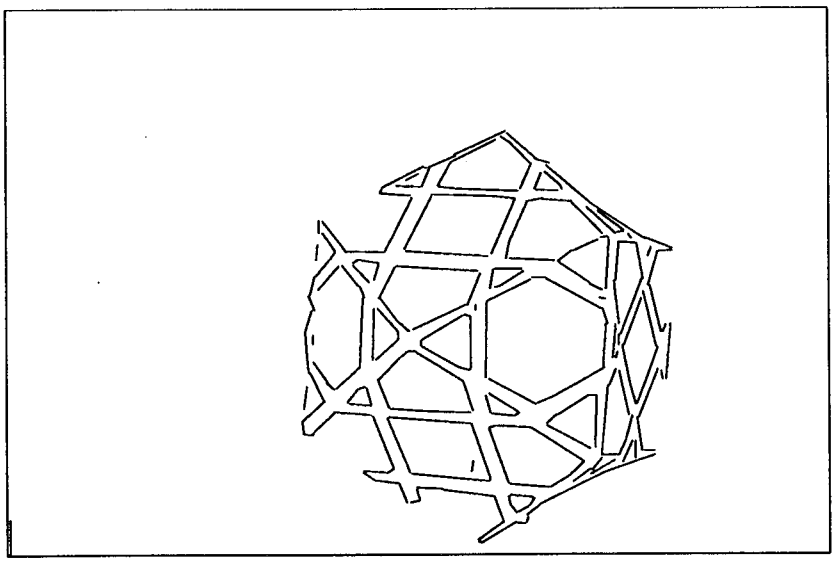


Figure 3.33: Segments de droites extraits par la méthode de l'INRIA sur la deuxième image

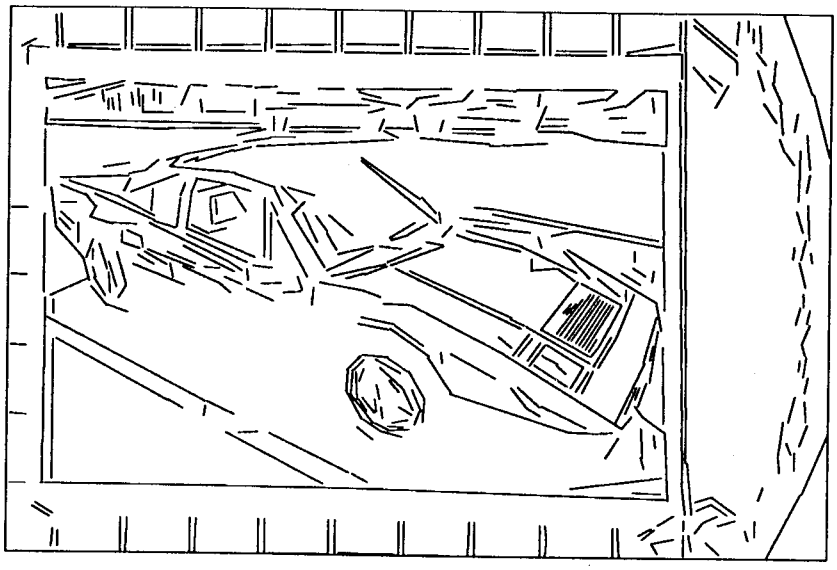


Figure 3.34: Segments pris en compte dans la figure 3.32

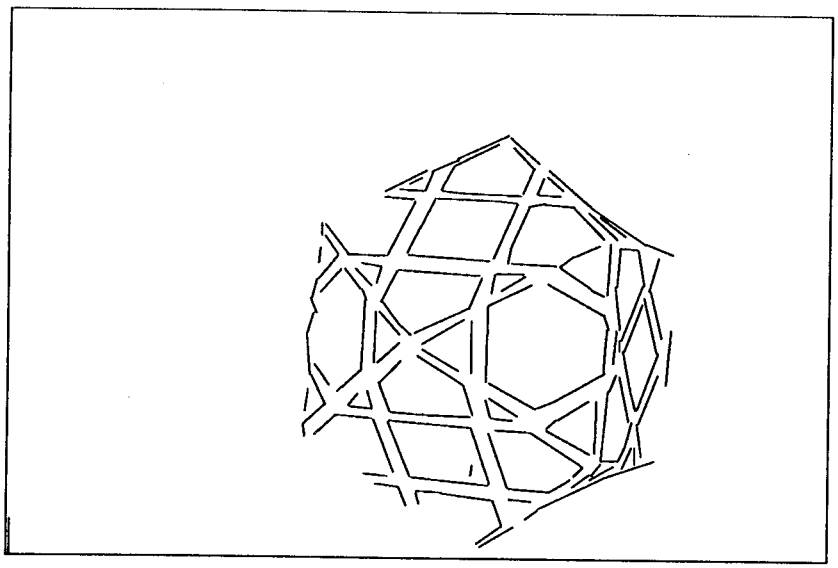


Figure 3.35: Segments pris en compte dans la figure 3.33

	Image 1		Image 2	
	méthode 1	méthode 2	méthode 1	méthode 2
Nombre de segments	900	957	222	202
Plus grand	384.0	416.01	67.07	70.01
Moyenne	15.84	16.95	19.35	23.19
Ecart type	20.69	25.99	11.54	15.58
Nombre de segments après élimination	516	481	180	157
Moyenne	22.08	28.32	21.98	28.19
Ecart type	25.56	32.83	11.29	14.22

Figure 3.36: Synthèse des résultats

Le tableau de la figure 3.36 présente une synthèse comparant pour chacune des deux images notre méthode (méthode 1) à celle de l'INRIA (méthode 2). Le tableau est séparé en deux parties : Dans la partie supérieure sont données les nombres de segments, la taille du plus grand, la longueur moyenne et l'écart type de ces longueurs. La partie du bas donne les mêmes résultats après l'élimination des plus petits segments.

3.3.2 Interprétation

On peut voir qu'en moyenne les segments extraits par notre méthode sont un peu plus courts que ceux de l'autre méthode, ceci en partie parce qu'il n'y a pas de fusion entre deux droites liées par un point et de caractéristiques similaires. En partie aussi parce qu'une coupure dans une chaîne coupe de même un segment.

Précision visuelle

On remarque que les segments fournis par notre méthode semble être moins précis que ceux de l'autre méthode. Ceci est principalement dû au fait que nous extrayons des segments s'appuyant sur des points extrémités réels. Une nette amélioration peut être faite en modélisant les extrémités des segments par des projections des

points extrémités réels sur la droite porteuse.

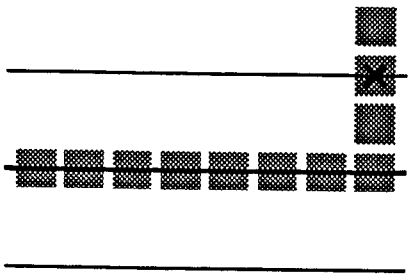


Figure 3.37: Les derniers points d'un segment se situent après la cassure

Un autre problème est celui des derniers points inclus dans un segment (figure 3.37). Ceci peut être amélioré de la manière suivante : Au lieu de recommencer une recherche de segments à partir du point précédent celui qui est sorti de la droite, considérer le vecteur de points en cours comme un petit segment dans lequel on recherche le point le plus éloigné. En prenant ce point comme départ de la nouvelle recherche, on a des chances de limiter l'erreur sur l'emplacement de la cassure (figure 3.38).

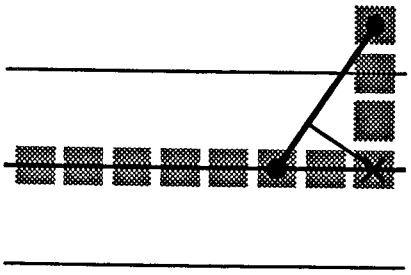


Figure 3.38: En recherchant le point le plus éloigné dans le vecteur, on peut retrouver la cassure

Problème de l'orientation

Lorsque les segments sont extraits, ils sont orientés dans le sens où ils apparaissent. Il serait souhaitable pour leur utilisation ultérieure qu'ils soient orientés en fonction de la direction du gradient (par exemple le côté sombre à droite). Ce problème n'est pas simple à résoudre car il supposerait de conserver l'information sur l'intensité lumineuse alors que notre algorithme est essentiellement basé sur l'utilisation d'une image binaire.

4.1.1 Principe du modèle dynamique

Le fonctionnement du suivi d'indices est basé sur la notion de "modèle dynamique". Le processus de suivi conserve constamment un modèle des indices image. Mais au lieu d'être simplement des copies de ce qui est observé, les indices contenu dans ce modèle possèdent une information d'existence, et l'absence d'une observation ne signifie pas que le modèle lui correspondant va disparaître. Par conséquent un indice modèle n'est pas lié à une seule image, mais se trouve sur plusieurs instants. Ceci permet de résoudre partiellement les problèmes rencontrés dans des images bruitées, comme celles provenant d'une caméra montée sur un robot mobile.

En utilisant une caméra montée sur un robot se déplaçant en translation, tout accident sur une trajectoire prévue peut faire disparaître momentanément des indices. Cela peut être les vibrations des moteurs, un petit objet sur lequel roule le robot et modifiant ainsi un court instant le mouvement et la prise de vue. Plus particulièrement si le processus suit des segments de droite, l'extracteur d'indices pourra, en fonction de l'éclairage et de l'orientation des chaînes de points de contraste dans l'image, donner des segments ayant leurs extrémités à des endroits différents. Dans d'autres cas il pourra couper certains segments en d'autres plus petits ou fusionner plusieurs segments pour n'en faire qu'un grand (figure 4.1). Ces incidents peuvent se combiner, et il est nécessaire d'avoir une technique qui garde en mémoire ce qui a été suivi.

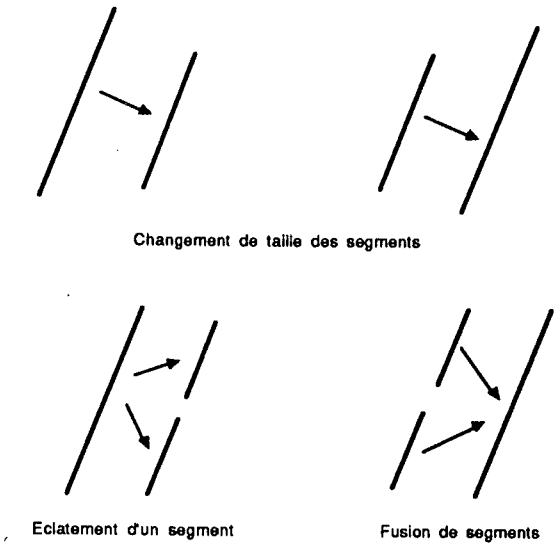


Figure 4.1: Segments

L'utilisation du modèle dynamique nous permet de mesurer les mouvement des indices malgré ces phénomènes de coupures; par exemple dans le cas où un segment se coupe d'abord en plusieurs autres qui vont se recombiner plus tard (figure 4.2).

Chapitre 4

Suivi 2-D de segments de droite

Ayant développé une méthode d'extraction de segments de droite, nous allons maintenant décrire comment utiliser ces segments afin d'extraire le mouvement visuel dans une séquence d'images[1] [Cro 88], [Ste 88].

4.1 Recherche du mouvement : le suivi d'indices

Nous avons exposé dans 2.3 les trois principaux problèmes du suivi d'indices. Nous avons d'abord soulevé celui de la modélisation des observations. Nous allons encore rappeler ici la nécessité d'une telle modélisation. Nous avons également vu l'intéret qu'il y avait à utiliser un filtre de Kalman pour l'estimation du mouvement dans le modèle, et nous expliquerons comment il s'applique à notre système. Enfin nous détaillerons comment nous avons cerné le problème de la mise en correspondance entre modèle et image.

[1] Ce travail a été effectué en collaboration avec O. Faugeras et R. Deriche de l'INRIA.

Si l'on suppose que le grand segment ne peut correspondre à aucun des plus petits

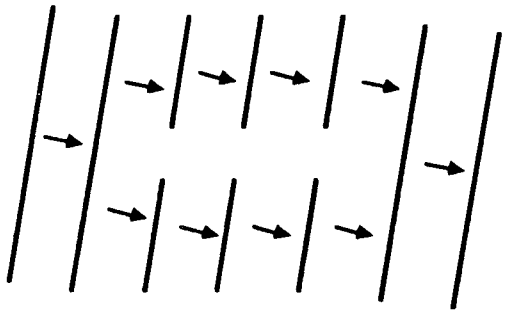


Figure 4.2: Un segment en plusieurs

segments, ces derniers vont créer autant de nouveaux modèles, tandis que le grand restera comme modèle. Lorsque l'on retrouve la configuration inverse, le grand segment étant toujours dans le modèle pourra être mis à nouveau en correspondance avec l'image. Quant aux petits segments ils disparaitront du modèle au bout d'un certain temps si ils ne réapparaissent pas (figure 4.3).

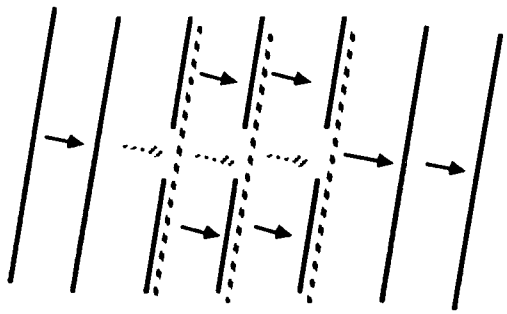


Figure 4.3: Comment résoudre le problème

Le cas inverse où des petits segments se recombinent momentanément se résout de façon similaire (figure 4.4). Mais le découpage d'une ligne en plusieurs petits segments étant moins stable que la reconstruction en un seul grand, il est moins évident que l'on puisse retrouver la même configuration qu'au départ et les anciens modèles disparaîtront.

Le fonctionnement du modèle dynamique peut être décrit de manière schématique dans la figure 4.5. Le modèle est composé de trois parties. La partie hypothèse contient tous les indices nouveaux trouvés dans une image. En effet, on ne considère pas comme sûrs les indices présents dans une seule image, et on en a éventuellement besoin de plusieurs pour confirmer leur existence. La partie modèle proprement dite contient les indices qui ont été confirmés par leur existence dans une séquence. Enfin le modèle rémanent est constitué de ceux qui ayant fait partie du modèle se

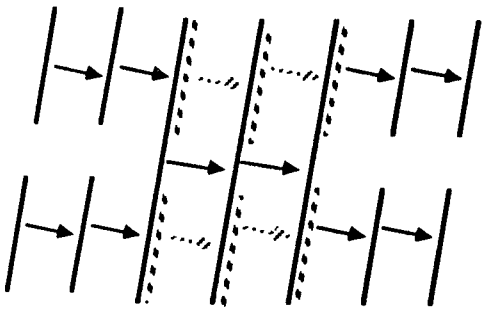


Figure 4.4: Plusieurs segments deviennent un seul

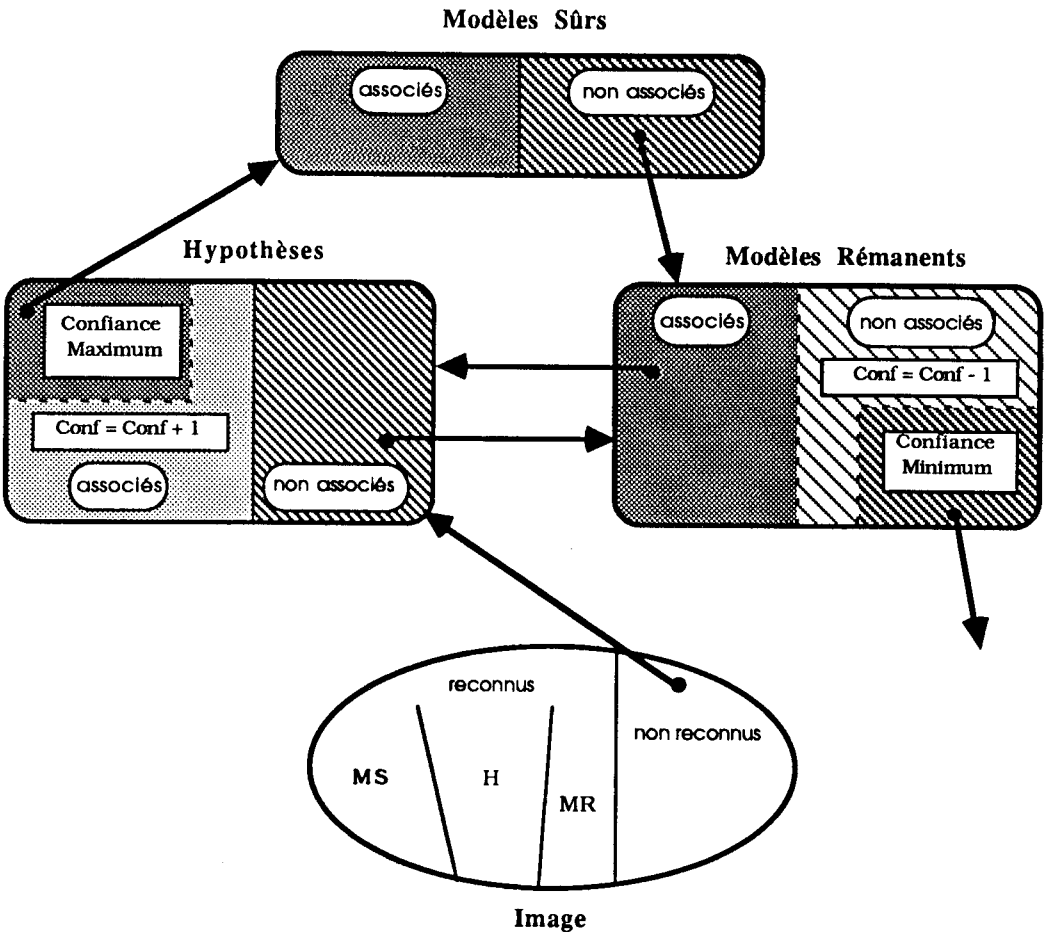


Figure 4.5: Fonctionnement du modèle dynamique

trouvent ne plus correspondre à des indices réels dans les images. Ils restent donc un certain temps ici avant de disparaître réellement.

Le passage dans les différents types de modèle est controlé par un facteur de confiance (CF). Les indices image ne correspondant avec aucun des indices des différents types de modèle deviennent des hypothèses avec un petit facteur de confiance. Chaque fois que l'on trouve cette hypothèse dans les images suivantes on augmente son facteur de confiance, jusqu'à atteindre une valeur maximale qui la fait devenir modèle sûr. De la même façon un modèle rémanent qui ne correspond à rien voit son facteur de confiance diminuer jusqu'à une valeur minimale où il est éliminé. Lorsqu'on fait passer une hypothèse dans les modèles rémanents et réciproquement, on peut laisser tel quel le facteur de confiance ou lui affecter une valeur arbitraire. Ainsi en lui donnant la valeur minimale pour une hypothèse qui devient modèle rémanent, on élimine tout de suite cet indice (si l'on a supposé qu'il fallait absolument un certain nombre d'images pour valider une hypothèse). Et en donnant la valeur maximale au facteur de confiance d'un indice rémanent, il revient immédiatemment dans le modèle (une seule observation aura suffit pour considérer qu'il n'a pas disparu).

Dans la pratique, ces différents types de modèles (modèle sûr, hypothèse, modèle rémanent) peuvent être représentés par une seule structure de données contenant les caractéristiques propres à chaque indice auxquelles sont associés un facteur de confiance et un indicateur permettant de savoir s'il s'agit d'une hypothèse ou d'un modèle (un modèle rémanent est un modèle avec CF $<$ CF maximum). Ce dernier ne sera d'ailleurs pas utilisé dans l'application que nous avons réalisée.

4.1.2 Description schématique du suivi

Le suivi tel que nous l'avons réalisé est représenté dans la figure 4.6. On entre continuellement de nouvelles images (ces images, dans le cas de segments de droite, ne proviennent pas directement d'une caméra mais ont d'abord subi un traitement qui a extrait les indices auxquels on s'intéresse) dans le processus. A l'intérieur se trouve un modèle des indices suivis et le traitement effectué peut se résumer en deux parties : Dans un premier temps on cherche à mettre en correspondance les indices modèle avec les indices image. C'est cette partie qui permet de dire si l'on suit vraiment quelque chose dans une séquence. Ensuite on met à jour le modèle. Outre les opérations qui ont été décrites dans la section précédente, cela consiste aussi à modifier les paramètres de la représentation des indices suivis :

- En comparant un indice modèle avec l'indice image lui correspondant, on raffine la connaissance (position, vitesse, etc.) que l'on a de cet indice en tenant

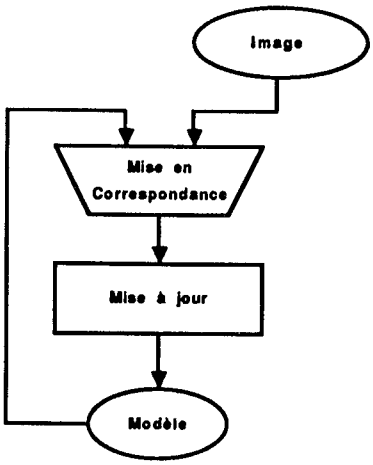


Figure 4.6: Schéma du suivi

compte aussi bien de ses caractéristiques dans l'image courante (données par l'indice image) que de celles des images précédentes (contenues implicitement dans le modèle).

- Un indice étant modélisé dynamiquement (vitesse, accélération) il faut aussi tenir compte du fait que la prochaine image sera acquise après un intervalle de temps Δt et donc faire progresser le modèle selon la loi d'évolution qu'on lui suppose avoir (prédiction).

La méthode du filtre de Kalman permet de regrouper la partie mise à jour proprement dite et la partie prédiction dans un même ensemble de calculs. Ceci explique que nous ayons représenté sur le schéma les deux parties de cette mise à jour en un seul bloc.

Pour l'estimation de la valeur supposée vraie d'un indice modèle, obtenue par comparaison entre ce modèle et l'indice image correspondant, on pourrait raisonnablement utiliser le facteur de confiance : Pour un indice nouvellement trouvé et ayant une très faible confiance, ces caractéristiques dans la nouvelle image seront ce que l'on possède de plus fiable à ce moment. Pour un indice plus ancien ces même caractéristiques image auront moins de poids, ce qui nous permettra d'éviter en partie les problèmes créés par le bruit. Dans la pratique, le filtre de Kalman possède son propre évaluateur de fiabilité du modèle et donc le facteur de confiance ne sera utilisé que pour gérer ce modèle.

Enfin la sortie des informations du suivi n'a pas été représentée sur le schéma car elle dépend de ce que l'on veut obtenir. Soit l'on recherche une estimation assez

fiable du mouvement dans une série d'images, et on s'intéresse alors au contenu du modèle qui lui peut intégrer une notion de vitesse (flot optique). Soit on veut simplement reconnaître des éléments dans plusieurs vues, les numéroter, et dans ce cas c'est le résultat de la mise en correspondance qu'il nous faut. Le modèle devient alors purement interne au processus et il ne sert qu'à optimiser la correspondance en tenant compte de l'évolution de la scène.

4.1.3 Indices visuels

Pour pouvoir suivre des objets dans une scène, il faut pouvoir extraire dans la vision que l'on a de ces objets des caractéristiques visuelles qui permettent soit d'identifier ces objets sans ambiguïté, soit de minimiser le nombre des ambiguïtés, de façon à avoir le moins de solutions possibles lors d'une mise en correspondance entre deux images différentes. Ces indices visuels peuvent être plus ou moins proches de la structure de l'image (pixels) pour aller vers la scène observée elle-même (objets réels). Une abstraction de l'image peut conduire à considérer des indices de niveaux de plus en plus élevés qui néanmoins ne concourrent pas à raffiner la description de la scène de départ. En effet ce que l'on cherche à faire n'est pas de "reconnaître" mais de "suivre" et donc au maximum de modéliser des trajectoires en deux dimensions.

