Stratégies de perception par vision active pour la reconstruction et l'exploration de scènes statiques

Eric Marchand

N° d'ordre : 1589

THÈSE

Présentée devant

L'UNIVERSITÉ DE RENNES I

Institut de Formation Supérieure en Informatique et Communication

Pour obtenir

Le Titre de Docteur de l'Université de Rennes I
Mention Informatique

par

Éric MARCHAND

Titre de la thèse

Stratégies de perception par vision active pour la reconstruction et l'exploration de scènes statiques

Soutenue le 26 Juin 1996, devant la commission d'examen composée de :

M.	Jean-Pierre	BANÂTRE	Président
MM.	Marc	RICHETIN	Rapporteurs
	Raja	CHATILA	
MM.	Patrick	RIVES	Examinateurs
	Paul	LE GUERNIC	
	Roger	MOHR	
	François	CHAUMETTE	

La vision est l'art de voir les choses invisibles.

Jonathan Swift, *Les voyages de Gulliver*, 1726

Remerciements

Le travail présenté dans ce mémoire a été réalisé à l'Institut de Recherche en Informatique et Système Aléatoire (IRISA, Rennes) au sein du projet TEMIS (Traitement, Exploitation et Modélisation d'Images Séquentielles).

Je souhaiterais remercier tout d'abord les membres du jury qui ont accepté de juger ce travail :

- Jean Pierre Banâtre, Directeur de l'IRISA, pour m'avoir fait l'honneur de présider le jury de cette thèse ;
- Marc Richetin, Professeur à l'Université Blaise Pascal de Clermont-Ferrand et Raja Chatila, Directeur de Recherche CNRS au LAAS à Toulouse, à qui j'adresse tous mes remerciements pour le temps passer à analyser, critiquer et commenter cette thèse en tant que rapporteurs ;
- Roger Mohr, Professeur à l'INPG/ENSIMAG de Grenoble, pour avoir accepté de juger cette thèse et pour la confiance qu'il m'a accordée ;
- Patrick Rives, Directeur de recherche à l'INRIA Sophia Antipolis, pour les conversations que nous avons eues et pour ses précieux conseils ;
- Paul Le Guernic, Directeur de recherche à l'INRIA Rennes (EP-ATR), pour avoir apporté son appréciation sur l'exploitation du langage SIGNAL dont il est le père.

Naturellement un travail de thèse n'est pas, comme pourrait le laisser croire la page de titre, celui d'une seule personne. Il est le fruit d'une longue collaboration entre plusieurs personnes.

C'est pourquoi, je voudrais tout particulièrement témoigner toute mon amitié à François Chaumette, Chargé de recherche à l'INRIA Rennes, mon directeur de thèse. Tu m'as confié un sujet de recherche passionnant. Grâce à la grande liberté donnée, à la confiance accordée, aux multiples conseils prodigués, aux critiques justifiées, à ta disponibilité de tous les instants, je te dois tout ce que j'ai pu apprendre pendant ces trois années de recherche que nous avons partagées.

Une partie de cette thèse est le résultat d'une collaboration avec le projet EP-ATR de l'IRISA. Mes premiers pas avec les langages synchrones ont été guidés par Éric Rutten. Je ne peux pas oublier Hervé, mon frère, j'espère que "notre collaboration" ne sera pas la dernière. J'en profite pour remercier l'ensemble de l'équipe

EP-ATR pour leur disponibilité et leur conseils.

J'ai aussi reçu une aide appréciable d'Alessandro Rizzo, Florent Martinez, Samuel Ketels et Jean-Francois Leroux, merci à eux. Cette thèse a été réalisée dans le cadre du projet Inter PRC VIA (*Vision Intentionnelle et Action*), j'adresse tous mes remerciements aux personnes qui dans le cadre de ce projet m'ont aidé à aboutir.

J'exprime toute ma gratitude à Claude Labit pour m'avoir accueilli dans son équipe. J'ai pu apprécier ses grandes qualités tant scientifiques qu'humaines pendant cette période. Je n'oublie pas Patrick Bouthemy qui m'a fait profiter de ses compétences et de ses conseils.

Je souhaite finalement saluer tous les membres du projet TEMIS, mes compagnons de route pendant ces trois années et en particulier Laurence, Jean-Marc, Yann, Virginie, Hervé, Fabien, Laurent, Cécile, Michael, Jean-Christophe et Myriam. Enfin, une pensée particulière pour $\frac{1}{2}$Jack et JPM, ils savent tout ce que je leur dois.

Table des matières

Introduction

> Est-il une satisfaction plus vraie, un plaisir plus réel que celui du navigateur qui pointe ses découvertes sur la carte du bord? Il voit les terres se former peu à peu sous ses regards, et, pour ainsi dire, émerger au sein des flots ! D'abord les lignes terminales sont vagues, brisées, interrompues ! Puis les découvertes se complètent, les lignes se rejoignent, les pointillés des cartes font place au trait, [...], enfin le nouveau continent se déplie sur le globe dans toute sa splendeur magnifique !
>
> Jules Verne, *Les enfants du capitaine Grant*,
> Bibliothèque d'éducation et de récréation J. Hetzel, Paris, 1868

De nombreux travaux menés en vision artificielle se sont fixés pour objectif la réalisation de systèmes puissants capables d'accéder à la géométrie spatiale d'une scène à partir de son observation par une caméra mobile. Ces systèmes doivent fournir une description géométrique claire et complète de la scène à partir d'une séquence d'images souvent bruitées et difficilement exploitables. L'étude proposée ici tente d'apporter sa contribution au problème de la reconstruction d'environnements tridimensionels assez restreints, avec comme ambition la reconstruction de scènes de plus en plus complexes.

Dans une optique d'analyse de scènes, les approches de vision inspirées du paradigme de Marr considèrent un capteur généralement statique, éventuellement mobile, mais non contrôlé. Cette approche s'avère insuffisante pour résoudre un grand nombre de problèmes où une modification pertinente des paramètres intrinsèques et/ou extrinsèques du capteur est nécessaire. C'est pourquoi Aloimonos [Aloimonos 87], [Aloimonos 90], Bajcsy [Bajcsy 88] ou encore Ballard [Ballard 91] ont proposé de modifier radicalement cet état de fait en élaborant le concept de vision active [Swain 93]. Les techniques de vision active tirent leur origine d'une tentative de simulation du système visuel animal et humain en essayant de recréer ses facultés d'adaptation. D'un point de vue méthodologique, la vision active, où les informations perçues sont utilisées au sein d'une boucle de rétroaction, tente surtout d'améliorer la qualité de la perception par rapport à l'approche passive classique, où l'on se restreint à observer, mesurer et interpréter les données issues du capteur. La vision active consiste en effet à élaborer des stratégies de perception intelligentes, en contrôlant les paramètres du capteur (position, vitesse, mise au point, etc.). Elle peut être définie comme un processus d'acquisition "intelligent" des données afin de résoudre les

problèmes soulevés lors de la conception d'un système de vision par ordinateur, à savoir leur sensibilité au bruit, leur faible précision et surtout leur manque de réactivité.

Dans le cadre de nos travaux, nous utilisons le principe de la vision active pour la reconstruction 3D et l'exploration de scènes à l'aide d'un système de vision robotique calibré. Plus précisément, notre objectif est d'obtenir une reconstruction complète de scènes statiques composées de segments, d'objets polyédriques et de cylindres droits circulaires sans aucune connaissance sur leur nombre, leurs dimensions et leur position spatiale. Pour cela, nous nous plaçons à deux niveaux différents : à un niveau local, les mouvements de la caméra sont contraints de manière à optimiser la qualité des résultats de reconstruction de primitives géométriques particulières. Nous avons utilisé une approche continue dont l'originalité consiste à contraindre les mouvements de la caméra de manière à obtenir une reconstruction 3D précise et robuste de primitives géométriques paramétrables telles que les points, les droites, les cylindres, etc. L'utilisation de la vision active à ce niveau est cependant assez contraignante puisque les mouvements de la caméra sont spécifiques à chaque primitive de la scène. La reconstruction d'une scène composée de plusieurs objets nécessite donc des focalisations successives sur les différentes primitives.

Pour cela, nous nous plaçons alors à un second niveau, plus global, en développant des **stratégies de perception** qui permettent l'acquisition d'une carte précise et complète de la scène : les mouvements de la caméra sont commandés de manière à observer l'ensemble de la scène tout en focalisant celle-ci sur des zones particulières, résolvant ainsi notamment les problèmes soulevés par les occlusions. Nous proposons donc une approche originale au problème de l'exploration/reconstruction 3D reposant sur un contrôle explicite des mouvements de la caméra. Cet aspect "stratégie de perception" est divisé en deux niveaux distincts. La première phase de l'exploration est locale au sens où seules les informations disponibles sont utilisées afin de déterminer sur quelle primitive la caméra doit se focaliser. Le second niveau vise à assurer la complétude de la reconstruction de la scène grâce à un calcul explicite de nouveaux points de vue. Le problème à résoudre consiste à choisir les positions et orientations de la caméra permettant d'obtenir le maximum d'informations possibles sur les zones encore inconnues de la scène. L'ensemble des points de vue choisis doit conduire à une modélisation complète de la scène, tout en tentant de minimiser le nombre de points de vue et/ou la distance parcourue par la caméra.

Un autre problème à traiter concerne l'intégration du système de vision proposé. Cet aspect intégration n'est que très rarement pris en compte dans la définition de tels systèmes, les problèmes purement informatiques d'application robotique n'étant malheureusement que trop rarement évoqués. Pour cela, nous proposons l'utilisation du langage synchrone SIGNAL qui permet de gérer de manière adéquate les aspects continus des algorithmes de commande et l'aspect événementiel des stratégies de perception. L'utilisation des langages synchrones en général, et de SIGNAL en particulier, permet de considérer l'intégration de ces différents aspects dans un même formalisme.

Ce mémoire est structuré en cinq chapitres qui portent respectivement sur les thèmes suivants :

- Le premier chapitre présente un panorama général des différentes approches rencontrées dans la littérature sur le thème de la **vision active**. Si les concepts de base sont

similaires, des motivations différentes ont amené les chercheurs à étudier la vision active sous différents aspects.

- Le second chapitre introduit les différentes méthodes existantes en matière de reconstruction tridimensionnelle. Nous essayerons de relever les avantages et les inconvénients de chacune d'entre elles. Nous insisterons plus particulièrement sur les techniques de reconstruction reposant sur l'analyse du mouvement des objets dans l'image (*vision dynamique*). En particulier, la méthode de reconstruction par vision active que nous avons utilisée sera décrite en détail. Finalement nous décrirons les techniques que nous avons développées pour améliorer et finaliser cette approche locale de reconstruction 3D : reconnaissance de primitives, détection d'objets occultants et estimation de la position 3D des extrémités des primitives.

- Le troisième chapitre décrit une approche locale de reconstruction de scènes. Nous proposerons un premier algorithme, simple mais efficace, qui permet de reconstruire de façon incrémentale toutes les primitives observées par la caméra. Cette modélisation reste cependant de bas niveau et est parfois incomplète. Nous proposerons donc une approche permettant d'obtenir une représentation de la scène de haut niveau tout en traitant les problèmes locaux d'occlusion.

  La méthode que nous avons développée pour parvenir à cet objectif vient se greffer sur l'algorithme de reconstruction incrémentale et repose sur des techniques de **prédiction / vérification d'hypothèses**. Du fait des incertitudes dans les mesures et les observations, nous avons utilisé une approche probabiliste reposant sur l'utilisation de réseaux Bayesiens. Ces réseaux permettent d'émettre des hypothèses sur l'existence et la localisation de nouveaux objets, puis de proposer l'exécution d'une action conduisant à vérifier ou à infirmer cette hypothèse, et enfin, en fonction du résultat de l'étape de vérification, de compléter le modèle 3D de la scène. L'étape de vérification s'appuie à la fois sur les observations déjà réalisées sur la scène, mais aussi sur une acquisition d'informations nouvelles nécessitant parfois un déplacement du capteur.

  On obtient ainsi une modélisation 3D des zones observées constituée d'objets reconstruits, de zones libres et de zones occultées (calculées par lancer de rayons).

- Le quatrième chapitre a pour objectif de présenter une méthode d'exploration globale visant à assurer la **complétude** de la reconstruction. Si, après une phase d'exploration *locale*, il est possible d'assurer que toutes les primitives observées par la caméra ont été reconstruites, il est par contre impossible d'affirmer à ce stade du processus de reconstruction que toutes les primitives composant la scène ont été traitées. Compléter le modèle géométrique requiert des mouvements exploratoires permettant d'observer les parties cachées et/ou les parties non encore observées de la scène. Cette exploration passe par le calcul d'un certain nombre de points de vue qui permettront l'observation de nouveaux objets ou au contraire qui permettront d'assurer que telle ou telle partie de l'espace est vide.

Le problème du **calcul de points de vue** ("*next best view problem*") est un problème difficile et peu étudié. Généralement, les solutions proposées reposent sur une connaissance forte de la scène étudiée (modèle CAO des objets de la scène). Dans notre cas, ces connaissances sont inexistantes. La stratégie de calcul de points de vue que nous avons mise en œuvre repose sur la modélisation et l'optimisation d'une fonction de coût représentant au mieux la tâche que nous souhaitons effectuer. Des algorithmes visant à optimiser la trajectoire du manipulateur sont également proposés.

- Le dernier chapitre traite plus particulièrement des aspects informatiques liés au développement d'applications en vision robotique. Il apparaît que l'une des difficultés réside, entre autres, dans la façon de gérer la dualité *continu/événementiel* relative aux mouvements de la caméra et aux stratégies de perception. Un langage asynchrone impératif requiert une prise en compte explicite des aspects bas niveau de la mise en œuvre (tel le séquencement des calculs imposés par les dépendances entre les données). Les **langages synchrones**, eux, sont particulièrement adaptés à la conception de systèmes réactifs comme le nôtre.

  L'exécution séquentielle ou parallèle des différentes phases de notre système a pu être réalisée efficacement en utilisant le langage synchrone SIGNAL et son extension récente SIGNAL*GTi*. Les méthodes proposées permettent de considérer de manière unifiée les différents aspects de l'application, c'est-à-dire depuis l'aspect "continu" avec la mise en œuvre des tâches flot de données (asservissement visuel et estimation 3D) jusqu'à l'aspect "événementiel" et discret avec la conception de systèmes autorisant la parallélisation de tâches et la hiérarchisation de tâches préemptives

- Signalons finalement que la méthode utilisée pour générer automatiquement les mouvements de la caméra, à savoir l'**asservissement visuel** également appelé commande référencée vision, est présentée en annexe de ce travail. Un certain nombre de tâches secondaires (suivis de trajectoires, évitement des butées articulaires du manipulateur) sont décrites.

Ajoutons que nous avons confronté l'ensemble des algorithmes proposés à des expérimentations réelles sur la cellule de vision robotique de l'IRISA. Quelques perspectives de recherche sont tracées en guise de conclusion.

Chapitre 1

La vision active

> Marr's contributions set the foundations for vision as a scientific discipline. [...] However, Marr left out of is theory a very important issue: the fact that all existing visual systems, from insect and frogs to fish, snakes, birds and humans, are active. Being active, they control the image acquisition process and thus introduce constraints that greatly facilitate the recovery of information about the 3D scene. "*I move, therefore I see*" is a fundamental true statement. And if I manage to make the human eye stationary, human start losing perception !
>
> Yannis Aloimonos, *What I have learned*, [Aloimonos 94]

L'objet de ce premier chapitre est de montrer les différences entre la vision classique, correspondant traditionnellement au paradigme de Marr qui suppose un observateur passif, et la vision active. Il apparaît cependant qu'il n'existe pas de définition unifiée pour la vision active faisant l'unanimité dans la communauté. Si les concepts de base sont les mêmes, des motivations différentes ont amené les chercheurs à étudier la vision active sous différents aspects.

1.1 La vision "traditionnelle" : le paradigme de Marr

Les systèmes de vision traditionnelle s'appuient généralement sur les concepts définis dans "le paradigme de Marr". David Marr [Marr 82] a proposé au début des années 80 une méthodologie complète devant permettre la conception d'un modèle calculatoire pour la vision artificielle. Les principales étapes devant mener à la conception d'un système de vision sont les suivants :

- la **segmentation** : l'extraction de primitives dans une image ou dans une séquence d'images (*primal sketch*) ;

- la **reconstruction** : la construction d'une représentation de l'espace centré sur l'observateur à partir de ces primitives de base : cette étape vise à calculer les caractéristiques tridimensionnelles de la scène par rapport à l'observateur (la caméra).
- la **reconnaissance** : cette représentation est ensuite fusionnée avec un autre type de connaissance 3D pour obtenir une représentation de l'espace centrée sur la scène. Cette connaissance 3D peut être la position de la caméra et/ou la connaissance de la position 3D d'objets dont on connait la position par rapport à la caméra, etc.

Ces trois étapes, segmentation, reconstruction, et reconnaissance, constituent une série de transformations permettant de transformer le signal image en une représentation symbolique de la scène. L'interprétation finale qui sera faite ne sera valide que si les différentes transformations sont valides, et donc si les modèles mathématiques sous-jacents sont également valides.

Cette méthodologie en trois étapes a guidé les recherches en vision artificielle pendant des années. Les progrès qui en ont découlé sont très significatifs : analyse des contours, des textures, des ombrages, des reflets ou encore analyse du mouvement ou stéréovision passive. Cependant, ce système a aussi montré dans bien des cas ses limites et, depuis une dizaine d'années, ce modèle est remis en cause car il ne prend pas en compte un facteur important dans le processus de perception : l'action.

1.2 La vision active

Les approches de vision inspirées du paradigme de Marr considèrent un capteur généralement statique, voire mobile, mais en aucun cas contrôlé. Cet aspect passif se montre vite problématique et une question s'est rapidement posée : les problèmes classiques de vision par ordinateur (*shape from X*, flot optique, etc) peuvent-ils tous se résoudre en utilisant cette approche? La réponse n'est évidemment pas toujours négative ; cependant, dans le cas de la vision passive, ces problèmes sont souvent difficiles à résoudre car mal posés. De fait, une solution unique à ces problèmes n'existe pas toujours et des contraintes supplémentaires sur les paramètres du capteur peuvent aider à trouver cette solution. C'est pourquoi Aloimonos, Bajcsy ou encore Ballard ont proposé de modifier radicalement cet état de fait en proposant le concept de vision active. Il est bon de rappeler ici que la vision active n'utilise pas de capteurs actifs (comme les lasers *range finder* par exemple), mais utilise des capteurs passifs de manière active.

Ces techniques tirent leur origine d'une tentative de simulation du système visuel humain. Concernant la vision humaine, le mouvement des yeux et de la tête, l'adaptation des pupilles aux variations d'illumination, jouent un rôle important dans la perception visuelle. C'est cette faculté d'adaptation que la vision active tente de recréer. En fonction de la tâche à effectuer et/ou de stimuli externes, un système de vision active peut être amené à modifier les paramètres extrinsèques et/ou intrinsèques d'une ou plusieurs caméras (position, orientation, zoom, etc.), mais aussi, à se focaliser sur certaines régions d'une image pour effectuer un traitement particulier. La vision active prend donc à la fois le contrôle du matériel et des ressources logicielles alloués au système [Swain 93]. Cette répartition

et ce contrôle intelligent de l'ensemble des ressources disponibles a pour objectif une optimisation du processus de perception, que ce soit au niveau de la qualité des résultats obtenus, de la complexité des algorithmes mis en jeu, ou de la quantité d'informations extraite de la séquence d'images.

Si la définition générale de la vision active est simple, les motivations des différents chercheurs ayant participé à l'introduction de ces nouvelles techniques sont cependant très différentes. De fait, ce que l'on appelle classiquement vision active peut se diviser en plusieurs sous classes[1].

- la **vision active** telle qu'elle est décrite par Aloimonos [Aloimonos 87] est une analyse théorique du processus de vision dont la motivation principale est l'optimisation de la tâche visuelle à l'aide d'un capteur actif ;
- la **perception active** est un concept introduit par Bajcsy [Bajcsy 88] tendant à élaborer des stratégies de perception afin d'améliorer la connaissance de l'environnement ;
- La **vision animée** [Ballard 91] est fondée sur l'analyse de la perception humaine. Reposant sur l'utilisation de têtes binoculaires actives, elle favorise les aspects de fixation et de focalisation sur des régions d'intérêt (*gaze control*). L'objectif de Ballard est de diminuer la complexité algorithmique du processus de perception ;
- la **vision intentionnelle** [Aloimonos 90] a pour objectif d'extraire uniquement les informations pertinentes en fonction d'une tâche donnée.

Nous allons à présent définir plus précisément ces quatre catégories en précisant les objectifs, les avantages et les inconvénients de chacune d'elles. La liste des travaux se rapportant à chacune des quatre "classes" de vision active qui sont présentées ici n'est bien sûr pas exhaustive. Leur classification peut parfois porter à discussion, certaines approches pouvant appartenir à plusieurs classes simultanément. Enfin, les applications se rapportant plus précisément au problème de la reconstruction 3D ou de l'exploration de scène seront décrites en détail dans les chapitres 2 et 5 de ce mémoire.

1.2.1 La vision active : Aloimonos

Aloimonos [Aloimonos 87] semble être l'un des premiers à avoir exploré cette voie de recherche. Il considère la vision active d'un point de vue théorique. L'objectif d'Aloimonos est de montrer que les problèmes fondamentaux de la perception sont moins complexes à résoudre en considérant un observateur actif. Un observateur actif est un observateur impliqué dans une certaine activité telle que le mouvement du capteur, le suivi d'objet, la focalisation, ou encore la vergence pour un système stéréoscopique actif. Les informations supplémentaires qui résultent de cette activité (ou les contraintes qu'elle impose) permettent de résoudre plus aisément un certain nombre de tâches visuelles.

1. Les noms "vision active", "perception active", "vision animée" et "vision intentionnelle" qui ont été retenus dans cette typologie tirent leur origine des titres des premiers articles parus dans chacun de ces domaines (à savoir respectivement *"Active Vision"* [Aloimonos 87], *"Active perception"* [Bajcsy 88], *"Animate Vision"* [Ballard 91], *"Purposive and qualitative active vision"* [Aloimonos 90])

Problème	Observateur passif	Observateur actif
Shape from shading	Problème mal posé Besoin d'une régularisation, mais pas de solution unique garantie car problème non linéaire	Problème bien posé Solution unique Equation linéaire Stabilité. **Contrainte**: Nombre important de points de vue
Shape from contour	Problème mal posé Existence de solutions sous certaines hypothèses restrictives	Problème bien posé Solution unique **Contrainte**: Mouvement connu
Shape from texture	Problème mal posé Hypothèse sur la texture	Problème bien posé Pas d'hypothèse **Contrainte**: Mouvement connu
Structure from motion	Problème instable mais bien posé Contraintes non linéaires	Problème stable et bien posé Contraintes quadratiques Solutions simples **Contrainte**: Tâche de fixation
Flôt optique	Problème mal posé Besoin de régularisation La régularisation peut impliquer des résultats érronés	Problème bien posé Solution unique Instabilité possible

TAB. 1.1 – *Comparaison des performances d'un système actif et d'un système passif. Extrait de* [Aloimonos 87]

Aloimonos a donc considéré et analysé, du point de vue d'un observateur actif, des problèmes de vision typique, difficiles à résoudre d'un point de vue passif. Il considère ainsi des problèmes aussi variés que le *shape from shading, shape from contour, shape from texture, structure from motion* ou encore le problème classique de la détermination du flot optique (*optical flow*). Il a entre autres prouvé qu'un observateur actif peut résoudre des problèmes de bas niveau de manière plus efficace qu'un observateur passif grâce à l'introduction de contraintes supplémentaires sur le mouvement ou les caractéristiques du capteur. De plus, des problèmes qui sont mal posés, non linéaires et instables pour un observateur passif, deviennent, pour un observateur actif, bien posés, linéaires et stables. Il constate en outre que l'approche vision active est beaucoup plus robuste au bruit qu'une approche passive comme la vision dynamique. Le tableau 1.1 propose une synthèse des conclusions d'Aloimonos [Aloimonos 87].

Parmi les travaux se rapportant à cette classe, on retrouve bien entendu les travaux réalisés à l'Université du Maryland dans le groupe d'Aloimonos [Aloimonos 87], [Aloimonos 88], [Hervé 92], [Huang 91] [Fermüller 95]. Ces travaux se focalisent surtout sur l'aspect mathématique de la vision active et sur la résolution des problèmes de stabilité, linéarité et unicité des solutions proposées. Les travaux d'Aloimonos sont impressionnants, et les

conclusions qu'ils apportent sont importantes. Il apparaît cependant regrettable que ces travaux se soient limités à une formulation mathématique et que les vérifications expérimentales n'aient pas été (ou rarement) réalisées. D'autre part, ces études ne prennent pas (ou peu) en compte le problème du contrôle du capteur (commande). De fait, nous pensons que les travaux d'Aloimonos se rapprochent plus de la *vision dynamique* que de la *vision active*.