Choix des paramètres

Pour reconnaître des indices les uns des autres, on compare les différents paramètres les identifiant, ces paramètres n'ayant pas tous la même portée :

- **Paramètres globaux** : Par exemple la position d'un indice dans l'image. L'utilisation la plus courante d'un tel type de paramètre consiste à découper les images en zones et ne chercher à mettre en correspondance entre deux images que les indices d'une même zone ou de deux zones adjacentes (Buckets).

- **Paramètres locaux** : ce sont des paramètres topologiques qui permettent de situer localement des indices par rapport à d'autres. En général leur représentation est de type booléenne comme "à gauche de", "à droite de", etc.

- **Paramètres cinématiques** : Toutes les informations de mouvement (vitesse, accélération) ainsi que la position (que l'on a aussi décrit comme paramètre global).

- **Paramètres intrinsèques** : Tout ce qui ne dépend pas de la position d'un indice, et que l'on espère le plus stable possible dans le temps. Par exemple pour un point de contraste ça peut être la valeur du gradient de l'intensité lumineuse.

Lorsqu'on compare des segments de droite, le premier problème est de trouver une représentation qui soit à la fois simple à manipuler et qui contienne le plus d'informations valables.

Représentation d'un segment de droite

La façon la plus courante de représenter un segment est de donner les coordonnées cartésiennes des deux points extrémités. C'est également sous cette forme que les extracteurs de contours fournissent les segments, c'est donc celle qui demande le moins de transformations pour être obtenue. Néanmoins cette représentation n'est pas la plus efficace pour évaluer la correspondance entre deux indices.

En reprenant la classification des paramètres que nous avons proposée plus haut, regardons quels sont les types de caractéristiques d'un segment :

- **position globale** : Positions de points caractéristiques. Les points caractéristiques les plus employés sont les suivants :
  - P_1 et P_2 les points extrémités.
  - P_m le point milieu.

  Tous ces points étant représentés par leurs coordonnées. On verra plus loin que le système de coordonnées employé a une grande importance dans la représentation.
- **Position locale** : Nous en décrivons deux :
  - L la longeur ou h_L la demi-longueur du segment.
  - θ l'orientation du segment.

  Ces deux paramètres sont fortement liés aux coordonnées des points caractéristiques.
- **Intrinsèques** : Par exemple g le gradient maximum de l'intensité lumineuse pour ce segment.

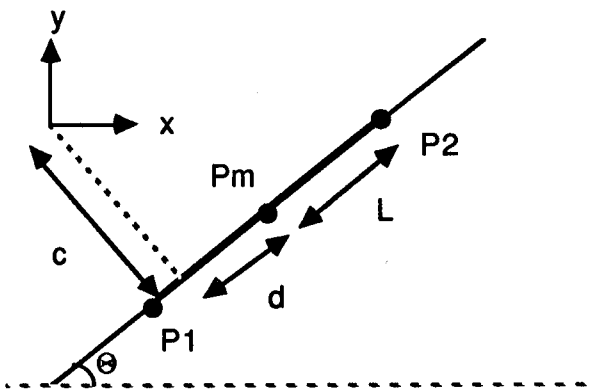


Figure 4.7: Représentation d'un segment

- **Cinématiques** : On peut aussi bien considérer les vitesses et accélérations des points caractéristiques que celles de θ. Pour le point milieu plus particulièrement, on peut représenter le mouvement selon ses coordonnées cartésiennes (x, y) ou bien dans un repère lié au segment avec une direction parallèle à ce segment et une perpendiculaire.

Représentation dans le repère du segment

On peut remarquer que si on exclut les paramètres cinématiques et le gradient de l'intensité, les deux points extrémités décrivent totalement un segment et donc qu'il peut être représenté au minimum par quatre valeurs. Nous proposons donc la représentation suivante (voir figure 4.7) qui est composée des paramètres c, d, θ, h_L. Si on prend x_m et y_m les coordonnées cartésiennes du point milieu, les valeurs c et d se calculent avec les équations suivantes :

$$c = -x_m \sin(\theta) + y_m \cos(\theta) \tag{4.20}$$

$$d = x_m \cos(\theta) + y_m \sin(\theta) \tag{4.21}$$

On observera que ces valeurs c et d sont respectivement les coordonnées perpendiculaires et parallèles du point milieu. Comme d'autre part la précision en position perpendiculairement à une ligne de contraste est bien plus grande que parallèlement, ces deux paramètres vont pouvoir être traités différemment de manière à extraire le plus d'informations sur un segment là où il en donne le plus.

Pour deux segments ayant leurs orientations θ très proches, leur distance perpendiculaire sera à peu près la différence en c. Quant à la différence en d elle représente

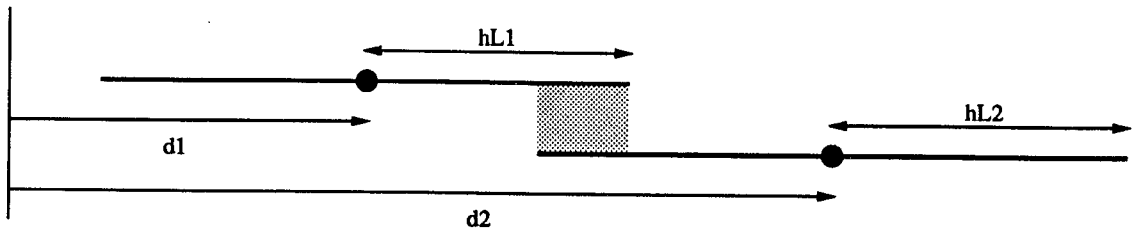


Figure 4.8: Recouvrement de deux segments ayant la même orientation

le "glissement" entre eux. pour deux segments ayant la même orientation, la relation :

$$| d_1 - d_2 | \leq h_{L_1} + h_{L_2} \tag{4.22}$$

est vérifiée quand ils se recouvrent (figure 4.8).

Problème de la corrélation entre les paramètres

Le choix des paramètres doit tenir compte de la distribution des informations entre eux. En effet si certains paramètres se partagent bien l'information, ils ont tendance à réagir de la même façon aux variations dues aux bruits des capteurs. Ce qui fait qu'en présence d'un bruit important il n'est pas possible de rattraper l'erreur sur un paramètre à l'aide d'un autre. De tels paramètres sont beaucoups moins discriminants lors de leur utilisation dans une mise en correspondance.

Dans pareil cas, ces deux paramètres sont dits corrélés. Lorsqu'on utilise un filtre de Kalman il est également préférable d'utiliser des paramètres décorrélés car cela permet d'avoir un filtre pour chacun des paramètres. Sinon il est nécessaire, pour utiliser correctement ce filtre, de filtrer ensemble les paramètres et donc de manipuler des matrices d'autant plus grandes.

Nous avons donc cherché à savoir dans quelle mesure tels ou tels paramètres de la représentation d'un segment étaient corrélés. Pour cela nous avons comparé le comportement de quatre segments observés délimitant un carré noir sur fond blanc. En faisant varier les conditions d'éclairage on obtient des images assez différentes, et les segments de droite extraits peuvent fortement varier, comme on peut le voir dans les deux cas extrêmes (mais parfaitement exploitables) de la figure 4.9.

Nous avons donc calculé pour différents couples de paramètres un coefficient de corrélation de la manière suivante :

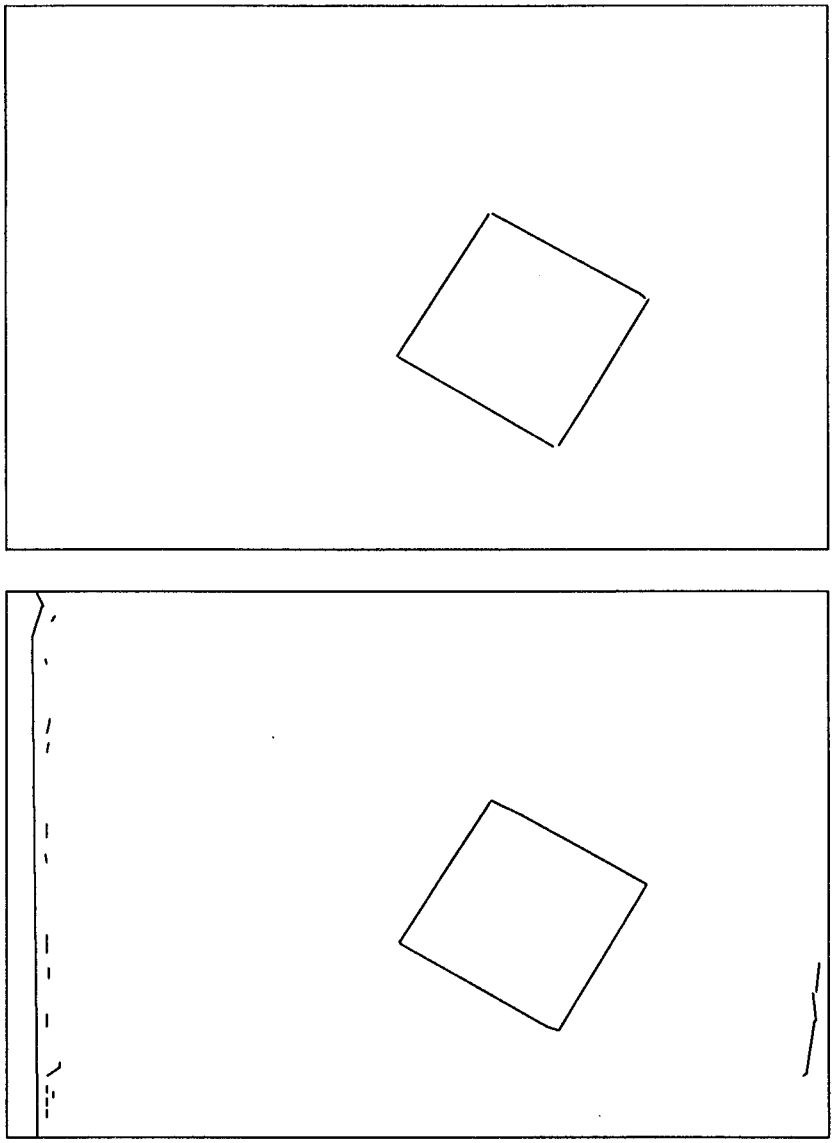


Figure 4.9: Deux aspects des images de segments de droite testées

$$\rho = \frac{\sigma_{xy}}{\sigma_x \sigma_y} \tag{4.23}$$

Où ρ est le coefficient de corrélation et les σ sont les éléments de la matrice de covariance : σ_x et σ_y les écarts-types des quantités x et y et σ_{xy} la covariance entre x et y.

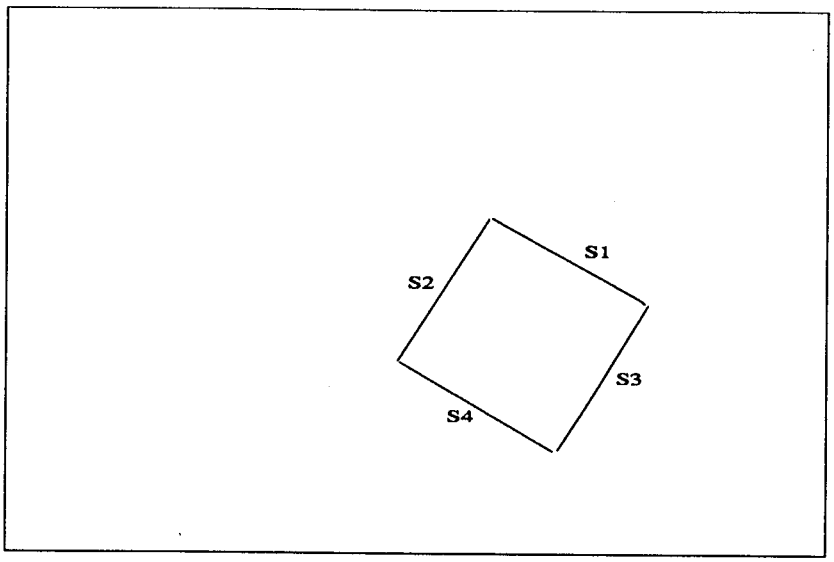


Figure 4.10: Les quatre segments étudiés

L'interprétation du facteur ρ est délicate. Si il est raisonnable de dire que si il est proche de 1 (cas de $y = x$) ou -1 (cas de $y = -x$) alors les deux variables sont corrélées, par contre lorsqu'il est proche de 0 cela ne peut pas garantir la décorrélation. On se sert donc de ce facteur seulement comme d'indication.

Le tableau de la figure 4.11 montre les valeurs de ρ pour chacun des quatre segments de la figure 4.10. Dans la colonne "interprétation" est indiquée la corrélation possible qui peut exister entre chacun des paramètres. On voit donc que le choix de c, d, θ et L pour la représentation paraît être le meilleur, même s'il semble exister une certaine dépendance entre d et L.

4.1.4 La mise à jour

Nous avons vu précédemment comment ont été regroupées en une seule entité la mise à jour du modèle, c'est à dire la comparaison entre un modèle et son correspondant dans l'image courante, et la prédiction, qui consiste à faire évoluer un modèle connaissant ses paramètres cinématiques et le temps qui va s'écouler jusqu'à

	S1	S2	S3	S4	Interprétation
x - y	.979703	-.99344	-.972757	.978272	Corrèlés
c - d	.100638	-.241527	-.233173	-.178868	Non corrèlés
θ - c	-.171017	-.248838	-.777006	.421389	Non Corrèlés ?
θ - d	.155073	.316927	.639353	-.500676	Non Corrèlés ?
c - x	.013988	-.145429	.075525	.244474	Non Corrèlés
c - y	.198124	.257072	-.303740	.064226	Non Corrèlés
d - x	.996132	.995123	-.986358	-.994972	Corrèlés
d - y	.990929	-.999861	.997026	-.992985	Corrèlés
d - L	-.974795	.659253	-.428477	.697757	Corrèlés ?
c - L	-.192687	-.308444	.719582	-.049434	Non Corrèlés ?
θ - L	-.10858	.727337	-.636601	-.44375	Non Corrèlés ?
x - θ	.183716	.289358	-.545383	.462116	Non Corrèlés ?
y - θ	.223145	-.315853	.694708	.478937	Non Corrèlés ?
x - L	-.961858	.637572	.32197	-.675647	Corrèlés ?
y - L	-.973812	-.659177	-.47044	-.70853	Corrèlés ?

Figure 4.11: Comparaison des facteurs de corrélations pour différents couples de paramètres

l'acquisition de la prochaine image. Pour réaliser cette mise à jour, nous allons en effet employer un estimateur basé sur le filtre de Kalman qui assure la réalisation des deux fonctions.

Le principe du filtre de Kalman-Bucy [Lab 78] est d'essayer de mesurer une valeur bruitée en intégrant plusieurs observations. Il fonctionne de manière récursive en utilisant un modèle de la valeur que l'on cherche : M_n le modèle combinant n observations est calculé à partir du modèle M_{n-1} et de la n^{ieme} observation. Au modèle est associé un bruit de modèle représentant le fait que l'on ne connait pas exactement l'évolution de ce modèle en fonction du nombre de mesures. A l'observation est associée un bruit de mesure. Ces deux bruits sont supposés être des bruits "blancs" de forme Gaussienne.

Dans le cas d'un suivi d'indices, nous n'avons aucune connaissance sur le bruit de mesure, et l'approximer par un bruit "blanc" semble un choix raisonnable. Quant au modèle, nous verrons plus tard dans quelle mesure l'hypothèse du bruit "blanc" est valable.

Après avoir exposé son principe, nous allons maintenant décrire plus formellement le filtre de Kalman :

- $\hat{X}$ est le vecteur d'état. Il contient tout ce que l'on cherche à mesurer. Son évolution en fonction des observations est régi par l'équation d'état :

$$\hat{X}_k = \Phi_{k/k-1}\hat{X}_{k-1} + W \tag{4.24}$$

  Où $\Phi_{k/k-1}$ est la matrice de transition représentant le passage de l'état $k-1$ à l'état k et W est le bruit du modèle. Ce bruit est connu par sa matrice de covariance Q.

- Z est le vecteur d'observation. Il contient tout ce qui est observable. Les liens entre l'observation et le modèle $\hat{X}$ sont donnés par l'équation de mesure :

$$Z = H\hat{X} + V \tag{4.25}$$

  Où H est la matrice de conditionnement permettant d'extraire du vecteur d'état ce qui est réellement observé, et V est le bruit de mesure qui est lui aussi connu par sa matrice de covariance R. Q et R ne sont pas corrélés.

Si l'on considère que chaque mesure correspond à un instant d'observation, le filtre de Kalman va nous donner pour chaque instant une estimation du vecteur

d'état $\hat{X}$, une erreur P sur ce vecteur (P est une matrice de covariance) tout en utilisant un intermédiaire de calcul, K appelé gain de Kalman. Les équations du filtre sont les suivantes :

$$\hat{X}_{k/k-1} = \Phi_{k/k-1}\hat{X}_{k-1} \tag{4.26}$$

$$P_{k/k-1} = \Phi_{k/k-1}P_{k-1}\Phi_{k/k-1}^{\top} + Q_{k-1} \tag{4.27}$$

$$K_k = P_{k/k-1}H_k^{\top}(H_kP_{k/k-1}H_k^{\top} + R_k)^{-1} \tag{4.28}$$

$$\hat{X}_k = \hat{X}_{k/k-1} + K_k(Z_k - H_k\hat{X}_{k/k-1}) \tag{4.29}$$

$$P_k = P_{k/k-1} - K_kH_kP_{k/k-1} \tag{4.30}$$

Lorsqu'une variable ou un ensemble de variables est décorrélé des autres, on peut la filtrer indépendamment. Cela permet d'utiliser autant de filtres de Kalman qu'il y a de valeurs indépendantes à mesurer, et donc de limiter la taille des matrices à manipuler. Pour simplifier le plus possible les calculs à effectuer, nous allons chercher à obtenir des paramètres de représentation d'un segment les plus indépendants possibles.

Nous avons vu qu'un segment de droite possède une plus grande précision en position perpendiculairement à son axe que parallèlement. Ceci nous amène à considérer indépendamment ces deux imprécisions, qui sont des imprécisions sur les grandeurs c et d et donc à postuler que ces deux valeurs sont décorrélées. Ce qui entraine que dans une représentation en x_m, y_m, x_m et y_m ne sont pas indépendants. D'autre part θ et L étant considérés comme paramètres intrinsèques, ils sont indépendants entre eux et des autres paramètres. Pour l'angle θ c'est vrai dans une représentation en x_m, y_m mais c'est moins vrai dans une représentation en c, d. Nous avons quand même choisi de le considérer indépendant de c et de d.

Une représentation sous la forme c, d, θ, L nous permet donc d'utiliser quatre filtres de Kalman indépendants. Dans chacun de ces filtres nous pouvons mesurer la valeur du paramètre et son mouvement. La seule contrainte que l'on puisse donner à un mouvement quelconque est que l'accélération d'un objet possédant une masse est bornée, par contre elle peut varier instantanément et une dérivée d'ordre supérieur n'a plus de sens physique. Il n'est donc pas intéressant d'essayer de calculer une valeur exacte de l'accélération mais plus de calculer la vitesse et de considérer l'accélération comme bruit du modèle. Donc pour chaque paramètre x on a les données suivantes :

- Le vecteur d'état :

$$\hat{X}_k = \begin{pmatrix} x \\ \dot{x} \end{pmatrix}$$

- La matrice de transition :

$$\Phi_{k/k-1} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

Si l'on prend comme unité de temps la durée qui s'écoule entre deux images prétraitées fournies au suivi.

- L'observation est donc constituée de la seule position :

$$Z = z$$

- La matrice de conditionnement est donc :

$$H = \begin{pmatrix} 1 & 0 \end{pmatrix}$$

- La matrice de covariance de l'observation se réduit à un scalaire :

$$R = r$$

- La matrice de covariance de l'erreur sur le modèle sera donnée par ses composantes :

$$P = \begin{pmatrix} \sigma_x^2 & \sigma_{x\dot{x}} \\ \sigma_{x\dot{x}} & \sigma_{\dot{x}}^2 \end{pmatrix}$$

- Il en est de même pour la matrice de covariance du bruit du modèle :

$$Q = \begin{pmatrix} \sigma_{q_x}^2 & \sigma_{q_{x\dot{x}}} \\ \sigma_{q_{x\dot{x}}} & \sigma_{q_{\dot{x}}}^2 \end{pmatrix}$$

- Et de la même façon pour le gain de Kalman :

$$K = \begin{pmatrix} k_x \\ k_{\dot{x}} \end{pmatrix}$$

Les équations simplifiées du filtre de Kalman qui découlent de ces contraintes sont les suivantes (pour ne pas alourdir l'ecriture de ces équations, nous noterons avec une prime certaines valeurs intermédiaires) :

$$4.26 \Rightarrow \hat{X}_{k/k-1} = \Phi_{k/k-1}\hat{X}_{k-1} = \begin{pmatrix} x+\dot{x} \\ \dot{x} \end{pmatrix} = \begin{pmatrix} x' \\ \dot{x}' \end{pmatrix} \tag{4.31}$$

$$4.27 \Rightarrow P_{k/k-1} = \Phi_{k/k-1}P_{k-1}\Phi_{k/k-1}^{\top} + Q_{k-1}$$

$$= \begin{pmatrix} \sigma_x^2 + 2\sigma_{x\dot{x}} + \sigma_{\dot{x}}^2 + \sigma_{q_x}^2 & \sigma_{x\dot{x}} + \sigma_{\dot{x}}^2 + \sigma_{q_{x\dot{x}}} \\ \sigma_{x\dot{x}} + \sigma_{\dot{x}}^2 + \sigma_{q_{x\dot{x}}} & \sigma_{\dot{x}}^2 + \sigma_{q_{\dot{x}}}^2 \end{pmatrix} = \begin{pmatrix} \sigma_x^{2\prime} & \sigma_{x\dot{x}}' \\ \sigma_{x\dot{x}}' & \sigma_{\dot{x}}^{2\prime} \end{pmatrix} \tag{4.32}$$

$$4.28 \Rightarrow K_k = P_{k/k-1}H^{\top}(HP_{k/k-1}H^{\top} + R)^{-1}$$

$$= \begin{pmatrix} \frac{\sigma_x^{2\prime}}{\sigma_x^{2\prime}+r} \\ \frac{\sigma_{x\dot{x}}'}{\sigma_x^{2\prime}+r} \end{pmatrix} = \begin{pmatrix} k_x \\ k_{\dot{x}} \end{pmatrix} \tag{4.33}$$

$$4.29 \Rightarrow \hat{X}_k = \hat{X}_{k/k-1} + K_k(Z - H\hat{X}_{k/k-1})$$

$$= \begin{pmatrix} x' + k_x(z-x') \\ \dot{x}' + k_{\dot{x}}(z-x') \end{pmatrix} \tag{4.34}$$

$$4.30 \Rightarrow P_k = P_{k/k-1} - K_k H P_{k/k-1}$$

$$= \begin{pmatrix} \sigma_x^{2\prime}(1-k_x) & \sigma_{x\dot{x}}'(1-k_x) \\ \sigma_{x\dot{x}}' - \sigma_x^{2\prime}k_{\dot{x}} & \sigma_{\dot{x}}' - \sigma_{x\dot{x}}^{2\prime}k_{\dot{x}} \end{pmatrix} = \begin{pmatrix} \frac{r\sigma_x^{2\prime}}{\sigma_x^{2\prime}+r} & \frac{r\sigma_{x\dot{x}}'}{\sigma_x^{2\prime}+r} \\ \frac{r\sigma_{x\dot{x}}'}{\sigma_x^{2\prime}+r} & \sigma_{\dot{x}}^{2\prime} - \frac{\sigma_{x\dot{x}}'^2}{\sigma_x^{2\prime}+r} \end{pmatrix} \tag{4.35}$$

4.1.5 La mise en correspondance

Nous avons vu dans 2.3.3 comment se faisait théoriquement la mise en correspondance entre deux ensembles par recherche de cliques maximales dans un graphe de compatibilités. Outre le coût algorithmique que nous avons déjà évoqué, cela n'est intéressant que si l'on possède réellement un graphe de compatibilité, ce qui n'est pas notre cas.

Nous allons donc dans notre mise en correspondance essayer de trouver des appariements univoques directement à partir de la comparaison des indices. Cette comparaison se fera en deux étapes. D'abord on va chercher à éliminer toute correspondance potentielle qui "semble" fausse. Dans le cas d'indices dans une image, on peut découper les images en régions et ne considérer que les correspondances qui se font entre des indices d'une même région ou de régions adjacentes. C'est ce qu'on appelle le découpage en "Buckets" et permet lorsque les indices sont rangés par zones de ne pas effectuer un certain nombre de comparaisons dont le résultat

serait de toute façon négatif. Il reste néanmoins une certaine quantité de comparaisons à effectuer pour ne garder qu'un petit nombre de correspondances potentielles.

Que signifie "sembler" faux ? Chaque indice est représenté par plusieurs paramètres, et chacun de ces paramètres est défini avec une certaine imprécision. Si on fait l'hypothèse que cette imprécision est de forme Gaussienne, un paramètre x est connu par sa valeur moyenne m et sa variance σ^2. Entre deux paramètres x_1 et x_2 de même type on peut calculer une distance $d(x_1, x_2)$:

$$d(x_1, x_2) = \sqrt{\frac{(m_1 - m_2)^2}{\sigma_1^2 + \sigma_2^2}} \tag{4.36}$$

On considère alors comme fausse la correspondance lorsque pour un type de paramètre il y a une distance entre les deux indices supérieure à une valeur V :

$$d(x_1, x_2) \geq V \Leftrightarrow (m_1 - m_2)^2 \geq V^2(\sigma_1^2 + \sigma_2^2) \tag{4.37}$$

Ne seront donc considérées comme valables que les correspondances entre deux indices dont chacun des paramètres x_i aura vérifié le test t_i :

$$t_i \Leftrightarrow (m_{i_1} - m_{i_2})^2 < V_i^2(\sigma_{i_1}^2 + \sigma_{i_2}^2) \tag{4.38}$$

Dans le cas du filtre de Kalman, les valeurs m_i sont celles qui sont calculées pour le modèle, et celles observées pour la mesure. Les variances ${\sigma_i}^2$ sont données pour le modèle par les ${\sigma_x}^2$ (qui sont également appelés a) extraits de la matrice de covariance P, et pour la mesure par r.

Après ce premier passage il peut encore rester des correspondances ambiguës où un indice est apparié avec plusieurs autres. Le but du deuxième passage est de ranger ces hypothèses de correspondance afin de ne garder que les meilleures, donc d'appliquer une heuristique. La façon la plus générale de procéder est d'appliquer une fonction de coût appelée distance de Mahalanobis :

$$D_M = \sum \frac{(m_{i_1} - m_{i_2})^2}{\sigma_{i_1}^2} \tag{4.39}$$

Cette formule permet de comparer plusieurs correspondances pour un même modèle (si le modèle est indicé 1). La division par σ_i permet de rendre comparable des grandeurs physiques différentes (comme par exemple une distance et un angle). Par contre dans le cas d'une même mesure (indicée 2) cette distance risque de favoriser une correspondance avec un modèle ayant des paramètres très imprécis (et

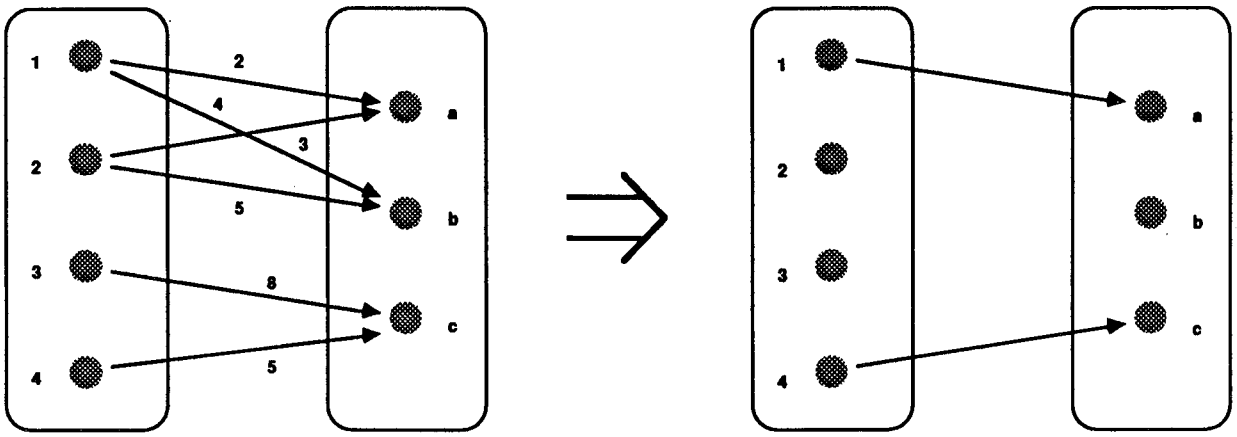


Figure 4.12: Autre méthode d'appariement

donc de grandes valeurs sur les σ_i).

Le choix des appariements une fois le coût calculé peut être faite par exemple par la méthode de "Greedy Algorithm" que nous avons présenté dans 2.3.3. Mais la nécessité de passer dans deux boucles algorithmiques, et aussi celle, dans certains cas, de conserver une très grande quantité d'informations (toutes les hypothèses de correspondance) nous a conduit à ne pas utiliser cette méthode. Nous avons préféré plutôt rejeter au fur et à mesure toute correspondance avec des indices déjà utilisés lorsque le coût est plus élevé (figure 4.12), au risque de perdre un certain nombre d'appariements.

4.2 Réalisation du suivi : Le "Token-Tracker"

Nous venons de présenter les outils mathématiques que nous avons employés pour réaliser un algorithme de suivi de segments de droite. Nous allons maintenant décrire plus précisément cet algorithme, et voir dans quelle mesure nous devrons adapter nos modèles théoriques, voire en abandonner certains, pour obtenir une solution qui fonctionne avec des contraintes raisonnables.

4.2.1 Les structures de données

Quelles structures de données doit posséder le suivi? Puisque l'algorithme est basé sur l'utilisation d'un modèle des indices des images, nous allons donc avoir un tableau d'objets représentant ces indices, leur dynamique et toutes les informations nécessaires à l'utilisation d'un fitre de Kalman. Nous avons regroupé en une seule structure la notion de modèle actif et le modèle de Kalman.

Nous avons simplifié au maximum la manipulation du modèle actif en associant à chaque objet uniquement un facteur de confiance; une hypothèse étant directement entrée à l'état de modèle. Il n'y a donc plus vraiment de différence entre une hypothèse nouvellement entrée et un modèle rémanent. Lorsqu'une nouvelle hypothèse arrive, son facteur de confiance $conf$ est affecté à la valeur $DEFAULTCONF$. Puis lorsqu'elle est suivie son facteur de confiance augmente jusqu'à la valeur $MAXCONF$ (ou elle est considérée comme modèle véritable). Si un modèle (ou une hypothèse) est perdu, son facteur de confiance diminue jusqu'à $MINCONF$ auquel cas il est rejeté. Ces trois valeurs sont telles que :

$$MINCONF < DEFAULTCONF < MAXCONF$$

On peut voir qu'une hypothèse fausse n'est pas immédiatement rejetée, et qu'un modèle récent sort plus vite qu'un modèle ancien lorsqu'ils ne sont plus validés.

L'application finale dans laquelle a été envisagée la réalisation du suivi d'indices est de suivre des indices d'une image à une autre, mais sans s'intéresser aux informations que l'on pourrait tirer d'une étude du mouvement. On va donc numéroter les indices à l'intérieur du modèle, en associant à chaque objet un *numéro* qui l'identifie sans ambiguïté : Le processus fournira en sortie un ensemble d'indices étiquetés, et si on compare des résultats obtenus à des moments différents, on pourra retrouver sans problème des indices présents aux deux instants. Pour les mêmes raisons seront conservés dans un indice modèle les paramètres de l'indice image, à savoir les coordonnées des deux points extrémités du segment.

Enfin, chaque objet dans le modèle contient les paramètres de la représentation interne avec un certain nombre de valeurs associées qui sont utilisées par le filtre de Kalman. Il y a donc les valeurs c, d, θ, h_L, et pour chacune en plus une vitesse et trois des composants de P (en constatant que les valeurs de l'antidiagonale sont toujours égales).

4.2.2 Composition de l'algorithme de suivi d'indices

Nous allons maintenant décrire le suivi de manière algorithmique en détaillant chacun des blocs principaux. La description est donnée par la figure 4.13.

L'initialisation de l'algorithme consiste à créer un modèle à partir de la première image. Il n'y a en fait aucune différence entre la façon de traiter cette première image et la façon dont sont traités les indices entrés comme hypothèses et provenant des autres images. Si l'on sépare ainsi l'entrée de la première image de celle des

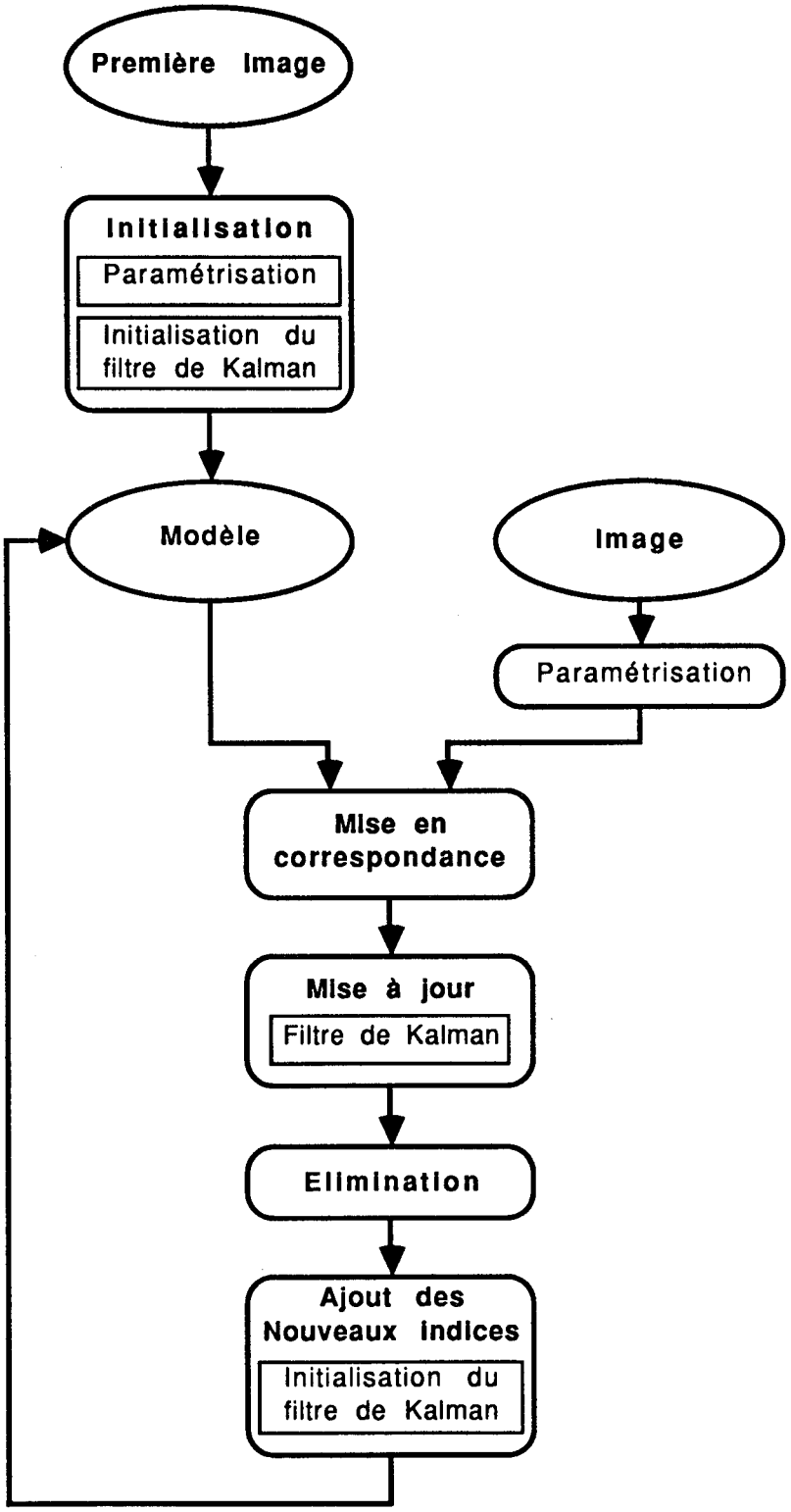


Figure 4.13: L'algorithme de suivi d'indices

autres, c'est simplement parce que les autres images ne sont pas entrées directement, et les indices de ces images ne sont pas tous traités de cette façon. C'est pourquoi nous ne décrirons qu'une seule fois la paramétrisation et l'initialisation du filtre de Kalman car ce sont les mêmes, aussi bien pour les indices de la première image que pour les indices ajoutés provenant des autres images.

Paramétrisation

L'extracteur de segments de droite nous fournissant les segments sous la forme des coordonnées des deux points extrémités, il faut tout d'abord extraire les paramètres utilisés par le suivi (c, d, θ, L) pour pouvoir les utiliser.

les coordonnées du point milieu sont données par :

$$\begin{cases} x_m = \frac{x_{P_1}+x_{P_2}}{2} \\ y_m = \frac{y_{P_1}+y_{P_2}}{2} \end{cases} \tag{4.40}$$

Nous en déduisons donc les paramètres c, d, θ, L :

$$\theta = \arctan 2(y_{P_2} - y_{P_1}, x_{P_2} - x_{P_1}) \tag{4.41}$$

$$c = -x_m \sin\theta + y_m \cos\theta \tag{4.42}$$

$$d = x_m \cos\theta + y_m \sin\theta \tag{4.43}$$

$$L = \sqrt{(x_{P_2} - x_{P_1})^2 + (y_{P_2} - y_{P_1})^2} \tag{4.44}$$

Utilisation du filtre de Kalman

Comme pour chacun des paramètres filtrés nous avons le même type de filtre, les opérations de mise à jour du filtre de Kalman ont été regroupées dans une procédure *Kalman* que le suivi utilise avec chacun des paramètres. Cette procédure s'utilise de la façon suivante :

$$Kalman(\sigma_z^2, z, \sigma_{xk-1}^2, \sigma_{x\dot{x}k-1}, \sigma_{\dot{x}k-1}^2, \sigma_{q_x}^2, \sigma_{q_{x\dot{x}}}, \sigma_{q_{\dot{x}}}^2, x_{k-1}, \dot{x}_{k-1})$$

$$\rightarrow \sigma_{xk}^2, \sigma_{x\dot{x}k}, \sigma_{\dot{x}k}^2, x_k, \dot{x}_k$$

Pour le paramètre L sur lequel on ne fait pas d'estimation de la vitesse, cette procédure peut être utilisée en faisant :

$$Kalman(\sigma_z^2, z, \sigma_x^2, 0, 0, \sigma_{q_x}^2, 0, 0, x, 0)$$

Initialisation du filtre de Kalman

L'initialisation est basée sur l'hypothèse simplificatrice que lors de sa première observation la vitesse d'un segment est nulle. Pour tout indice nouvellement entré dans le modèle on exécute un pas du filtre de Kalman en prenant comme valeur pour le modèle et pour l'observation la valeur de cet indice. Ce premier pas est réalisé avec une matrice initiale de covariance du bruit du modèle Q_0 :

$$Q_0 = \begin{pmatrix} \sigma_m^2 & \sigma_m^2 \\ \sigma_m^2 & \sigma_m^2 \end{pmatrix} \tag{4.45}$$

Ceci revient à exécuter la procédure $Kalman$ de la façon suivante :

$$Kalman(\sigma^2_{observation}, observation, \sigma_m^2, \sigma_m^2, \sigma_m^2, \sigma^2_{q_x}, \sigma_{q_{x\dot{x}}}, \sigma^2_{q_{\dot{x}}}, observation, 0)$$

Ainsi la matrice P se trouve initialisée. Comme la vitesse est supposée nulle, la vitesse réelle est implicitement bornée par la valeur de cette matrice P. Le choix de la matrice Q qui conditionne en grande partie la valeur de cette matrice P est donc important : Si l'erreur sur le modèle est trop grande, le filtrage sera moins efficace et trop d'indices images pourront être appariés avec cet indice modèle par la suite. Si elle est trop petite, le filtre peut diverger et si la vitesse n'est pas nulle cet indice modèle risque de ne pas trouver de correspondant dans les images suivantes.

Par la suite, on utilise une matrice de covariance du bruit du modèle Q :

$$Q = \begin{pmatrix} \sigma_q^2 & \frac{\sigma_q^2}{2} \\ \frac{\sigma_q^2}{2} & \frac{\sigma_q^2}{4} \end{pmatrix} \tag{4.46}$$

Cette matrice sert à modéliser l'accélération que nous avons choisi de représenter par un bruit. La valeur σ_q^2 peut être très petite, mais pas nulle car dans ce cas le filtre diverge.

Entrée des images

Outre la paramétrisation des segments de droite, il faut estimer pour chaque indice une incertitude d'observation sur chacun des paramètres qui fournira la valeur r du filtre de Kalman.

Pour le paramètre c, on peut estimer que l'erreur de l'extracteur est d'un pixel : Certains programmes de segmentation peuvent fournir une plus grande précision sur la droite porteuse, mais les coordonnées des points extrémités sont entrées sous

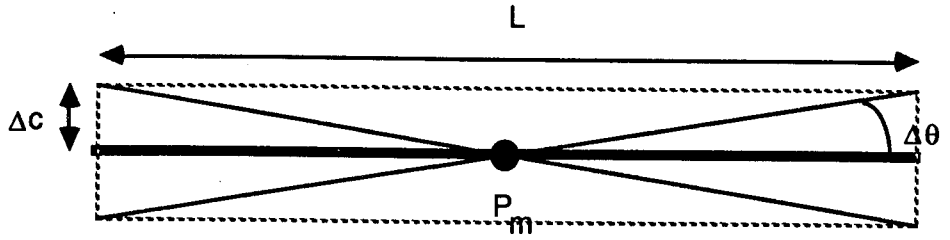


Figure 4.14: Estimation de l'erreur en θ

forme entière, et donc avec une erreur proche du pixel.

Pour θ on peut voir sur la figure 4.14 que l'erreur en θ est liée à l'erreur sur c et à la longueur du segment :

$$\Delta\theta = \arctan(\frac{\Delta c}{h_L}) \tag{4.47}$$

Comme l'erreur sur c est constante et proche de un pixel on a donc :

$$\Delta\theta \simeq \arctan(\frac{1}{h_L}) \tag{4.48}$$

h_L étant également mesurée en pixels.

La précision d'un segment dépend donc principalement de sa longueur. Cela a entre autre pour conséquence, étant donné qu'il est plus facile de suivre des segments peu bruités, que tous les petits segments vont être difficiles à suivre et à la limite géner le suivi des autres. C'est pourquoi nous n'effectuerons le suivi que sur une partie des segments, éliminant ceux qui sont trop petits. Ce n'est pas génant car outre leur imprécision intrinséque, ces petits segments sont les plus susceptibles de se modifier d'une image à une autre. Ils sont d'autre part d'un moindre intérêt pour des processus de plus haut niveau utilisant les résultats du suivi (par exemple une reconstruction 3D).

L'erreur en position sur d ne dépend pas uniquement de l'erreur de position des points extrémités. En effet si nous supposons que la position du point milieu du segment est totalement imprécise le long de la droite porteuse, nous pouvons raisonnablement situer ce point milieu n'importe où sur le segment. Dans ce cas nous faisons l'hypothèse que l'erreur sur d est h_L :

$$\Delta d = h_L \tag{4.49}$$

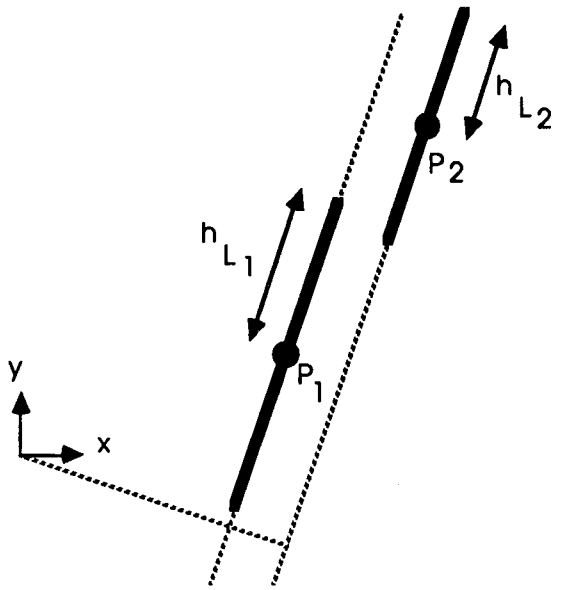


Figure 4.15: Cas de suivi avec de grands déplacements en d

en faisant des tests sur une image réelle, on peut voir que selon l'éclairage, un segment peut effectivement changer de taille, mais aussi se couper en plusieurs segments. Comme dans pareil cas nous voulons forcer le suivi d'indice à suivre au moins un de ces segments (en général le plus grand) la position observée de d est effectivement très incertaine. Dans le cas particulier décrit par la figure 4.15 les segments observés **2** et **3** sont effectivement les mêmes si l'on suppose que l'erreur Δd vaut bien h_L.

Pour les mêmes raisons, la mesure L (ou de h_L) est très imprécise. On peut remarquer d'autre part qu'elle n'est pas centrée : Dans le cas où il est acceptable pour L de passer du simple au double, ΔL vaut L en plus et $\frac{L}{2}$ en moins. En fait le choix de cette erreur dépend de l'utilisation qui est faite de L ou de h_L. Si on veut que ce paramètre soit suffisamment discriminant il ne faut pas que l'erreur soit trop large. Il est clair par contre que cette erreur dépend de la valeur de L ou de h_L.

Nous venons de décrire pour chaque paramètre une erreur de mesure Δp. Cette erreur est telle que :

$$\begin{cases} P(x \in [p - \Delta p, p + \Delta p]) = 1 \\ P(x \notin [p - \Delta p, p + \Delta p]) = 0 \end{cases} \tag{4.50}$$

Où p est la valeur observée du paramètre et x sa valeur réelle. Or la variance σ_p que nous devons fournir au filtre de Kalman est celle d'une Gaussienne. On a donc :

$$P(x \in [a,b]) = \int_a^b \frac{1}{\sqrt{2\pi}\sigma_p} e^{-\frac{(x-p)^2}{2\sigma_p^2}} dx \tag{4.51}$$

Ici la taille de la zone d'incertitude est infinie. On peut seulement considérer une zone dans laquelle x a une forte probabilité de se trouver. Mais on peut aussi considérer que toute "l'énergie" d'une Gaussienne est limitée à un intervalle donné :

$$P(x \in [p - 3\sigma_p, p + 3\sigma_p]) \simeq 100\% \tag{4.52}$$

On peut donc écrire :

$$\Delta p \simeq 3\sigma_p \Rightarrow \sigma_p \simeq \frac{\Delta p}{3} \tag{4.53}$$

Dans la pratique les valeurs de σ sont entrées au début du processus par l'utilisateur. Pour c il est donc fixé tout au long du processus. Il en est de même pour θ, bien que l'erreur sur θ dépende de L. Pour simplifier les calculs à l'intérieur du suivi, nous avons préféré éliminer les petits segments ayant une grande imprécision en θ et laisser l'utilisateur choisir une valeur qui corresponde à peu près correctement pour tous les autres segments plus grands.

En ce qui concerne d et L, dont l'incertitude dépend de L, on laisse l'utilisateur choisir un facteur multiplicatif pour chacun de ces deux paramètres : $scale_d$ et $scale_L$. Les variances se déduisent donc pour chaque indice entré de la manière suivante :

$$\sigma_d = scale_d * L \tag{4.54}$$

$$\sigma_L = scale_L * L \tag{4.55}$$

Les incertitudes de mesure des différents paramètres sont donc déterminées arbitrairement à l'initialisation du processus. Celles sur d et L sont néanmoins toujours proportionnelles à L.