D'autres travaux plus récents ont cependant rejoint les points de vue d'Aloimonos, soulignant l'apport incontestable de la vision active. On pourra par exemple, citer les travaux de S. Boukir [Boukir 93a], [Chaumette 96] sur le délicat problème de l'estimation de la structure 3D à partir du mouvement ("*structure from controlled motion*"). Dans ce cas le contrôle du mouvement de la caméra est explicitement pris en compte afin d'optimiser la reconstruction 3D (nous reviendrons en détail sur cette méthode dans le chapitre suivant).

1.2.2 La perception active : Bajcsy

"Agir pour voir et voir pour agir", tel pourrait être une définition de la perception active introduite par Bajcsy [Bajcsy 88]. Contrairement à Aloimonos, elle ne considère donc pas la vision passive comme un problème mal posé, mais cherche à élaborer des stratégies de perception afin d'améliorer la connaissance de l'environnement. La perception active va donc consister à acquérir des informations sur la scène depuis des points de vue différents, à procéder à leur intégration et enfin à asservir le ou les capteurs en utilisant les résultats tirés des informations précédemment acquises. Cette "activité" du capteur concerne aussi bien la commande de ses paramètres variables (mise au point, position,...) que le choix d'un certain nombre de modules de traitement. Ce sont les informations fournies par le capteur à l'instant t qui décideront de l'emplacement de la caméra à l'instant $t+1$ ainsi que du traitement à effectuer sur ces informations.

La perception active passe par l'étude de la modélisation et des stratégies de la perception visuelle. La modélisation concerne les capteurs, ainsi que les modules de traitement. Les modèles manipulés sont de deux types : les modèles locaux, qui permettent de prédire le comportement du capteur ou le résultat des modules de traitement, et les modèles globaux (superviseur) qui représentent les interactions entre les différents modules (un modèle de ce type doit intégrer tous les paramètres globaux du système ainsi que son état final). L'introduction d'une boucle de rétroaction dans ces modèles globaux doit permettre au système d'acquérir les données au fur et à mesure de leur nécessité. Les stratégies de perception, elles, consistent en la recherche d'une séquence d'actions de perception et de traitement qui mèneront à l'obtention d'un maximum d'informations moyennant un coût minimal (ces stratégies sont donc fortement liées à la formalisation des modèles globaux supervisant le système). La perception active, telle que la définit Bajcsy, inclut donc des processus de raisonnement, de décision et de contrôle.

Notons enfin que si Aloimonos se contente d'utiliser des systèmes à base de caméras, Bajcsy, elle, considère que la perception active peut utiliser des capteurs passifs (caméras) mais aussi des capteurs actifs (radiation électromagnétique, radar, sonar, laser, etc.)

Parmi les travaux se rapportant au concept de perception active, on trouvera bien sûr ceux menés par Bajcsy au GRASP Lab. Ces travaux portent essentiellement sur le déve-

loppement des modèles globaux des systèmes, sur la définition de stratégies de perception [Maver 93], [Pito 96] ou encore sur la conception des superviseurs [Košeká 95]. En outre, Whaite et Ferrie [Whaite 94] cherchent à reconstruire et explorer une scène 3D en utilisant des boucles de rétroaction fondées sur l'incertitude associée aux données acquises.

Si les auteurs précités utilisent des informations de type photométrique ou géométrique 2D ou 3D, l'utilisation d'informations de type mouvement peut aussi être envisagée. Pour Sandini et Tistarelli, l'objet fondamental de la vision active est le mouvement et plus particulièrement les interactions entre l'observateur et l'environnement [Tistarelli 92]. Sandini [Sandini 86] utilise des informations issues de la stéréovision et du mouvement en s'appuyant sur un contrôle "intelligent" du mouvement de la caméra pour fiabiliser l'analyse stéréoscopique et l'analyse du mouvement. Des approches permettant d'obtenir des cartes de profondeur à partir d'un système monoculaire en mouvement sont également développées [Sandini 90]. Ces méthodes visent à stabiliser le point de fixation pendant le déplacement de la caméra.

1.2.3 La vision animée : Ballard

La vision animée, proposée par Danna Ballard [Ballard 91], a des motivations très proches de celles de la vision active. Comme pour Aloimonos, l'objectif est de contraindre les problèmes mal posés de la vision. La vision animée se fonde sur les études du mouvement de l'œil humain pendant une tâche visuelle [Yarbus 67], [Noton 71]. Cette approche repose donc sur l'utilisation d'une *tête binoculaire active*. Elle a pour objectif de contrôler le mouvement du capteur afin de réaliser des tâches de fixation ou de focalisation d'attention (*gaze control*) afin de diminuer la complexité du processus de perception et d'analyse. L'idée qui semble la plus prometteuse dans les théories de Ballard est l'utilisation d'un système de coordonnées exocentriques (par opposition à un système de coordonnées égocentriques, *i.e.*, centré sur l'observateur). Ce système de coordonnées centré sur l'objet fixé a l'avantage d'être invariant vis-à-vis du mouvement de l'observateur.

Pour Ballard, la vision animée est un sur-ensemble de la vision active (elle en intégrerait en effet toutes les capacités). Notre sentiment est cependant que la vision animée, qui considère principalement des tâches de fixation (simulant uniquement le mouvement des yeux et les saccades en particulier) est moins générale que la vision active puisque cette dernière permet de considérer une seule caméra, des mouvements beaucoup plus généraux et l'ensemble des paramètres du capteur.

Parmi les travaux se rapportant à la vision animée on trouvera principalement les travaux effectués à l'Université de Rochester sous la direction de Ballard et Brown [Ballard 91], [Wixson 94a], [Brown 92], [Rimey 91] mais aussi les travaux de Tsosos à l'Université de Toronto [Tsosos 87], [Tsosos 92], [Milios 93] ou en Europe, ceux de [Murray 95], [Brunnström 96], [Viéville 95], [Bobet 94].

1.2.4 La vision intentionnelle : Aloimonos

L'idée de la vision intentionnelle [Aloimonos 90] part du constat suivant : la plupart des systèmes de vision ont à répondre à une question précise ou à effectuer une seule tâche

décrite elle aussi de manière bien précise. Il n'apparaît donc pas utile de créer (mais est-ce seulement possible?) des systèmes "universels" ayant pour motivation la réalisation de tout type d'actions. Afin de résoudre le problème posé, il convient donc de déterminer la nature de l'information dont on a besoin, puis d'extraire cette information (et uniquement cette information), et enfin, après une phase plus ou moins complexe d'interprétation, de donner une réponse à la question posée ou d'effectuer la tâche demandée.

Au centre de la notion de vision intentionnelle, on retrouve le besoin d'une analyse et d'une modélisation précise et complète de la tâche à effectuer. "Dans quel but le système doit-il être construit?" et, connaissant ce but, "quelles sont les connaissance nécessaires pour l'atteindre?", et enfin "comment obtenir ces connaissances?". Telles sont les questions que doit se poser le concepteur d'un tel système. Toute action qui n'est pas fondamentale pour atteindre le but recherché est considérée comme inutile et ne sera donc pas effectuée. D'une certaine manière, on obtient un système minimaliste : ainsi, par exemple, la reconstruction 3D de la scène n'est pas toujours nécessaire. La vision intentionnelle consiste donc souvent à décomposer le problème initial en plusieurs sous-problèmes, à définir les modules de traitement qui doivent être associés à chacune de ces sous-tâches et enfin, à concevoir le superviseur qui activera/désactivera telle ou telle tâche.

Dans le cadre d'applications bien précises et parfaitement définies à l'avance, il est certain que la vision intentionnelle donne d'excellents résultats. Cependant, ce qui peut apparaître d'un côté comme un avantage (l'adéquation parfaite des modules de traitement à la tâche à réaliser) peut aussi être considéré comme son principal défaut. En effet, une modification même minime du but à atteindre peut nécessiter des modifications radicales dans la conception du système. Une bonne analyse des avantages et des inconvénients de la vision intentionnelle est faite dans la discussion proposée par Tarr et Black [Tarr 94] et en particulier dans les réponses de Brown [Brown 94], Sandini et Grosso [Sandini 94].

Depuis la définition du concept de vision intentionnelle au début des années 90, un grand nombre d'applications se réclame de cette approche. Citons entre autres le système TEA-1 de Rimey et Brown [Rimey 94] qui repose sur l'utilisation de réseaux Bayésiens pour la modélisation de la connaissance de l'environnement et la décision des actions à effectuer, le système de navigation d'Aloimonos, Rivlin et Huang [Aloimonos 93], l'approche de reconstruction 3D proposée par Kutulakos et Dyer [Kutulakos 95], etc.

Enfin, l'asservissement visuel [Feddema 91] [Espiau 92] [Hashimoto 93] peut aussi être classé dans cette catégorie. Seules les informations nécessaires à la tâche spécifiée sont extraites et utilisées. L'asservissement visuel consiste à introduire directement, et en boucle fermée, les informations extraites de l'image dans une boucle de commande. Le problème est spécifié en termes de régulation dans l'image et ne nécessite pas de calibration précise de la caméra ou de reconstruction tridimensionnelle de la scène à chaque itération de la boucle de commande. Des tâches de positionnement, de préhension, de suivi d'objets peuvent ainsi être effectuées en utilisant un minimum d'informations visuelles. Dans le cadre de la reconstruction de scène, l'asservissement visuel peut aussi être utilisé en considérant des trajectoires qui permettent de remonter de manière optimale aux caractéristiques de l'objet considéré. De plus, dans ce cas, les mouvements n'apportant pas d'informations supplémentaires peuvent être totalement annihilés [Boukir 93a]. Plus récemment, des méthodes d'asservissement visuel reposant sur la mesure du mouvement 2D

ont vu le jour pour des tâches d'alignement [Sundareswaran 94], d'accostage [SantosVictor 95], ou de suivi.

1.3 Conclusion

Comme on peut le constater, Aloimonos et Bajcsy (pour ne parler que des principaux auteurs) ont des objectifs radicalement différents. L'un cherche à simplifier les problèmes fondamentaux de la vision en utilisant un observateur actif et l'autre cherche à augmenter la connaissance sur l'environnement observé en définissant des stratégies de perception adéquates. Si ces deux approches ont pour point commun l'utilisation d'un ou de plusieurs capteurs mobiles, les motivations, les méthodes, et les outils utilisés sont très éloignés.

Pour notre part, nous estimons que ces différents aspects sont fortement liés. Ainsi, dans le cadre du problème qui nous intéresse : la reconstruction et l'exploration de scènes, il nous semble que les différents aspects de la vision active doivent être mis en œuvre pour réaliser cette tâche de manière efficace. Ainsi nous considérons deux niveaux : un niveau local dédié à la reconstruction 3D d'une primitive particulière et un niveau global dédié à la reconstruction complète de la scène.

- À un niveau local, les mouvements de la caméra sont contraints de manière à optimiser la qualité des résultats de reconstruction de primitives géométriques particulières. À ce niveau, on se place alors plutôt dans l'optique d'Aloimonos. Cependant, les contraintes nécessaires à une estimation robuste et non biaisée des paramètres des primitives 3D sont, comme nous le verrons dans le chapitre suivant, des contraintes sur la position de la primitive dans l'image et sur la trajectoire de la caméra par rapport à cette primitive. La génération automatique de ces mouvements se fait en utilisant l'asservissement visuel. Cet aspect est à rapprocher du concept de vision intentionnelle puisque seules les informations nécessaires à l'estimation des paramètres de cette primitive sont acquises et utilisées. De plus, les lois de commande utilisées ne considèrent *que* des mouvement de caméra permettant d'acquérir une information 3D. Ainsi tout mouvement n'apportant pas d'information n'est pas exécuté.

- L'utilisation de la vision active au niveau local est assez contraignante puisque les mouvements de la caméra sont fortement contraints. Pour pallier ce problème, nous nous plaçons alors à un second niveau en développant des **stratégies de perception** : les mouvements de la caméra sont commandés de manière à permettre la reconstruction de scènes composées de plusieurs objets inconnus. Nous développons des stratégies de déplacement de la caméra permettant de traiter l'exploration complète et la focalisation sur des zones particulières de la scène. Dans cette optique, nous rejoignons plutôt l'aspect "perception active" de Bajcsy et/ou l'aspect vision intentionnelle où l'intention est alors explicitement la reconstruction et l'exploration 3D.

Comme on peut le constater, dissocier les différents aspects de la vision active n'est pas toujours possible, ni même souhaitable. Pour nous, la vision active, consiste donc à définir l'ensemble des **stratégies de perception** permettant d'assurer la réalisation d'une

tâche donnée. Ceci concerne aussi bien les stratégies de bas niveau (*locales et continues*), permettant d'aboutir à des méthodes d'acquisition stables et robustes des données, que des stratégies de haut niveau (*globales et événementielles*), visant à définir l'ensemble des actions nécessaires à la réalisation de la tâche nominale.

Ces deux aspects nous semblent fortement complémentaires. Les stratégies de haut niveau doivent reposer sur des bases fiables et utiliser des données aussi précises que possible. De l'autre coté, les données issues des stratégies de bas niveau, ne sont pas directement exploitables car elles ne donnent qu'une vision partielle et incomplète de l'environnement. Elles n'ont donc un intérêt que si l'on considère un contexte plus vaste qui est donné par l'aspect stratégie de perception de haut niveau.

Chapitre 2

La reconstruction tridimensionnelle

> Toute forme a son origine dans le mouvement qui la trace. La forme n'est que le mouvement enregistré.
>
> Henri Bergson

Dans ce chapitre, nous introduisons les différentes méthodes existantes en reconstruction tridimensionnelle, tout en relevant les avantages ainsi que les limitations de chacune d'elles. Nous insisterons tout particulièrement sur les méthodes reposant sur l'utilisation d'une caméra en mouvement sur laquelle s'est porté notre choix. Nous décrirons ensuite une approche de reconstruction continue fondée sur le concept de vision active. Finalement nous décrirons les techniques que nous avons développées pour améliorer et finaliser cette approche locale de reconstruction 3D : reconnaissance de primitives, détection d'objets occultants et estimation de la position 3D des extrémités des primitives.

2.1 Reconstruction tridimensionnelle : état de l'art

Les données tridimensionnelles d'une scène sont acquises, en pratique, soit directement, avec des capteurs mesurant la profondeur, soit indirectement, en exploitant par exemple les informations 2D issues d'une ou plusieurs caméras. Il existe diverses méthodes en vision par ordinateur permettant de remonter à une représentation 3D de l'environnement perçu :

- les approches mono-image comme la reconstruction à partir de l'intensité lumineuse [Pentland 84], [Horn 89], la reconstruction à partir de la texture [Aloimonos 88], [Witkin 81] ou encore la reconstruction à partir des contours [Brady 84]. Des hypothèses *a priori* sur la scène sont alors nécessaires (comme, par exemple, la connaissance de la forme et des dimensions des objets).

- les approches multi-images qui font appel à au moins deux images pour déterminer la structure 3D d'une scène. On distingue alors deux catégories d'approches qui diffèrent de par la structure même du système de vision utilisé : la reconstruction 3D par stéréovision qui utilise un système constitué de plusieurs caméras, et la reconstruction 3D à partir du mouvement qui utilise une seule caméra mobile. Cette dernière approche est aussi communément appelée vision dynamique.

- enfin, plus récemment, certains auteurs ont développé des méthodes de reconstruction sans calibrage préalable de la caméra [Mohr 91][Faugeras 92a]. Ces méthodes s'appuient sur des propriétés invariantes par projection perspective. Les résultats sont obtenus dans un espace projectif, voire affine, mais en aucun cas, sans connaissances sur la scène, dans un espace euclidien.

Nous allons maintenant présenter plus en détail les approches multi-images en insistant plus particulièrement sur l'utilisation d'une caméra mobile.

2.1.1 Reconstruction par stéréovision

La motivation initiale de cette technique, maintenant bien connue, est de simuler le système visuel de la plupart des êtres vivants : la vision en relief obtenue par triangulation après mise en correspondance des images issues de l'œil droit et de l'œil gauche.

Stéréovision passive Les méthodes de reconstruction par stéréoscopie passive sont fondées sur l'utilisation d'un système composé de deux caméras ou plus, liées rigidement entre elles [Faugeras 93]. Les points de vue étant différents, les images de la scène transmises par ces caméras ne sont évidemment pas semblables. L'écart entre les positions de deux points dans les images de la projection d'un même point physique est appelé disparité. L'objectif des systèmes stéréoscopiques est de mesurer cette disparité afin d'estimer par triangulation la profondeur entre les différents objets de la scène et le système de caméras.

De manière générale, le processus se divise en trois parties : sélection de primitives particulières dans l'une des images, recherche de leurs homologues dans l'autre image (mise en correspondance, calcul de leur disparité), et enfin triangulation. La principale difficulté des techniques stéréoscopiques réside dans la mise en correspondance des primitives. Différentes techniques d'appariement ont été proposées en fonction du type de primitive retenu. Il en résulte différents types de cartes de disparité (éparse [Ayache 89a] ou dense [Okutomi 91]). Afin de faciliter la mise en correspondance et d'obtenir une solution unique, une série de contraintes sur la scène et sur la position des primitives dans les deux images est nécessaire. La plus importante de ces contraintes est la contrainte épipolaire. Cette contrainte spécifie que le correspondant d'un point m d'une image se trouve nécessairement sur la droite épipolaire associée à ce point dans l'autre image (voir Figure 2.1). Le problème de la mise en correspondance se ramène alors à un problème monodimensionnel de recherche le long de la droite épipolaire. On citera aussi, entre autres, la contrainte d'unicité (un point d'une image admet un correspondant et un seul dans l'autre image) et la contrainte de continuité (la disparité dans l'image doit varier lentement s'il n'y a pas de discontinuité importante dans la profondeur des surfaces observées).

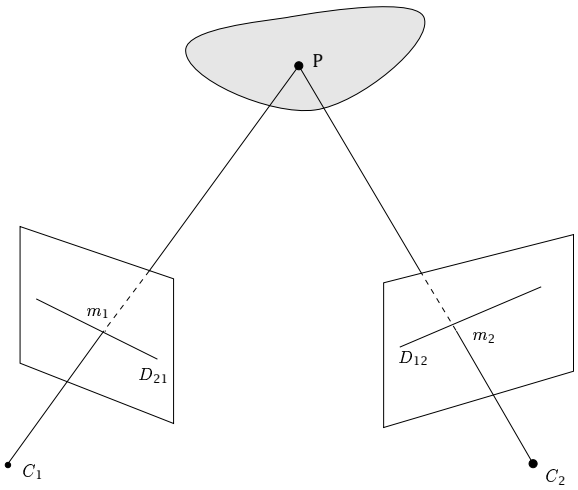


FIG. 2.1 – *La stéréovision et les droites épipolaires (système binoculaire)*

La mise en correspondance est simplifiée si le nombre de caméras augmente. Des systèmes à trois [Ayache 89a], [Zhang 92], quatre ou cinq caméras [Okutomi 91], [Kanade 95] sont donc apparus. Cependant, si multiplier le nombre de capteurs permet de lever des ambiguités en renforçant la contrainte épipolaire, de tels systèmes posent de nouveaux problèmes : le champ de vision commun devient plus restreint, le calibrage du système stéréoscopique devient plus difficile et le choix de la disposition des caméras n'est pas immédiat.

Notons enfin qu'une étape de calibrage est nécessaire pour le calcul, par triangulation, de la profondeur 3D des objets mis en correspondance. Les méthodes de calibrage sont nombreuses, citons par exemple [Tsai 86], [Toscani 87] ou encore [Chaumette 89], [Wei 94], [Robert 95].

Alliance stéréovision - mouvement Le handicap majeur des systèmes stéréoscopiques est leur rigidité. Cependant, l'association d'un capteur stéréoscopique et du mouvement donne des résultats intéressants. En effet, le mouvement du capteur permet de faciliter la mise en correspondance en levant les ambiguités d'appariement. La perception n'est donc plus statique mais dynamique (on ne parlera pas encore de vision active, car le mouvement des caméras n'est pas explicitement commandé). Ces systèmes sont surtout utilisés en coopération avec des plateformes robotiques mobiles [Weng 92a], [Ayache 89b], [Zhang 90].

Ils reposent généralement sur une approche récursive. Dans la méthode proposée par Weng [Weng 92a], les primitives de base sont des points 3D. Les informations extraites des séquences de vues stéréoscopiques sont fusionnées à l'aide d'un filtrage récursif, en l'occurrence un filtre de Kalman qui permet de prendre en compte la redondance ainsi que les incertitudes liées à l'acquisition des données. La méthode développée par Ayache et Faugeras [Ayache 89b] a pour but la reconstruction incrémentale d'environnement statique d'un robot mobile. Dans ce cas, les primitives de base sont des segments et le système,

en l'occurrence trinoculaire [Ayache 89a], donne les équations des segments 3D observés. Chaque paire de vues, qui permet d'obtenir une description locale de l'environnement, est fusionnée avec les précédentes pour obtenir une vision globale de la scène de façon incrémentale. Là encore, la mise à jour du modèle de la scène est formulée en terme de schéma récursif. L'inclusion d'une nouvelle primitive dans le modèle (ou bien la fusion avec une ancienne) fait l'objet d'un processus de prédiction-vérification intégrée dans un filtrage de Kalman. L'étude proposée par Zhang [Zhang 90], qui repose initialement sur une approche similaire à la précédente, permet en plus de considérer dans la scène des objets rigides en mouvement. Un schéma de prédiction-vérification est à la base du système d'appariement et permet de déterminer le mouvement des objets considérés. Chaque segment 3D est ensuite suivi le long de la séquence et son mouvement est estimé en utilisant, là encore, un filtrage de Kalman.

Les techniques de stéréovision sont maintenant relativement bien maîtrisées et il existe désormais un grand nombre de systèmes stéréoscopiques permettant d'obtenir des cartes de profondeur denses ou éparses en temps réel. Si la stéréovision semble être très bien adaptée à la perception globale de l'environnement, elle n'a plus les mêmes performances lorsqu'il s'agit d'analyser une zone bien précise en vue d'une reconnaissance ou d'une manipulation en robotique. Ces techniques sont très rigides et la lourdeur du système (aux sens propre et figuré) les destinent surtout à une observation globale des scènes étudiées et non à une observation précise nécessitant une mobilité importante du capteur. Par contre, l'utilisation d'une caméra mobile est beaucoup plus souple d'emploi et présente moins de limitations dans le mouvement.

2.1.2 Reconstruction à partir du mouvement

L'objectif de cette classe de techniques est de réaliser la reconstruction 3D à partir d'une séquence d'images acquises par une seule caméra. On peut classer les approches traitant de ce problème en deux catégories principales : les techniques continues et les techniques discrètes.

- Les techniques discrètes sont basées sur la mesure de la disparité de primitives particulières dans les images [Aggarwal 87], [Boukarri 89], [Chien 89], [Viala 92], [Crowley 92], [Weng 92b], [Wells 89], [Zhao 94], [Zhang 95]. Elles s'appuient pour la reconstruction sur une formulation en termes de déplacement de la caméra. Si l'on considère le cas d'un point de coordonnée $\underline{X}$, la variation de position de ce point dans le repère de la caméra est donnée par :

  $$\underline{X}_{t+1} = R\underline{X}_t + T$$

  où la matrice de rotation R et la translation T représentent le changement de repère de la caméra entre les positions t et $t+1$.

  Ces techniques reposent sur trois étapes fondamentales : extraction dans la séquence d'images d'un ensemble de primitives 2D suffisamment pertinentes, appariement

des primitives sélectionnées d'image en image, reconstruction par triangulation. On retrouve ainsi exactement le même schéma que celui de la stéréovision.

- Les approches continues s'appuient sur une formulation en termes de vitesse [Adiv 85], [Adiv 89], [Vernon 90], [Waxman 87], [Espiau 87], [Boukir 93a], [Negahdaripour 87], [Xie 89a]. Ainsi, dans le cas d'un point, on ne considèrera plus son déplacement $\underline{X}_{t+1} - \underline{X}_t$ mais sa vitesse donnée par :

$$\dot{\underline{X}} = -V - \Omega \wedge \underline{X}$$

où Ω et V sont respectivement les composantes de rotation et de translation du torseur cinématique de la caméra.

Les images sont alors acquises à une cadence proche de la cadence vidéo. D'un point de vue traitement d'images, cette approche repose soit sur une estimation du champ des vitesses apparent (flôt optique), qui s'avère généralement peu robuste et instable, notamment autour des contours d'occultation [Adiv 89], [Waxman 87], soit sur le suivi d'objets d'intérêt (*token trackers*) [Espiau 87], [Xie 89b], [Boukir 93a]. Ces approches sont le plus souvent très sensibles au bruit en raison du faible déplacement qu'elles imposent entre deux images successives.

2.1.2.1 Techniques discrètes

Comme évoqué précédemment, les techniques discrètes sont très proches, dans les concepts de base, de la stéréovision. Cependant, des différences notables existent. La première, fondamentale (et qui peut paraître évidente), est le mouvement qui anime la caméra. Plus ce mouvement sera important, plus l'estimation de la structure 3D de la scène sera robuste et stable. Il faut cependant noter qu'un déplacement "non contrôlé" important de la caméra soulèvera d'autres problèmes : mise en correspondance plus délicate, faible recoupement du contenu des images acquises. Ces méthodes utilisent également la contrainte de rigidité : la scène étant supposée statique, cette contrainte exprime que la distance entre deux primitives 3D reste constante durant le mouvement de la caméra. Cette contrainte signifie aussi que tous les points d'un même objet subissent le même mouvement relatif 3D dans le repère de la caméra. L'utilisation de cette contrainte, sous l'une de ces deux formes, permet souvent la génération d'un système non linéaire, dont la résolution permet la reconstruction 3D.

Certaines des techniques proposées considèrent un nombre très limité d'images (deux ou trois uniquement) [Faugeras 87], [Weng 89], [Wu 88], [Zhang 95]. Une estimation précise de la structure est cependant dans ce cas très difficile, surtout si les images sont bruitées. C'est pourquoi, afin d'améliorer la robustesse de l'estimation, des séquences plus longues d'images sont utilisées [Viala 92] [Boukarri 89], [Crowley 92]. Une méthode originale de factorisation, valide uniquement dans le cadre d'un modèle de projection parallèle, a également été proposée par Tomasi et Kanade [Tomasi 92].

Parmi ces méthodes, certaines reposent sur une connaissance du mouvement de la caméra et d'autres le considèrent comme inconnu. Parmi les premières, les travaux de

Crowley [Crowley 92], qui supposent une connaissance parfaite du mouvement de la caméra, proposent une solution à la reconstruction de droites. La reconstruction s'effectue de façon incrémentale en trois étapes : prédiction, appariement (grâce à la méthode dite du "*tocken tracker*") et enfin mise à jour du modèle de l'environnement de manière récursive grace à un filtrage de Kalman. Safaee-Rad et al. [Safaee-Rad 92] proposent une méthode permettant la reconstruction de primitives plus complexes que des points ou des droites, à savoir des cercles et des ellipses. La méthode est constituée de deux étapes : la détermination de l'orientation de la primitive, puis ensuite la détermination de sa position. L'orientation est obtenue de manière analytique en résolvant un système complexe. Le mouvement de la caméra étant connu *a priori*, le problème revient à déterminer l'intersection de deux cônes généralisés, correspondant à deux points de vue, définis par leur base correspondant à la projection des primitives dans l'image et leur sommets correspondant au centre optique des deux caméras. Ensuite, la position de la primitive est obtenue en réalisant l'intersection du cône correspondant à l'une des vues avec le plan définissant l'orientation. Il faut noter que cette méthode relativement complexe est très sensible au bruit.

La connaissance du mouvement de la caméra peut cependant être considérée comme un *a priori* trop important, surtout dans le cas de la robotique mobile où les erreurs sur l'estimation de la position du robot peuvent être très grandes. Il est donc naturellement apparu des approches plus complexes qui visent non seulement à déterminer la structure 3D de la scène mais aussi le déplacement de la caméra [Aggarwal 88]. Sans connaissance du mouvement de la caméra, le problème de la reconstruction 3D par vision dynamique ne peut être résolu qu'à un facteur d'échelle près. Si la rotation du déplacement peut être estimée, la norme de la translation ne peut pas l'être : il est bien connu en effet que le mouvement rapide d'une caméra éloignée de la scène est équivalent à un mouvement lent d'une caméra proche de celle-ci. Deux types d'algorithmes existent :

- des algorithmes non linéaires fondés sur les équations reliant le déplacement (discret) de la caméra au déplacement des primitives dans l'image (en raison des méthodes mises en œuvre pour la résolution, ces algorithmes sont coûteux en temps de calcul, et peuvent aboutir à une solution fausse) ;
- des algorithmes linéaires qui calculent un ensemble de paramètres intermédiaires (par résolution d'un système linéaire) dont on déduit les paramètres de mouvement. Ces algorithmes sont rapides et mènent généralement à une solution unique mais souvent sous-optimale en raison des approximations réalisées .

Parmi les méthodes proposées, citons les travaux de Weng et al. [Weng 89] reposant sur la mise en correspondance de points dans la séquence d'images. Un petit nombre de points est nécessaire (huit) mais pour augmenter la robustesse au bruit un plus grand nombre de points est utilisé (une estimation de la structure 3D aux moindres carrés est ensuite réalisée). Ces travaux ont été poursuivis en utilisant des droites [Weng 92b] : l'algorithme s'appuie encore sur la redondance dans les données afin d'améliorer la robustesse au bruit. Cependant, un grand nombre de droites doit alors être apparié (13 au minimum), ce qui n'est pas une tâche particulièrement aisée.

Liu et Huang [Liu 86] utilisent également des droites appartenant à une scène rigide pour estimer le déplacement de la caméra. Ils montrent que seul le suivi de deux droites dans deux images est nécessaire pour obtenir les paramètres de rotation. Le système non linéaire qui est obtenu est résolu par une méthode itérative. Les paramètres de translation découlent ensuite de la résolution d'un système linéaire (six droites et trois images sont nécessaires dans le cas général).

Faugeras et al. [Faugeras 87] proposent une méthode plus robuste qui prend explicitement en compte les incertitudes sur les mesures. Les estimations des paramètres de mouvement sont filtrées en utilisant un filtre de Kalman étendu et la structure des droites 3D est ensuite déterminée. Boukarri [Boukarri 89] propose une méthode similaire mais considère des points correspondant aux jonctions de droites 3D (des hypothèses sont en plus posées sur la profondeur moyenne de la scène ainsi que sur le mouvement général de la caméra).

La plupart des études proposées utilisant ces techniques portent sur la reconstruction d'objets simples (points [Faugeras 87], [Wu 88], [Weng 89] ou segments [Faugeras 87], [Weng 92b] [Mitiche 89]). Plusieurs approches traitent cependant du problème de la reconstruction à partir des contours occultants. Les premiers travaux dans ce domaine ont été réalisés par Koenderink [Koenderink 90]. Il a montré qu'il était possible d'obtenir une carte de profondeur des points du contour observé, et de calculer les propriétés intrinsèques de la surface (courbure de Gauss, courbure moyenne) qui donneront l'allure de la surface par la détection des points elliptiques, hyperboliques ou bien paraboliques. Giblin et Weiss [Giblin 87] ont montré que la reconstruction était possible dans le cas d'une projection orthographique. Ils ont établi un ensemble d'équations permettant la reconstruction de la surface comme enveloppe de ses plans tangents. Le plan tangent de la surface en chaque point du contour est défini par le rayon optique passant par ce point et la tangente au contour apparent. Dans le cas d'une projection perspective, Cipolla et Blake [Cipolla 90], [Blake 90] ont montré la faisabilité de la reconstruction à l'aide de trois vues sous la contrainte d'un mouvement linéaire. Une approche originale a été également proposée récemment par Zhao et Mohr [Zhao 94]. La méthode consiste à reconstruire les carreaux de surface 3D à l'aide de B-splines et en utilisant l'observation des contours d'occultation dans une séquence d'images. Zhao et Mohr introduisent aussi une régularisation sur la surface 3D au lieu de réaliser le lissage sur les contours occultants 2D comme cela a été fait dans les approches antérieures. L'objectif final est de fusionner les carreaux de surface reconstruits afin d'obtenir la surface complète. Cette dernière étape est présentée dans [Zhao 93]. Le raccordement des morceaux de surfaces B-Splines reconstruits est effectué en utilisant un maillage triangulaire et une interpolation de surface. Une approche similaire est proposée par Boyer [Boyer 96], la différence principale avec l'approche de Zhao et Mohr réside dans la technique de régularisation de la surface qui, ici, est dans un premier temps triangularisée puis régularisée (les positions des point 3D reconstruits sont optimisées par minimisation d'une certaine fonctionnelle).

2.1.2.2 Techniques continues

Ces techniques s'appuient principalement sur l'observation du mouvement apparent de l'environnement (ou champ de vitesse apparent), ou sur le mouvement des objets dans la séquence d'images. Les variations de positions des primitives dans l'image fournissent un moyen d'estimer d'une part le mouvement parfois inconnu de la caméra et d'autre part la structure 3D de la scène. Si la détermination du champ des vitesses apparentes est indispensable à la résolution du problème, il faut cependant introduire des contraintes supplémentaires (comme dans le cas des approches discrètes) afin de remonter à la structure 3D de la scène et au mouvement de la caméra si celui-ci est inconnu [Aggarwal 88], [Subbarao 87].

Cependant, tout comme le problème de la mise en correspondance de primitives entre deux images successives, celui de la détermination du champ des vitesses apparentes est un problème difficile. En effet, le bruit présent dans les images entraîne des erreurs et une instabilité dans l'estimation du champ de vitesses, qui impliquent des erreurs, parfois importantes, dans l'estimation de la structure 3D de la scène. Pour réaliser une estimation de la structure de la scène, le mouvement de la caméra doit être connu ou estimé. Dans le cas où cette connaissance n'est pas disponible ou pas mesurable, il faudra préalablement procéder à une étape d'estimation de ce mouvement. Parmi les études réalisées dans cette optique, on trouvera les travaux de [Negahdaripour 87] et [Adiv 85].

Les recherches de Negahdaripour et Horn [Negahdaripour 87] ont pour objectif de déterminer, outre le mouvement de la caméra, l'orientation d'une surface plane en utilisant, non pas le champ de vitesse apparent, mais les dérivées spatiale et temporelle de l'intensité lumineuse f (c'est-à-dire $\nabla f = (\frac{\partial f}{\partial X}, \frac{\partial f}{\partial Y})^T$ et $\frac{\partial f}{\partial t}$). La résolution d'un système non linéaire est nécessaire pour obtenir la solution du problème. Adiv [Adiv 85] considère lui un environnement non statique, puisqu'il prend en compte la présence potentielle de plusieurs objets mobiles. Le champ des vitesses apparentes est dans un premier temps segmenté en régions élémentaires dans lesquelles le mouvement est cohérent avec celui d'une surface plane. Les paramètres de mouvement de chacune de ces régions sont ensuite calculés, puis, si besoin est, des régions ayant des mouvements proches sont fusionnées entre elles. Chacune de ces régions est supposée correspondre à un même objet rigide. La segmentation du champ de vitesse étant réalisée, les positions relatives entre chaque objet et la caméra peuvent être déterminées.

Ces méthodes considèrent le mouvement de la caméra totalement inconnu. Or, si l'on se place dans un contexte robotique, ce mouvement est, sinon contrôlable (ou contrôlé), au moins mesurable avec une certaine précision. En considérant des mouvements particuliers de la caméra, Vernon et Tistarelli [Vernon 90] simplifient fortement le calcul du mouvement apparent. En utilisant le champ de vitesse ainsi calculé et les paramètres de mouvement de la caméra, une carte de profondeur est ensuite créée.

Une approche utilisant un système de vision monoculaire en mouvement a été développée par E. Arbogast [Arbogast 91] afin de reconstruire des objets non polyédriques. Il introduit la notion de surface spatio-temporelle : il s'agit de la surface décrite par l'ensemble des contours qui se déplacent sur le plan image, au cours du temps. Les propriétés différentielles de cette surface spatio-temporelle à l'ordre deux sont analysées, après avoir

choisi une paramétrisation adéquate de cette surface. Pour cela, il utilise une modélisation sphérique de la caméra en reprenant la formulation de Blake et Cipolla [Blake 90]. Cela conduit à une paramétrisation très souple et permet de simplifier les équations obtenues. Bien que la méthode soit attrayante du point de vue théorique, elle présente l'inconvénient d'être très sensible au bruit du fait de son aspect différentiel. Ainsi la précision demandée sur les paramètres du mouvement est importante. Par ailleurs, cette méthode ne permet pas de reconstruire les surfaces ne présentant aucun contour apparent (surfaces planes ou concaves).

Des méthodes "hybrides" permettent de combiner les avantages des deux approches discrètes et continues. Fondamentalement cependant, elles appartiennent à la classe des approches continues puisqu'elles s'appuient sur une formulation en termes de vitesse ; par contre, elles utilisent, comme les approches discrètes, une mise en correspondance de primitives particulières (*token trackers*). La détermination problématique du flot optique n'est donc pas à réaliser et, comme les approches continues impliquent de faibles déplacements de la caméra, la mise en correspondance est simplifiée et se ramène à un problème de suivi 2D de primitives. Trois approches similaires ont été proposées en ce sens [Rives 87], [Espiau 87], [Xie 89b], [Boukir 93a]. Espiau et Rives [Rives 87], [Espiau 87] considéraient le cas de primitives de type point. La mesure instantanée du mouvement 2D d'un point dans la séquence d'image est utilisée pour raffiner l'estimation de ses paramètres 3D au fur et à mesure du déplacement du capteur dans la scène en utilisant un filtrage récursif de type filtrage de Kalman. Xie [Xie 89a], [Xie 89b] a étendu cette méthode aux segments. Enfin, Boukir et Chaumette [Boukir 93a], [Chaumette 96] ont généralisé cette approche à tout type de primitive géométrique paramétrable. En se plaçant dans un contexte de vision active, ils ont en plus déterminé les mouvements de la caméra assurant une reconstruction optimale des primitives 3D en termes de suppression des erreurs de discrétisation et minimisation des erreurs de mesure. En raison de la qualité des résultats obtenus, cette méthode a été retenue pour effectuer la reconstruction des primitives que nous considérons. Elle est donc décrite en détail dans la section suivante.

2.2 Reconstruction de primitives 3D par vision dynamique active

2.2.1 Modélisation de l'information visuelle

La caméra est modélisée de manière classique par une projection perspective (voir Figure 2.2). Sans perte de généralité, la distance focale de la caméra est fixée égale à 1. Un point M de coordonnées $\underline{X} = (X\,Y\,Z)$ se projette donc en m de coordonnées $\underline{x} = (x\,y\,1)$ avec :

$$\underline{x} = \frac{1}{Z}\,\underline{X}. \tag{2.1}$$

Soit $\mathcal{P}_s$ une primitive géométrique paramétrable décrite par une équation de la forme :

$$h(\underline{X}, \underline{P}) = 0\ ,\ \forall \underline{X} \in \mathcal{P}_S \tag{2.2}$$

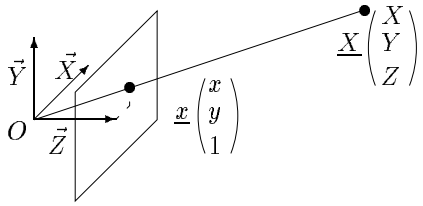


FIG. 2.2 – *Modélisation perspective de la caméra.*

où h définit la nature de la primitive et $\underline{P}$ son vecteur de paramètres. La représentation de $\mathcal{P}_S$, définie par les paramètres $\underline{P}$ (vecteur de dimension n), est choisie complète et minimale de manière à ce que toutes les positions possibles de la primitive puissent être représentées par une seule valeur de $\underline{P}$. Le but de la reconstruction est d'estimer la valeur des paramètres $\underline{P}$ afin de reconstruire et localiser la primitive $\mathcal{P}_S$ définie par h. Cette méthode de reconstruction repose sur une connaissance explicite de la nature de la primitive. Dans le cas où cette nature est inconnue, une étape préalable de reconnaissance est nécessaire. Nous présentons dans le paragraphe 2.2.6 une telle méthode de reconnaissance capable de distinguer les segments 3D des cylindres droits.

Soit $\mathcal{P}_i$ la projection dans l'image de $\mathcal{P}_s$. La primitive $\mathcal{P}_i$ peut s'écrire :

$$g(\underline{x}, \underline{p}) = 0 \ , \ \forall \underline{x} \in \mathcal{P}_i \tag{2.3}$$

où g définit la nature de la primitive et où la valeur des paramètres $\underline{p}$, dépendant de $\underline{P}$, décrit sa configuration dans l'image. La représentation de $\underline{p}$, de dimension m, est choisie complète et minimale de manière à ce que toutes les positions de $\mathcal{P}_i$ puissent être représentées par une seule valeur de $\underline{p}$.

Par ailleurs, en utilisant les équations de projection perspective (2.1), l'équation (2.2) devient :

$$h'(\underline{x}, 1/Z, \underline{P}) = 0. \tag{2.4}$$

Si l'on exclut les cas dégénérés (tels que $\frac{\partial h'}{\partial Z} = 0$, ce qui se produit par exemple lorsqu'un cercle se projette dans l'image sous la forme d'un segment au lieu d'une ellipse), le théorème des fonctions implicites assure l'existence d'une unique fonction μ telle que :

$$1/Z = \mu(\underline{x}, \underline{P}_L) \tag{2.5}$$

où la représentation $\underline{P}_L$, fonction des paramètres $\underline{P}$, est choisie complète et minimale (sa dimension est notée n_L).

Pour des primitives planes (cercle,...), la fonction μ représente le plan dans lequel la primitive se situe. Dans le cas plus général de primitives volumiques (cylindre (voir Figure 2.3), sphère, tore,...), la fonction $g(\underline{x}, \underline{p})$ représente la projection dans l'image des limbes de la primitive et la fonction μ, alors appelée *surface des limbes*, exprime la relation entre les points de $\mathcal{P}_i$ et les points de $\mathcal{P}_s$ correspondants. Les paramètres $\underline{P}_L$, dépendant uniquement de $\underline{P}$, décrivent la configuration de cette surface dans le repère de la caméra.

Soit $T = (V(O), \Omega)$ le torseur cinématique de la caméra où $V(O) = (V_x, V_y, V_z)$ représente la vitesse de translation de la caméra et $\Omega = (\Omega_x, \Omega_y, \Omega_z)$ sa vitesse de rotation. La

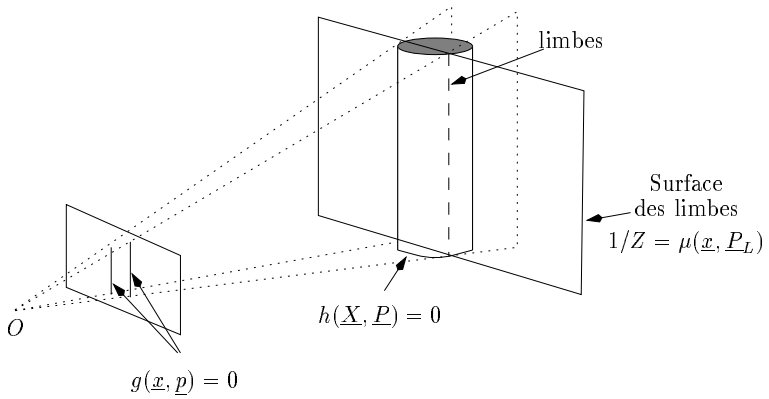


FIG. 2.3 – *Projection de la primitive dans l'image (g) et surface des limbes (μ) dans le cas d'un cylindre (h)*

variation de $\underline{p}$ qui relie le mouvement apparent de la primitive dans l'image au mouvement de la caméra T peut être calculée explicitement et s'exprime par :

$$\dot{\underline{p}} = L_{\underline{p}}^{T}(\underline{p}, \underline{P}_L)\ T \tag{2.6}$$

où $L_{\underline{p}}^{T}(\underline{p}, \underline{P}_L)$ appelée matrice d'interaction associée à $\underline{p}$, caractérise les interactions entre le capteur et la primitive considérée [Espiau 92].

À partir de cette modélisation, il est possible de développer une méthode de reconstruction reposant sur l'analyse du mouvement apparent de la primitive dans l'image et la mesure du mouvement de la caméra.

2.2.2 Reconstruction 3D par vision dynamique

Notons $\mathcal{H}(\underline{p}, \dot{\underline{p}}, \underline{P}_L, T)$ la fonction suivante obtenue à partir de (2.6):

$$\mathcal{H}(\underline{p}, \dot{\underline{p}}, \underline{P}_L, T) = \dot{\underline{p}} - L_{\underline{p}}^{T}(\underline{p}, \underline{P}_L)\ T = 0. \tag{2.7}$$

Si $\dfrac{\partial \mathcal{H}}{\partial \underline{P}_L}$ (de dimension $m \times n_L$) est de rang plein n_L, le théorème des fonctions implicites permet d'exprimer $\underline{P}_L$ en fonction des autres paramètres présents dans l'équation (2.7). La dimension n_L de $\underline{P}_L$ étant minimale, il en résulte une unique solution :

$$\underline{P}_L = \underline{P}_L\ (T, \underline{p}, \dot{\underline{p}}). \tag{2.8}$$

Plus précisément, pour l'ensemble des primitives considérées dans [Boukir 93a] (*i.e.*, points, droites, cercles, sphères et cylindres - voir section 2.2.2), les paramètres $\underline{P}_L$ sont en fait simplement obtenus en résolvant un système linéaire obtenu à partir de l'équation (2.7).

On peut noter qu'il est possible de déterminer les mouvements de la caméra qui ne permettent pas de remonter à la configuration spatiale de la primitive : ils sont tels que $\dfrac{\partial \mathcal{H}}{\partial \underline{P}_L}$ n'est pas de rang plein.

Enfin, connaissant $g(\underline{x}, \underline{p})$ et $\mu(\underline{x}, \underline{P}_L)$, grâce aux contraintes géométriques sur le type de la primitive à reconstruire (droite, cylindre,...), il est possible de remonter aux paramètres $\underline{P}$ qui caractérisent pleinement la configuration de la primitive 3D considérée. La dimension n de $\underline{P}$ ayant été choisie minimale, une solution unique est obtenue :

$$\underline{P} = \underline{P}\,(\underline{p}, \underline{P}_L). \tag{2.9}$$

Cette méthode est basée sur une approche continue puisqu'elle repose sur la mesure de la vitesse de la caméra. De fait, elle est fondamentalement différente des méthodes discrètes [Chien 89], [Mitiche 89], [Viala 92], [Weng 92b] qui considèrent un déplacement de la caméra (décrit par une translation et une rotation). En effet, d'un point de vue géométrique, cette méthode revient à calculer l'intersection entre un cône généralisé (de centre O et de génératrice $g(\underline{X}, \underline{P}) = 0$) et la surface des limbes, alors que les méthodes discrètes, équivalentes au problème de la stéréovision, reposent sur le calcul de l'intersection de deux cônes (dont le centre est situé aux deux positions successives de la caméra - voir Figure 2.4).

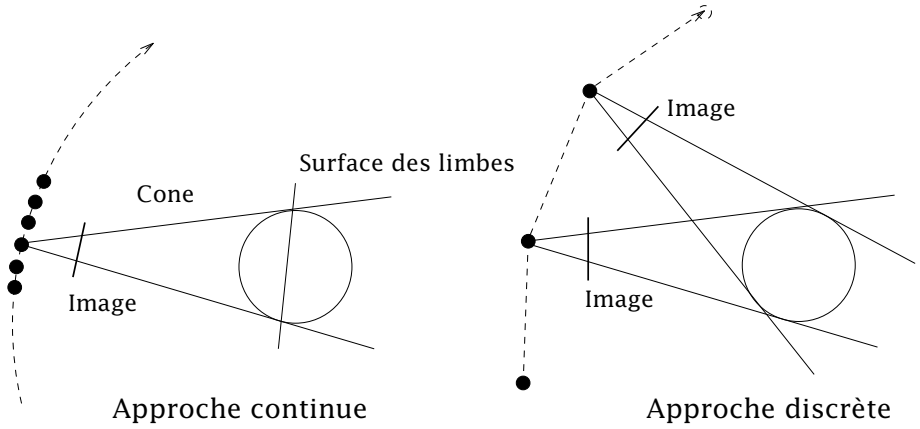


FIG. 2.4 – *Différence entre l'approche continue et l'approche discrète pour l'estimation 3D.*

Nous décrivons à présent, de manière plus précise, les équations de reconstruction associées à quelques primitives : le point, la droite et le cylindre. Pour plus de détails, on peut se reporter à [Boukir 93a].