Mise en correspondance

Pour rechercher quels indices image mettre en correspondance avec chacun des indices du modèle, le processus de suivi procède de la façon qui a été décrite dans la section 4.1.5.

Chacun des indices modèle est d'abord comparé à tous les indices images. Une correspondance potentielle entre un modèle et une image est validée si elle satisfait une série de tests portant sur chacun des paramètres, dans l'ordre : θ, c, d, L. θ

est testé en premier car c'est le paramètre le plus sélectif, ce qui permet de ne pas exécuter les autres tests lorsque ce premier test est négatif. Les tests effectués sont ceux décrits dans l'équation 4.38, avec $V_i = V$ une constante choisie arbitrairement.

Pour chaque appariement ayant vérifié les quatre tests, un coût est calculé basé sur la distance de Mahalanobis (cf 4.39). Seuls θ et L sont considérés dans cette distance, car les valeurs $\Delta\theta$ et ΔL représentent plus la ressemblance entre deux segments que Δc et Δd qui décrivent eux un éloignement spatial. Le coût est donc donné par la formule :

$$coût = \frac{(\Delta L)^2}{\sigma^2_{L_m}} + \frac{(\Delta\theta)^2}{\sigma^2_{\theta_m}} \tag{4.56}$$

Pour chaque indice modèle le suivi obtient donc plusieurs indices image correspondants. Après quoi il choisi celui ayant le moindre coût. Lorsque tous les segments du modèle ont été traités, si un segment image a correspondu à plusieurs modèles, seule la correspondance de coût le plus faible est conservée (voir figure 4.12).

La mise en correspondance fournit donc un pointeur sur indice image pour chaque modèle, ou alors un indicateur spécifiant qu'il n'y a pas de correspondant.

Mise à jour

La mise à jour du modèle consiste dans un premier temps à faire évoluer le filtre de Kalman sur chaque paramètre. Dans le cas où un indice modèle n'a pas trouvé de correspondant dans l'image, le filtrage est effectué avec lui même. C'est à dire que :

$$z = x$$

$$r = \sigma_x^2$$

Ceci à pour effet de diminuer les valeurs de la matrice de covariance de l'erreur du modèle. Cet effet n'a pas trop d'incidence car dans la plupart des cas si les indices n'ont pas trouvé de correspondant c'est parce qu'ils ne sont plus physiquement dans l'image et peuvent disparaître. Pour ceux qui n'ont disparu que momentanément, cette façon de procéder permet de poursuivre le suivi dans la plupart des cas.

Le modèle, lorsqu'il a été mis en correspondance, hérite d'autre part des coordonnées des points extrémités du segment image. Ceci afin de conserver en mémoire les caractéristiques des indices observés.

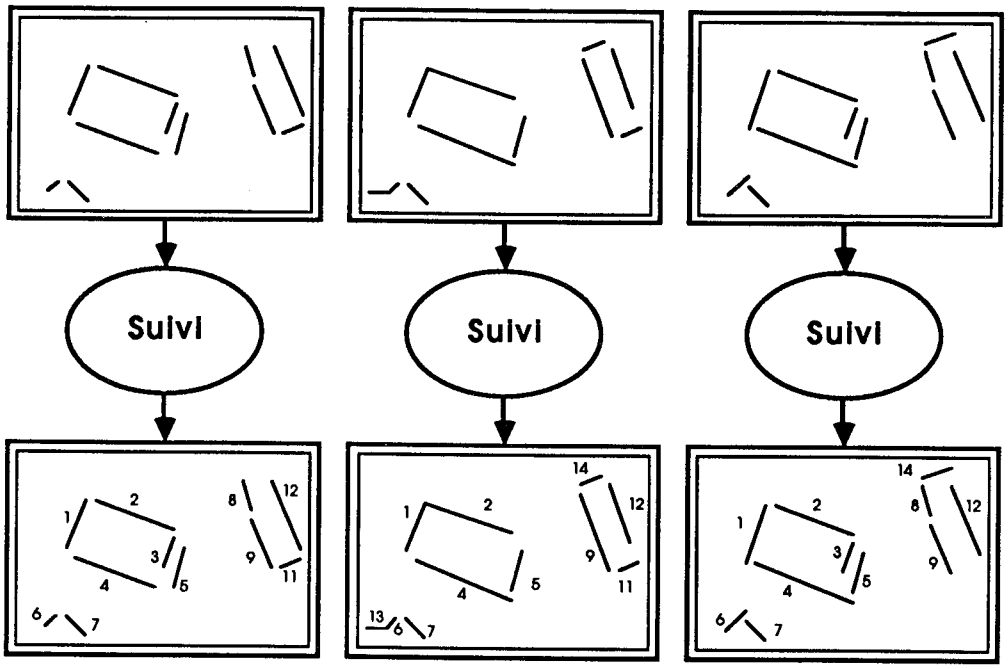


Figure 4.16: Le suivi vu comme un filtre

Enfin le facteur de confiance évolue, en augmentant si le modèle a un correspondant ou en diminuant sinon.

Elimination / Ajout

Lorsqu'un indice modèle possède un facteur de confiance inférieur ou égal à $MINCONF$ il est éliminé de la liste des indices modèle. Les indices image ne correspondant à aucun modèle sont quant à eux ajoutés à la fin du modèle. Leur filtre de Kalman est initialisé avec les données observées, leur facteur de confiance est mis à $DEFAULTCONF$ et un numéro d'identification leur est assigné.

Exploitation des résultats

Comme le suivi est utilisé pour reconnaître des segments dans une succession d'images, les résultats fournis sont uniquement les coordonnées des points extrémités conservées dans les modèles et les numéros d'identification, et ce uniquement pour les indices modèle qui ont été appariés dans la dernière image. Cela revient en fait à fournir les segments qui étaient présents dans la dernière image : en effet on peut voir ce suivi d'indices comme un filtre prenant en entrée des segments de droite et fournissant en sortie ces mêmes segments munis d'un numéro permettant de les

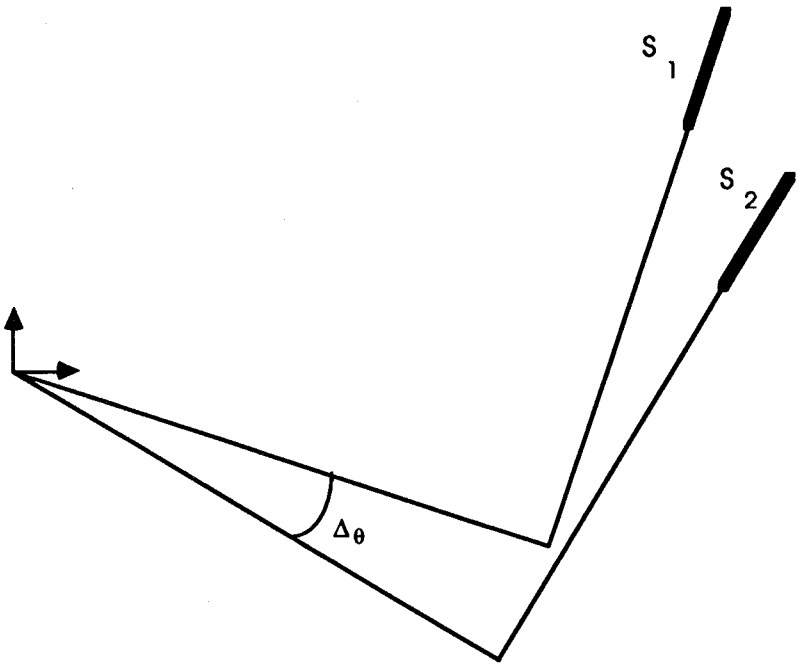


Figure 4.17: L'effet de bras de levier

identifier tout au long d'une séquence (figure 4.16).

4.2.3 Contraintes supplémentaires

Dans la pratique le suivi tel que nous venons de le décrire ne s'avère pas totalement satisfaisant par rapport à ce que l'on souhaiterait. En fait ce n'est pas tellement le nombre des erreurs qui est génant mais plutôt la façon dont elles se présentent.

Ces erreurs proviennent principalement du fait que ni c ni d ne sont vraiment indépendants de θ, malgré l'hypothèse que nous avons faite et sur laquelle est basé l'algorithme. Nous avons représenté ce type d'erreur dans la figure 4.17 : On peut voir que les deux segments S_1 et S_2 ne diffèrent que par θ, Δc et Δd étant négligeables. Dans ce cas nous ne vérifions plus l'hypothèse que les deux valeurs de θ sont très proches : Les assertions selons lesquelles c représente la distance perpendiculaire et d le glissement sont fausses.

Une première idée pour résoudre ce problème serait de ne considérer les correspondances entre des segments ayant des valeurs de θ vraiment très voisines. Malheureusement, ceci entraine l'élimination d'une grande partie du suivi des segments petits et moyens.

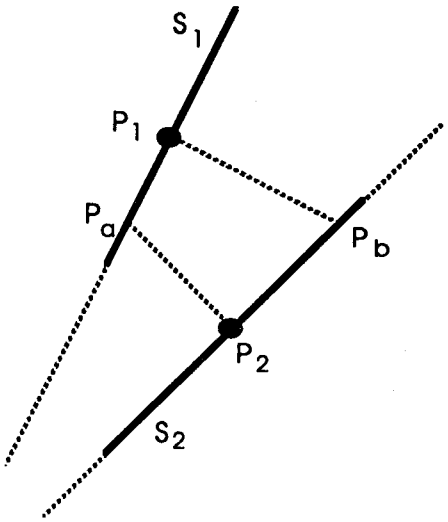


Figure 4.18: Distance perpendiculaire et glissement

L'effet de "bras de levier" n'intervient en fait que pour des valeurs de c ou d assez élevées, lorsqu'un petit déplacement en θ entraine un grand déplacement en x et y. Une deuxième idée est donc de modifier le repère dans lequel sont fournies les coordonnées des segments. Le repère de base est généralement situé en haut à gauche de l'image. En prenant un repère au centre de l'image, on diminue de moitié les valeurs possibles pour c et d. Cette solution diminue le problème sans totalement le résoudre. Par contre elle améliore sensiblement le comportement global du suivi.

Si l'on reprend le schéma du problème (figure 4.18), on voit que la distance perpendiculaire et le glissement sont donnés par les distances entre les points milieux P_1 et P_2 et les points P_a et P_b. Outre le fait que la notion de distance perpendiculaire n'est plus très bien définie (est-ce la distance du point P_1 à la droite S_2 ou du point P_2 à la doite S_1 ?), le calcul exact de cette distance va nous demander beaucoup de calculs et on risque de perdre l'avantage des simplifications que nous avons apportées jusqu'à maintenant.

Nous avons plutôt choisi un test simple à écrire et résolvant totalement le problème. Ce test consiste à comparer la distance cartésienne entre les points milieux avec la somme des demi-longueurs. Il s'écrit de la façon suivante :

$$t \Leftrightarrow \| \vec{P_{m_1} P_{m_2}} \| < h_{L_1} + h_{L_2} \tag{4.57}$$

Ce qui est équivalent à :

$$t \Leftrightarrow \vec{P_{m_1} P_{m_2}}^2 < (h_{L_1} + h_{L_2})^2$$

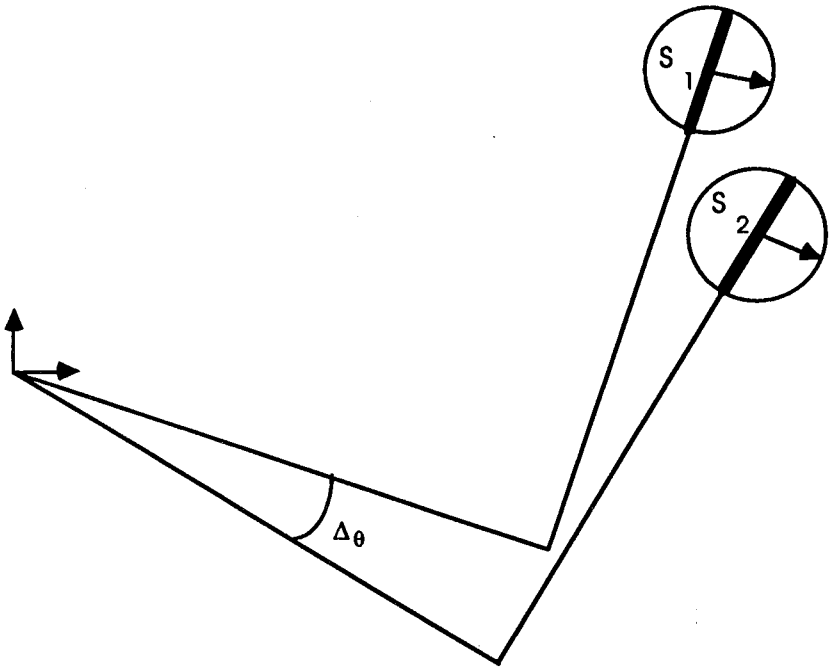


Figure 4.19: Visualisation du test sur x et y

$$\Leftrightarrow (x_{m_1} - x_{m_2})^2 + (y_{m_1} - y_{m_2})^2 < (h_{L_1} + h_{L_2})^2 \tag{4.58}$$

Où P_{m_1} est le point milieu du segment modèle et P_{m_2} celui du segment image, P_{m_1} est donné par :

$$\begin{cases} x_{m_1} = d_1 \cos\theta_1 - c_1 \sin\theta_1 \\ y_{m_1} = d_1 \sin\theta_1 + c_1 \cos\theta_1 \end{cases} \tag{4.59}$$

Tandis que P_{m_2} peut être directement extrait des valeurs observées :

$$P_{m_2} = \frac{P_{1_2} + P_{2_2}}{2} \tag{4.60}$$

Ce test est visuellement représenté dans la figure 4.19 : chaque segment est considéré comme un disque de diamètre L et le test vérifie que les deux segments, c'est à dire les deux disques, se recouvrent. Comme il n'est pas tenu compte de la forme des segments, entre autre de la différence d'incertitude entre c et d, les tests sur c et d sont conservés, et les résultats sont satisfaisants.

C'est la version découlant de l'ajout de ce test qui a servi de base pour l'évaluation.

4.2.4 Réalisation cablée d'un tel suivi : nécessité de simplifications pour alléger la structure matérielle

La première simplification que l'on puisse faire concerne l'utilisation du filtre de Kalman sur L. On peut remarquer que L n'a pas de vitesse et que la variance $\sigma_{\dot{L}}$ et les covariances $\sigma_{L\dot{L}}$ sont initialisées à 0. Les équations du filtre deviennent dans ce cas :

$$\sigma^2_{L_{n+1}} = \sigma^2_{L_n}(1 - k_x)$$

$$\sigma^2_{L_{n+1}} = \frac{\sigma^2_{L_n}\sigma^2_{L_o}}{\sigma^2_{L_n} + \sigma^2_{L_o}} \tag{4.61}$$

$$L_{n+1} = L_n + k_x(L_o - L_n)$$

$$L_{n+1} = \frac{\sigma^2_{L_o}L_n + \sigma^2_{L_n}L_o}{\sigma^2_{L_n} + \sigma^2_{L_o}} \tag{4.62}$$

Lorsque $\sigma^2_{L_n} \ll \sigma^2_{L_o}$ seul $\sigma^2_{L_o}$ sert à déterminer la zone de recherche. $\sigma^2_{L_n}$ sert uniquement à calculer L_{n+1} à partir de L_n et L_o. En utilisant une formulation de 4.62 où les paramètres sont fixés, il n'est pas nécessaire d'utiliser et de calculer σ_L. On peut utiliser une équation du type :

$$L_{n+1} = \frac{7L_n + L_o}{8} \tag{4.63}$$

Où L évolue lentement en fonction de L_o, ce qui correspond à peu près au comportement du filtre de Kalman après plusieurs mesures.

Une autre simplification consiste à ne plus faire dépendre les zones de recherche des valeurs calculées par le filtre, mais prendre des zones constantes, en particulier pour θ et c. Ces zones peuvent être constantes avec une seule valeur ou par des paliers de taille de plus en plus restreinte pour simuler partiellement le comportement des zones obtenues par l'utilisation du filtre de Kalman.

Le test que nous avons introduit dans la section 4.2.3 nécessitent le calcul d'un cosinus et d'un sinus, deux fonctions coûteuses à réaliser matériellement. Malheureusement les évaluations montrent que ce test est nécessaire au bon fonctionnement du suivi. Nous avons donc refait un suivi utilisant les paramètres c, θ, L, x_m et y_m. Malgré la forte dépendance qu'il y a entre x_m et y_m, nous avons utilisé deux filtres indépendants pour ces paramètres. d n'est plus nécessaire : on peut remarquer que le test de similarité en d est contenu dans le test de proximité en x et y. Cette transformation consiste donc à remplacer d par x_m et y_m. c est peu

corrélé avec x_m ou y_m alors que d l'est beaucoup. c et x_m et y_m sont donc estimés indépendamment par le filtre de Kalman ce qui permet de ne pas utiliser les formules permettant de passer des uns aux autres.

Malgrè l'hypothèse fausse utilisée – l'indépendance entre x_m et y_m – le suivi ainsi réécrit fonctionne correctement, c'est à dire avec à peu près les mêmes résultats que dans la version précédente utilisant c, d, θ et L.

L'utilisation de ces nouveaux paramètres pose le problème du choix des incertitudes utilisées par le filtre de Kalman. Pour la valeur de la matrice initialisant le filtre on peut prendre pour x_m et y_m les mêmes valeurs que celles utilisées pour l'initialisation de d. Quant à la variance sur la mesure, plutôt que d'être proportionnelle à la longueur, ce qui est le cas pour d, nous avons choisi de l'entrer directement en paramètre à l'initialisation du processus, comme dans le cas de c.

Enfin la dernière simplification, et pas la moindre, concerne le fonctionnement du filtre de Kalman. Dans les équations que nous avons formulées, on peut remarquer qu'à chaque itération la matrice de covariance de l'erreur du modèle P et le gain K ne dépendent que des anciennes valeurs de P, des valeurs de la matrice de covariance du bruit du modèle Q et de la valeur de la variance du bruit de mesure r. Or les valeurs de Q que nous utilisons restent les mêmes tout le long du suivi, et pour Θ et c nous avons fixé les valeurs de r à l'initialisation. Dans ce cas P et K ne dépendent plus des observations et leurs valeurs successives peuvent être calculées hors ligne (et même cablées). Le fonctionnement du filtre de Kalman se réduit alors au calcul des formules 4.31 et 4.34.

4.3 Expérimentations

Nous présentons ici les résultats obtenus avec deux séquences d'images. Dans ces expérimentations nous avons modélisé le suivi uniquement pour numéroter les segments de droite, et nous montrons le résultat de façon visuelle en matérialisant la liaison qui existe entre les segments de deux images différentes de la séquence.

4.3.1 Conditions d'expérimentation

Pour réaliser ces expérimentations nous avons placé une caméra dans la pince d'un robot manipulateur qui a effectué un déplacement autour d'objets placés sur un plateau. Ces conditions correspondent à celles des expériences sur la méthode de reconstruction que nous exposons dans le chapitre suivant.

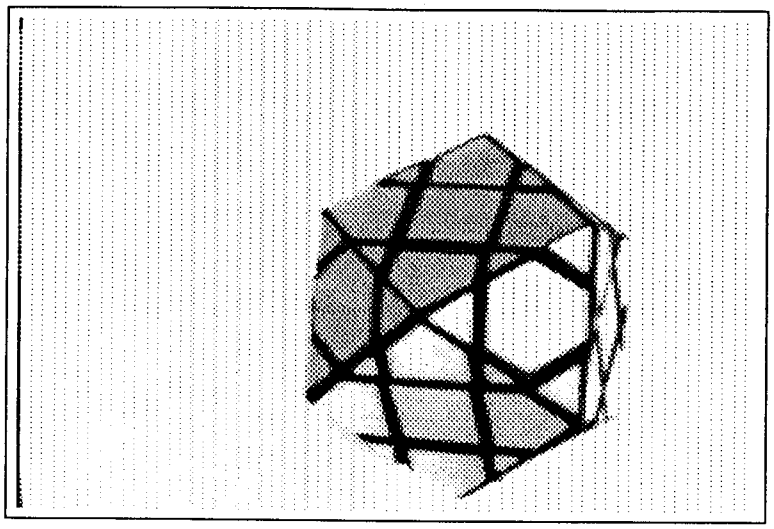


Figure 4.20: Une des images de la première séquence (image 5).

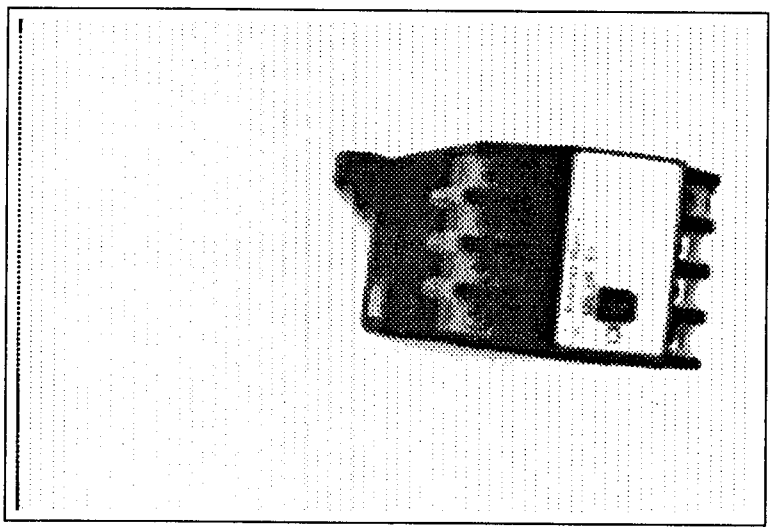


Figure 4.21: Une des images de la deuxième séquence (image 5).

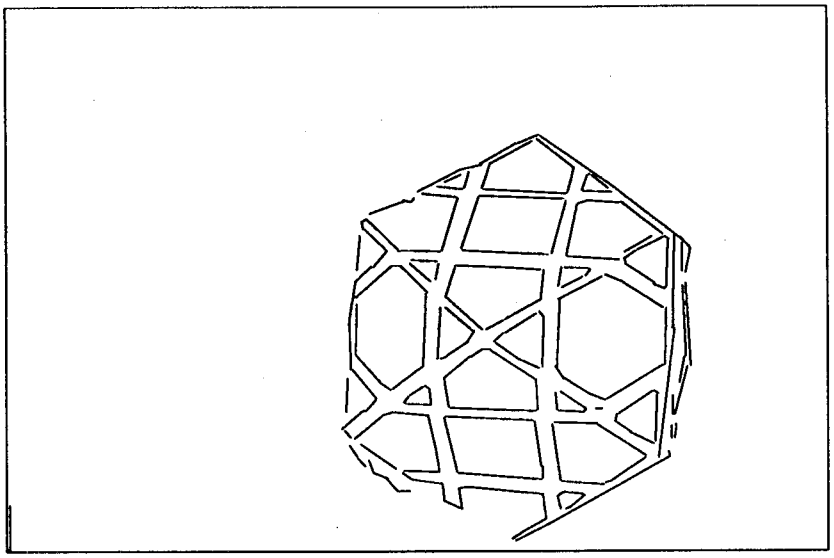


Figure 4.22: Première série : image 1.

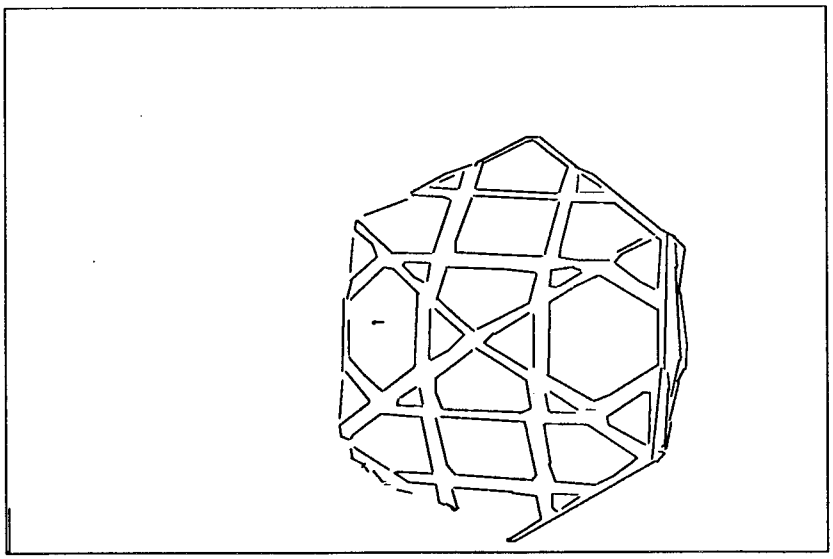


Figure 4.23: Première série : image 2.