Le cas du point L'estimation de la position (X_0, Y_0, Z_0) d'un point peut bien entendu être obtenue à partir de cette approche. Dans ce cas simple, l'équation h de la primitive est donnée par :

$$h(\underline{X}, \underline{P}) = \begin{cases} X - X_0 = 0 \\ Y - Y_0 = 0 \\ Z - Z_0 = 0 \end{cases} \tag{2.10}$$

La projection du point dans l'image permet d'écrire l'équation de la fonction g :

$$g(\underline{x}, \underline{p}) = \begin{cases} x - x_0 = 0 \\ y - y_0 = 0 \end{cases} \text{ avec } \begin{cases} x_0 = \frac{X_0}{Z_0} \\ y_0 = \frac{Y_0}{Z_0} \end{cases} \tag{2.11}$$

Enfin, la surface μ est définie par :

$$1/Z = \mu(\underline{x}, \underline{P_L}) = 1/Z_0. \tag{2.12}$$

La profondeur Z_0 est obtenue en utilisant l'équation (2.6) relative au point :

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} -1/Z_0 & 0 & x/Z_0 & xy & -(1+x^2) & y \\ 0 & -1/Z_0 & y/Z_0 & 1+y^2 & -xy & -x \end{pmatrix} T \tag{2.13}$$

qui peut se réécrire sous la forme :

$$A\frac{1}{Z_0} = B \tag{2.14}$$

avec :

$$A = \begin{pmatrix} V_x - x_0 V_z \\ V_y - y_0 V_z \end{pmatrix}, \text{ et } B = \begin{pmatrix} \alpha_x \\ \alpha_y \end{pmatrix} = \begin{pmatrix} x_0 y_0 \Omega_x - (1+x_0)^2 \Omega_y + y_0 \Omega_z - \dot{x}_0 \\ (1+y_0)^2 \Omega_x - x_0 y_0 \Omega_y - x_0 \Omega_z - \dot{y}_0 \end{pmatrix}$$

La solution de l'équation (2.14) est alors donnée par l'équation (2.8):

$$1/Z_0 = \frac{\alpha_x(x_0 V_z - V_x) + \alpha_y(y_0 V_z - V_y)}{(x_0 V_z - V_x)^2 + (y_0 V_z - V_y)^2} \tag{2.15}$$

Les paramètres X_0 et Y_0 sont ensuite aisément déterminés (équation (2.9)) :

$$\begin{aligned} X_0 &= x_0 Z_0 \\ Y_0 &= y_0 Z_0. \end{aligned} \tag{2.16}$$

Il faut noter qu'un mouvement dans la direction du point ($V_x = x_0 V_z, V_y = y_0 V_z$) ne permet pas de remonter à sa position 3D.

Le cas de la droite. Une droite peut être représentée par l'intersection de deux plans orthogonaux (voir Figure 2.5):

$$h(\underline{X}, \underline{P}) = \begin{cases} A_1 X + B_1 Y + C_1 Z = 0 \\ A_2 X + B_2 Y + C_2 Z + D_2 = 0 \end{cases} \text{ avec } \begin{cases} A_1^2 + B_1^2 + C_1^2 = 1 \\ A_2^2 + B_2^2 + C_2^2 = 1 \\ A_1 A_2 + B_1 B_2 + C_1 C_2 = 0. \end{cases} \tag{2.17}$$

Notons que dans ce cas la longueur de la droite est supposée infinie. Dans le cas d'un segment de droite, un processus de calcul de cette longueur ainsi que de la position spatiale des extrémités devra être considéré (voir paragraphe 2.2.5).

Une représentation minimale et complète de la droite 2D correspondante est donnée par :

$$g(\underline{x}, \underline{p}) = x\cos\theta + y\sin\theta - \rho = 0 \text{ avec } \begin{cases} \cos\theta &= A_1/\sqrt{A_1^2 + B_1^2} \\ \sin\theta &= B_1/\sqrt{A_1^2 + B_1^2} \\ \rho &= -C_1/\sqrt{A_1^2 + B_1^2}. \end{cases} \tag{2.18}$$

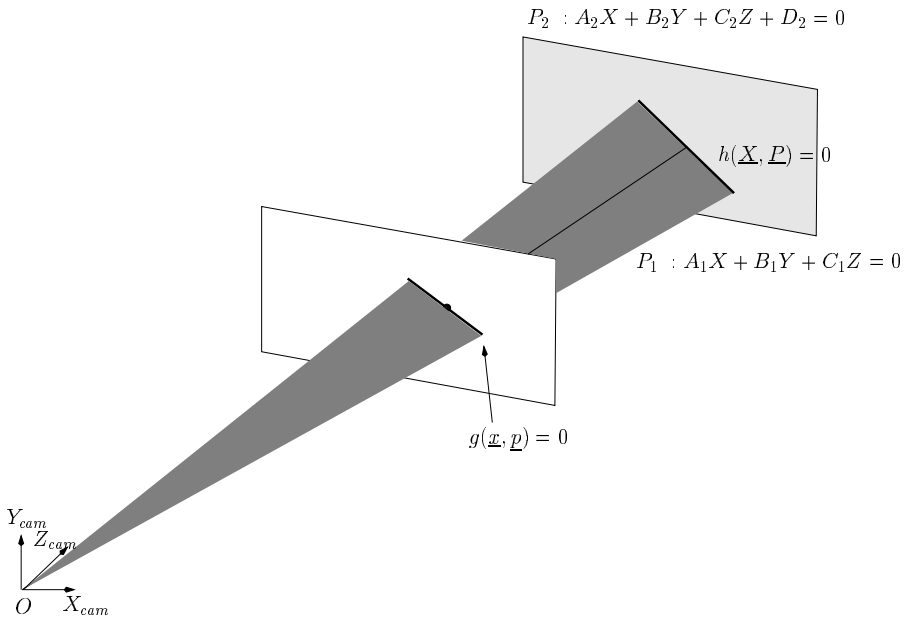


FIG. 2.5 – *Projection d'une droite dans l'image et fonction μ associée*

De plus, la fonction μ définie par (2.5) est obtenue à partir de (2.17) de la manière suivante :

$$1/Z = \mu(\underline{x}, \underline{P_L}) = Ax + By + C \text{ avec } \left\{ \begin{array}{lcl} A & = & -A_2/D_2 \\ B & = & -B_2/D_2 \\ C & = & -C_2/D_2 \end{array} \right. \quad (2.19)$$

Enfin, la relation entre le mouvement de la droite dans l'image (défini par $(\dot{\rho}, \dot{\theta})$) et la vitesse de la caméra T est donnée par la matrice d'interaction associée aux paramètres $\underline{p} = (\rho, \theta)$. Plus précisément nous avons [Espiau 92]:

$$\underline{\dot{p}} = \left(\begin{array}{cccccc} \lambda_\rho \cos\theta & \lambda_\rho \sin\theta & -\lambda_\rho \rho & (1+\rho^2)\sin\theta & -(1+\rho^2)\cos\theta & 0 \\ \lambda_\theta \cos\theta & \lambda_\theta \sin\theta & -\lambda_\theta \rho & -\rho\cos\theta & -\rho\sin\theta & -1 \end{array} \right) T \quad (2.20)$$

avec $\lambda_\rho = -A\rho\cos\theta - B\rho\sin\theta - C$, et $\lambda_\theta = B\cos\theta - A\sin\theta$.

A partir des valeurs mesurées de $\underline{p} = (\rho, \theta)$, $\underline{\dot{p}} = (\dot{\rho}, \dot{\theta})$ et T, l'objectif est donc d'estimer les paramètres des deux plans qui définissent la droite considérée.

Les paramètres A_1, B_1 et C_1 sont immédiatement déduits de ρ et θ. Ensuite, comme décrit précédemment, les paramètres décrivant la fonction μ sont déterminés en utilisant la mesure de la vitesse de la caméra et la vitesse apparente de la droite dans l'image qui en résulte. Ainsi, les paramètres A, B et C sont donnés par la résolution du système linéaire suivant :

$$\left\{ \begin{array}{lcl} -A\rho\cos\theta - B\rho\sin\theta - C & = & \lambda_\rho \\ -A\sin\theta + B\cos\theta & = & \lambda_\theta \\ A\cos\theta + B\sin\theta - C\rho & = & 0 \end{array} \right. \quad (2.21)$$

où λ_ρ et λ_θ sont obtenus à partir de (2.20):

$$\left\{ \begin{array}{rcl} \lambda_\rho & = & \dfrac{\dot{\rho} + (1+\rho^2)(\Omega_Y \cos\theta - \Omega_X \sin\theta)}{V_X \cos\theta + V_Y \sin\theta - \rho V_Z} \\ \\ \lambda_\theta & = & \dfrac{\dot{\theta} + \rho(\Omega_X \cos\theta + \Omega_Y \sin\theta) + \Omega_Z}{V_X \cos\theta + V_Y \sin\theta - \rho V_Z}. \end{array} \right. \tag{2.22}$$

On obtient finalement $D_2 = 1/\sqrt{A^2 + B^2 + C^2}$, $A_2 = -AD_2$, $B_2 = -BD_2$ et $C_2 = -CD_2$.

Par ailleurs, les mouvements de la caméra qui ne permettent pas de remonter aux caractéristiques 3D de la droite sont tels que $V_X \cos\theta + V_Y \sin\theta - \rho V_Z = 0$ (voir (2.22)). Ces mouvements sont tels que le centre de projection O appartienne constamment au plan défini par la position initiale de O et la projection de la droite dans l'image.

Le cas du cylindre La même méthode peut être utilisée pour réaliser la reconstruction d'un cylindre droit circulaire. L'équation du cylindre est donnée par :

$$h(\underline{X}, \underline{P}) = (X - X_0)^2 + (Y - Y_0)^2 + (Z - Z_0)^2 - (UX + VY + WZ)^2 - R^2 = 0 \tag{2.23}$$

où

- R est le rayon du cylindre ;
- (U, V, W) représente le vecteur directeur de l'axe du cylindre (de norme 1) ;
- (X_0, Y_0, Z_0) sont les coordonnées du point de l'axe du cylindre qui est le plus proche du centre optique O de la caméra.

Les paramètres à estimer sont donc ce cas : $\underline{P} = (U, V, W, X_0, Y_0, Z_0, R)$. Notons que dans ce cas aussi, la longueur du cylindre est supposée infinie, un processus de calcul de la position spatiale des extrémités devra être considéré (voir paragraphe 2.2.5).

Dans les cas non dégénérés, la projection d'un cylindre dans l'image est constituée de deux droites $\mathcal{D}_1(\rho_1, \theta_1)$ et $\mathcal{D}_2(\rho_2, \theta_2)$. La fonction relative à la surface des limbes μ et définie par (2.5) est donnée par [Boukir 93a] :

$$1/Z = \mu(\underline{x}, \underline{P}_L) = A\,x + B\,y + C \text{ avec } \left\{ \begin{array}{l} A = X_0/(X_0^2 + Y_0^2 + Z_0^2 - R^2) \\ B = Y_0/(X_0^2 + Y_0^2 + Z_0^2 - R^2) \\ C = Z_0/(X_0^2 + Y_0^2 + Z_0^2 - R^2) \end{array} \right. \tag{2.24}$$

où A, B et C sont les composantes du vecteur normal à la surface des limbes du cylindre.

Le mouvement apparent du cylindre dans l'image est décrit par la matrice d'interaction relative aux paramètres $\underline{p} = (\rho_1, \theta_1, \rho_2, \theta_2)$:

$$\dot{\underline{p}} = \begin{pmatrix} \lambda_{\rho_1}\cos\theta_1 & \lambda_{\rho_1}\sin\theta_1 & -\lambda_{\rho_1}\rho_1 & (1+\rho_1^2)\sin\theta_1 & -(1+\rho_1^2)\cos\theta_1 & 0 \\ \lambda_{\theta_1}\cos\theta_1 & \lambda_{\theta_1}\sin\theta_1 & -\lambda_{\theta_1}\rho_1 & -\rho_1\cos\theta_1 & -\rho_1\sin\theta_1 & -1 \\ \lambda_{\rho_2}\cos\theta_2 & \lambda_{\rho_2}\sin\theta_2 & -\lambda_{\rho_2}\rho_2 & (1+\rho_2^2)\sin\theta_2 & -(1+\rho_2^2)\cos\theta_2 & 0 \\ \lambda_{\theta_2}\cos\theta_2 & \lambda_{\theta_2}\sin\theta_2 & -\lambda_{\theta_2}\rho_2 & -\rho_2\cos\theta_2 & -\rho_2\sin\theta_2 & -1 \end{pmatrix} T \tag{2.25}$$

$$\text{avec} \begin{cases} \lambda_{\rho_1} = -(A\rho_1 \cos\theta_1 + B\rho_1 \sin\theta_1 + C) \\ \lambda_{\theta_1} = B\cos\theta_1 - A\sin\theta_1 \\ \lambda_{\rho_2} = -(A\rho_2 \cos\theta_2 + B\rho_2 \sin\theta_2 + C) \\ \lambda_{\theta_2} = B\cos\theta_2 - A\sin\theta_2. \end{cases} \tag{2.26}$$

Les paramètres $\underline{P_L} = (A, B, C)$ peuvent être calculés en résolvant le système linéaire (2.26), λ_{ρ_1}, λ_{θ_1}, λ_{ρ_2} et λ_{θ_2} étant directement obtenus à partir de (2.25), comme dans le cas de la droite.

On peut déduire de (2.25) que les mouvements de la caméra qui ne permettent pas de résoudre le système linéaire (2.26) sont tels que le centre de projection O se maintienne constamment dans l'un des plans définis par la position initiale de O et les projections des limbes du cylindre (*i.e.* $V_X \cos\theta_1 + V_Y \sin\theta_1 - \rho_1 V_Z = 0$, ou $V_X \cos\theta_2 + V_Y \sin\theta_2 - \rho_2 V_Z = 0$). Dans ce cas, seulement deux équations de (2.26) sont valides alors que trois sont nécessaires pour obtenir une solution unique. De plus, si le mouvement de la caméra est tel que O appartienne à une droite résultant de l'intersection de deux plans (*i.e.* tel que $V_X \cos\theta_1 + V_Y \sin\theta_1 - \rho_1 V_Z = 0$ et $V_X \cos\theta_2 + V_Y \sin\theta_2 - \rho_2 V_Z = 0$), alors λ_{ρ_1}, λ_{θ_1}, λ_{ρ_2} et λ_{θ_2} sont tous indéterminés.

Enfin, les paramètres $\underline{P}$ décrivant la position et le rayon du cylindre peuvent aisément être obtenus à partir de l'intersection de la surface des limbes $\mathcal{P}_L$ et des plans $\mathcal{P}_{t1}$ et $\mathcal{P}_{t2}$, définis à partir O et des deux droites $\mathcal{D}_1$ et $\mathcal{D}_2$ observées dans l'image (cf Figure 2.6). Cette intersection est formée de deux droites $\mathcal{D}_{\mathcal{P}_L 1}$ et $\mathcal{D}_{\mathcal{P}_L 2}$, à partir desquelles il est possible de déterminer la position de l'axe du cylindre. Son équation, suffisante pour déterminer les paramètres U, V, W, X_0, Y_0 et Z_0, est en effet donnée en calculant l'intersection des deux plans $\mathcal{P}_{n1}$ et $\mathcal{P}_{n2}$ orthogonaux à $\mathcal{P}_{t1}$ et $\mathcal{P}_{t2}$ et contenant respectivement $\mathcal{D}_{\mathcal{P}_L 1}$ et $\mathcal{D}_{\mathcal{P}_L 2}$ [Boukir 93a]. Enfin, le rayon du cylindre est obtenu en calculant la distance entre l'axe du cylindre et $\mathcal{D}_{\mathcal{P}_L 1}$ ou $\mathcal{D}_{\mathcal{P}_L 2}$.

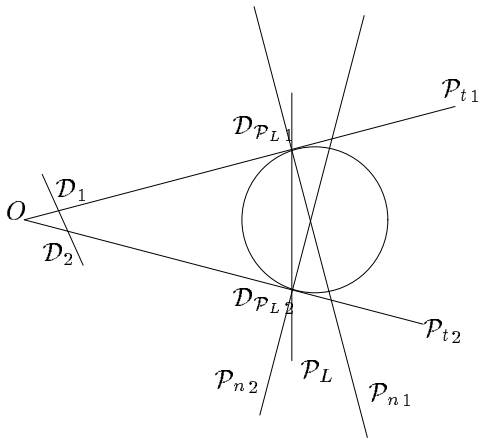


FIG. 2.6 – *Vue de dessus d'un cylindre.*

Notons que la reconstruction d'un cylindre peut aussi s'effectuer en utilisant la projection d'un seul limbe. L'utilisation de la matrice d'interaction associée à l'unique limbe

considéré pour deux positions de la caméra permet de remonter aux paramètres 3D du cylindre [Boukir 93a]. Cette méthode sera utilisée lors de la phase de reconnaissance de primitive décrite en 2.2.6.

2.2.3 Reconstruction tridimensionnelle par vision active

Malgré son élégance, cette approche s'avère inefficace pour reconstruire précisément une primitive géométrique. De par son aspect continu, elle est confrontée aux erreurs de discrétisation et est, de plus, très sensible au bruit. Classiquement, pour tenter de résoudre ces problèmes, et pour augmenter la robustesse des solutions, des processus de filtrage récursif [Matthies 89], [Boukarri 89], [Viala 92], [Crowley 92] ou des processus d'optimisation non linéaire [Weng 92b] sont mis en œuvre. Une autre solution pour résoudre ce problème est d'utiliser le paradigme de la vision active [Aloimonos 90][Bajcsy 88][Swain 93].

2.2.3.1 Reconstruction 3D par vision active : état de l'art

Contrairement à la vision dynamique où l'on se contente d'observer le mouvement de la caméra, dans le cas de la vision active, le mouvement est commandé. Si les travaux portant sur la vision active se multiplient depuis quelques années, peu de chercheurs ont tenté de savoir si, en ce qui concerne le problème de la structure à partir du mouvement, il existait des trajectoires de l'observateur qui soient plus bénéfiques que d'autres.

Sandini et Tistarelli ont proposé deux approches, l'une reposant sur une coopération entre la stéréovision et le mouvement [Sandini 86] et l'autre fondée sur l'utilisation d'un système monoculaire en mouvement [Sandini 90]. La première approche, visant à l'estimation robuste d'une carte de profondeur [Sandini 86], utilise un capteur stéréoscopique en mouvement. L'objectif est de définir des stratégies de déplacement du capteur qui permettent au système stéréoscopique d'acquérir des informations fiables. De plus, la stéréovision doit aider l'acquisition de l'information 3D obtenue à partir du mouvement. En effet, un capteur stéréoscopique est inadapté pour acquérir des informations fiables sur les contours dont l'orientation est proche des droites épipolaires, alors que l'analyse à partir du mouvement est, elle, problématique sur les contours dont l'orientation dans l'image est proche de la projection dans l'image de la direction du mouvement de translation. Une optimisation de ces deux systèmes est obtenue en contrôlant la caméra de telle manière que le point de fixation reste immobile le long de la séquence d'images, la trajectoire du mouvement étant elle-même contrainte sur un plan vertical. Le champ de vitesse apparent est obtenu en deux étapes : les vitesses de rotation sont obtenues en mesurant l'angle de rotation de la caméra autour de l'axe x, puis les composantes de translation sont obtenues en utilisant cet angle et la distance au point de fixation obtenu par stéréovision. Ces informations ainsi que les composantes de la vitesse normale aux contours permettent une interprétation complète du champ des vitesses apparentes qui conduit à la détermination d'une carte de profondeur. Cette carte est ensuite fusionnée avec la carte de profondeur obtenue par stéréovision.

La deuxième approche proposée par Sandini et Tistarelli [Sandini 90] utilise une approche monoculaire active en ce sens que, là encore, la caméra est contrôlée afin de stabiliser

le point de fixation le long de la séquence d'image (stratégie proche de celle d'Aloimonos [Aloimonos 87] dans l'étude du problème de "*structure from motion*"). Ces contraintes imposées sur le mouvement de la scène dans l'image permettent de remonter plus aisément aux paramètres de mouvement de la caméra. L'estimation de la structure 3D de la scène repose sur la mise en correspondance de points, l'analyse du mouvement apparent de ces points dans la séquence d'images ainsi que l'estimation du mouvement de la caméra. Il reste que l'apport de la vision active dans ces deux approches n'est pas des plus évident. Une étude quantitative (voire même qualitative) aurait été intéressante.

Huang et Aloimonos [Huang 91] se placent plutôt dans le cadre de la vision intentionnelle pour résoudre le problème de l'estimation de la structure à partir du mouvement. Le problème traité porte sur la localisation d'objets mobiles les uns par rapport aux autres. La solution proposée permet d'éviter les étapes difficiles de l'estimation du mouvement 3D et de l'estimation du champ de mouvement apparent. Elle repose uniquement sur l'utilisation des dérivées spatiales et temporelles de la fonction d'intensité lumineuse et donc sur les composantes perpendiculaires aux contours des vitesses apparentes. Cette estimation est en effet beaucoup plus facile à réaliser. Cependant, la carte de profondeur ainsi déterminée ne sera pas complète. La stratégie de déplacement de la caméra est très simple puisqu'elle consiste en un mouvement de translation le long de l'axe optique. Cette approche donne de bons résultats relativement robustes aux bruits excepté bien sûr dans la zone proche de l'axe optique.

Dans le cas de la reconstruction 3D de primitives géométriques, Espiau et Rives [Rives 87], [Espiau 87] avaient montré, dans le cas des points, que la qualité de l'estimation est très sensible à la nature des mouvements de la caméra. S. Boukir a donc déterminé quels étaient les mouvements de la caméra qui, pour une primitive donnée, permettaient d'obtenir une estimation optimale de ces paramètres [Boukir 93a]. On verra que le modèle de l'asservissement visuel [Hashimoto 93], [Chaumette 90], [Espiau 92] est parfaitement adapté à ce formalisme.

2.2.3.2 Reconstruction 3D optimale d'une primitive

La qualité de l'estimation des paramètres d'une primitive à partir d'une séquence d'images étant très sensible aux mouvements de la caméra, l'objectif présenté dans [Boukir 93a], [Chaumette 96], et que nous rappelons ici, est de déterminer les mouvements adéquats de celle-ci afin d'obtenir une estimation optimale des caractéristiques de la primitive. Ce problème d'optimisation a été abordé sous deux aspects différents :

- la suppression des erreurs de discrétisation ;
- la minimisation des effets des erreurs de mesure (erreurs sur les mesures dans les images et sur la mesure du mouvement de la caméra).

A. Suppression des erreurs de discrétisation La méthode de reconstruction 3D présentée précédemment appartient à la famille des techniques continues. Elle repose donc sur la mesure de la vitesse apparente de la primitive dans l'image (*i.e.* $\underline{\dot{p}}$, la vitesse des

paramètres représentant la projection de la primitive). Or cette valeur $\dot{\underline{p}}$ n'est pas directement mesurable : les valeurs mesurées dans la séquence d'images n'autorisent que le calcul de $\Delta\underline{p}$ qui représente la variation des paramètres $\underline{p}$ pendant l'intervalle de temps Δt entre deux images. L'utilisation de $\Delta\underline{p}/\Delta t$ à la place de $\dot{\underline{p}}$ dans l'estimation des paramètres 3D de la primitive entraîne des erreurs importantes. Une manière de résoudre ce problème est d'assurer l'égalité suivante :

$$\dot{\underline{p}} = \frac{\Delta \underline{p}}{\Delta t}, \forall t. \tag{2.27}$$

Dès lors la discrétisation n'aura plus d'effet sur la qualité de la reconstruction. Une telle condition ne sera satisfaite que si :

$$\ddot{\underline{p}} = \cdots = \underline{p}^{[n]} = 0, \forall t. \tag{2.28}$$

De (2.6), notée ici $\dot{\underline{p}} = L_{\underline{p}}^T(\underline{p}, \underline{P}_L)\ T = f(\underline{p}, \underline{P}_L, T)$, il est possible de déduire :

$$\ddot{\underline{p}} = L_{\underline{p}}^T(\underline{p}, \underline{P}_L)\ \dot{T} + \frac{\partial f}{\partial \underline{p}}\dot{\underline{p}} + \frac{\partial f}{\partial \underline{P}_L}\dot{\underline{P}}_L. \tag{2.29}$$

Une condition suffisante pour satisfaire (2.28) est de contraindre les mouvements la caméra de manière à assurer :

$$\dot{\underline{p}} = \dot{\underline{P}}_L = 0, \forall t. \tag{2.30}$$

Dans ce cas en effet, $T \in$ Ker $L_{\underline{p}}^T, \forall t$. En utilisant (2.30) on peut alors montrer que $\dot{T} \in$ Ker $L_{\underline{p}}^T, \forall t$, d'où on déduit que $\ddot{\underline{p}} = 0, \forall t$. Une méthode récurrente permet alors de montrer que (2.28) est toujours vérifiée.

En d'autres termes, une solution pour supprimer les erreurs de discrétisation est que la surface des limbes de la primitive demeure immobile dans le repère mobile de la caméra et que, de plus, la projection de la primitive 3D apparaisse à la même position dans l'image quand la caméra est en mouvement. Il est possible de montrer que, excepté dans le cas du point et des droites, la première condition $\dot{\underline{p}} = 0$ implique la deuxième $\dot{\underline{P}}_L = 0$, ce qui réduit alors le problème à un problème de *fixation*.

Remarque : La condition proposée dans [Boukir 93a], [Chaumette 96] pour supprimer les erreurs de discrétisation est seulement suffisante, mais pas forcement nécessaire. En effet il existe des mouvements de caméra tel que $\ddot{\underline{p}} = 0$ avec $\dot{\underline{p}} \neq 0$. Pour un point par exemple, on peut montrer [Chaumette 96] que $\ddot{\underline{p}} = 0$ quand la caméra se déplace suivant un mouvement de translation parallèle au plan image à vitesse constante (*i.e.*$V_X = V_1, V_Y = V_2$ et $V_Z = \Omega_X = \Omega_Y = \Omega_Z = 0$). De manière plus générale, calculer toutes les solutions du système non linéaire (2.28) semble hors de portée. De plus, comme ces solutions reposent sur la connaissance de $\frac{\partial f}{\partial \underline{p}}$ et $\frac{\partial f}{\partial \underline{P}_L}$, elles dépendent fortement de la nature de la primitive considérée. D'un autre coté, la condition (2.30) est valide quelle que soit la nature de la primitive. Elle présente en outre l'avantage de maintenir la primitive dans le champ de vision de la caméra tout au long du processus d'estimation.

B. Minimisation des effets des erreurs de mesure La configuration d'une primitive dans l'image influe sur la qualité de son estimation. Il est ainsi montré dans [Boukir 93a] que certaines contraintes sur la position de la primitive dans l'image ainsi que des contraintes sur le mouvement de la caméra sont nécessaires à une bonne estimation des paramètres de la primitive considérée.

Soit P l'un des paramètres représentant la primitive à reconstruire, les erreurs sur l'estimation de P sont étroitement liées aux erreurs de mesure sur les informations visuelles $\underline{p}$ et $\underline{\dot{p}}$ ainsi que sur le mouvement de la caméra T. Si les erreurs de mesure sur $\underline{p}$, $\underline{\dot{p}}$ et T sont supposées décorrélées, l'incertitude σ_P sur l'estimation de P peut alors s'exprimer de la manière suivante :

$$(\sigma_P)^2 = \sum_{i=1}^{m}(\frac{\partial P}{\partial p_i})^2(\sigma_{p_i})^2 + \sum_{j=1}^{m}(\frac{\partial P}{\partial \dot{p}_j})^2(\sigma_{\dot{p}_j})^2 + \sum_{k=1}^{6}(\frac{\partial P}{\partial T_k})^2(\sigma_{T_k})^2. \quad (2.31)$$

où $p_{i(i=1..m)}$, $\dot{p}_{j(j=1..m)}$, et $T_{k(k=1..6)}$ sont respectivement les différentes composantes de $\underline{p}$, $\underline{\dot{p}}$ et T. Minimiser σ_P revient alors à minimiser chaque terme $\left(\frac{\partial P}{\partial a}\right)^2$ où a représente chacune des variables $p_i, \dot{p}_j$ et T_k. Il reste alors à déterminer les valeurs de $\underline{p}$ telles que :

$$\left(\frac{\partial \left(\frac{\partial P}{\partial a}\right)^2}{\partial p_j}\right) = 0 \ , \quad \forall a \text{ et } \forall j = 1 \text{ à } m. \quad (2.32)$$

Déterminer la solution d'un tel système de manière analytique semble impossible. Cependant la dérivation de solutions particulières intéressantes reste possible. Ainsi, des solutions ont été déterminées pour les principales primitives géométriques (point, droite, cylindre, sphère, cercle) [Boukir 93a]. Par exemple :

- Dans le cas du **point**, les effets des erreurs de mesure sur sa profondeur Z sont minimisés si le point apparaît constamment au centre de l'image (*i.e.*, $x = \dot{x} = y = \dot{y} = 0, \forall t$) et si de plus on a $V_Z = \Omega_Z = 0$. Ceci signifie que la caméra doit se déplacer sur une sphère dont le centre est le point à reconstruire.

- Dans le cas d'une **droite**, les effets des erreurs de mesure sont minimisées si celle-ci apparait centrée et verticale (ou horizontale) dans l'image, et si, de plus, $V_Y = V_Z = \Omega_X = 0$ (ou $V_X = V_Z = \Omega_Y = 0$) (voir Figure 2.7.a).

- Enfin, dans le cas du **cylindre**, les effets des erreurs de mesure sont minimisés s'il se projette sous la forme de deux droites symétriques parallèles verticales (ou horizontales) dans l'image avec en outre $V_Y = 0$ (ou $V_X = 0$). La caméra doit alors se déplacer sur un cercle dont le centre est traversé par l'axe du cylindre et perpendiculaire à celui-ci (cf Figure 2.7.b).

Les deux tâches nécessaires au processus d'"estimation optimale" des paramètres 3D d'une primitive sont donc une tâche de fixation et une tâche de focalisation. Ces deux tâches sont similaires à celle spécifiée par Sandini [Sandini 90] ou par Aloimonos [Aloimonos 87] dans les solutions qu'ils proposaient au problème de reconstruction à partir du mouvement

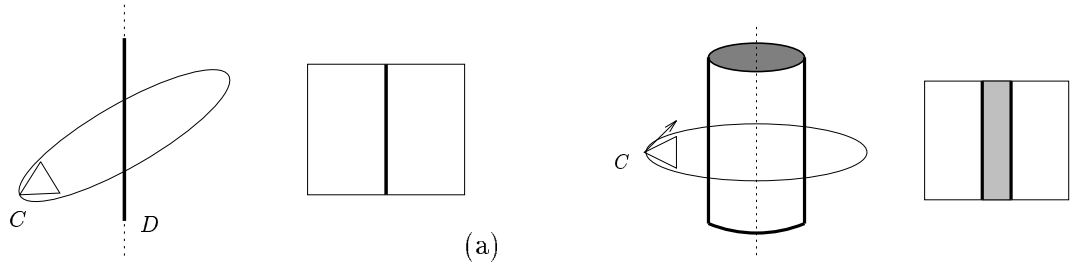


FIG. 2.7 – *Mouvements optimaux de la caméra pour l'estimation (a) d'une droite et (b) d'un cylindre*

(rappelons que cette tâche consistait à stabiliser le point de fixation ("*gaze control*") le long de la séquence d'images). Les conclusions de Samia Boukir vont cependant plus loin puisque dans chaque cas des contraintes supplémentaires ont été imposées sur la trajectoire de la caméra (suppression des mouvements en direction et autour de l'axe optique dans le cas du point ($V_Z = \Omega_Z = 0$), mouvement le long de l'axe dans le cas de la droite ($V_Y = V_Z = \Omega_X = 0$) ou du cylindre ($V_Y = 0$). En effet, ces mouvements n'apportent aucune information dans le cadre du processus de reconstruction 3D. On rejoint ici le principe de la vision intentionnelle [Aloimonos 90], où les mouvements (ou actions) inutiles sont supprimés.

Les techniques d'asservissement visuel [Espiau 92], [Hashimoto 93] permettent d'effectuer automatiquement les différents mouvements satisfaisant les contraintes nécessaires à une estimation optimale (tâches de fixation et de focalisation). L'asservissement visuel consiste en effet à introduire directement et en boucle fermée les informations extraites de l'image dans une boucle de commande. Dans notre cas, les tâches s'expriment comme la régulation à zéro d'une fonction de tâche combinant une tâche primaire (telle que la primitive apparaisse dans l'image à sa position désirée) et une tâche secondaire (ici un suivi de trajectoire prenant en compte l'évitement des butées articulaires du robot). Cette tâche secondaire est construite afin de permettre des mouvements de la caméra tout en assurant la réalisation de la tâche primaire (voir [Espiau 92], [Boukir 93a], [Chaumette 96] et l'Annexe D pour de plus amples détails).

Si l'approche retenue pour la reconstruction permet une estimation très précise et robuste des paramètres des primitives concernées (voir les résultats donnés au paragraphe 2.2.7), elle présente un certain nombre de défauts :

- elle implique une connaissance *a priori* sur la nature de la primitive observée. Dans le cas qui nous intéresse, nous avons fait l'hypothèse que la scène est constituée uniquement de cylindres ou de segments 3D. Nous proposons donc dans le paragraphe 2.2.6 un test statistique reposant sur l'analyse du rapport du maximum de vraisemblance qui permet de déterminer, en parallèle avec l'estimation de ses paramètres, la nature de la primitive considérée.

- elle suppose, dans le cas des primitives qui nous intéressent, que ces primitives ont une longueur infinie. Nous proposons donc dans le paragraphe 2.2.5 une méthode permettant de déterminer précisément la position 3D des extrémités de la primitive. Cette méthode repose sur la réalisation de mouvements complémentaires de la caméra réalisés par asservissement visuel.

- elle ne prend pas en compte les autres objets de la scène, ni les contraintes mécaniques du manipulateur pendant les phases de reconstruction. Nous proposons dans le paragraphe suivant une méthode permettant d'éviter les occlusions pendant la reconstruction. Une technique permettant d'éviter les butées articulaires du manipulateur est présentée en annexe D.

2.2.4 Évitement des occlusions pendant la reconstruction

Pour effectuer l'estimation des paramètres d'une primitive, un mouvement particulier de la caméra autour de celle-ci est nécessaire. Malheureusement, pendant ce mouvement, une occlusion peut se produire et faire disparaître l'objet que l'on est en train de reconstruire (un segment dans le cas de la Figure 2.8).

Si l'on considère de manière locale la reconstruction des primitives 3D (c'est-à-dire sans connaissances sur la position spatiale d'autres primitives), il convient de prendre en compte uniquement les informations présentes dans la séquence d'images pour détecter ces occlusions avant leur occurrence.

Il est possible de prédire une prochaine occlusion en cas d'apparition d'un objet dans la périphérie de la primitive considérée. De manière à détecter l'apparition de ces objets autour de la primitive sur laquelle est focalisée la caméra, nous utilisons des "observateurs" [Djian 96], disposés dynamiquement autour de la primitive suivie et sur lesquels un traitement d'image particulier est effectué (voir Figure 2.8). Cette notion d'observateur dynamique est assez classique en vision active [Djian 96] puisqu'elle consiste à rechercher une information à l'endroit où elle est éventuellement attendue. Plus précisément, afin de simplifier le problème, et de respecter la cadence vidéo, nous considérons que ces objets sont des segments 2D. L'algorithme utilisé pour cette détection est similaire à celui qui permet de suivre les segments 2D considérés pour la reconstruction (cet algorithme est décrit en annexe B). Dans le cas où un tel objet/segment est découvert, et se rapproche de la primitive à reconstruire, la caméra est animée d'un mouvement dans la direction opposée pour éviter l'occlusion potentielle (voir Figure 2.8).

2.2.5 Position spatiale d'une primitive le long de son axe

Le processus de reconstruction optimale proposé dans [Boukir 93a] et décrit précédemment considère que les droites et cylindres droits ont une longueur infinie. Il est cependant nécessaire de déterminer la longueur d'un segment 3D ou d'un cylindre ainsi que leur position spatiale le long de leur axe. De manière à déterminer précisément la position 3D des extrémités d'une primitive, celles-ci doivent être successivement observées au centre de l'image, ce qui nécessite un mouvement de la caméra (voir les Figures 2.9 et 2.10). Quand la caméra a atteint la position désirée, la position du point 3D correspondant est

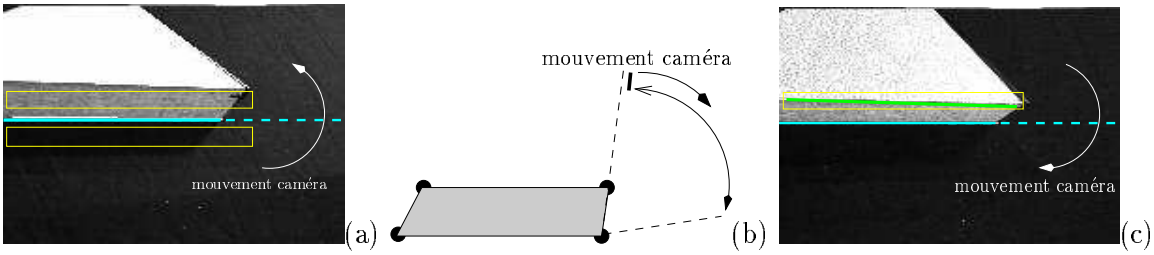


FIG. 2.8 – *Évitement des occlusions pendant la phase de reconstruction (a) segment suivi pour la reconstruction et position des observateurs dans l'image (b) mouvement de la caméra pendant le processus (c) détection d'un segment dans le voisinage du segment suivi par l'un des observateurs, ce qui entraîne un mouvement de rotation de la caméra dans la direction opposée.*

simplement calculée par l'intersection de l'axe de la primitive avec l'axe optique de la caméra.

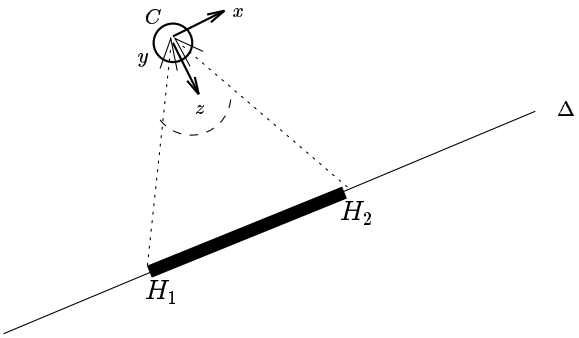


FIG. 2.9 – *Calcul de la longueur d'un segment par rotation autour d'un axe de la caméra*

Dans le cas d'un segment, la méthode utilisée est la suivante :

- Après la phase d'estimation, le segment que l'on reconstruit est centré verticalement ou horizontalement dans l'image. La droite porteuse du segment se projette selon un des axes $\vec{x}$ ou $\vec{y}$ de la caméra.
- On effectue alors une rotation autour de l'axe $\vec{y}$ (respectivement $\vec{x}$) si le segment à reconstruire est centré horizontalement (respectivement verticalement) jusqu'à ce que l'extrémité du segment apparaisse au centre de l'image.
- À partir de la position de la caméra correspondant à une des extrémités du segment et des paramètres de la droite qui ont déjà été estimés lors de la phase précédente, on détermine les coordonnées 3D de l'extrémité H_1 du segment en effectuant l'intersection de l'axe optique et de Δ.
- Les coordonnées de la deuxième extrémité H_2 sont déterminées de manière semblable.

La génération des mouvements répondant à ces spécifications est réalisée en utilisant l'asservissement visuel. Les lois de commande utilisées sont données en annexe D.

Une seconde méthode consistant à réaliser un mouvement de translation le long de l'axe de la primitive à distance constante de celui-ci peut aussi être appliquée dans le cas du segment. Dans un premier temps, il convient alors, afin d'obtenir une meilleure estimation de la position des extrémités, d'effectuer une correction de l'orientation de la caméra de manière à ce que l'axe optique soit perpendiculaire au segment considéré. Dans un deuxième temps, des mouvements de translation dans la direction de l'axe du segment sont effectués de manière à observer les extrémités au centre de l'image, le calcul de la position 3D des extrémités H_1 et H_2 se faisant alors de la même manière que dans la méthode précédente. Cette méthode est théoriquement plus robuste puisque l'intersection portera sur des droites perpendiculaires (aux erreurs de reconstruction 3D près). Cependant, elle est plus coûteuse en temps d'exécution et n'a donc pas été utilisée en pratique dans le cas des segments.

Par contre, cette méthode a été utilisée pour estimer la longueur d'un cylindre. En effet, avec un mouvement de rotation autour des axes $\vec{x}$ ou $\vec{y}$, la projection des limbes du cylindre n'apparaît plus parallèle verticale centrée dans l'image, ce qui suppose de modifier la tâche primaire du processus d'asservissement (en s'asservissant sur la bissectrice des deux droites perçues dans l'image par exemple). Pour simplifier les lois de commande, nous avons donc choisi d'effectuer un mouvement de translation le long de l'axe du cylindre (voir Figure 2.10).

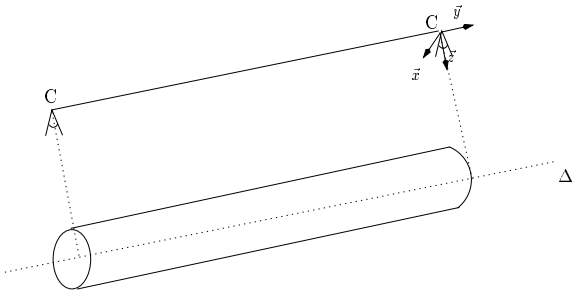


FIG. 2.10 – *Calcul de la longueur d'un cylindre par translation de la caméra le long de l'axe du cylindre*

2.2.6 Reconnaissance de primitives

La seule hypothèse relative à la scène, outre les dimensions d'un volume l'englobant, est qu'elle est constituée uniquement de segments et de cylindres.

La méthode de reconstruction présentée ci-dessus implique une connaissance *a priori* sur la nature de la primitive observée (segment ou cylindre). Un processus de reconnaissance est donc nécessaire. D'un point de vue géométrique, on peut facilement distinguer laquelle de ces deux primitives est observée en fonction du mouvement projeté dans l'image. En effet, ce mouvement est différent puisque les paramètres de la surface définie par μ

(équation (2.19) dans le cas de la droite, équation (2.24) dans le cas du cylindre) apparaissent dans la matrice d'interaction associée à ces primitives. Pour obtenir un critère robuste, nous supposons dans un premier temps que le segment considéré correspond à l'image d'un limbe d'un cylindre. Une estimation de ses paramètres 3D (en particulier de son rayon R), basée sur ce segment 2D, est alors réalisée. Quand cette estimation est effectuée, deux hypothèses sont considérées :

$\boldsymbol{H_1}$: la primitive observée était en fait un segment.

$\boldsymbol{H_2}$: la primitive observée était effectivement un cylindre de rayon R ;

Un test au maximum de vraisemblance est alors utilisé pour déterminer laquelle de ces deux hypothèses est correcte. Notons L_0 et L_1 les fonctions de vraisemblance associées aux hypothèses H_0 et H_1. En supposant que les estimations successives R_i du rayon du cylindre suivent une loi Gaussienne de moyenne R et de variance σ^2, nous obtenons :

$$L_0 = \left(\frac{1}{2\pi\sigma^2}\right)^{\frac{N}{2}} e^{-\frac{\sum_{i=1}^{N} {R_i}^2}{2\sigma^2}}, \qquad L_1 = \left(\frac{1}{2\pi\sigma^2}\right)^{\frac{N}{2}} e^{-\frac{\sum_{i=1}^{N} (R_i - R)^2}{2\sigma^2}} \tag{2.33}$$

Le choix de l'hypothèse H_1 contre l'hypothèse H_0 repose sur l'optimisation du rapport de vraisemblance ξ donné par $\xi = \log \frac{L_1}{L_0}$. En substituant dans cette équation les expressions données dans (2.33), on arrive à :

$$\xi = -\frac{1}{2\sigma^2}\left(\sum_{i=1}^{N} (R_i - R)^2 - \sum_{i=1}^{N} {R_i}^2\right) \tag{2.34}$$

Le critère pour déterminer la nature de la primitive peut alors être défini par :

$$\max_R \xi \quad \begin{matrix} H_1 \\ > \\ < \\ H_0 \end{matrix} \quad \lambda \tag{2.35}$$

où λ est un seuil prédéterminé.

Le paramètre optimal $\widehat{R}$ doit satisfaire la relation $\frac{\partial \xi}{\partial R} = 0$, ce qui amène à $\widehat{R} = \frac{1}{N}\sum_{i=1}^{N} R_i$. En utilisant cette relation, ξ peut finalement s'exprimer de la manière suivante :

$$\xi = \frac{NR^2}{2\sigma^2} \tag{2.36}$$

L'hypothèse H_1 (cylindre) est donc retenue contre l'hypothèse H_0 (segment) si la valeur obtenue du rapport de vraisemblance ξ est plus grande que λ. En fait, quand la primitive est un segment, le processus de reconstruction d'un cylindre donne une valeur très faible du rayon, avec un écart type assez élevé (ce qui implique une valeur petite de ξ). *A contrario*, quand la primitive est réellement un cylindre, la valeur estimée du rayon est très proche de sa valeur réelle avec une faible variance (ce qui implique une valeur importante

de ξ). Cette méthode s'est avérée être plus robuste en pratique que le critère proposé dans [Vaillant 90] (un segment étant là aussi supposé avoir un "rayon nul", cette méthode est basée sur le calcul de la probabilité pour la valeur zéro d'être dans l'intervalle de confiance $R - 2\sigma < 0 < R$).

Si la primitive est reconnue comme étant un cylindre, une estimation plus précise de ses paramètres, basée sur les deux limbes du cylindre est ensuite réalisée. Avec l'estimation obtenue par la méthode basée sur un seul limbe, il est en effet possible de déterminer la position dans l'image du segment correspondant au deuxième limbe du cylindre.

2.2.7 Résultats

Nous présentons dans ce paragraphe des résultats expérimentaux concernant les divers points abordés dans ce chapitre : reconstruction d'un cylindre, reconstruction d'une droite, reconnaissance d'une primitive et calcul de la longueur d'une primitive. Enfin, nous avons effectué des tests de robustesse et de stabilité afin de valider définitivement l'approche retenue. En préalable à ce processus de reconstruction, une étape de calibration a été effectuée. Cette calibration est réalisée en utilisant la méthode proposée dans [Chaumette 89] et décrite dans l'Annexe C.

Le système expérimental avec lequel nous avons validé les différentes étapes de ce travail est composé d'un robot cartésien à six degrés de liberté (pour plus de détails, voir l'Annexe A). Le traitement d'images est effectué sur une carte spécialisée EDIXIA IA 1000 et consiste à suivre un segment dans la séquence d'images, deux dans le cas d'un cylindre en utilisant la projection de ses deux limbes (voir Annexe B. Les observateurs dynamiques décrits dans le paragraphe 2.2.4 ont également été implantés sur cette carte. Le système de traitement d'image permet de suivre un maximum de trois droites dans la séquence. La fréquence d'acquisition, de traitement des images et d'estimation 3D est de 12.5 Hz, ce qui signifie que chaque itération du processus d'estimation 3D est réalisée en 80 millisecondes. Une itération k consiste à :

- acquérir une image ;
- extraire de l'image les paramètres $\underline{p}_k$ définissant la configuration de la primitive, et en déduire leur vitesse $\dot{\underline{p}}_k$;
- mesurer la vitesse de la caméra T_k ;
- estimer les paramètres $\underline{P}_k$ décrivant la configuration spatiale de la primitive 3D considérée ;
- enfin, calculer la commande T_{k+1} de la caméra.

La vitesse $\dot{\underline{p}}_k$ de la primitive dans l'image est calculée en utilisant l'expression suivante :

$$\dot{\underline{p}}_k \approx \frac{\underline{p}_k - \underline{p}_{(k-n)}}{n\Delta t}$$

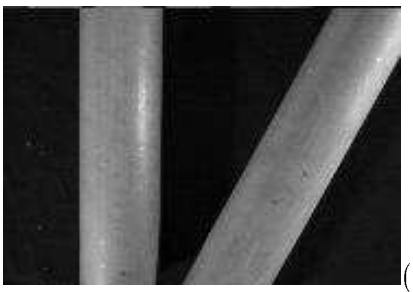
où Δt est l'intervalle de temps entre deux itérations (dans notre cas 80 ms) et où $n \geq 1$ est un paramètre définissant la taille du pas de discrétisation. Notons que si $n = 1$ (*i.e.*, si on

utilise deux image successives), les résultats sont relativement instables (voir [Boukir 93a], [Chaumette 96]). Par contre, avec un n plus grand (typiquement $n = 10$ ou $n = 20$), les résultats sont plus stables. Cependant un biais dû à la discrétisation apparaît si l'on ne contraint pas les mouvements de la caméra comme décrit précédemment. Si l'on utilise la vision active, l'estimation devient non biaisée, stable et robuste. Pour une comparaison effective entre la reconstruction par vision active et la reconstruction par vision dynamique (sans contrôle du mouvement de la caméra), on peut se rapporter aux résultats présentés dans [Boukir 93a] et [Chaumette 96].

2.2.7.1 Reconstruction d'un cylindre

Nous présentons tout d'abord des résultats portant sur la reconstruction de deux cylindres (voir Figure 2.11.a) afin de montrer la précision de cette méthode de reconstruction par vision active. La reconstruction de ces deux cylindres se fait de manière indépendante. La méthode utilisée pour l'enchaînement des reconstructions est décrite dans les chapitres suivants.