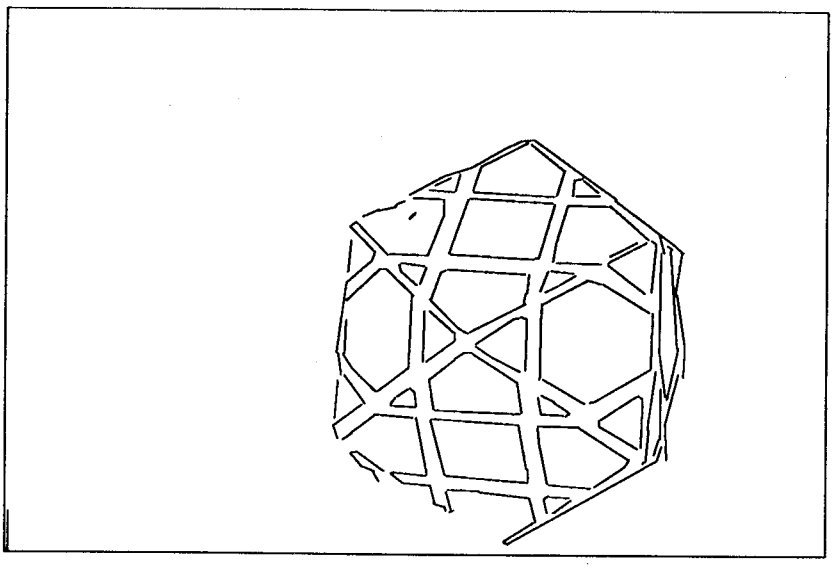


Figure 4.24: Première série : image 3.

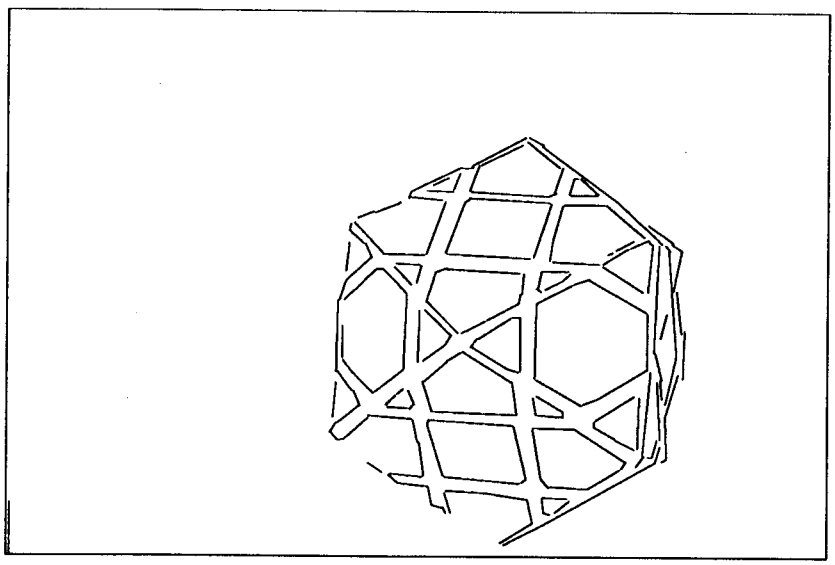


Figure 4.25: Première série : image 4.

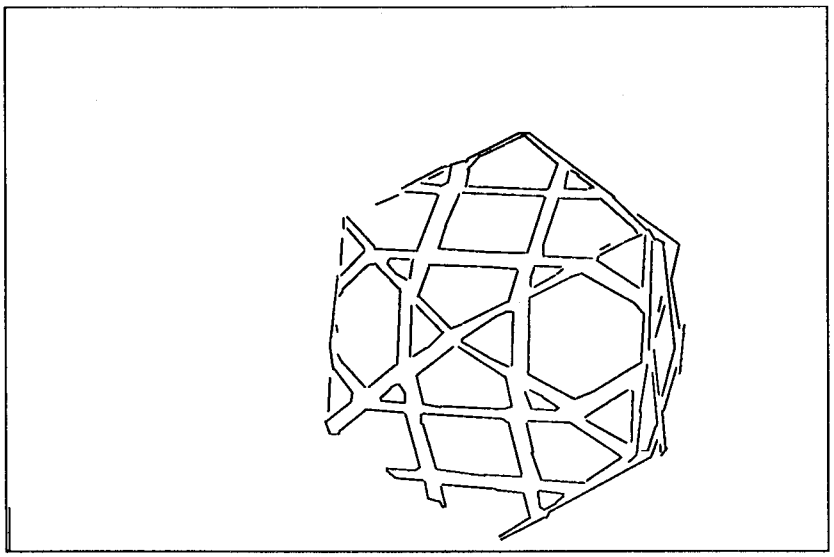


Figure 4.26: Première série : image 5.

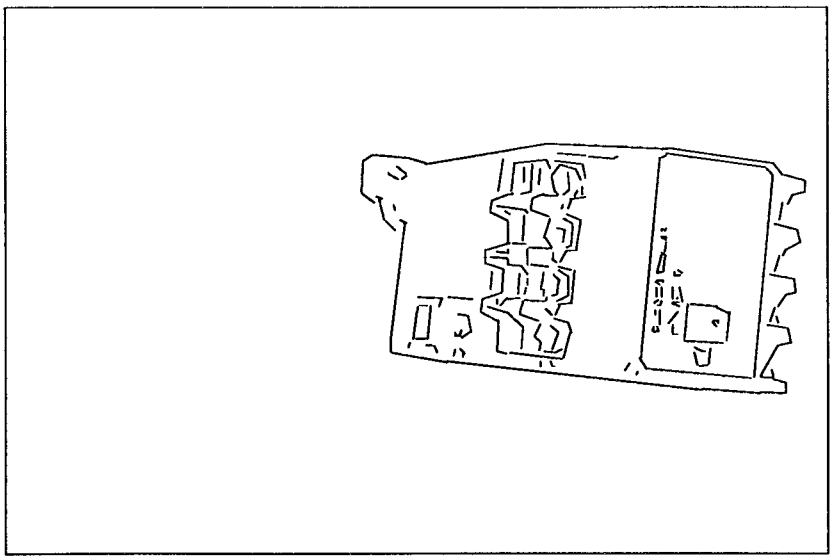


Figure 4.27: Deuxième série : image 1.

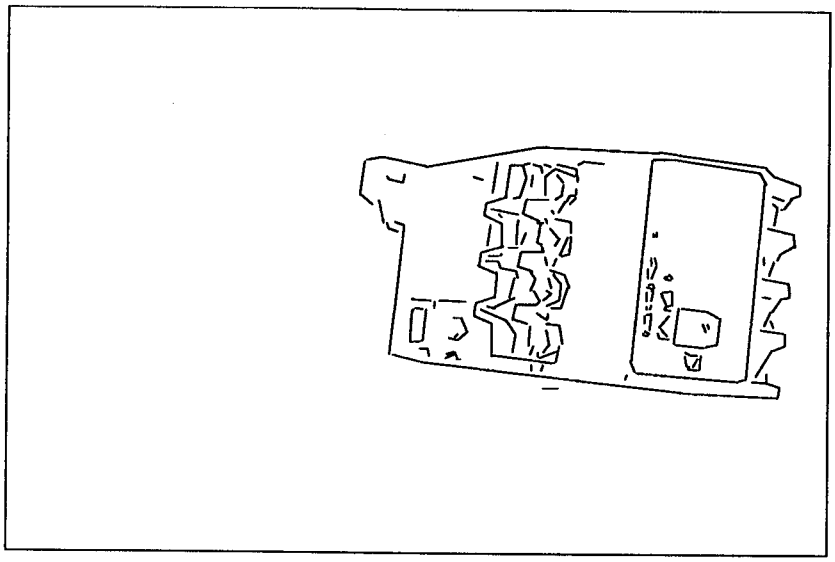


Figure 4.28: Deuxième série : image 2.

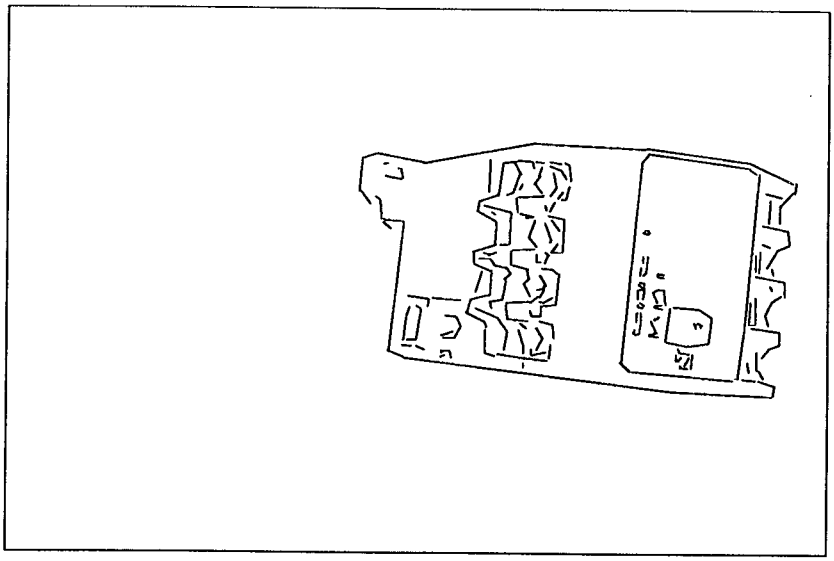


Figure 4.29: Deuxième série : image 3.

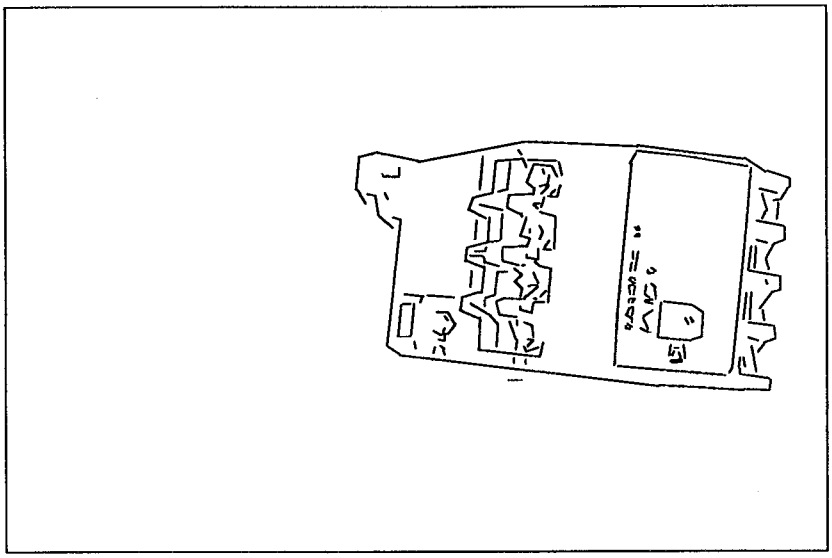


Figure 4.30: Deuxième série : image 4.

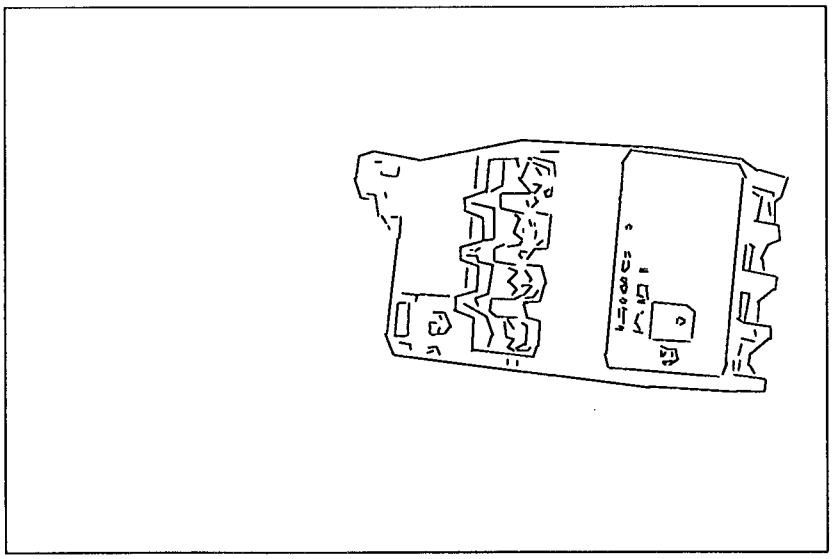


Figure 4.31: Deuxième série : image 5.

Nous montrons ici deux séquences, chacune avec un objet différent : La première série (figure 4.20) avec un "Rubik's cube" à coins coupés et la seconde (figure 4.21) avec un disjoncteur. Les figures 4.22 à 4.26 montrent, pour la première série, la séquence des cinq premières images de segments de droite sur lesquelles nous allons travailler. De même les figures 4.27 à 4.31 montrent les cinq premières images de la deuxième série.

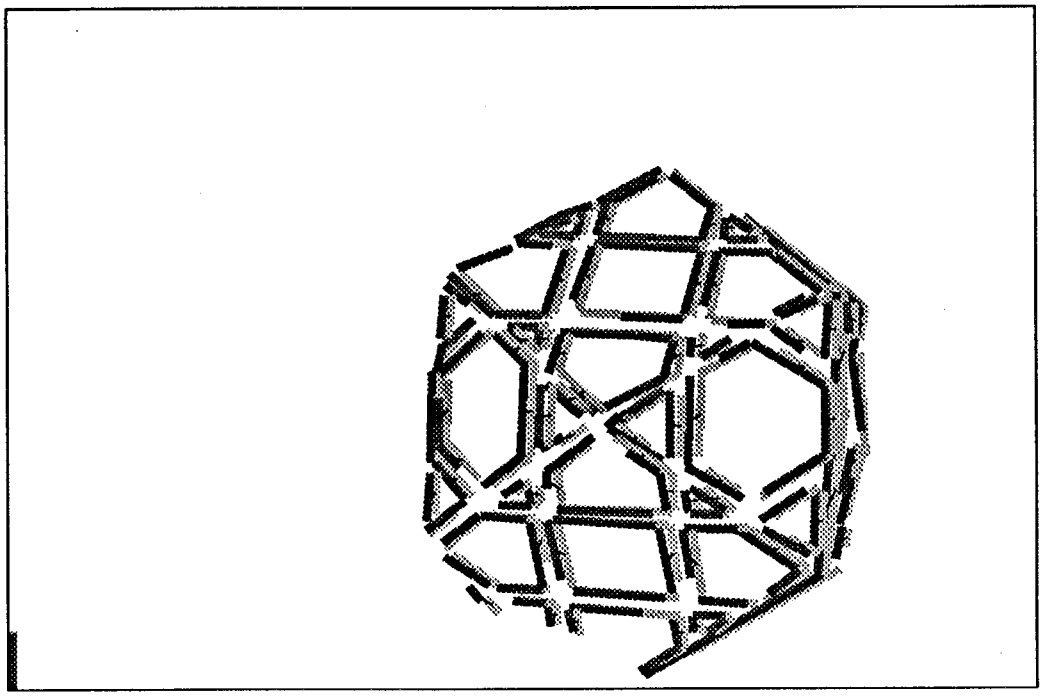


Figure 4.32: Première série : suivi entre l'image 1 (141 segments) et l'image 2 (148 segments). 119 segments ont été suivis.

Pour la première série, les figures 4.32 à 4.35 montrent le suivi entre deux images successives (de la première à la deuxième, de la deuxième à la troisième, etc.) et la figure 4.36 le suivi résultant entre la première et la cinquième. Les figures 4.37 à 4.40 et 4.41 montrent le même type de résultats pour la seconde série.

4.3.2 Suivi sur une longue séquence

Jusqu'à maintenant nous avons seulement montré des résultats sur des séquences assez courtes (cinq images). Il est intéressant de voir comment se comporte le suivi sur une plus longue séquence. Pour la première série (celle du "Rubik's cube"), la figure 4.42 montre le suivi entre la cinquième et la quinzième image et la figure 4.43 entre la cinquième et la vingt-cinquième. On peut remarquer que sur une aussi longue séquence, l'objet ayant tourné, de nombreux indices sont cachés, et que ceci est la principale cause de perte du suivi de nombreux segments.

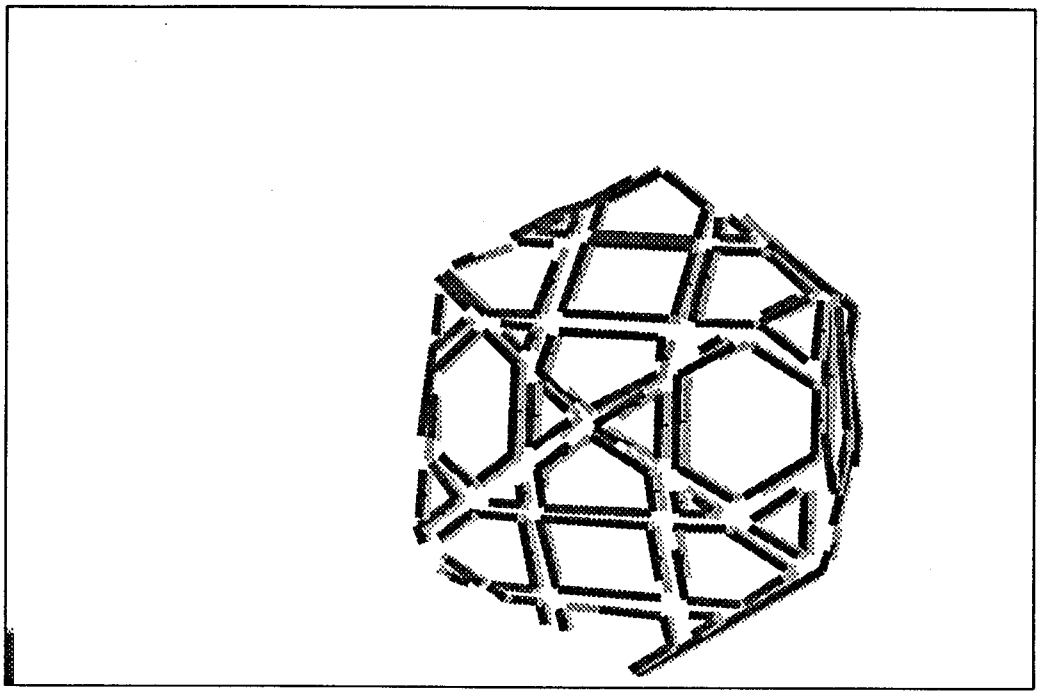


Figure 4.33: Première série : suivi entre l'image 2 (148 segments) et l'image 3 (139 segments). 117 segments ont été suivis.

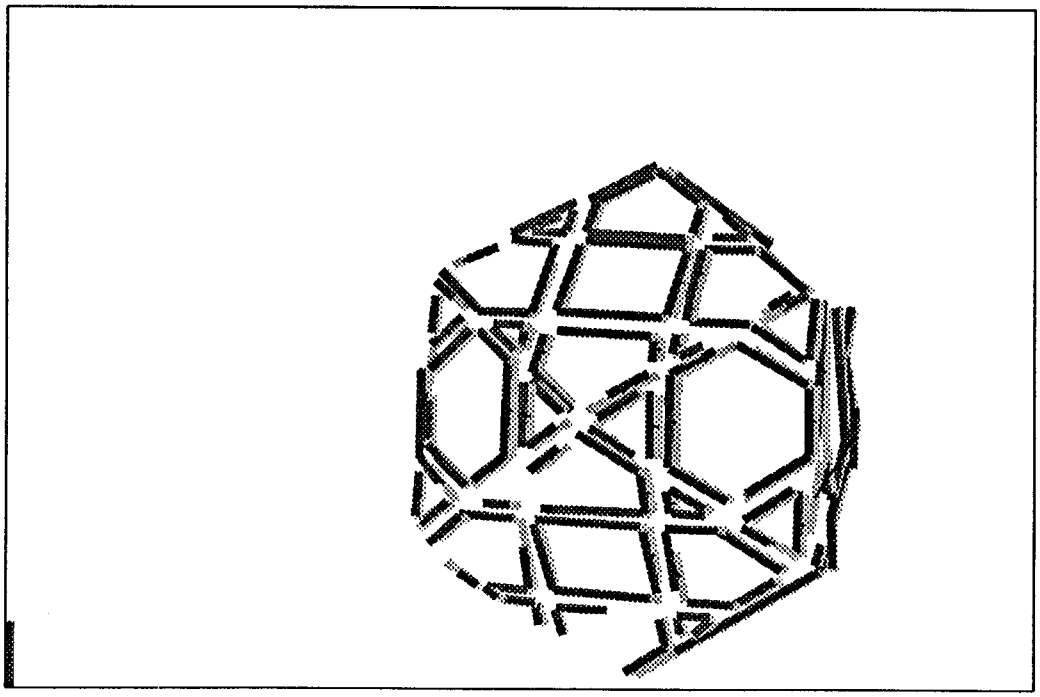


Figure 4.34: Première série : suivi entre l'image 3 (139 segments) et l'image 2 (143 segments). 115 segments ont été suivis.

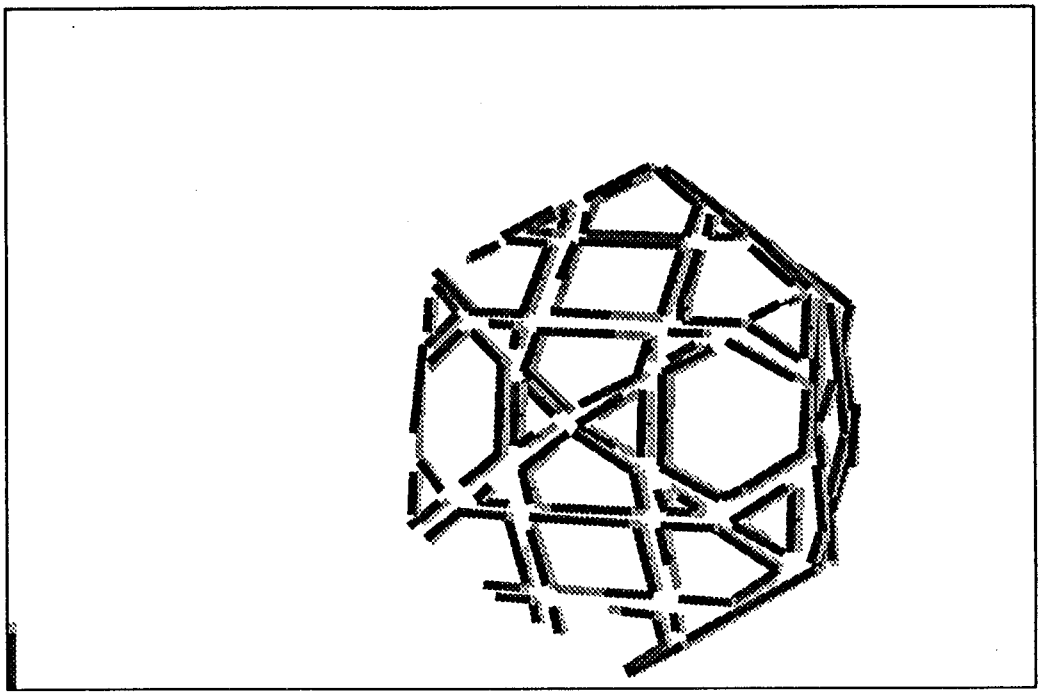


Figure 4.35: Première série : suivi entre l'image 4 (143 segments) et l'image 5 (138 segments). 114 segments ont été suivis.