De manière à obtenir une estimation robuste et non biaisée de ses paramètres, un cylindre doit apparaître centré et vertical ou horizontal dans l'image (cf figure 2.11.b) pendant le mouvement de la caméra qui consiste en une rotation à distance constante de l'axe du cylindre. Les techniques d'asservissement visuel ont donc été employées pour réaliser cette tâche en temps réel. La méthode de reconstruction utilisée est basée sur l'utilisation des deux limbes du cylindre (cette méthode est décrite à la section 2.2.2). Un "peigne" de 15 itérations ($n = 15$) a été utilisé assurant une bonne stabilité dans l'estimation des paramètres.


(a)

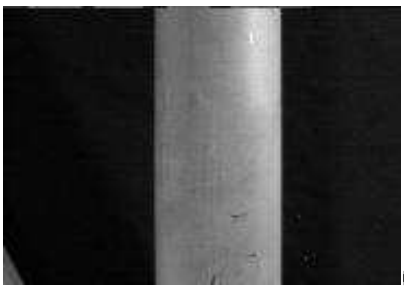

(b)

FIG. 2.11 – *Reconstruction de cylindres par vision active : Image acquise (a) au début du processus de reconstruction et (b) à la fin du processus de focalisation (sur le cylindre de droite)*

La Figure 2.12.a présente l'évolution des paramètres estimés d'un des deux cylindres en fonction du nombre d'itérations. On peut noter la stabilité des résultats obtenus. Un filtre moyennant les résultats des trois dernières estimations a été utilisé. Les dix premières estimations n'apparaissent pas dans ces courbes car les premières valeurs sont souvent bruitées, la commande n'étant pas encore stabilisée. La Figure 2.12.b montre l'erreur entre la valeur estimée du rayon et sa valeur réelle (*i.e.*, $R_i - R^*$ où $R^* = 40$ mm). On constate que cette erreur ne dépasse pas le 1/2 mm et se situe généralement autour du

1/10 de mm.

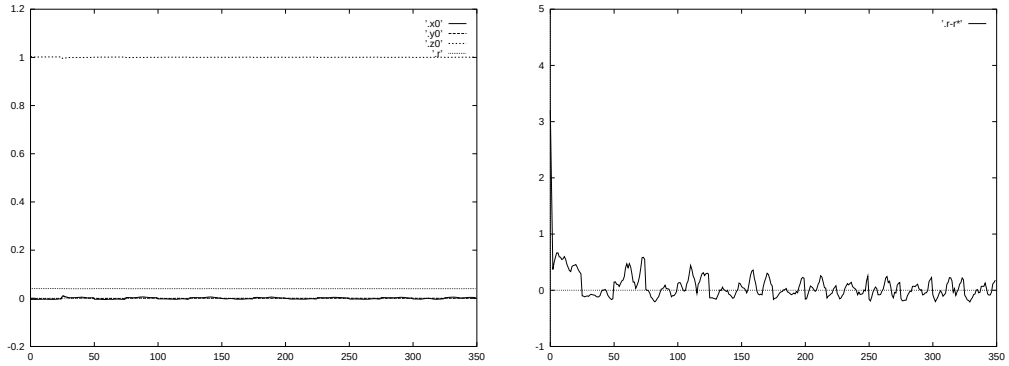


FIG. 2.12 – *Reconstruction d'un cylindre. Evolution des paramètres estimés du cylindre dans le repère de la caméra en fonction du nombre d'itérations (a) paramètres X_0, Y_0, Z_0 et R du cylindre (en mètre), (b) Estimation de l'erreur sur le rayon $R_i - R^*$ (en mm)*

Cylindre	Diamètre mesuré	Rayon mesuré	Rayon estimé
1	80.8	40.4	40.3
2	80.4	40.2	40.1

TAB. 2.1 – *Reconstruction de deux cylindres: rayon mesuré et rayon estimé par l'approche vision active des deux cylindres (ces mesures sont en mm)*

Le tableau 2.1 donne la valeur du rayon de chacun des deux cylindres mesuré avec une technique classique (pieds à coulisse), et sa valeur estimée par vision active. Une erreur d'environ seulement un dixième de millimètre est à noter (qui correspond à l'ordre de grandeur de l'écart-type observé sur cette même estimation, cf tableau 2.2). Les résultats obtenus sont donc extrêmement précis.

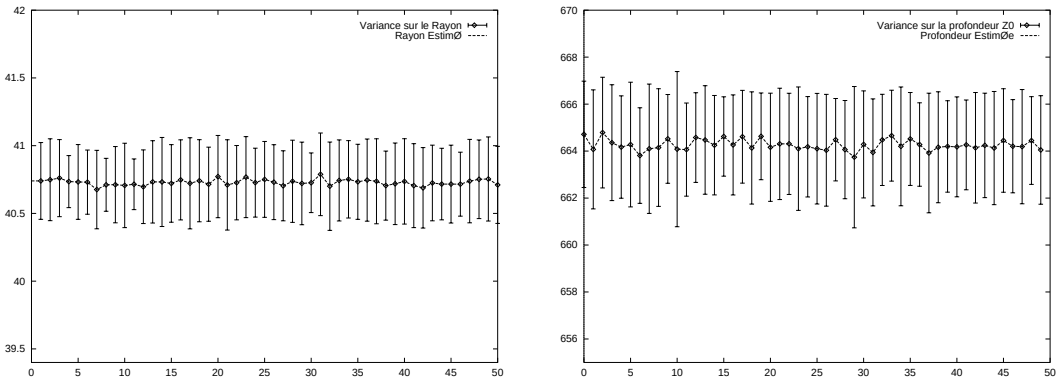
S'il est facile de juger de la qualité et de la précision de la méthode sur le rayon du cylindre, il est beaucoup plus difficile de juger de la qualité de l'estimation des autres paramètres. Notons cependant que les deux cylindres sont situés dans un plan parallèle au plan Oxy du repère dans lequel sont exprimées les positions des deux cylindres. Les paramètres Z_0 des deux cylindres devraient donc être identiques. Sur les résultats donnés dans le tableau 2.2, une erreur d'environ 1 mm est observée, soit une erreur relative de 1‰. Cette différence correspond là-encore à l'ordre de grandeur de l'écart-type sur Z_0. Il faut cependant reconnaître que si cette donnée tend à montrer que les positions relatives des deux cylindres sont correctement estimées, elle n'apporte cependant pas de vérification concernant la position absolue des cylindres dans l'espace. D'autres expériences consistant à observer la primitive depuis diverses positions et à rétro-projeter le modèle 3D du cylindre dans l'image acquise montrent que l'estimation des paramètres de position et d'orientation est aussi très précise. Le résultat de ces dernières expériences est présenté dans le chapitre suivant.

Des tests de stabilité ont également été réalisés. Les paramètres du cylindre ont été

	cylindre 1		cylindre 2	
Paramètre	Moyenne	Écart type	Moyenne	Écart type
U	0.762	0.0002	0.346	0.0004
V	0.752	0.0002	0.938	0.0001
W	-0.002	0.0013	-0.009	0.0016
X_0	136.5	3.4	-45.5	2.5
Y_0	-131.4	1.1	41.8	1.5
Z_0	1358.2	0.5	1359.3	0.5
R	40.3	0.2	40.1	0.1

TAB. 2.2 – *Reconstruction des deux cylindres: statistique sur les paramètres 3D estimés des deux cylindres (U, V et W définissent l'orientation d'un cylindre, X_0, Y_0 et Z_0 sa position et R son rayon, en mm)*

estimés 50 fois en partant de positions initiales différentes. Les résultats reportés sur la Figure 2.13.a montrent pour chacune des 50 estimations le rayon $\widehat{r}$ estimé ainsi que l'écart-type $\sigma_{\widehat{r}}$ sur cette estimation. Pour chaque estimation, l'erreur entre la valeur estimée et la valeur réelle est inférieure à 0.1 mm et l'écart-type sur l'ensemble des moyennes est inférieur à 0.02 mm. Les mêmes tests ont été réalisés sur la profondeur $\widehat{Z}_0$. L'écart type sur l'ensemble des moyennes $\sigma_{\widehat{Z}_0}$ estimées est de l'ordre de 0.25 mm. Ceci démontre que l'algorithme de reconstruction est robuste, stable et précis.



(a) Rayon (b) Profondeur

FIG. 2.13 – *Test de stabilité sur: (a) le rayon R, (b) le paramètre Z_0 du cylindre*

2.2.7.2 Reconstruction de segments

Considérons maintenant la reconstruction de deux segments. Dans le cas présenté (voir Figure 2.14), ces deux segments appartiennent à une feuille de papier standard (de format 210x297 mm). Comme précédemment, pour obtenir une estimation robuste et non biaisée,

la droite à reconstruire doit apparaître verticale ou horizontale centrée dans l'image (voir Figure 2.14.b). Un "peigne" de 15 itérations ($n = 15$) a, ici aussi, été utilisé.

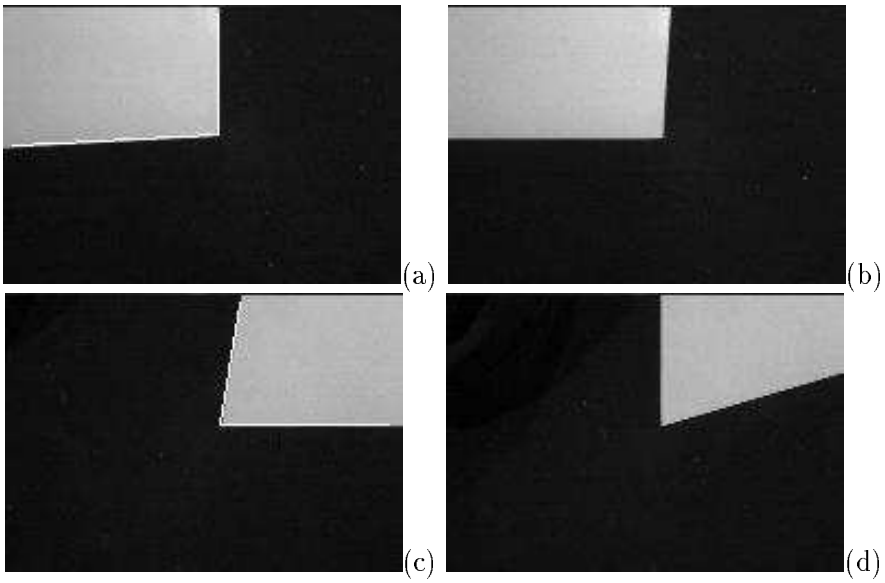


FIG. 2.14 – *Reconstruction de droites par vision active : Image acquise (a) au début du processus de reconstruction de la première droite, (b) à la fin du processus de focalisation (c) et (d) : situation indentique pour la deuxième droite*

Le tableau 2.3 montre les statistiques obtenues en ce qui concerne ces deux droites (les valeurs sont exprimées dans le repère initial de la caméra et le paramètre D_2 est exprimé en mm). La Figure 2.15 présente l'évolution des paramètres des deux droites au cours du temps (première droite de l'itération 1 à 33 et seconde droite de l'itération 34 à 62). Là encore, l'utilisation de la vision active permet une grande stabilité dans l'estimation des paramètres des deux plans caractérisant la droite. On peut noter que l'écart type sur la profondeur de la droite (paramètre D_2) est de l'ordre du mm (soit environ une erreur de 1‰).

Comme nous l'avons dit, ces deux droites appartiennent à une feuille A4 et sont donc perpendiculaires. Le produit scalaire des deux vecteurs directeurs doit donc être égal à 0. La valeur trouvée est de 0.0005 (soit 89.97 dg), ce qui est très proche de la valeur attendue. D'autre part, si on calcule la distance maximale d'un des 4 points au plan contenant les 3 autres, celle-ci est de 0.8 mm, ce qui montre bien que les segments estimés sont quasiment coplanaires.

Comme décrit dans le paragraphe 2.2.5, pour obtenir la position des extrémités d'un segment 3D, il convient de les observer au centre de l'image (voir les Figures 2.14.b et 2.14.c pour la première droite) et de calculer ensuite l'intersection entre la droite 3D préalablement estimée et l'axe optique de la caméra. Les résultats obtenus sont donnés dans le tableau 2.4. On peut noter la faible erreur de reconstruction (de l'ordre du mm) sur le sommet commun des deux segments. Par ailleurs, leurs longueurs réelle et estimée sont reportées dans le Tableau 2.5 où l'on peut noter un écart ici encore de l'ordre du mm.

	droite 1		droite 2	
Paramètre	Moyenne	Écart type	Moyenne	Écart type
A_1	0.207	0.0006	-0.992	0.0002
B_1	-0.978	0.0001	-0.125	0.0017
C_1	-0.029	0.0002	-0.001	0.00064
A_2	-0.277	0.006	0.0901	0.0019
B_2	-0.031	0.001	-0.713	0.816
C_2	-0.960	0.002	-0.695	0.002
D_2	529.74	0.95	373.56	0.87

TAB. 2.3 – *Reconstruction d'une droite : statistique sur les paramètres 3D estimés des deux droites (A_1, B_1 et C_1 définissent le plan passant par le centre optique, A_2, B_2, C_2 et D_2 définissent le second plan caractérisant la droite)*

segment	X	Y	Z
1	165.91	-417.99	889.01
	313.80	-159.52	891.03
2	312.40	-159.43	891.82
	130.62	-55.25	888.72

TAB. 2.4 – *Position 3D des points extrémités de chacun des deux segments (en mm)*

droite	longueur réelle	longueur estimée
1	297	297.8
2	210	209.5

TAB. 2.5 – *Calcul de la longueur des droites : longueur réelle et estimée (en mm)*

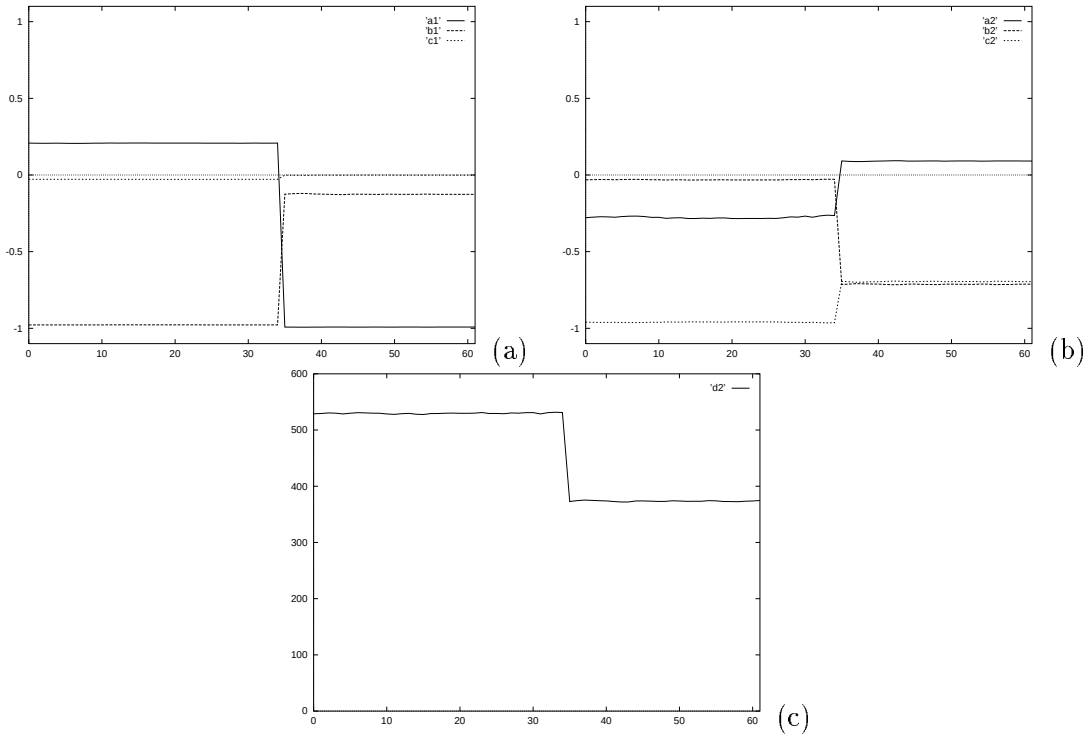


FIG. 2.15 – *Reconstruction de deux droites (de l'itération 1 à 33 pour la première et 34 à 62 pour la seconde). Evolution des paramètres estimés des droites en fonction du nombre d'itérations (a) paramètres A_1, B_1 et C_1 (b) paramètres A_2, B_2 et C_2 (c) paramètre D_2 (en mm)*

Cette erreur provient des imprécisions de l'estimation des paramètres géométriques de la caméra, issue du processus de calibrage, mais aussi du traitement d'image. La précision de l'estimation de la longueur est en effet fortement dépendante des paramètres intrinsèques de la caméra ainsi que de la distance et de l'orientation entre le segment et la caméra. Par ailleurs, la précision du traitement d'images pour la mesure d'un point extrémité d'un segment est seulement d'un pixel. Or, une erreur d'un pixel entraîne une erreur d'environ 0.4 mm (dans cette expérience, la distance focale est de 12.5 mm et la distance entre la caméra et le point d'environ 400 mm). Cette erreur pouvant se retrouver sur l'estimation de la position des deux points extrémités, une erreur finale de 0.8 mm est envisageable, ce qui est cohérent avec les erreurs obtenues.

2.2.7.3 Reconnaissance de primitives

Nous considérons maintenant une scène plus complexe comportant des primitives de nature différente: un cylindre, un triangle, et deux rectangles (voir Figure 2.16). L'un d'eux est une longue plinthe de couleur grise (qui ressemble fortement dans l'image à un cylindre) et l'autre est une plaque texturée. L'objectif ici est de montrer l'efficacité du test de reconnaissance de primitives présenté dans le paragraphe 2.2.6. Pour cela nous

nous intéressons uniquement dans cette scène au cylindre, à la plinthe ainsi qu'à un des segments du triangle.

FIG. 2.16 – *Scène "edixia" : Vue initiale de la scène à reconstruire*

Les objets de la scène sont successivement reconstruits en utilisant la méthode de reconstruction décrite au chapitre suivant. La difficulté dans le traitement de cette scène réside ici principalement dans la présence du cylindre et de la plinthe.

Le critère de reconnaissance est évalué après 40 estimations successives des paramètres d'un cylindre basée sur la projection d'un seul limbe. Le tableau 2.6 montre les différentes valeurs servant à calculer le critère ξ. Le seuil λ a été fixé à 400. Comme on le constate, la valeur du critère ξ par rapport à ce seuil est assez discriminante. Même dans le cas de la plinthe, toutes les primitives ont été correctement reconnues.

	moyenne $\bar{r}$	écart-type σ	critère ξ	décision
cylindre	40.9	4.4	1715.8	cylindre
plinthe 1	27.3	15.4	63.1	droite
plinthe 2	14.3	11.1	33.3	droite
segment 1	23.7	20.2	27.4	droite

TAB. 2.6 – *Reconnaissance de primitives*

Les résultats sont conformes à l'interprétation intuitive que nous avions faite du rapport de vraisemblance (équation 2.36). Un cylindre a une valeur estimée de son rayon proche de la valeur réelle avec un faible écart-type. Dans le cas d'un segment (qui a normalement un rayon nul) la valeur estimée est plus faible mais avec un écart-type très important (de l'ordre de grandeur de ce rayon). Cela entraîne une faible valeur de ξ. On peut noter qu'il est normal d'obtenir une mauvaise estimation du "rayon" du segment. En effet, le mouvement généré pendant la reconnaissance est optimal pour la reconstruction d'un cylindre, et non pour celle d'un segment.

2.3 Conclusion

Nous avons présenté dans ce chapitre les principales approches permettant de remonter à la structure 3D d'une scène. Nous nous sommes principalement intéressés aux approches

monoculaires multi-images. La méthode de reconstruction 3D pour laquelle nous avons opté repose sur une technique continue de reconstruction par vision dynamique active [Boukir 93a] [Chaumette 96]. Les travaux de S. Boukir ont donc souligné l'apport incontestable de la vision active à la reconstruction 3D (rejoignant ainsi les points de vue de Aloimonos [Aloimonos 90] sur le problème du *"structure from motion"*).

Si l'approche retenue pour la reconstruction permet une estimation très précise et robuste des paramètres des primitives concernées, elle présente cependant un certain nombre de défauts :

- elle implique une connaissance *a priori* sur la nature de la primitive observée.
- elle suppose, dans le cas des primitives qui nous intéressent, que ces primitives aient une longueur infinie.

L'approche que nous proposons pour reconstruire une primitive sans connaissance préalable sur sa position ou sa nature repose sur le séquencement de processus de perception (représenté sur la Figure 2.17 par un automate parallèle ; les transitions n'apparaissent pas sur cet automate, elles sont données dans le chapitre 5).

Un première étape de reconnaissance est nécessaire afin de déterminer la nature de la primitive considérée. Dans le cas qui nous intéresse, nous avons fait l'hypothèse que la scène est constituée uniquement de cylindres ou de segments 3D. Nous avons donc proposé un test statistique qui permet de déterminer, en parallèle avec l'estimation de ses paramètres, la nature de la primitive considérée. Quand la primitive est reconnue, une estimation optimale de ses paramètres ainsi que le calcul de la position de ses extrémités sont réalisés. Dans le cas d'un cylindre, cette estimation optimale passe par l'utilisation de son second limbe. En parallèle de ce processus d'estimation, nous avons développé une technique permettant d'éviter les occlusions et les butées articulaires du manipulateur.

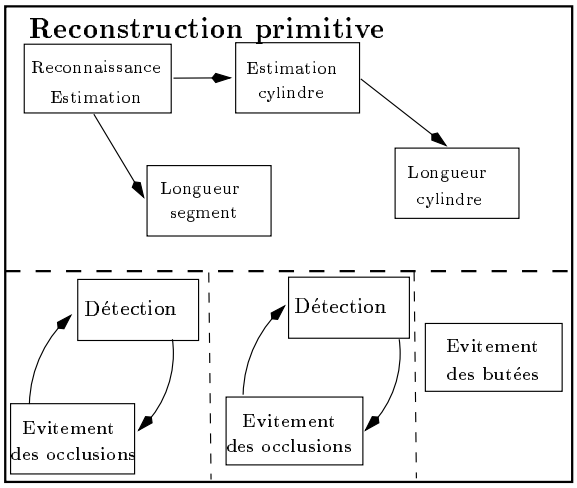


FIG. 2.17 – *Reconstruction d'une primitive : principe général*

Les résultats obtenus par cette approche sont fiables, robustes et précis. Cette précision a cependant un prix : l'approche vision active retenue contraint fortement les mouvements

de la caméra, ce qui implique que l'on ne peut reconstruire qu'une seule primitive à la fois. Or, nous souhaitons obtenir une carte de l'environnement sans hypothèse sur le nombre et la disposition relative des objets de la scène. Il faut donc s'abstraire des contraintes imposées par la méthode de reconstruction en définissant des stratégies de perception qui permettent l'acquisition d'une carte précise et complète de la scène. Les stratégies que nous avons développées sont décrites dans les deux chapitres suivants.

Chapitre 3

Exploration locale de la scène

> – C'est exact dit le géographe, mais je ne suis pas explorateur. Je manque absolument d'explorateurs. Ce n'est pas le géographe qui va faire le compte des villes, des fleuves, des montagnes, des mers, des océans et des déserts. [...] Mais, il reçoit les explorateurs. Il les interroge, et il prend en note leurs souvenirs. Et si les souvenirs de l'un deux lui paraissent intéressants, le géographe fait faire une enquête sur la moralité de l'explorateur.
>
> – Pourquoi ça? fit le petit prince.
>
> – Parce qu'un explorateur qui mentirait entraînerait des catastrophes dans les livres de géographie. [...] On exige de l'explorateur qu'il fournisse des preuves.
>
> Antoine de Saint-Exupéry, *Le Petit Prince*,
> Gallimard, 1943

Si l'approche retenue pour la reconstruction permet une estimation très précise et robuste des paramètres des primitives concernées, elle impose de considérer la reconstruction de plusieurs primitives de façon indépendante. Nous présentons donc des stratégies de perception permettant la reconstruction de scène contenant plusieurs objets.

De manière schématique, l'approche utilisée consiste à sélectionner automatiquement les informations images pertinentes puis à focaliser successivement la caméra sur les différents objets de la scène afin de les reconnaître et ensuite de les reconstruire. Des phases d'exploration sont ensuite nécessaires afin d'assurer une reconstruction aussi complète que possible de la scène. La reconstruction de la scène s'effectuera donc en deux étapes principales :

- La première étape que nous allons décrire dans ce chapitre se propose de reconstruire de manière incrémentale l'ensemble des primitives qui apparaissent dans le champ de vision de la caméra. Nous appellerons cette phase **exploration locale** car elle ne fait appel qu'à des informations disponibles localement.