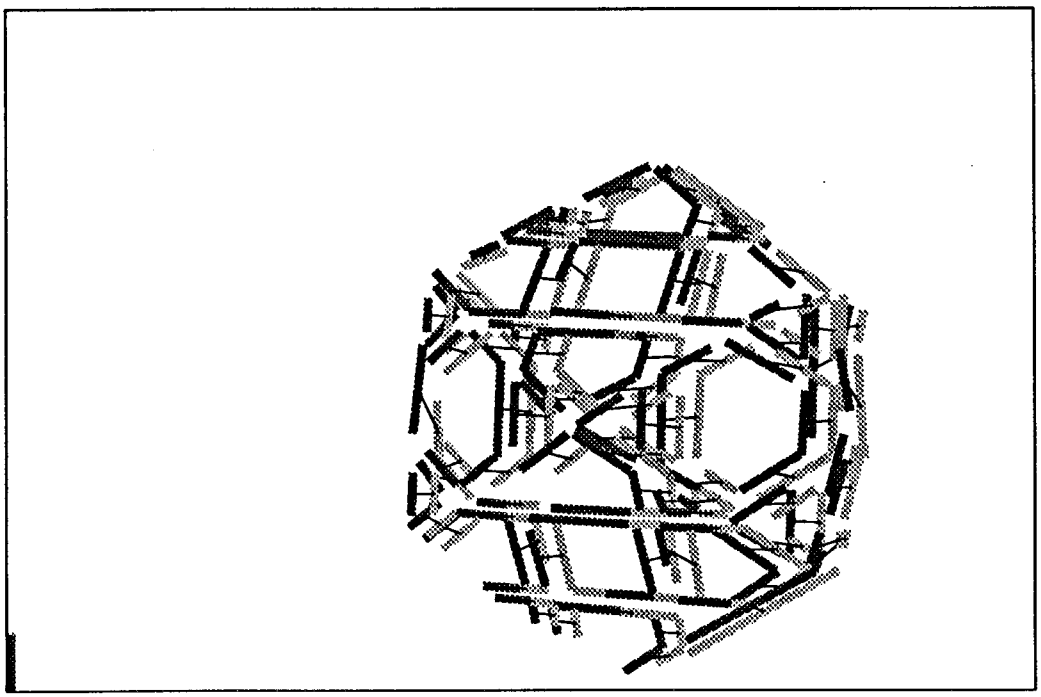


Figure 4.36: Première série : suivi entre l'image 1 (141 segments) et l'image 5 (138 segments). 99 segments ont été suivis.

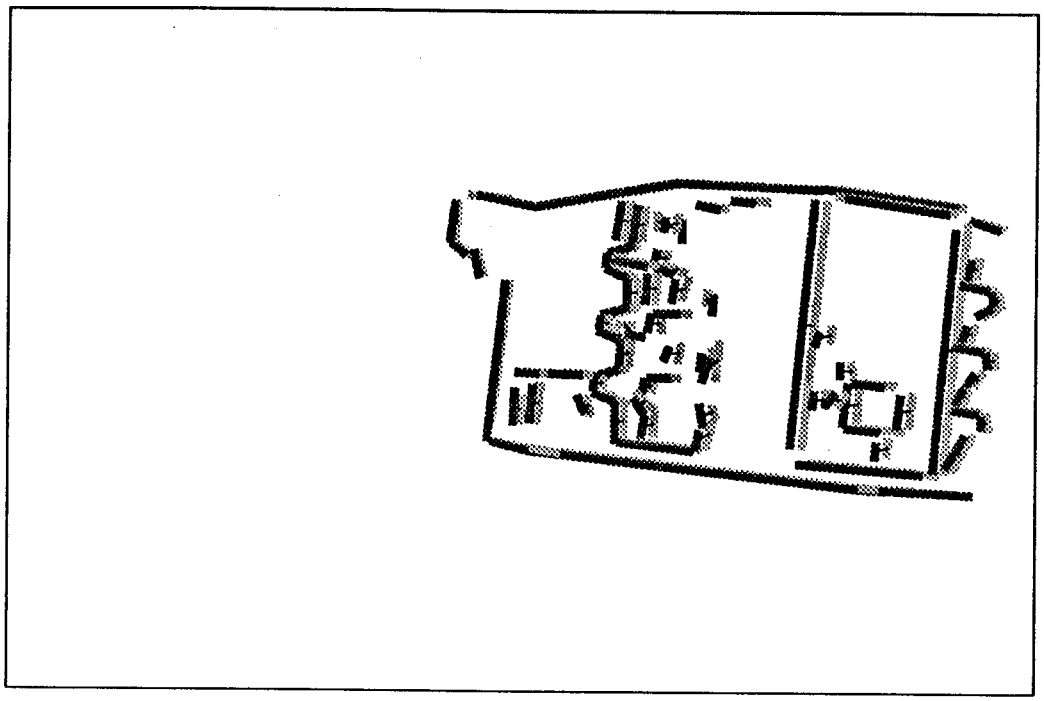


Figure 4.37: Deuxième série : suivi entre l'image 1 (110 segments) et l'image 2 (107 segments). 79 segments ont été suivis.

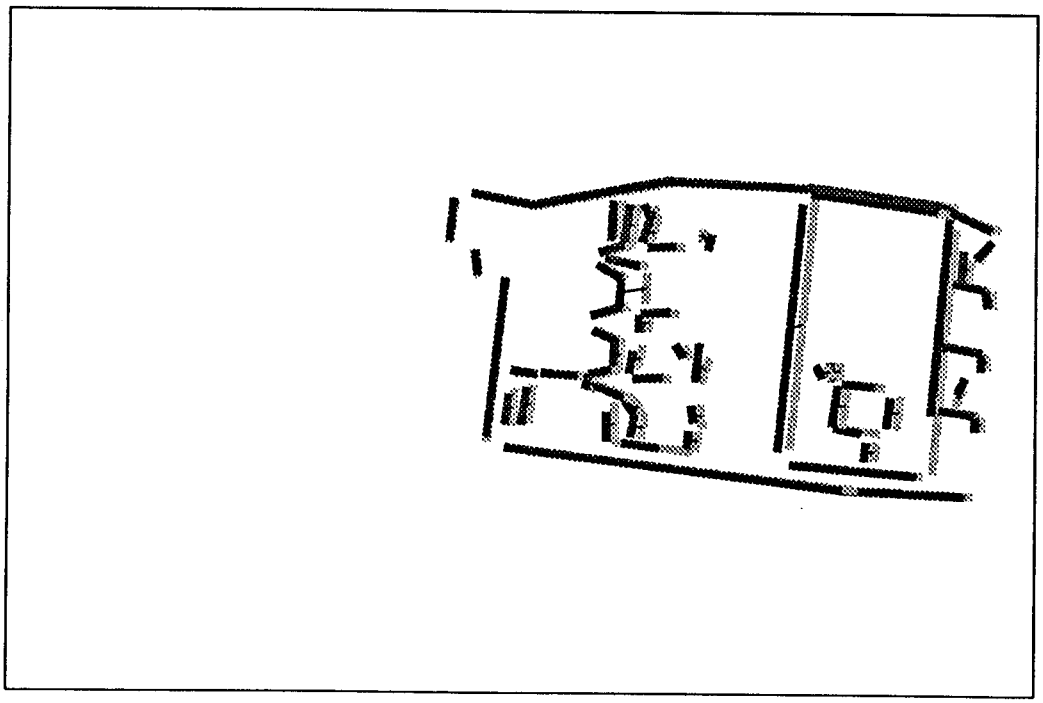


Figure 4.38: Deuxième série : suivi entre l'image 2 (107 segments) et l'image 3 (103 segments). 63 segments ont été suivis.

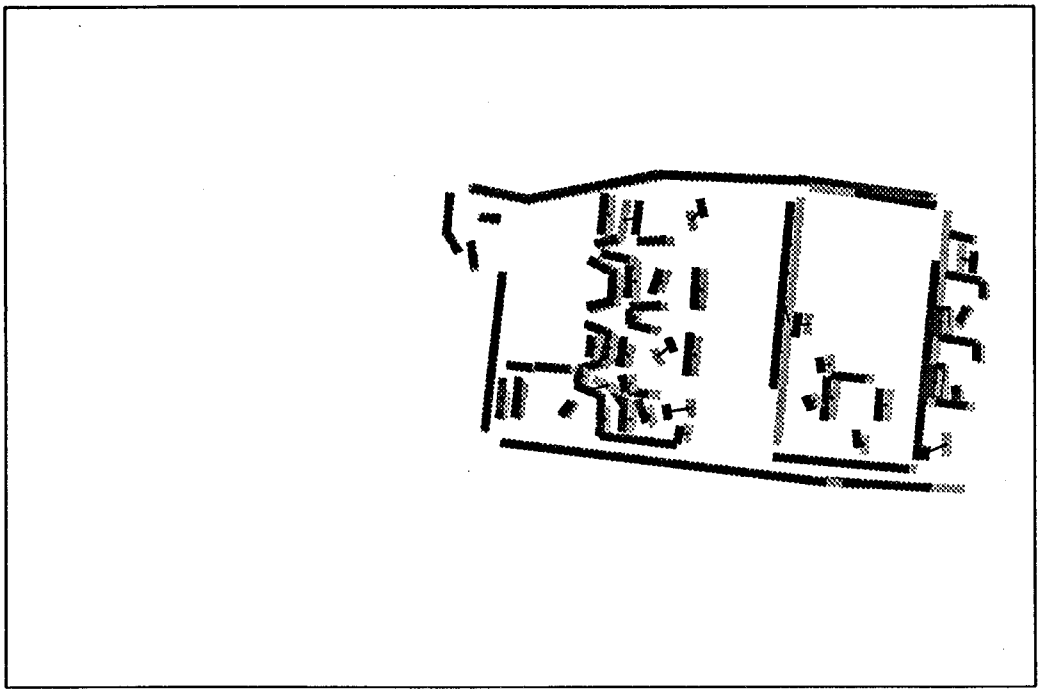


Figure 4.39: Deuxième série : suivi entre l'image 3 (103 segments) et l'image 4 (110 segments). 74 segments ont été suivis.

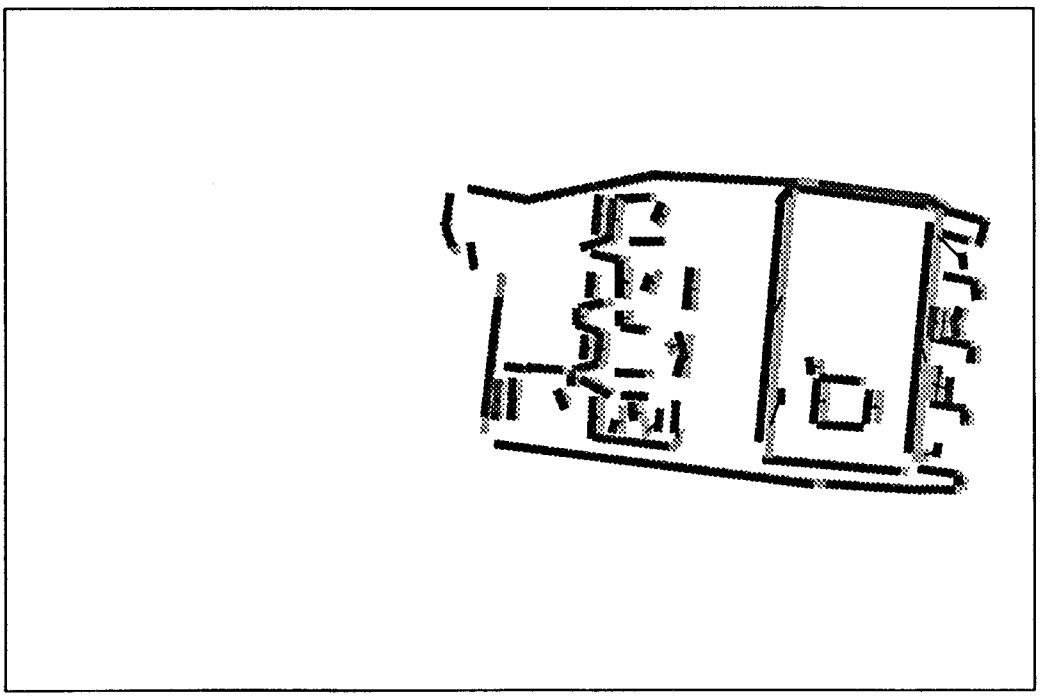


Figure 4.40: Deuxième série : suivi entre l'image 4 (110 segments) et l'image 5 (111 segments). 78 segments ont été suivis.

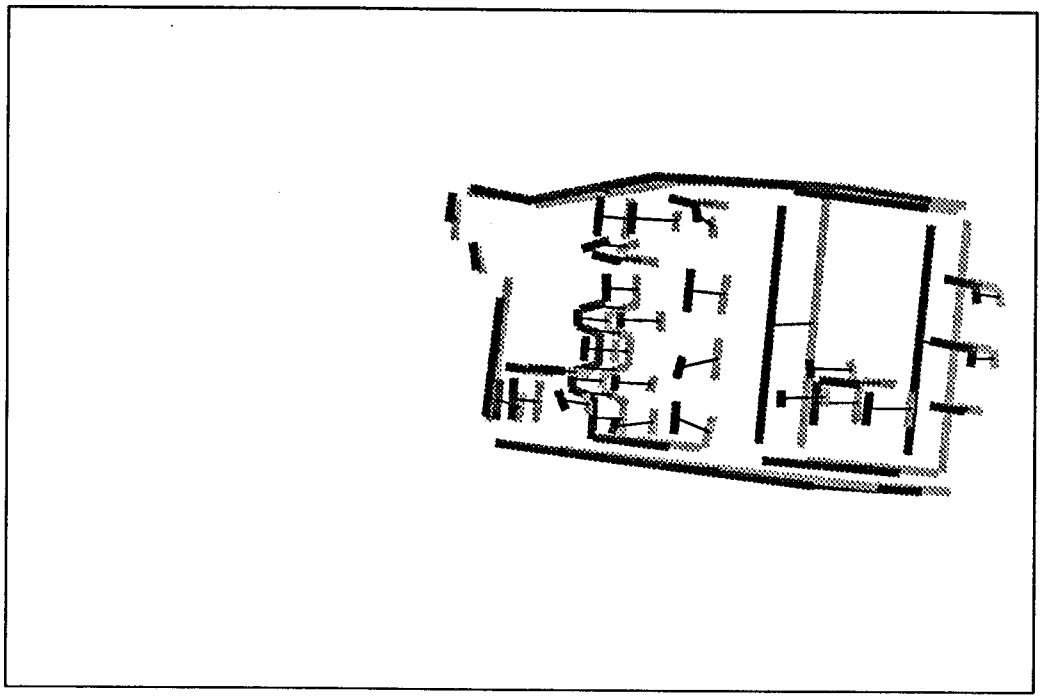


Figure 4.41: Deuxième série : suivi entre l'image 1 (110 segments) et l'image 5 (111 segments). 51 segments ont été suivis.

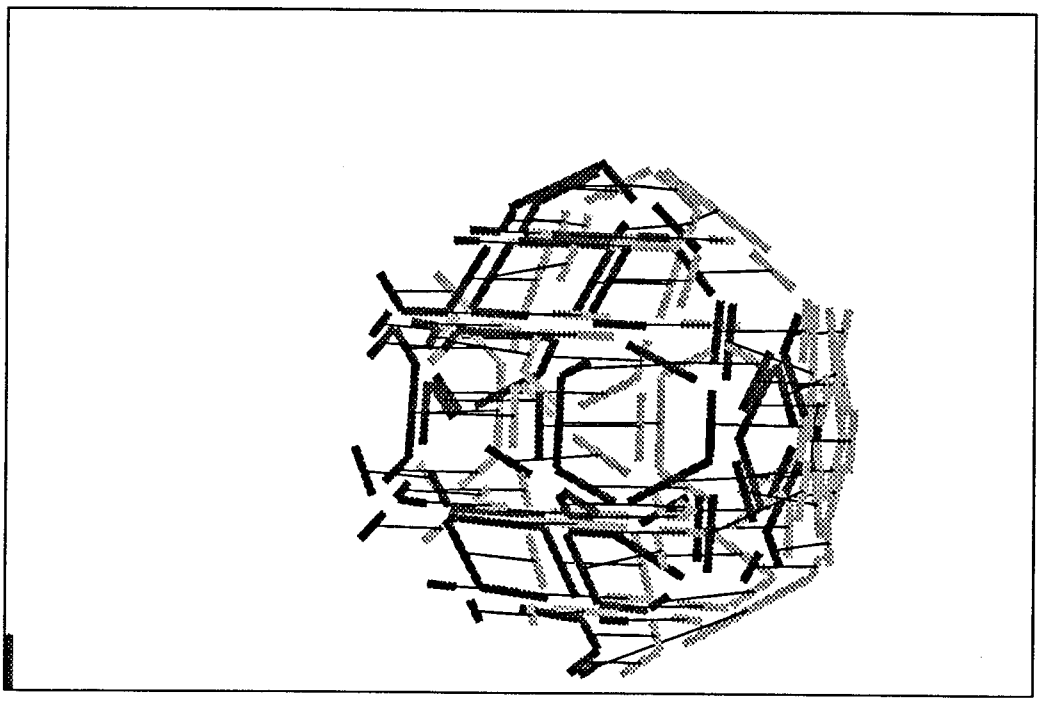


Figure 4.42: Première série : suivi entre l'image 5 (138 segments) et l'image 15 (161 segments). 86 segments ont été suivis.

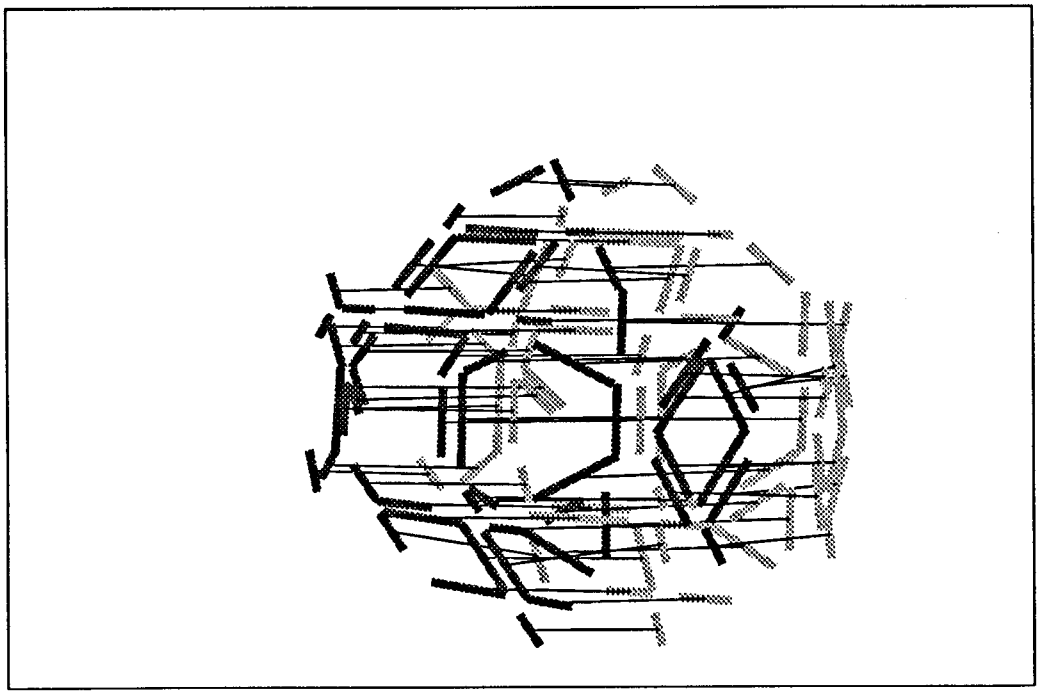


Figure 4.43: Première série : suivi entre l'image 5 (138 segments) et l'image 25 (159 segments). 59 segments ont été suivis.

Chapitre 5

Reconstruction 3-D

Le processus de suivi d'indices que nous venons de décrire est quelque chose d'assez fondamental dans l'analyse du mouvement. Il peut n'être qu'un simple module, parfois en plusieurs exemplaires, d'un système de vision plus global. Nous avons plus simplement voulu tester sa validité en utilisant la numérotation des indices ainsi fournie pour estimer les coordonnées 3-D de ces indices.

5.1 Quelle reconstruction ?

Nous avons vu dans 2.4 que de nombreuses méthodes utilisaient un suivi directement dans l'espace. Notre suivi ayant été effectué dans le plan image, nous allons maintenant regarder quelles informations vont être nécessaires pour mener à bien l'estimation des coordonnées 3-D[1].

5.1.1 Utilisation du suivi d'indices

De nombreuses données peuvent être fournies par le suivi d'indices, chacune permettant d'extraire différents types d'information sur le mouvement apparent.

[1] Ces travaux ont été effectués avec la société ITMI.

- Le suivi estime le mouvement de chacun des segments. Ce mouvement étant filtré il est très fiable et lors d'un déplacement à vitesse constante, une bonne estimation de cette vitesse est fournie. Ces valeurs étant calculées constamment elles peuvent être analysées parallèlement au fonctionnement du suivi. Parmi les différents paramètres représentant un segment, certains sont plus ou moins valables :

  - Le mouvement parallèle à la droite porteuse et la longueur sont estimés avec une très grande incertitude. Même si la précision réelle est dans la plupart des cas plus grande que celle trouvée par le suivi, il est préférable de ne pas utiliser directement ces données.
  - Les valeurs du déplacement angulaire sont assez fiables. Mais comme un bon fonctionnement du suivi n'est garanti qu'avec de faibles déplacements angulaires, ces valeurs sont en général de peu d'intérêt.
  - La valeur du déplacement perpendiculaire à la droite porteuse est réellement très fiable. C'est à partir de cette valeur que l'étude du mouvement peut être envisagée.

- Le suivi permet d'autre part de faire se correspondre des indices entre images successives, en fournissant à partir d'une image de segments de droite une autre image dont les segments sont numérotés et identifiables (figure 4.16). Cette information permet d'une part de retrouver les informations que nous venons d'évoquer plus haut et qui sont estimées par le suivi (position, vitesse). Mais dans ce cas le mouvement est calculé à partir des indices mesurés dans les images. Dans un certain nombre d'utilisations du suivi, comme l'aide à la stéréovision ou encore une étude du mouvement fondée sur l'utilisation des équations de la stéréovision, ces seules informations de correspondance suffisent.

L'utilisation des informations du mouvement en 2 dimensions dépend du type de mouvement en 3 dimensions que l'on envisage possible. Selon les types de mouvement 3D on peut ajouter des contraintes permettant de diminuer la sous-détermination des problèmes que l'on cherche à résoudre. On distingue les cas suivants de mouvement :

- Le capteur (caméra) et les objets observés sont en mouvement. C'est le cas le plus général et avec lequel il est difficile de retrouver beaucoup d'informations, sauf localement à un objet en faisant des hypothèses de rigidité.

- Le capteur est fixe, les objets sont en mouvement. Si on ne peut avoir aucun contrôle sur le mouvement des objets, nous avons les mêmes difficultés que dans le cas précédent.

- le capteur est mobile et les objets sont fixes. Par rapport au premier cas, la contrainte sur les objets observés n'est pas trop génante si l'on considère que dans la plupart des scènes réelles, très peu d'objets, pour ne pas dire aucun, sont vraiment en mouvement. C'est la raison pour laquelle cette contrainte est la plus utilisée, et c'est d'ailleurs celle que nous prendrons.

Dans le cas du capteur en mouvement dans un univers fixe, si ce mouvement n'est pas connu, l'étude des déplacements en 2 dimensions dans l'image permet de définir la direction du déplacement en 3 dimensions. D'autre part l'étude de la vitesse dans l'image de différents indices permet de retrouver la profondeur relative de leurs correspondants en 3 dimensions. Il n'est par contre pas possible de déterminer l'amplitude du mouvement du capteur sans informations sur l'univers observé, et la détermination de la profondeur réelle (ainsi que de toutes les coordonnées si la distance focale de la projection associée au capteur est connue) est tributaire de la connaissance de l'amplitude du déplacement de la caméra.