- Par contre, dans les autres cas, quand toutes les primitives précédemment observées ont été reconstruites, une stratégie différente doit être mise en œuvre de manière à focaliser la caméra sur des zones de la scène n'ayant pas encore été observées. Nous

parlerons alors d'**exploration globale**. La description de cette étape fera l'objet du chapitre suivant.

Un premier algorithme simple mais efficace de reconstruction incrémentale de la scène est proposé dans la section suivante, puis nous présentons dans le paragraphe 3.2 une amélioration de cet algorithme reposant sur des méthodes de génération et vérification d'hypothèses.

3.1 Reconstruction incrémentale de la scène

3.1.1 Extraction et classement des informations visuelles

Comme nous l'avons déjà dit, la scène est supposée composée uniquement d'objets polyédriques et de cylindres. De fait, les contours de ces objets se projettent dans le plan image sous la forme de segments. La première étape du processus de reconstruction consiste donc à extraire ces segments de l'image. Nous appellerons par la suite la liste des segments non reconstruits $\omega_{\phi_t} = \{\mathcal{S}_i, i = 1 \ldots N\}$ où ϕ_t est la position de la caméra depuis laquelle les N segments $\mathcal{S}_i$ sont visibles.

Pour obtenir cet ensemble de segments, nous avons utilisé un extracteur de contours (à savoir un filtre de Shen Castan [Shen 92] puis une approximation polygonale des contours, voir Figure 3.1). Les segments qui ont une longueur supérieure à un seuil donné et qui n'ont pas encore été reconstruits sont ensuite placés dans l'ensemble ω_{ϕ_t}. Les segments de trop petite taille ne sont pas introduits dans la liste de segments car il n'est pas possible de les reconstruire en utilisant la méthode proposée dans le chapitre précédent (la génération par asservissement visuel des mouvements de la caméra autour de ces segments est trop instable en raison de leur taille). Nous verrons cependant dans la section suivante comment prendre ces segments en compte.

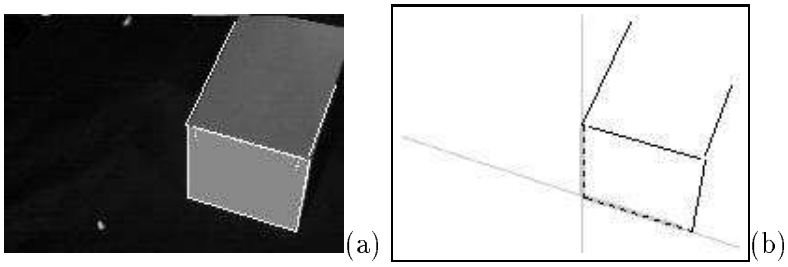


FIG. 3.1 – *Extraction des segments: (a) Image acquise par la caméra (les segments extraits de l'image sont superposés sur celle-ci) (b) Construction de l'ensemble ω_{ϕ_t} (les segments en traits pleins correspondent aux segments non reconstruits et forment donc l'ensemble ω_{ϕ_t}, les segments en traits pointillés correspondent à la projection de segments déjà reconstruits, les segments en traits gris représentant la droite porteuse de ces segments reconstruits).*

L'algorithme de traitement d'image que nous utilisons pour l'extraction et le suivi des segments pendant les phases d'estimation active ne nous permet de suivre qu'un petit nombre de primitives simultanément. Par conséquent, nous ne pouvons pas acquérir ces

ensembles à chaque itération du processus de reconstruction des primitives. Leur création a donc lieu après la fin de chaque reconstruction.

Un autre ensemble que l'on notera $\Omega_{\mathcal{T}_{t_1}^{t_2}}$ est aussi utilisé. Il contient l'ensemble des segments non reconstruits qui ont été précédemment observés ainsi que les positions de la caméra depuis laquelle ils ont été observés. Plus précisément, nous avons :

$$\Omega_{\mathcal{T}_{t_1}^{t_2}} = \{(\mathcal{S}_i, \phi_k), i = 1 \ldots N, k \in [t_1, t_2]\}$$

où

- $\mathcal{T}_{t_1}^{t_2} = \{\phi_{t_1}, \phi_{t_1+1}, \ldots, \phi_{t_2}\}$ représente l'ensemble des positions occupées par la caméra entre les instants t_1 et t_2 [1]. Initialement, $\mathcal{T}_0^0 = \{\phi_0\}$ représente la première position de la caméra.
- $\mathcal{S}_i$ représente un segment 2D non reconstruit et ϕ_k la position depuis laquelle il a été vu. ϕ_k appartient à $\mathcal{T}_{t_1}^{t_2}$
- N représente le nombre de segments présents dans cet ensemble.

En résumé, nous utilisons les deux ensembles de segments suivants :

- $\boldsymbol{\omega_{\phi_t}}$ qui contient les segments non reconstruits, visibles depuis la position courante ϕ_t de la caméra ;
- $\boldsymbol{\Omega_{\mathcal{T}_{t_1}^{t_2}}}$ qui contient les segments non reconstruits observés à un moment ou un autre du processus de reconstruction incrémentale.

3.1.2 Algorithme de reconstruction incrémentale

Ces définitions posées, nous proposons à présent un algorithme permettant de reconstruire incrémentalement l'ensemble des primitives observées par la caméra. Cet algorithme est composé de trois étapes principales : la reconstruction d'une primitive 3D, la mise à jour des informations 2D disponibles, et le choix de la prochaine primitive à reconstruire.

- **Étape 0 : Initialisation.**

  Nous considérons que la caméra est positionnée en ϕ_0. Depuis cette position, la caméra observe l'ensemble de segments ω_{ϕ_0}. À ce stade, la reconstruction de la scène n'a pas encore commencé, nous n'avons donc aucune information sur la nature, la taille ou la position des primitives qui sont présentes dans la scène.

  Initialement, nous avons $\Omega_{\mathcal{T}_0^0} = \omega_{\phi_0} = \{(\mathcal{S}_i, \phi_0), i = 1 \ldots n\}$ où $\mathcal{T}_0^0 = \{\phi_0\}$ et où n est le nombre de segments visibles depuis ϕ_0.

  Un des segments $\mathcal{S}_i$ est ensuite extrait de l'ensemble ω_{ϕ_0} afin d'estimer la primitive 3D qui lui est associée. Comme nous l'avons vu dans le paragraphe 2.2.3.2, pour

1. Par la suite, nous appellerons abusivement $\mathcal{T}_{t_1}^{t_2}$ la "trajectoire" de la caméra entre t_1 et t_2.

minimiser les effets des erreurs de mesure pendant le processus d'estimation, la primitive doit apparaître verticale ou horizontale centrée dans l'image. Afin d'accélérer l'étape de reconstruction, nous choisissons donc le segment qui est le plus proche de l'une de ces deux positions.

- **Étape 1 : Estimation active des paramètre 3D et mise à jour de la carte.**

  Considérons que la caméra est positionnée en ϕ_t ($\phi_t = \phi_0$ si on commence la reconstruction).

  Une reconstruction de la primitive correspondant au segment $\mathcal{S}_i$ sélectionné est réalisée. Cette phase de reconstruction inclut la phase de reconnaissance (segment ou cylindre,voir section 2.2.6), le processus d'estimation active permettant d'obtenir une estimation des paramètres $\underline{\widehat{p}}$ (décrit en 2.2), et l'estimation de la longueur de la primitive (voir paragraphe 2.2.5). Les caractéristiques de la primitive sont ensuite introduites dans la carte de la scène notée $\mathcal{O}(\mathcal{T}_0^t)$.

  Il convient aussi de retirer de la liste des segments $\Omega_{\mathcal{T}_0^t}$ l'ensemble des segments 2D correspondant à cette primitive. En effet, cette primitive a pu être observée depuis d'autres positions précédemment occupées par la caméra. Cette mise à jour sera décrite dans le paragraphe 3.1.3.

- **Étape 2 : Création des listes locales et globales de segments 2D.**

  Après la reconstruction de la primitive, la position de la caméra a été modifiée. Elle est maintenant en ϕ_{t+1}. Une nouvelle liste de segments $\omega_{\phi_{t+1}}$, correspondant à cette position, est créée. Cette liste est ensuite fusionnée avec l'ensemble $\Omega_{\mathcal{T}_0^t}$:

  $$\Omega_{\mathcal{T}_0^{t+1}} = \Omega_{\mathcal{T}_0^t} \cup \omega_{\phi_{t+1}}$$

  Par construction, $\Omega_{\mathcal{T}_0^{t+1}}$ contient tous les segments associés à des primitives non reconstruites.

- **Étape 3 : Focalisation sur le segment suivant.** Trois cas de figure peuvent alors se présenter :

  1. Il y au moins un segment dans l'ensemble $\omega_{\phi_{t+1}}$ qui correspond à une primitive non reconstruite (*i.e.* cet ensemble n'est pas vide, voir Figure 3.2). Un segment $\mathcal{S}_i$ est alors extrait de cet ensemble. Dans le cas où $\omega_{\phi_{t+1}}$ contient plusieurs segments, la sélection est effectuée de la manière suivante : en utilisant tous les segments visibles dans l'image, nous construisons un graphe de voisinage. Dans ce graphe, chaque nœud représente une jonction et chaque arête représente un segment 2D. Ces arêtes sont marquées en fonction des caractéristiques du segment associé (à savoir (T) si l'arête correspond à une primitive déjà reconstruite ou (U) sinon, voir Figure 3.2). En utilisant ce graphe, nous recherchons un segment non traité (U) connexe au dernier segment ayant servi de base à une reconstruction (nous réalisons un tel choix afin de limiter le temps de focalisation et de reconstruction). Si un tel segment existe, il est sélectionné

(s'il existe plusieurs segments de ce type, celui dont l'angle avec le segment que l'on vient de reconstruire est le plus faible est choisi ; par exemple dans le cas de la Figure 3.2, le segment $\mathcal{S}_2$ est sélectionné). Dans le cas contraire, on sélectionne le segment le plus proche d'une position adéquate pour l'estimation active (exactement comme dans le cas de l'étape d'initialisation).

Le processus est ensuite itéré par un retour à l'étape 1 (reconstruction du segment sélectionné).

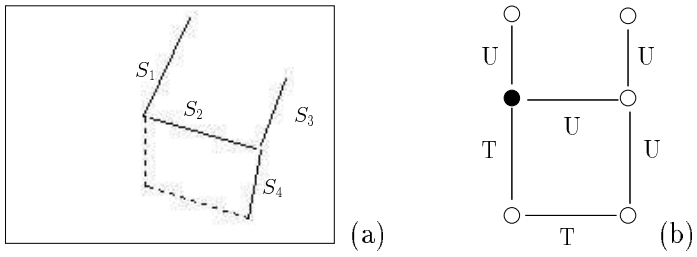


FIG. 3.2 – *(a) Ensemble des segments perçus dans l'image, les traits pleins correspondent à l'ensemble ω_{ϕ_t} et (b) graphe de voisinage associé*

2. Le deuxième cas de figure concerne le cas où tous les segments $\omega_{\phi_{t+1}}$ correspondent à des primitives déjà reconstruites (voir Figure 3.3). Dans ce cas, il se peut que des segments précédemment observés (et qui sont donc dans $\Omega_{\mathcal{T}_0^{t+1}}$) n'aient pas encore été traités.

   Nous exécutons alors une phase de retour arrière (*backtracking*). On recherche dans l'ensemble $\Omega_{\mathcal{T}_0^{t+1}}$ le couple $(\mathcal{S}_i, \phi_k)$ pour lequel la distance entre la position courante de la caméra ϕ_{t+1} et ϕ_k est la plus faible (nous cherchons à réduire les déplacements de la caméra). La caméra se replace à cette position ϕ_k ($\phi_{t+2} = \phi_k$) et une estimation active du segment $\mathcal{S}_i$ est ensuite réalisée (retour à l'étape 1).

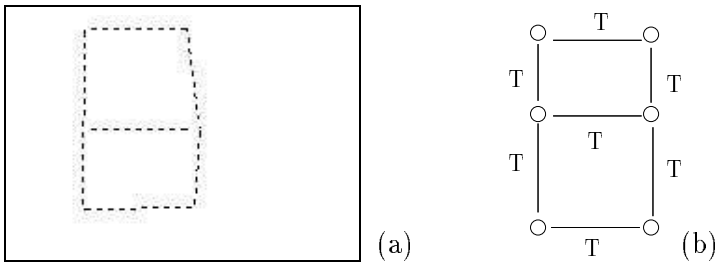


FIG. 3.3 – *(a) Ensemble des segments perçus dans l'image (b) graphe de voisinage associé*

3. Finalement, si l'ensemble $\Omega_{\mathcal{T}_0^{t+1}}$ est lui aussi vide (c'est-à-dire que l'ensemble des segments 2D observés entre la date 0 et la date $t+1$ ont fait l'objet d'une reconstruction), le processus de reconstruction incrémentale de la scène prend fin.

Nous décrivons à présent comment sont mises à jour les listes ω_{ϕ_t} et $\Omega_{\mathcal{T}_0^t}$.

3.1.3 Mise à jour des informations visuelles

L'un des aspects fondamentaux de la construction des deux ensembles de segments est, pour ω_{ϕ_t}, la détection dans l'image des segments correspondant à la projection de nouvelles primitives et, pour $\Omega_{\mathcal{T}_{t_1}^{t_2}}$, la suppression des segments correspondant à la projection des primitives déjà reconstruites. Dans les deux cas, la technique employée est la même :

- **Création de** ω_{ϕ_t} : dans ce cas, il convient de savoir si les segments observés dans l'image correspondent à la projection de primitives 3D déjà reconstruites (T) ou non (U).

  Pour effectuer la mise en correspondance des primitives 3D déjà reconstruites et des segments 2D observés, une première étape consiste à exprimer l'équation de la primitive dans le repère de la caméra correspondant à sa position ϕ_t.

  Pour chaque primitive 3D déjà reconstruite, le problème se limite alors à :

  - calculer, à partir des paramètres $\underline{P}$ de cette primitive exprimé dans le repère caméra, les paramètres $\underline{p}$ de sa projection dans le plan image.
  - trouver dans l'ensemble ω_{ϕ_t} des segments 2D ceux qui correspondent à la projection de cette primitive.

- **Mise à jour de** $\Omega_{\mathcal{T}_{t_1}^{t_2}}$. Le problème est très similaire. Connaissant les paramètres 3D de la dernière primitive reconstruite (étape 1 de l'algorithme de reconstruction incrémentale), ils convient de repérer dans $\Omega_{\mathcal{T}_{t_1}^{t_2}}$ l'ensemble des couples $(\mathcal{S}_i, \phi_i)$ correspondant à cette primitive. Pour cela, il faut exprimer les paramètres $\underline{P}$ de cette primitive dans le repère caméra associé à chaque position ϕ_i, puis, après avoir calculé les paramètres $\underline{p}$ de sa projection dans le plan image, vérifier si chaque segment $\mathcal{S}_i$ correspond à la projection de cette primitive.

Dans les deux cas, la technique est similaire et repose sur :

- le calcul de la projection d'une ou plusieurs primitives 3D dans le plan image ;
- le calcul d'un test d'appariement avec un ou plusieurs segments.

Les équations de tranformations des paramètres d'une primive 3D d'un repère à un autre, ainsi que les transformations des données d'un espace métrique à un espace pixel sont données en Annexe C.

Nous présentons à présent les deux autres étapes du processus.

Projection d'un cylindre et d'une droite dans l'image : Rappelons que l'équation d'un cylindre est donnée par :

$$h(\underline{X}, \underline{P}) = (X - X_0)^2 + (Y - Y_0)^2 + (Z - Z_0)^2 - (UX + VY + WZ)^2 - R^2 = 0, \quad (3.1)$$

$$\text{avec} \begin{cases} U^2 + V^2 + W^2 & = & 1 \\ UX_0 + VY_0 + WZ_0 & = & 0, \end{cases}$$

où U, V, W représente le vecteur directeur de l'axe du cylindre, X_0, Y_0, Z_0 sont les coordonnées d'un point passant par son axe et R son rayon.

Connaissant les paramètres $\underline{P}$ du cylindre exprimés dans le repère de la caméra, il est possible de déterminer l'équation de sa projection dans l'image. On obtient [Chaumette 90]:

$$\begin{array}{rcl} D_1 & : & x\cos\theta_1 + y\sin\theta_1 - \rho_1 = 0 \\ D_2 & : & x\cos\theta_2 + y\sin\theta_2 - \rho_2 = 0 \end{array} \tag{3.2}$$

$$\text{avec} \left\{ \begin{array}{lcllcl} \cos\theta_1 & = & \frac{RX_0/\sqrt{a}-\alpha}{\sqrt{(RX_0/\sqrt{a}-\alpha)^2+(RY_0/\sqrt{a}-\beta)^2}} & \cos\theta_2 & = & \frac{RX_0/\sqrt{a}+\alpha}{\sqrt{(RX_0/\sqrt{a}+\alpha)^2+(RY_0/\sqrt{a}+\beta)^2}} \\ \sin\theta_1 & = & \frac{RY_0/\sqrt{a}-\beta}{\sqrt{(RX_0/\sqrt{a}-\alpha)^2+(RY_0/\sqrt{a}-\beta)^2}} & \sin\theta_2 & = & \frac{RY_0/\sqrt{a}+\beta}{\sqrt{(RX_0/\sqrt{a}+\alpha)^2+(RY_0/\sqrt{a}+\beta)^2}} \\ \rho_1 & = & \frac{RZ_0/\sqrt{a}-\gamma}{\sqrt{(RX_0/\sqrt{a}-\alpha)^2+(RY_0/\sqrt{a}-\beta)^2}} & \rho_2 & = & \frac{RZ_0/\sqrt{a}+\gamma}{\sqrt{(RX_0/\sqrt{a}+\alpha)^2+(RY_0/\sqrt{a}+\beta)^2}} \end{array} \right.$$

$$\text{où} \left\{ \begin{array}{l} \alpha = WY_0 - BZ_0 \\ \beta = UZ_0 - WX_0 \\ \gamma = VX_0 - UY_0 \\ a = X_0^2 + Y_0^2 + Z_0^2 - R^2 \end{array} \right.$$

Dans le cas d'une droite, l'équation est donnée par :

$$h(\underline{X}, \underline{P}) = \left\{ \begin{array}{l} A_1X + B_1Y + C_1Z = 0 \\ A_2X + B_2Y + C_2Z + D_2 = 0 \end{array} \right. \text{ avec } \left\{ \begin{array}{l} A_1^2 + B_1^2 + C_1^2 = 1 \\ A_2^2 + B_2^2 + C_2^2 = 1 \\ A_1A_2 + B_1B_2 + C_1C_2 = 0. \end{array} \right. \tag{3.3}$$

et la droite résultant de sa projection dans l'image a pour équation :

$$D_0 : x\cos\theta_0 + y\sin\theta_0 - \rho_0 = 0 \text{ avec } \left\{ \begin{array}{lcl} \theta_0 & = & \arctan(\frac{A_1}{B_1}) \\ \rho_0 & = & -\frac{C_1}{\sqrt{A_1^2+B_1^2}} \end{array} \right. \tag{3.4}$$

Remarque : on retrouve bien entendu la même équation (à un changement de variables près) en considérant une droite comme un cylindre de rayon nul.

Contraintes d'appariement : En utilisant les équations précédentes, il est possible de mettre en correspondance la projection d'une primitive 3D avec un segment D présent dans un des deux ensembles ω_{ϕ_t} ou $\Omega_{\mathcal{T}_{t_1}^{t_2}}$.

Pour deux segments (ρ, θ) et (ρ', θ'), il est possible de faire un premier appariement si les deux conditions suivantes sont vérifiées :

$$\left\{ \begin{array}{l} \mid \rho - \rho' \mid < \varepsilon \\ \mid \theta - \theta' \mid < \Theta \end{array} \right.$$

où ε et Θ sont deux seuils prédéterminés. Plus la calibration du système caméra-robot est précise, plus le processus d'estimation active est précis et plus ces seuils peuvent être bas. Dans la pratique, nous avons fixé $\varepsilon = 5$ pixels [2] et $\Theta = 1.5$ degré (ces seuils pourraient être dans la plupart des cas abaissés mais il subsiste des imperfections, en particulier dans le processus d'extraction des segments de l'image).

Cependant, cette comparaison ne suffit pas toujours à assurer que l'appariement est correct. Ainsi sur la Figure 3.4, les deux segments $\mathcal{S}_1$ et $\mathcal{S}_2$ répondent au critère proposé mais ne doivent pas être appariés. C'est pourquoi nous testons aussi, par une méthode similaire, la position des points extrémités.

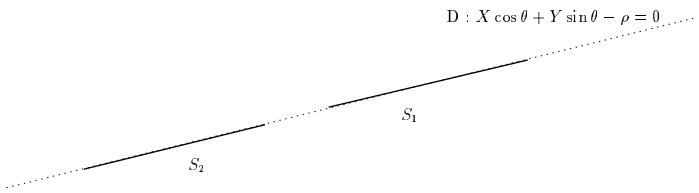


FIG. 3.4 – *Mise en correspondance : cas d'un double appariement possible*

Cette méthode de mise en correspondance des segments 2D et des primitives 3D est très simple. Son efficacité repose essentiellement sur la qualité de l'estimation des paramètres 3D des primitives. D'autres techniques reposant sur l'utilisation de la distance de Mahalanobis ont été proposées dans la littérature et il est certain que, dans le cas de scènes fortement bruitées, de telles techniques trouveraient avantageusement leur place.

Résultats de l'appariement La Figure 3.5 montre le résultat de la mise en correspondance des segments 2D et des primitives 3D sur trois scènes différentes. La première scène (Figure 3.5.a) est composée de deux rectangles situés dans deux plans perpendiculaires, la seconde (Figure 3.5.b) est composée d'un rectangle et d'un cylindre (deux des segments du rectangle n'ont pas encore été reconstruits, ce qui explique qu'aucune mise en correspondance n'ait eu lieu sur ces deux segments. La troisième scène enfin est composée d'un polyèdre plus complexe à 5 faces (là-encore un segment n'a pas encore été estimé). Dans ces images, les segments 2D en trait continus représentent des segments qui ne correspondent pas à une primitive 3D déjà reconstruite, il convient donc soit de les mettre dans l'ensemble ω_{ϕ_t}, soit de les laisser dans l'ensemble $\Omega_{\mathcal{T}_{t_1}^{t_2}}$. Les segments en pointillés correspondent eux aux segments mis en correspondance avec la projection d'une primitive 3D déjà reconstruite, ils ne devront donc pas apparaître dans ω_{ϕ_t} et être retirés de $\Omega_{\mathcal{T}_{t_1}^{t_2}}$. Les lignes grises fines correspondent à la projection soit de la droite porteuse de segments 3D, soit des limbes d'un cylindre, alors que la ligne grise plus épaisse correspond à la projection de la primitive elle-même.

Comme on le constate sur ces trois images, la projection dans l'image des primitives 3D correspond parfaitement aux informations 2D présentes dans l'image. Il en découle que l'opération de mise en correspondance 3D-2D est satisfaisante.

2. La taille des images traitées est de 512 lignes et 730 colonnes.

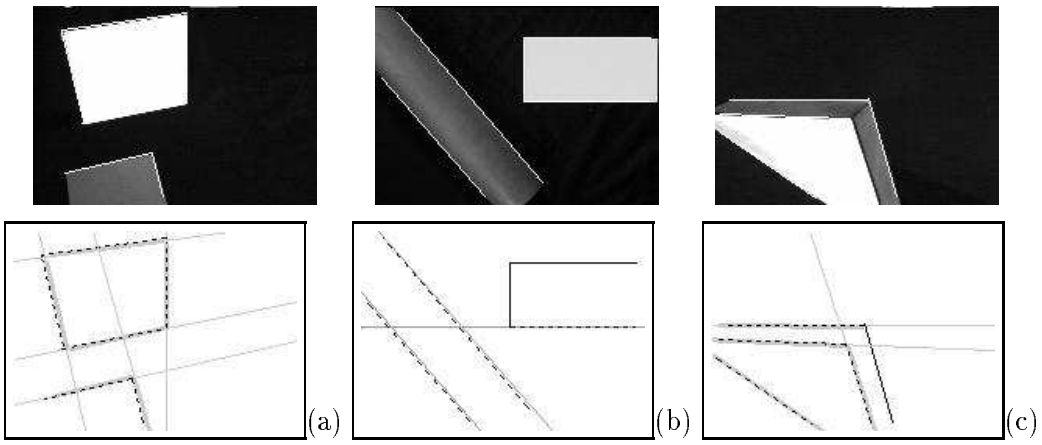


FIG. 3.5 – *Résultat de l'appariement sur trois types de scènes différentes*

3.1.4 Résultat de la reconstruction incrémentale

Nous avons validé la première phase de la reconstruction sur un grand nombre de scènes. Nous présentons ici les résultats de reconstruction de deux scènes différentes. Chacune nous permettra d'illustrer les points forts et les points faibles de cette première méthode.