Lorsqu'une caméra est montée sur un robot mobile, il est difficile d'imposer que le déplacement commandé à ce robot soit bien le même que celui qui est effectué. Ceci à cause de la configuration du terrain sur lequel se déplace le robot, ainsi que des incertitudes dans le mécanisme de déplacement (roues). Tandis que si la caméra est montée sur un bras manipulateur, même si le mouvement effectué n'est pas exactement le même que celui qui est commandé, les axes du robot sont dotés de capteurs de position qui permettent de connaître avec beaucoup de précision le déplacement réel de la caméra.

C'est pourquoi nous avons envisagé de réaliser le montage d'une caméra sur un robot manipulateur, ce qui, le robot nous fournissant les coordonnées de la pince dans son repère, nous permet de connaître parfaitement le mouvement effectué.

5.1.2 Description du processus mis en oeuvre

Nous avons fixé une caméra dans la pince d'un bras manipulateur, puis nous faisons effectuer à ce bras un mouvement autour d'un objet. tout au long de ce mouvement, le suivi d'indices fournit des images de segments numérotés. En faisant une analyse du type de celle de la stéréovision entre deux instants différents, nous pouvons reconstruire en trois dimensions les segments extraits. Ce processus répété plusieurs fois au cours du mouvement donne toute une série de reconstructions différentes, ceci dans le but ultérieur de fusionner ces résultats 3D (figure 5.1).

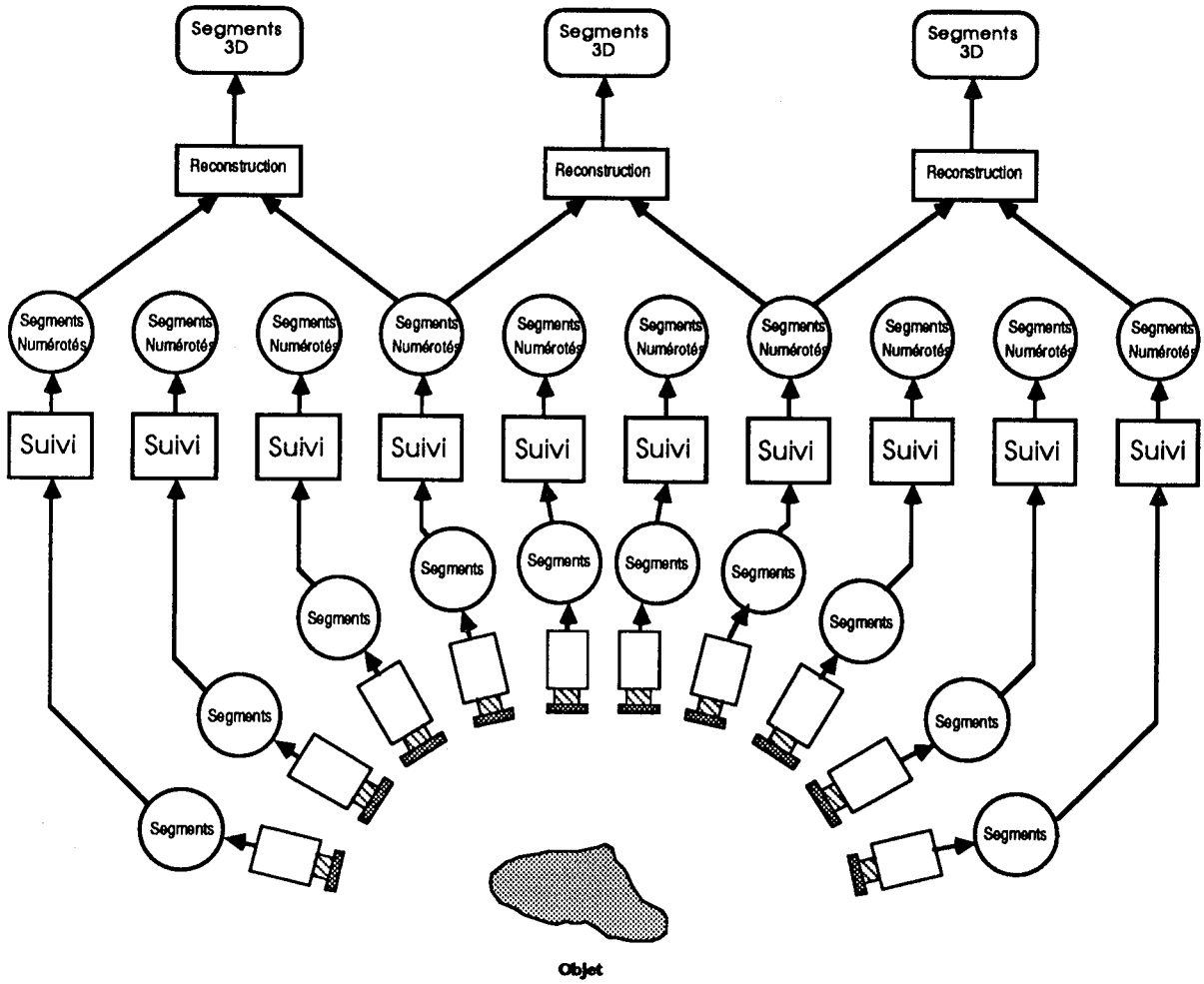


Figure 5.1: Processus effectué avec une caméra fixée dans la pince d'un robot

Calibrage

Avant de lancer ce processus, il faut procéder au calibrage du système robot + caméra. Cette partie est très importante car tous les résultats que nous fournirons dépendront totalement de ce calibrage. Il y a deux types d'informations dont nous avons besoin et donc deux calibrages à effectuer :

- Le calibrage de la caméra : Ce calibrage ([Tsa 86]) est effectué en observant à travers la caméra une mire dont les caractéristiques géométriques sont connues (emplacement, dimension) et fournit deux types de paramètres de la caméra :
  - Les paramètres intrinsèques : La distance focale, les facteurs d'échelle (qui permettent de retrouver la taille réelle des pixels), la position du centre de l'image.
  - Les paramètres extrinsèques : La rotation et la translation du déplacement rigide, qui permet de passer du repère de la caméra au repère de la scène.
- Le calibrage entre la caméra et le robot : les capteurs de position du robot nous donnent avec précision l'emplacement de la pince par rapport à unrepère fixe du robot (généralement situé dans le socle). Mais comme ce que nous observons l'est à travers de la caméra, il nous faut déterminer quel est le déplacement entre le repère de la caméra et celui de la pince.

Equations stéréo

Parmit les résultats fournis par le calibrage, les paramètres extrinsèques sont représentés sous la forme d'une matrice 3 X 4 T_i, où i est l'indice de l'image. Nous utiliserons par la suite trois vecteurs t^i_j extraits de T_i que nous définissons de la manière suivante [Aya 88] :

$$T_i = \begin{pmatrix} t^i_{11} & t^i_{12} & t^i_{13} & t^i_{14} \\ t^i_{21} & t^i_{22} & t^i_{23} & t^i_{24} \\ t^i_{31} & t^i_{32} & t^i_{33} & t^i_{34} \end{pmatrix}$$

$$t^i_j = \begin{pmatrix} t^i_{j1} \\ t^i_{j2} \\ t^i_{j3} \end{pmatrix} \tag{5.64}$$

Un point P de coordonnées x, y et z de l'espace se projette dans le plan de l'image i en un point I_i de coordonnées u_i et v_i. On utilise pour cela un intermédiaire de calcul $I^*_i = (U_i, V_i, S_i)$ appelé coordonnées projectives de I_i :

$$I^*_i = \begin{pmatrix} U_i \\ V_i \\ S_i \end{pmatrix} = T_i \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{5.65}$$

$$I_i = \begin{pmatrix} u_i \\ v_i \end{pmatrix} = \begin{pmatrix} U_i/S_i \\ V_i/S_i \end{pmatrix} \tag{5.66}$$

Lorsqu'on recherche les coordonnées d'un point P de l'espace connu par ses coordonnées images u_i et v_i, la formule 5.66 fournit deux équations par images :

$$\begin{cases} (t_1^i - u_i t_3^i)^\top P = u_i t_{34}^i - t_{14}^i \\ (t_2^i - v_i t_3^i)^\top P = v_i t_{34}^i - t_{24}^i \end{cases} \tag{5.67}$$

A partir de deux images on obtient un système surdéterminé. Ce système peut être résolu par une méthode des moindres carrés ou en utilisant le filtre de Kalman, et dans ce dernier cas il peut être intéressant d'utiliser plus que deux images.

Informations épipolaires

Lorsque nous allons reconstruire un segment en trois dimensions à partir de deux observations en deux dimensions d'un même segment, nous n'allons pas directement utiliser les segments observés. En effet la longueur et la position le long de la droite porteuse n'étant pas très sûres, nous n'effectuerons la reconstruction que sur une partie des segments 2D que l'on sait être commune en 3D. Pour cela nous utiliserons les informations épipolaires entre les deux images.

Un point dans une image est l'observation d'un point de la scène situé sur une droite passant par le centre de projection de la caméra et le point de l'image. La projection de cette ligne dans une seconde image forme ce que l'on appelle la ligne épipolaire du point de la première image dans la seconde.

De la même façon un segment dans une image fournit dans une seconde image une bande épipolaire qui est la zone comprise entre les deux lignes épipolaires des points extrémités. Nous ne prendrons donc que la partie commune entre un segment et la bande épipolaire de son correspondant de l'autre image pour la reconstruction.

Pour extraire les lignes épipolaires, nous recherchons d'abord les épipoles de chacune des images. L'épipole d'une image est la projection du centre de projection de l'autre image, et toutes les lignes épipolaires s'intersectent en ce point. Nous devons donc d'abord calculer les coordonnées des centres de projection en résolvant les systèmes suivants pour chacune des deux images :

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = T_i \begin{pmatrix} x_{C_i} \\ y_{C_i} \\ z_{C_i} \\ 1 \end{pmatrix} \tag{5.68}$$

En utilisant les formules 5.65 et 5.66 nous en déduisons :

$$\begin{pmatrix} U_{E_2} \\ V_{E_2} \\ S_{E_2} \end{pmatrix} = T_2 \begin{pmatrix} x_{C_1} \\ y_{C_1} \\ z_{C_1} \\ 1 \end{pmatrix} \quad et \quad \begin{pmatrix} U_{E_1} \\ V_{E_1} \\ S_{E_1} \end{pmatrix} = T_1 \begin{pmatrix} x_{C_2} \\ y_{C_2} \\ z_{C_2} \\ 1 \end{pmatrix} \tag{5.69}$$

$$\begin{pmatrix} u_{E_2} \\ v_{E_2} \end{pmatrix} = \begin{pmatrix} U_{E_2}/S_{E_2} \\ V_{E_2}/S_{E_2} \end{pmatrix} \quad et \quad \begin{pmatrix} u_{E_1} \\ v_{E_1} \end{pmatrix} = \begin{pmatrix} U_{E_1}/S_{E_1} \\ V_{E_1}/S_{E_1} \end{pmatrix} \tag{5.70}$$

Ces valeurs n'ont de sens que si ni S_{E_1} ni S_{E_2} est nul, ce qui peut arriver lorsque le centre optique d'une image se trouve dans le plan focal de l'autre. Dans la pratique les mouvements que nous effectuerons autour d'un objet ne nous conduiront pas dans une telle situation.

A partir de la connaissance de ces épipoles, on peut calculer les vecteurs directeurs des lignes épipolaires. Le vecteur directeur $(\Delta u_2, \Delta v_2)$ de la ligne épipolaire correspondant au point (u_1, v_1) est donnée par :

$$\begin{pmatrix} \Delta u_2 \\ \Delta v_2 \end{pmatrix} = M_{21} \begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix} \tag{5.71}$$

M_{21} est une matrice 2 X 3 qui est calculée de la façon suivante :

$$M_{21} = \begin{pmatrix} S_{E_2} & 0 & -U_{E_2} \\ 0 & S_{E_2} & -V_{E_2} \end{pmatrix} * T_2' * \begin{bmatrix} t_2^1 \wedge t_3^1 & t_3^1 \wedge t_1^1 & t_1^1 \wedge t_2^1 \end{bmatrix} \tag{5.72}$$

T_2' étant la matrice 3 X 3 extraite de T_2 et ne comportant que les vecteurs t_1^2, t_2^2 et t_3^2. On calcule de la même façon la matrice M_{12}. Les épipoles E_1 et E_2 et les matrices M_{12} et M_{21} sont évaluées une seule fois et sont ensuite utilisées pour calculer chacune des lignes épipolaires.

5.2 Réalisation

5.2.1 Recherche de la projection associée à une position de la pince

Les calibrages étant effectués, nous pouvons maintenant faire tourner notre processus de reconstruction entre des paires d'images prises au cours d'un mouvement du robot. Chacune des reconstructions effectuées sera fournie dans le repère de la mire de calibrage. La figure 5.2 montre comment peut être calculée la projection perspective pour chacune des positions du robot.

Pour chaque position de la pince, le repère R_i qui lui est lié est connu grâce à la transformation P_i qui le lie au repère de base R_s (repère station) du robot :

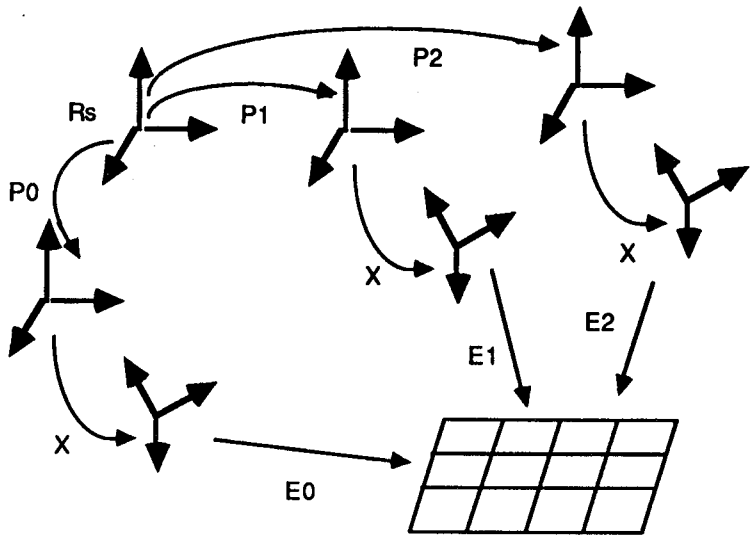


Figure 5.2: Comment retrouver la projection pour une position donnée de la pince

$$R_i = R_s * P_i \tag{5.73}$$

Le repère R_{C_i} de la caméra est lié au repère de la pince par la transformation X qui provient du calibrage du lien entre le robot et la caméra. Enfin on passe du repère de la caméra à celui de la mire R_m par la transformation E_i. On a donc :

$$R_m = R_s * P_i X E_i \tag{5.74}$$

E_i est ce que l'on cherche à obtenir, car comme l'on connait les paramètres intrinsèques de la caméra, on peut en déduire la projection perspective à partir de cette transformation.

Lors du calibrage de la caméra, la position de la pince est connue. Nous avons donc une ou plusieurs position P_0 connues où E_0 peut être déduit des paramètres de la caméra, ce qui nous donne l'équation suivante :

$$P_i X E_i = P_0 X E_0$$

$$\Leftrightarrow E_i = X^{-1} P_i^{-1} P_0 X E_0 \tag{5.75}$$

Dans le calcul 5.75 les valeurs X^{-1} et $P_0 X E_0$ peuvent être évaluées une fois pour toutes le calibrage fait. Trouver E_i revient donc à calculer :

$$5.75 \Leftrightarrow E_i = A P_i^{-1} B \text{ avec } \begin{cases} A = X^{-1} \\ B = P_0 X E_0 \end{cases} \tag{5.76}$$

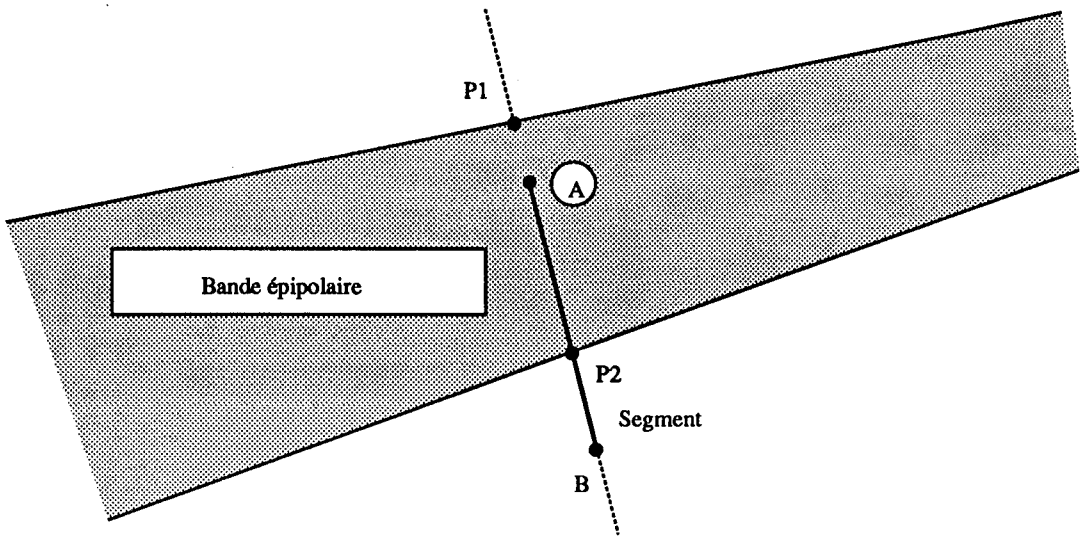


Figure 5.3: Intersection entre un segment et la bande épipolaire de son correspondant

5.2.2 Recouvrement des segments

Comme nous l'avons déjà dit plus haut, nous ne prendrons dans un couple de segments 2D que les parties qui sont communes dans le segment 3D correspondant. Voici comment nous allons procéder : Chacun des segments 2D est représenté par une paire de points extrémités. En utilisant l'information épipolaire pour un segment d'une image dans l'autre image, nous recherchons les extrémités de la partie commune dans cette deuxième image. Nous procédons de même pour la première image et nous reconstruisons ensuite les deux points extrémités d'un segment 3D.

Pour trouver l'intersection entre un segment et la bande épipolaire de son correspondant, nous utilisons quatre points : A et B les points extrémités du segment, P_1 qui est le point intersection entre la droite porteuse du segment et la première épipolaire et P_2 le point intersection avec la deuxième épipolaire (figure 5.3).

Pour comparer A, B, P_1 et P_2 nous calculons leurs abscisses sur la droite porteuse du segment, droite sur laquelle ils sont tous alignés. Puis nous regardons tous les cas de figure :

si x_{P_1} entre x_A et x_B alors
  si x_{P_2} entre x_A et x_B alors
    P_1 et P_2 sont les extrémités
  sinon

si x_A entre x_{P_1} et x_{P_2} alors
  A et P_1 sont les extrémités
sinon
  P_1 et B sont les extrémités
sinon
  si x_{P_2} entre x_A et x_B alors
    si x_A entre x_{P_1} et x_{P_2} alors
      A et P_2 sont les extrémités
    sinon
      P_2 et B sont les extrémités
  sinon
    si x_A entre x_{P_1} et x_{P_2} alors
      A et B sont les extrémités
    sinon
      Il n'y a pas de partie commune

La relation "A entre B et C" est définie de la manière suivante :

$$A \; entre \; B \; et \; C \Leftrightarrow \begin{cases} A = B \; ou \; A = C \\ ou \\ A > B \; et \; C \geq A \\ ou \\ A \leq B \; et \; CleA \end{cases} \tag{5.77}$$

5.2.3 Reconstruction

La reconstruction des segments consiste en fait à reconstruire chacun des points des extrémités communes. Pour simplifier au maximum cette partie, nous utilisons trois des quatre équations fournies par 5.67 ce qui nous donne le système suivant :

$$\begin{cases} (t_1^1 - u_1 t_3^1)^\top P = u_1 t_{34}^1 - t_{14}^1 \\ (t_2^1 - v_1 t_3^1)^\top P = v_1 t_{34}^1 - t_{24}^1 \\ (t_1^2 - u_2 t_3^2)^\top P = u_2 t_{34}^2 - t_{14}^2 \end{cases} \tag{5.78}$$

5.2.4 Résultats fournis

Le résultat de la combinaison de deux listes de segments 2D numérotés est donc une liste de segments 3D. Chacun de ces segments est représenté par les six coordonnées des deux points extrémités. Pour être compatible avec les utilisations à un niveau supérieur de ces segments 3D, nous fournissons également une matrice de covariance pour chacun des deux points extrémités. Nous prenons par défaut une

matrice diagonale.

Il serait intéressant, connaissant les caractéristiques de la projection, de calculer ces matrices de covariance en donnant une plus grande incertitude dans la direction de projection.

5.3 Résultats

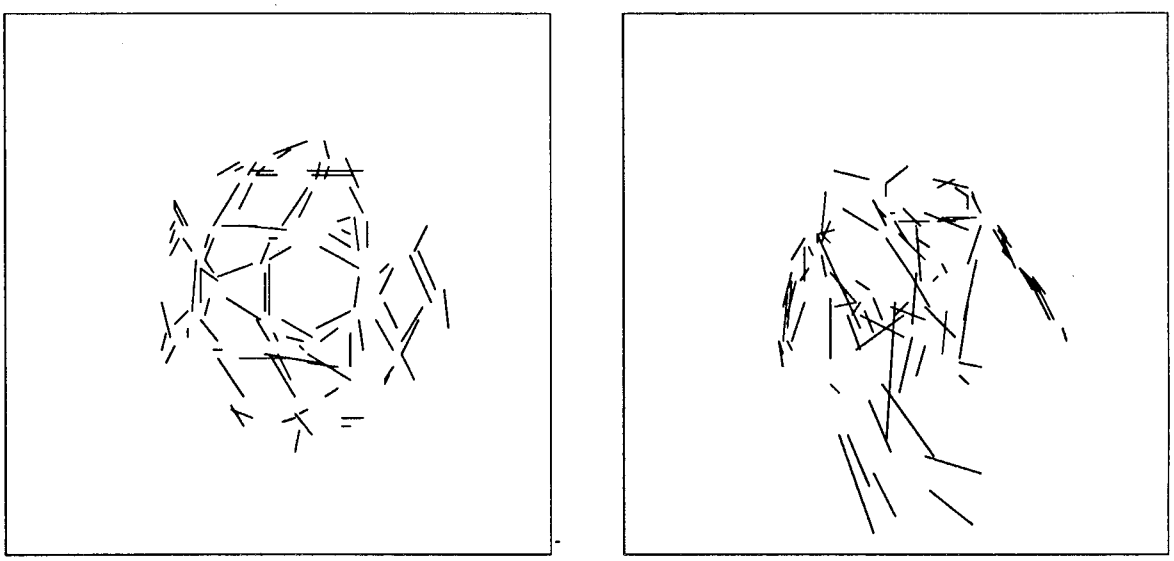


Figure 5.4: Reconstruction entre l'image 5 et l'image 10.

Pour les expérimentations nous reprenons les séquences d'images que nous avons déjà étudiées dans le chapitre précédent, et en particulier la première série (celle du "Rubik's cube"). Afin de posséder un écart suffisant entre les images et ne pas générer de trop grandes erreurs, nous effectuons des reconstructions sur des écarts de cinq images.

5.3.1 Visualisation des résultats

Nous montrons les reconstructions entre les images 5 et 10 (figure 5.4), 10 et 15 (figure 5.5), 15 et 20 (figure 5.6) et 20 et 25 (figure 5.7). Sur chacune de ces figures la partie gauche montre la projection des segments 3-D le long de l'axe z et la partie droite leur projection sur l'axe y (Repère du robot). La caméra ayant un axe de visée assez proche de la verticale, on peut remarquer que les estimations le long de l'axe z ne sont pas très fiables. C'est pourquoi nous montrons également une reconstruction sur un plus grand écart – entre l'image 5 et l'image 25 (figure 5.8)

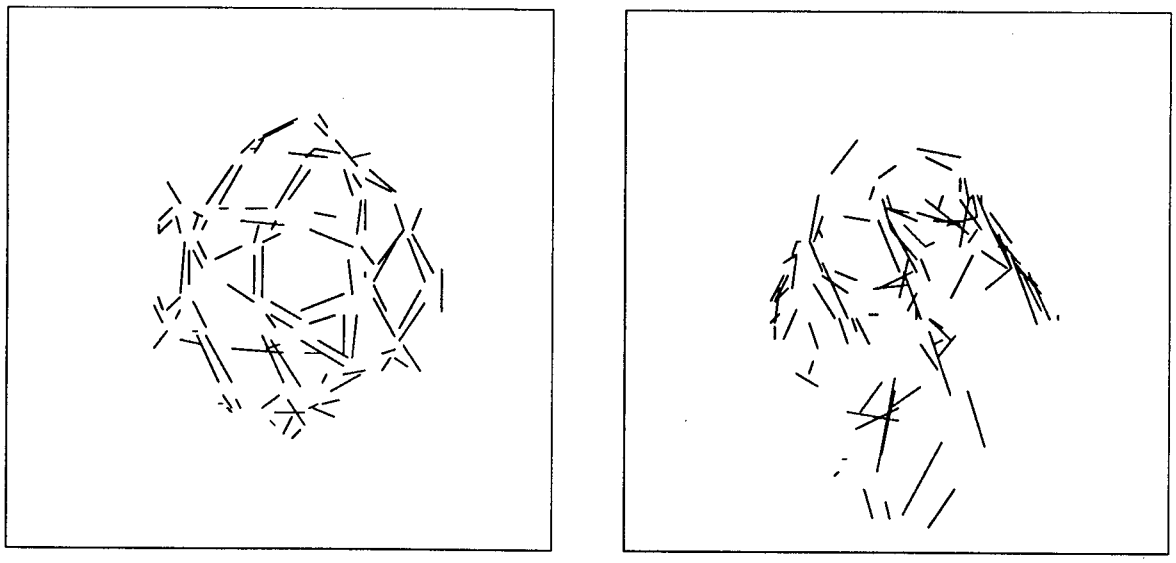


Figure 5.5: Reconstruction entre l'image 10 et l'image 15.

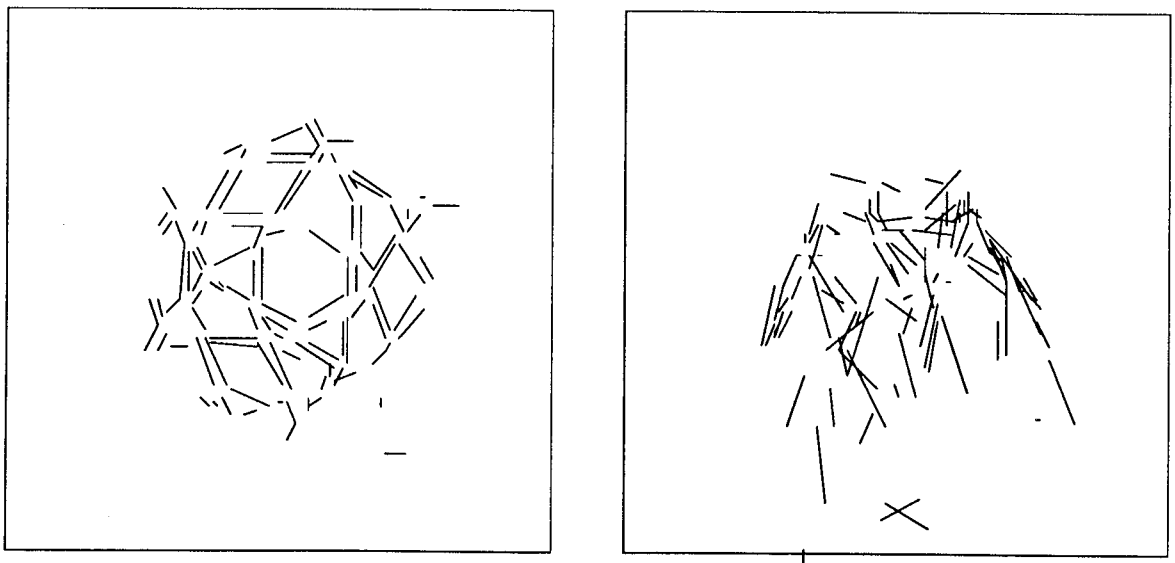


Figure 5.6: Reconstruction entre l'image 15 et l'image 20.

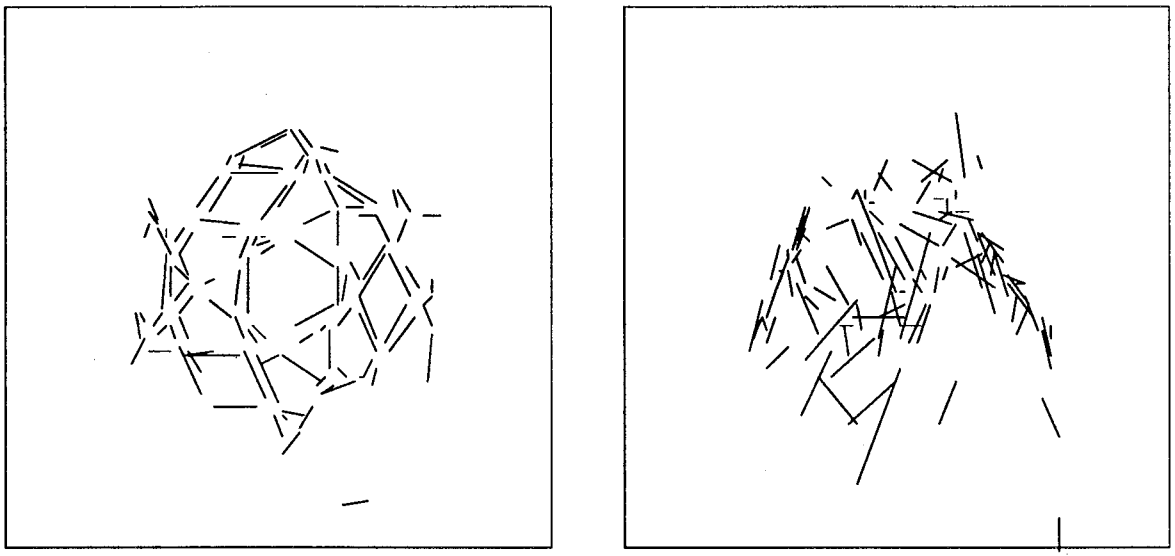


Figure 5.7: Reconstruction entre l'image 20 et l'image 25.

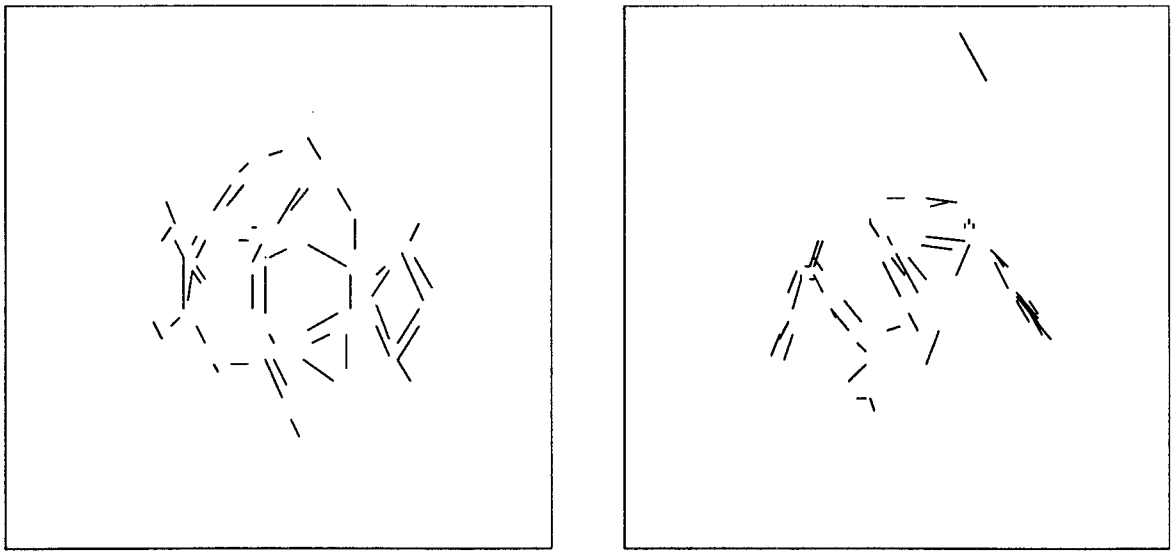


Figure 5.8: Reconstruction entre l'image 5 et l'image 25.

– où l'on peut voir que les estimations sur z se sont améliorées (mais le nombre d'indices visibles a diminué).

5.3.2 Utilisation

La numérotation des segments 2-D peut être récupérée pour les segments 3-D. On peut ainsi fournir des données tridimensionnelles facilement utilisables par un processus de fusion multisensorielle comme celui développé dans [Ram 89].

Chapitre 6

Conclusion

6.1 Evaluation

Nous venons donc de décrire un système complet de vision mobile, depuis des images brutes jusqu'à des indices 3-D de la scène. Nous avons décomposé notre approche en trois parties indépendantes se succédant depuis un bas niveau jusqu'à un niveau relativement élevé du type d'information traitée :

- Une partie d'extraction d'indices visuels dans une image 2-D.
- Une partie de suivi d'indices dynamiques 2-D.
- Une partie de reconstruction d'indices 3-D à partir de la connaissance du mouvement d'indices bidimensionnels.

En ce qui concerne l'extraction d'indices, notre contribution a porté sur la définition d'une méthode d'extraction de segments de droite incrémentale, tout à fait adaptée à une transposition sous forme cablée. L'intérêt d'un tel processus est tout à fait évident dans la combinaison avec notre suivi d'indices, où le temps, et donc la vitesse à laquelle sont fournies les données, intervient.

Pour réaliser cette extraction de segments, nous avons découpé le problème en deux parties. Dans un premier temps nous avons élaboré une méthode de chaînage

de points de contraste utilisant un balayage de l'image. Ensuite, partant d'une méthode incrémentale d'extraction de segments de droite dans une chaîne de points, nous avons intégré cette recherche de droite dans la méthode de chaînage.

Notre apport se situe principalement dans le suivi d'indices. Cette partie est un des constituants principal de tout système de vision mobile. Notre étude basée sur l'utilisation des segments de droite ne doit pas faire oublier que le principe fondamental est le même pour tous les types d'indices. Notre étude nous a conforté dans la nécessité d'utiliser certains outils :

- Une modélisation de l'univers perçu.
- Une prédiction du mouvement et une estimation tenant compte à la fois de ce qu'on observe et de ce qu'on connait dans le modèle. Le filtre de Kalman nous a semblé être la meilleure solution pour exécuter cette tâche.
- Une mise en correspondance simple et robuste.

Pour la réalisation proprement dite de ce suivi d'indices, nous avons étudié quels types de paramètres il fallait utiliser avec les indices choisis. En effet un choix judicieux de ces paramètres permet de simplifier sensiblement le fonctionnement du filtre de Kalman. Dans notre cas nous n'utilisons pas de structure plus grandes que des matrices 2*2.

Nos expériences nous ont permis de connaître les limites d'un tel système. Et en particulier les limites de notre mise en correspondance : Telle qu'elle est, elle ne permet pas d'effectuer des appariements sur des images très dissemblables (elle n'utilise pas de connaissance contextuelle globale) mais elle permet au système de fonctionner relativement rapidement. Etant données les conditions d'utilisations d'un suivi d'indices, il ne semble pas nécessaire d'utiliser une méthode plus complexe d'appariement qui, si elle était plus efficace, ralentirait néanmoins le suivi.

Enfin nous avons développé une méthode de reconstruction de segments 3-D à partir des informations fournies par le suivi, ce qui nous a permis de vérifier le bon comportement de notre suivi. Nous avons fondé cette méthode sur les techniques de la stéréovision : Connaissant les mouvements du robot sur lequel est fixée la caméra, la reconstruction est effectuée par la comparaison entre les positions observées d'un indice à deux instants donnés, le suivi ayant fourni la correspondance entre ces indices.

6.2 Ce qu'il reste à faire

En réalisant notre système, nous avons également soulevé un certain nombre de problèmes, et des zones d'ombres subsistent toujours.

Notre extracteur de segments n'est pas encore parfait. Il y a d'abord le problème de la précision des résultats. Nous avons fondé notre méthode sur l'utilisation de calculs en nombres entiers. Une meilleure approximation des valeurs passerait certainement par l'utilisation de calculs en flottant, ce qui changerait en partie la philosophie de notre approche. Mais cela permettrait d'utiliser de manière efficace l'équation de la droite telle qu'elle est décrite dans le système, et de ne plus utiliser pour points extrémités des points existants. En effet l'utilisation de tels points engendre des résultats pas très fiables.

Il y a d'autre part le problème de l'orientation arbitraire des segments. On s'est aperçu que la connaissance de l'orientation d'une ligne par rapport au gradient de l'intensité lumineuse était très utile pour éliminer certaines ambiguïtés de mise en correspondance. Résoudre ce problème signifierait utiliser autre chose qu'une image binaire comme donnée, et modifierait sensiblement la méthode.

En ce qui concerne le suivi d'indices, on peut lui reprocher de nécessiter des images trop rapprochées pour fonctionner correctement. Mais c'est aussi le prix à payer à son efficacité. Et il est difficile de l'améliorer sans envisager des solutions se rapprochant des méthodes classiques de la mise en correspondance, avec une forte combinatoire.

On pourrait par contre limiter le nombre de comparaisons dans la mise en correspondance en séparant a priori les indices en différentes classes selon leur paramètres. Par exemple le premier test effectué étant la similarité en orientation, on pourrait séparer les segments en "plutôt horizontaux" et en "plutôt verticaux" (ou en plus de classes encore) et ne chercher les correspondances qu'entre indices de même classe (sachant que les classes se recouvrent partiellement).

On peut aussi se poser la question de savoir si il aurait fallu modéliser l'accélération dans le modèle. En fait il y a plusieurs raisons de ne pas la modéliser complètement :

- L'accélération ne possède dans le monde physique que la caractéristique d'être bornée et il n'est pas possible de la supposer continue. Il est de plus très difficile de la filtrer car elle est très bruitée. Il est donc plus avantageux de la considérer comme du bruit.
- Les mouvements observés sont les projections de mouvements dans l'espace. De ce fait des mouvements simples dans l'espace, comme la rotation d'un solide, n'apparaissent pas comme des fonctions à accélération constante.

Dans nos expériences le mouvement est typiquement un mouvement qui n'est pas modélisable de façon simple (rotation autour d'un objet avec l'axe de visée de la caméra qui pointe toujours vers cet objet) et il peut être nuisible d'utiliser un filtre pour estimer une accélération constante. Dans un tel cas la meilleure approche est de supposer l'accélération nulle et fortement bruitée.

Nous avons développé notre reconstruction dans le but de vérifier le suivi. Il reste à faire une reconstruction qui va à la fois plus loin dans l'utilisation de ses données (moins de contraintes) et dans la réutilisabilité de ses résultats (principalement la connection avec les travaux de [Ram 89]). Nous n'avons pas d'autre part utilisé toute les possibilités du suivi d'indice. Nous n'avons utilisé que l'étiquetage des segments, mais à l'intérieur du modèle les positions et les vitesses des indices sont évaluées et filtrées. Il aurait été intéressant de tirer partie de ces données moins bruitées que ce que l'on observe.

6.3 Perspectives

Nous avions dans l'introduction schématiquement situé nos travaux à l'intérieur du domaine de la vision par ordinateur (figure 1.1). Nous avons effectivement réalisé un tel système qui, à partir d'une séquence d'images 2-D fournit une estimation d'indices de scène tri-dimensionnels.

Indépendamment de sa situation à l'intérieur de l'ensemble, la partie de suivi d'indices est un module qui peut être utilisé partout où on a besoin d'obtenir des indices 2-D dynamiques à partir d'une séquence d'indices 2-D. Ce module se révèle à l'usage relativement robuste et rapide. On peut donc le considérer comme une partie fondamentale de tout système de vision fondé sur la mesure du mouvement.

Il resterait, pour le rendre encore plus universel, d'étendre ses capacités à tous les types d'indices. En fait le problème est là du choix des paramètres représentant un indice : Nous avons vu dans le cas des segments de droite comment ce choix était crucial pour les performances du suivi. La solution est donc d'étudier les corrélations pouvant exister entre chacun des paramètres utilisables, et définir ceux qui semblent nécessaires.

Bibliographie

[Agg 88] J. K. Aggarwal and N. Nandhakumar. *On the computation of motion from sequences of images. A review.* Technical Report TR-88-2-47, The University of Texas at Austin, Computer and Vision Research Center, Austin, Texas 78712, April 1988.

[Agi 80] Gerald J. Agin. Computer vision systems for industrial inspection and assembly. In *IEEE Computer*, May 1980.

[Agi 82] Gerlad J. Agin. Image processing algorithms for industrial vision. April 1982. The Robotics Institute, Carnegie Mellon University, Pittsburgh, Pa.

[Arn] Jens Arnspang. Optic acceleration. local determination of absolute depth and velocity, time to contact and geometry of an accelerating surface.

[Aub 89] Didier Aubert. *Mise en correspondance d'indices d'images en résolution multiple.* Doctorat de l'INPG, Institut National Polytechnique de Grenoble, Janvier 1989.

[Aya 86] Nicholas Ayache and Olivier D. Faugeras. HYPER : a new approach for the recognition and positioning of two-dimensional objects. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 44, 54, January 1986.

[Aya 88] Nicolas Ayache. *Construction et Fusion de Représentations Visuelles 3D, Applications à la Robotique Mobile.* Thèse d'état, Université de Paris-Sud, Centre d'Orsay, Inria, 1988.

[Bal 82] D. H. Ballard and C. M. Brown. *Computer Vision.* Prentice Hall, 1982.

[Bol 86] Robert C. Bolles and Patrice Horaud. 3DPO : a three-dimensional part orientation system. *The International Journal of Robotics Research*, 5(3):3–26, 1986.

[Bol 87] Robert. C. Bolles, H. Harlyn Baker, and David H. Marimont. Epipolar-plane image analysis : an approach to determining structure from motion. *International Journal of Computer Vision*, 1:7–55, 1987.

[Bou 87] P. Bouthemy and J. Santillana Rivero. A hierarchical likelihood approach for region segmentation according to motion-based criteria. In *First International Conference on Computer Vision*, pages 463–467, IEEE, 1987.

[Bur 82] Peter J. Burt, Chihsung Yen, and Xinping Xu. Local correlation measures for motion analysis, a comparative study. In *IEEE Conference Publication*, pages 269–274, Vol 14, Las Vegas, june 1982.

[Can 86] John Canny. A computational approach to edge detection. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 679–698, vol PAMI-8 no 6, november 1986.

[Cro 85] James L. Crowley. Navigation for an intelligent mobile robot. *IEEE Journal of Robotics and Automation*, RA-1(1):31–41, March 1985.

[Cro 88] J. L. Crowley, P. Stelmaszyk, and C. Discours. Measuring image flow by tracking edge-lines. In *2nd International Conference on Computer Vision*, Tarpon Springs, Fla., December 1988.

[Dem 86] Yves Demazeau. *Niveaux de représentation pour la vision par ordinateur. Indices d'image et indices de scène.* Doctorat de l'INPG, Institut National Polytechnique de Grenoble, Décembre 1986.

[Der 87a] Rachid Deriche. Optimal edge detection using recursive filtering. In *IEEE*, pages 501–505, 1987.

[Der 87b] Rachid Deriche. Separable recursive filtering for efficient multi-scale edge detection. In *International Workshop on Industrial Applications of Machine Vision and Machine Intelligence*, pages 18–23, Tokyo, Japan, February 1987.

[Der 87c] Rachid Deriche. Using canny's criteria to derive a recursively implemented optimal edge detector. In *International Journal of Computer Vision*, pages 167–187, KluwerAcademic Publishers, Boston, 1987.

[Gar] P. Garcia and C. Doncarli. Localisation of a 2-dimensionnal object for tracking in a series of images. *?*, ?.

[Gen 82] Donald B. Gennery. Tracking known 3-dimensional objects. In *The National Conference on Artificial Intelligence AAAI 82*, pages 13–17, Carnegie-Mellon University, Pittsburgh, Pennsylvania, August 1982.

[Gir 87a] Gérard Giraudon. *Chaînage efficace de contour.* Technical Report 605, INRIA, February 1987.

[Gir 87b] Gérard Giraudon. An efficient edge following algorithm. In *5th Scandinavian Conference on Image Analysis*, pages 547–554, Stokholm, June 1987.

[Har 88] C. G. Harris. Using a sequence of more than 2 images. In *IEE Colloquium on Motion and Stereopsis in Machine Vision*, page 44 ..., 1988. digest #7.

[Hil 84] Ellen Catherine Hildreth. *The mesurement of visual motion.* The MIT Press, 1984.

[Hor 81] Berthold K.P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[Hor 88] R. Horaud and T. Skordas. Structural matching for stereo vision. In *Proc. 9th International Conference on Pattern Recognition*, pages 439–445, Rome, Italy, November 1988.

[Hor 89] R. Horaud and T. Skordas. Stereo Correspondence Through Feature Grouping and Maximal Cliques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-11(11):1168–1180, November 1989.

[Lab 78] M. Labarrere, J. P. Krief, and B. Gimonet. *Le filtrage et ses applications.* Cepadus Edition, 1978.

[Law 81] Daryl T. Lawton. Optic flow field structure and processing image motion. In *Int. Conf. of Artificial Intelligence*, 1981.

[Loo 86] L. Pocztar, G. Loopuyt and H. Maître. Restitution d'un flot de particules dans un espace à 4 dimensions. In *Semaine internationale de l'image électronique, deuxième colloque image*, Nice, Avril 1986.

[Lux 85] Augustin Lux. *Algorithmique et contrôle en vision par ordinateur.* Thèse d'état, Université Scientifique et Médicale de Grenoble et Institut National Polytechnique de Grenoble, septembre 1985.

[Mar 82] D. Marr. *Vision.* W. H. Freeman, San Francisco, 1982.

[Nag 87] Hans-Hellmut Nagel. On the estimation of optical flow : relations between different approaches and some new results. *Artificial Intelligence*, 33:299–324, 1987.

[Pra 83] K. Prazdny. A simple method for recovering relative depth map in the case of a translating sensor. In *Int. Conf. of Artificial Intelligence*, 1983.

[Ram 89] Fano Ramparany. *Perception Multisensorielle de la Structure Géométrique d'une Scène.* Thèse de l'INPG, Institut National Polytechnique de Grenoble, Février 1989.

[Riv 87] Patrick Rives and Bernard Espiau. Estimation de primitives 3-d au moyen d'une came'ra mobile. *Traitement du Signal*, 4(3), 1987.

[Sko 88] Thomas Skordas. *Mise en correspondance et reconstruction stéréo utilisant une description structurelle des images.* Thèse de l'INPG, Institut National Polytechnique de Grenoble, septembre 1988.

[Ste 88] P. Stelmaszyk, C. Discours, and A. Chehikian. A fast and reliable token tracker. In *IAPR Workshop on Computer Vision*, Tokyo, Japan, October 1988.

[Tho 83] Charles Thorpe and Steven Shafer. Correspondance in line drawings of multiple view of objects. In *Int. Conf. of Artificial Intelligence*, 1983.

[Tos 87] G. Toscani and O. D. Faugeras. Structure and motion from two noisy perspective views. In *IEEE*, 1987.

[Tsa 86] Roger Y. Tsai. *An efficient and accurate camera calibration technique for 3D machine vision.* Technical Report, IBM T.J. Watson Research Center, 1986.

[Vei 78] Françoise Veillon. One pass computation of morphological and geometrical properties of objects in digital pictures. In *4th IJCAI*, Kyoto, Japan, November 1978.

[You 88] Gem-Sun Young and Rama Chellappa. 3-d motion estimation using a sequence of noisy stereo images. In *Proceedings of computer vision and pattern recognition*, pages 710–716, Ann Arbor, MI, June 5-9 1988.