Scène "Cylindre et polygones" La scène que l'on considère est composée d'un cylindre et de plusieurs polygones disposés dans des plans différents. L'image 3.6 montre une vue extérieure de la scène et des différents objets qui la composent.

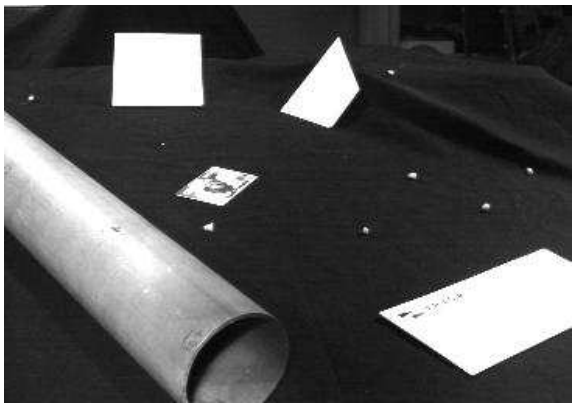


FIG. 3.6 – *Vue extérieure de la scène*

La figure 3.7 représente les images acquises avant chaque reconstruction optimale. À chacune de ces images est associée la liste de segment ω_{ϕ_t} correspondante. Les lignes pleines montrent les éléments de cette liste (correspondant à des primitives qui n'ont pas encore été traités). Les lignes en pointillés représentent les segments correspondant à des primitives 3D déjà reconstruites.

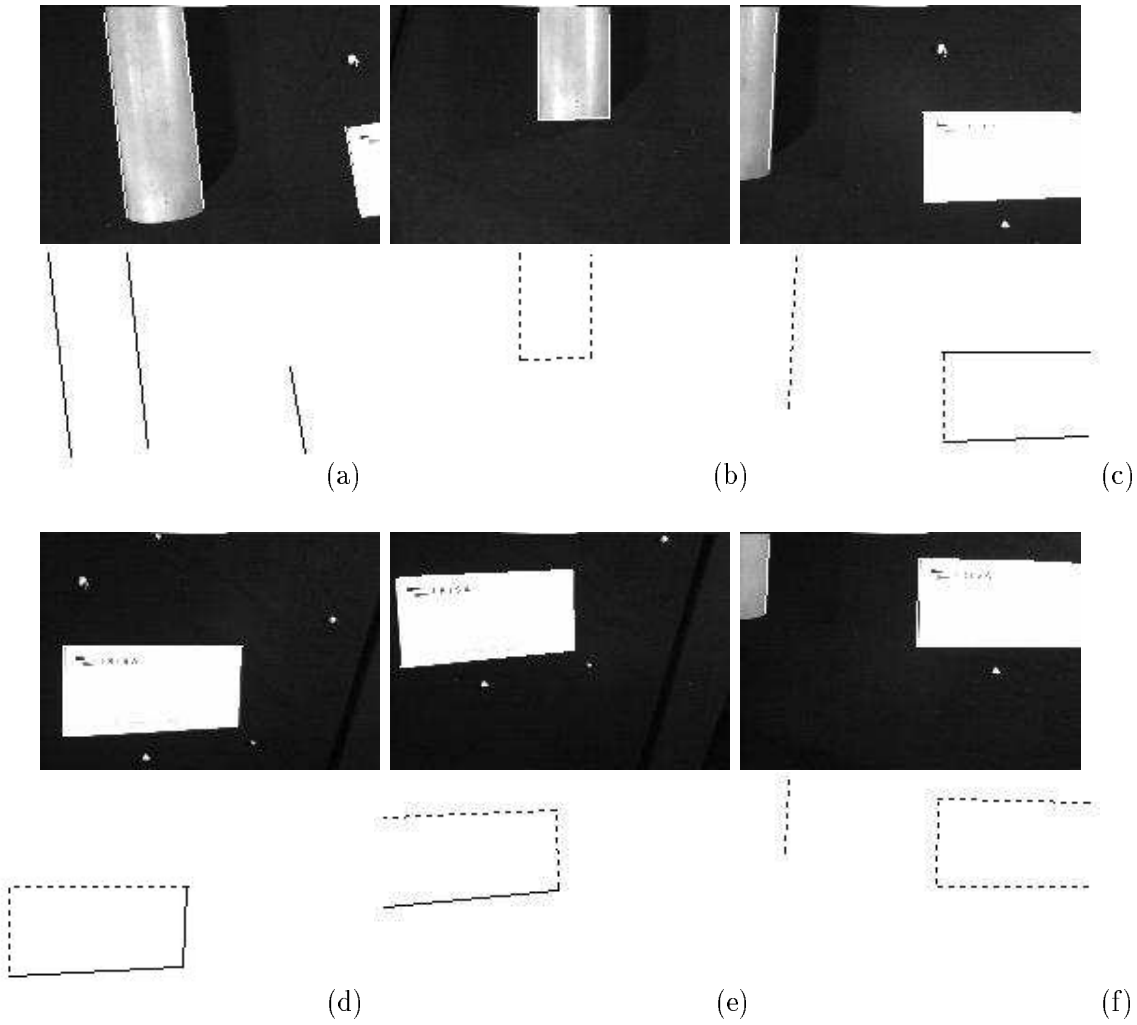


FIG. 3.7 – *Exploration locale de la scène "Cylindre et polygones"*

La figure 3.7.a montre l'image acquise depuis la position ϕ_0 de la caméra. Aucune reconstruction n'a encore été effectuée et trois segments seulement sont visibles depuis cette position. On peut noter que la scène complète n'est pas visible depuis cette position de la caméra. Le segment extrait de la liste de segments ω_{ϕ_0} est celui correspondant au limbe de droite du cylindre. Après la phase de reconnaissance et de reconstruction du cylindre, la caméra est positionnée en ϕ_1 (image 3.7.b). Les segments de ω_{ϕ_1} ont tous été traités. Après consultation de la liste de segments globale Ω_Φ, on constate qu'un segment a été observé depuis la position ϕ_0 et n'a pas encore été reconstruit. La caméra se déplace donc en $\phi_2 = \phi_0$ et se focalise sur le segment retenu. Après l'estimation de cette primitive, la caméra est positionnée en ϕ_3 (image 3.7.c). Deux segments correspondant à des primitives non reconstruites apparaissent dans la base de données ω_{ϕ_3}. Le segment le plus proche du centre de l'image est sélectionné et reconstruit. Ce processus est renouvelé jusqu'à ce que toutes les primitives observées pendant cette phase d'exploration locale aient été estimées (image 3.7.f correspondant à la position ϕ_7 de la caméra). La scène estimée à ce stade du processus de reconstruction est présentée sur la figure 3.8.

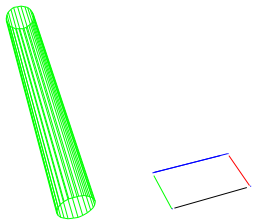


FIG. 3.8 – *Scène "Cylindre et polygones" : scène reconstruite*

Dans cette expérience, il est intéressant de noter que des primitives qui n'apparaissaient pas dans le champ de vision initial de la caméra ont été découvertes et reconstruites. Il n'en demeure pas moins que sur les quatre polygones constituant la scène, seuls les segments constituant l'un d'entre eux ont été reconstruits. Les autres polygones ne sont jamais apparus dans le champ de vision de la caméra et n'ont donc pas pu être traités. Ceci justifie l'introduction dans le chapitre suivant des phases d'exploration globale de la scène visant à amener la caméra à observer ces objets.

Scène "polyèdre" La seconde et dernière scène que nous considérons ici est composée d'un polyèdre formé lui-même à partir de cinq polygones non coplanaires. La Figure 3.9.a montre la première image de cet objet. Si l'algorithme se comporte de la façon attendue en exécutant une reconstruction très précise de l'ensemble des primitives présentes dans l'ensemble Ω (voir Figures 3.9 et 3.10), cette scène nous permet d'illustrer les limitations de cette approche.

Comme on le constate sur les Figures 3.9 et 3.10, un grand nombre de segments de petites tailles n'ont pas été introduits dans les listes de segments et n'ont donc pas été

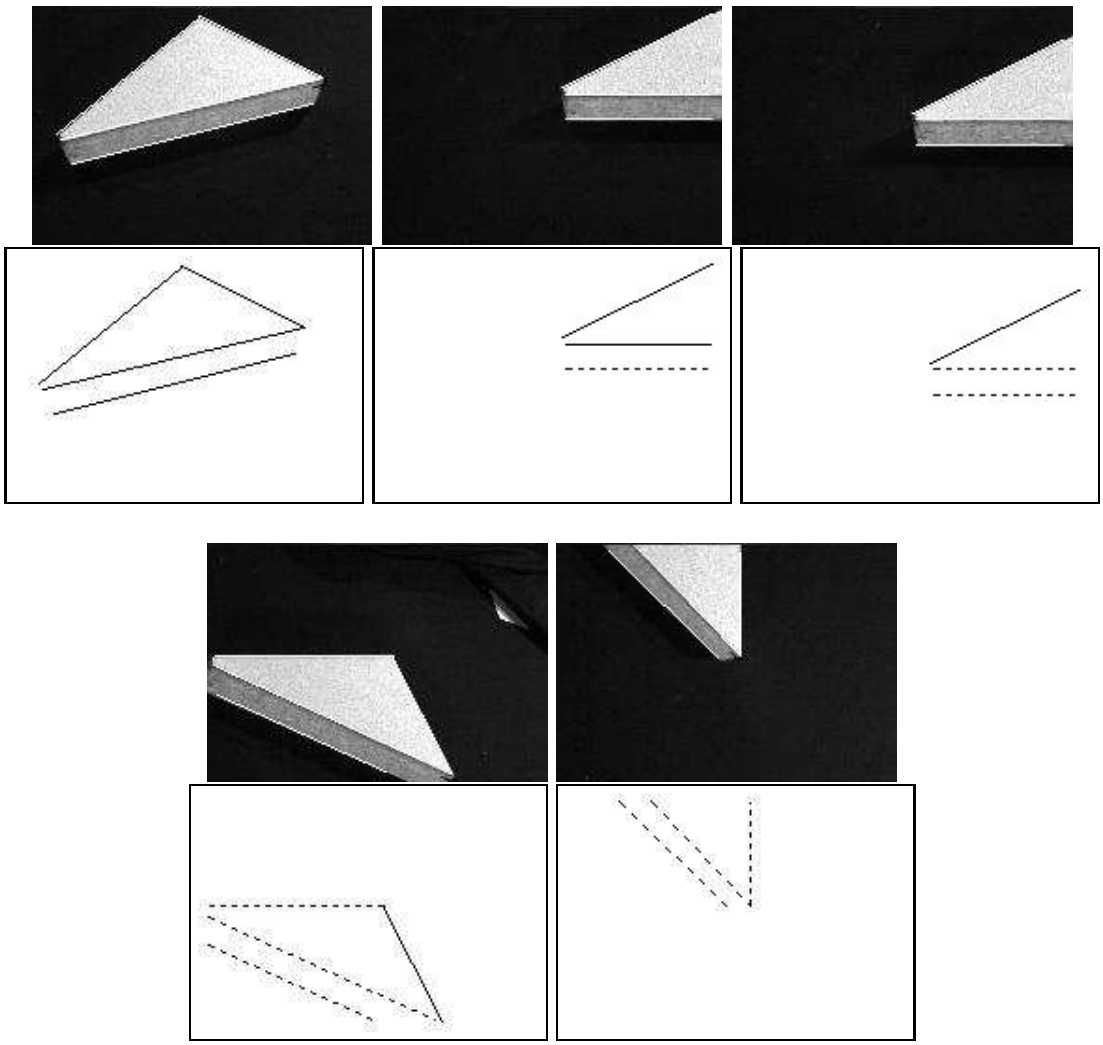


FIG. 3.9 – *Scène "polyèdre"*

reconstruits. D'autre part, il existe des segments composant ce polyèdre qui ont une longueur importante et qui n'ont jamais été observés en raison d'occlusions. Finalement, et comme dans le cas de la scène précédente, nous disposons d'un ensemble de segments 3D, mais pas de polygones, ni d'informations de plus haut niveau.

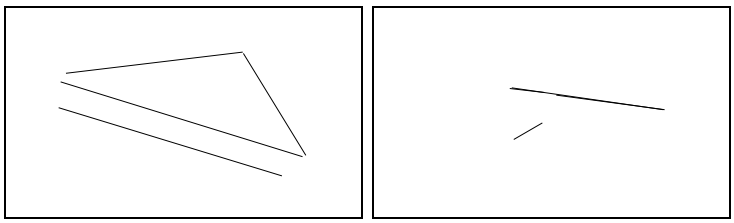


FIG. 3.10 – *Scène "polyèdre" : représentation 3D des primitives reconstruites (a) vue de dessus et (b) vue de coupe (on peut noter que les 3 arêtes du "triangle" sont parfaitement coplanaires)*

3.1.5 Conclusion partielle

L'algorithme que nous venons de présenter permet de reconstruire toutes les primitives qui se sont trouvées dans le champ de vision de la caméra. Cependant, au vu des résultats obtenus, il présente les inconvénients suivants :

- La description de la scène se situe à un niveau primitive alors qu'il serait intéressant de passer au niveau objet (jonctions, polygones, faces, etc.).
- La reconstruction de la scène n'est pas complète pour les deux raisons suivantes :
  - La projection dans l'image de certaines primitives a une taille trop petite pour être retenue dans les listes de segments. Par conséquent, ces primitives n'ont pas été reconstruites.
  - Ne traitant que les segments observés, cet algorithme se contente d'une vision locale de la scène. Par conséquent, certains objets peuvent ne pas apparaître dans le champ de vision de la caméra soit à cause d'occlusion, soit parce qu'ils sont situés dans des zones totalement inconnues. Ces segments non vus n'ont évidemment pas pu faire l'objet d'une reconstruction.

Nous avons donc développé deux méthodes complémentaires afin de remédier à ces inconvénients. Le paragraphe suivant propose une approche reposant sur des techniques de prédiction / vérification d'hypothèses et qui permet d'obtenir une modélisation complète et de haut niveau des zones observées. Le chapitre suivant propose, lui, un algorithme d'exploration globale de la scène assurant une reconstruction complète de celle-ci grâce au calcul explicite de nouveaux points de vue.

3.2 Génération / Vérification d'hypothèses

L'objectif ici est de montrer comment, à partir d'une modélisation 3D incomplète de la scène, il est possible de remonter à une description plus complète et de plus haut niveau de cette même scène. Afin d'éclairer cet objectif, nous présentons dès maintenant un exemple de résultat obtenu dans le cadre de la reconstruction de la scène "polyèdre". Le processus de reconstruction incrémentale permet d'obtenir le modèle de la scène présenté sur la Figure 3.11.a. La représentation de la scène que nous souhaitons obtenir doit entre autres tenir compte des petits segments qui n'avaient pas été reconstruits. De plus, elle doit permettre, dans la mesure du possible, d'inférer la présence de nouveaux segments (voir Figure 3.11.b). Enfin, nous souhaitons pouvoir passer d'une représentation en termes de segments 3D à une représentation en terme d'objets : segments, polygones, faces.

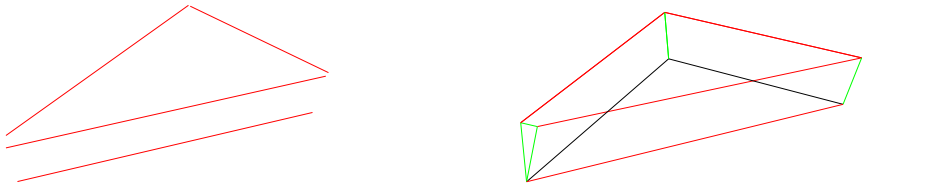


FIG. 3.11 – *(a) modèle de la scène "polyèdre" acquis en se servant uniquement du module de reconstruction incrémentale et (b) modèle reconstruit en se servant du module de prédiction / vérification d'hypothèses*

La méthode que nous avons développée pour parvenir à cet objectif vient se greffer sur l'algorithme de reconstruction incrémentale et repose sur des techniques de prédiction / vérification d'hypothèses. Ce type d'approche vise généralement à résoudre le problème de l'adéquation des modèles aux données. Elle est principalement utilisée pour la reconnaissance d'objets 2D [Ayache 83] [Ayache 86] et 3D [Bolles 86], mais aussi pour la reconstruction incrémentale de scènes 3D à partir d'un ensemble d'images [Herman 86]. Ces techniques consistent à émettre des hypothèses sur la présence et la position spatiale des primitives dans la scène et à vérifier ensuite que des appariements sont compatibles avec ces hypothèses.

Dans notre cas, l'objectif n'est pas de reconnaître l'objet considéré mais d'obtenir une modélisation complète d'un objet à partir de données tridimensionnelles incomplètes et d'informations visuelles disponibles ou à acquérir. La manipulation de l'incertitude inhérente à la manipulation de données réelles ainsi que les stratégies de perception nécessaires à l'accomplissement de cette tâche sont gérées à l'aide de réseaux Bayésiens. À la fin de ce chapitre, nous présenterons des résultats acquis sur différentes scènes. En particulier, nous reviendrons sur certaines des scènes présentées dans la section précédente et nous verrons comment il a été possible de résoudre certains des problèmes qui avaient été alors soulevés. La méthode que nous allons présenter ici porte exclusivement sur des primitives 3D de type segment. Cependant, nous pensons qu'elle est adaptable à d'autres types de primitives comme par exemple des réseaux complexes de tuyaux du type de ceux que l'on

trouve dans des centrales nucléaires.

3.2.1 Principe général

3.2.1.1 Décision en présence d'incertitude : les réseaux Bayesiens

Dans des applications comme la nôtre, l'incertitude se retrouve à plusieurs niveaux : au niveau de l'extraction des informations présentes dans les images et au niveau de la qualité de la reconstruction 3D des primitives. Cette incertitude a généralement pour conséquence principale la confrontation de plusieurs alternatives possibles quant aux décisions à prendre pour guider l'exploration et la reconstruction de la scène.

La théorie de la décision a pour objectif l'étude des techniques mathématiques utilisées pour prendre une décision en présence de plusieurs alternatives. Différents cadres mathématiques sont envisageables pour traiter ce problème. Citons par exemple la théorie de Dempster Shafer [Shafer 76] (utilisée en vision par Hutchinson et Kak [Hutchinson 89] pour la reconnaissance d'objets), les chaînes de Markov cachées (utilisées par Rimey et Brown [Rimey 91] pour simuler le mouvement des yeux) ou l'utilisation de réseaux Bayesiens [Pearl 88]. Nous avons choisi d'utiliser le cadre Bayesien car il nous paraît bien adapté à notre problème. En effet, l'utilisation des réseaux Bayesiens permet de modéliser à travers la structure du réseau les connaissances d'un "expert". De plus, ils se prêtent bien à l'exécution d'actions au cours des raisonnements, ce qui permet d'introduire les stratégies de perception au sein même du processus d'interprétation de la scène.

Réseaux Bayesiens : définition. Les réseaux Bayesien permettent de représenter les distributions de probabilités jointes d'un ensemble de variables en utilisant un ensemble de connaissances *a priori* sur les relations entre ces variables. Un réseau Bayesien est un graphe orienté acyclique où les nœuds représentent une proposition (ou une variable) associée, dans le cas des variables aléatoires discrètes, à un ensemble discret de valeurs. Les liens entre les nœuds (arcs orientés) indiquent les relations de dépendance (causalité) entre variables. Une quantification de cette dépendance est donnée par des tables définissant la probabilité conditionnelle qu'un nœud ait telle valeur sachant que son prédécesseur a telle autre valeur.

Un tel réseau peut être utilisé pour représenter une connaissance disponible sur un domaine particulier. La structure du graphe ainsi que les connaissances *a priori* qu'il contient (sous la forme de table de probabilité conditionnelle) doivent être introduites par le concepteur de l'application. Les étapes nécessaires à la création d'un réseau Bayesien sont les suivantes :

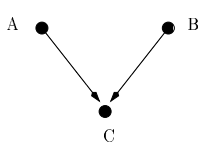
- Déterminer les relations de causalité entre les différentes variables mises en jeu et représenter ces relations par un graphe orienté acyclique. Chaque nœud du graphe représente une des variables.
- Déterminer l'ensemble des états possibles que peut adopter chaque variable.
- Enfin, déterminer les probabilités conditionnelles qu'une variable soit dans un état sachant que les variables associées aux parents du nœud sont dans tel état.

Propagation. La propagation dans un réseau Bayesien consiste à propager sur toutes les variables du réseau l'influence des variables mises à jour. La description de tout le processus de propagation n'est pas l'objectif de ce paragraphe. Nous en donnerons uniquement le principe élémentaire (propagation d'un nœud d'un réseau à un autre). Le célèbre théorème de Bayes [Bayes 63] (équation (3.5) ci-dessous) est à la base de ce mécanisme de propagation.

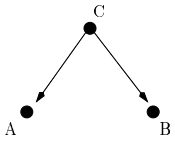
$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{3.5}$$

Il permet de mettre à jour la croyance en une hypothèse connaissant une certain nombre d'observations. $P(A)$ représente la probabilité *a priori* de l'hypothèse A, $P(A|B)$ la probabilité *a posteriori* de cette même hypothèse A sachant les observations B. Enfin $P(B|A)$ représente la croyance que l'on a dans les observations en présence de l'hypothèse A. Le dénominateur $P(B)$ n'est qu'une constante de normalisation, elle peut être calculée en utilisant la contrainte $P(A|B) + P(\neg A|B) = 1$ (où $\neg A$ est le complémentaire de A).

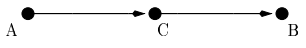
La propagation de l'influence d'une variable aux variables du réseau constitue l'un des principaux inconvénients à l'utilisation des réseaux Bayesiens. En effet, résoudre ce problème revient à résoudre un problème NP-complet [Pearl 86], [Pearl 88]. Des algorithmes permettant une propagation optimale en se basant sur des techniques de propagations locales, et reposant sur les relations d'indépendance entre les différentes variables du réseau, ont cependant été développées [Lauritzen 90]. Une description simple de ces mécanismes est donnée dans [Krause 93] et une description détaillée dans [Neapolitan 90]. La propagation dans un réseau passe par le calcul de la distribution de probabilités jointes associée aux différentes variables du réseau. Ainsi, si l'on considère les trois réseaux élémentaires de la Figure 3.12, les distributions de probabilités jointes associées à ces réseaux sont définies par $P(A, B, C)$ avec respectivement :



$P(A,B,C) = P(C \mid A,B)P(A \mid B)P(B)$
$= P(C \mid A,B)P(A)P(B)$



$P(A,B,C) = P(A \mid B,C)P(B \mid C)P(C)$
$= P(A \mid C)P(B \mid C)P(C)$



$P(A,B,C) = P(B \mid A,C)P(C \mid A)P(A)$
$= P(B \mid C)P(C \mid A)P(A)$

FIG. 3.12 – *Réseaux élémentaires et distributions de probabilités associées*

De manière générale, si $x_1, x_2, \ldots, x_n$ sont les variables présentes dans le réseau, cette distribution de probabilité est définie par $P(x_1, x_2, \ldots, x_n)$ telle que :

$$P(x_1, x_2, \ldots, x_n) = P(x_n|x_{n-1}, \ldots, x_1)P(x_{n-1}|x_{n-2}, \ldots, x_1) \ldots P(x_2|x_1)P(x_1) \tag{3.6}$$

La Figure 3.13 montre un réseau Bayesien constitué de 8 nœuds. La distribution de probabilités jointes associée à l'ensemble des variables $x_1, \ldots, x_8$ est donnée par :

$$\begin{aligned} P(x_1, \ldots, x_8) &= P(x_6|x_5)P(x_7|x_5)P(x_5|x_4)P(x_8|x_2) \\ &\quad P(x_4|x_2, x_3)P(x_2|x_1)P(x_3|x_1)P(x_1) \end{aligned} \tag{3.7}$$

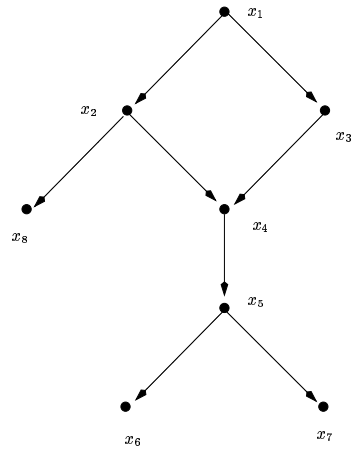


FIG. 3.13 – *Exemple de réseau Bayesien*

Cette distribution de probabilités jointes permet de calculer la probabilité associée à chaque variable (nœud) du graphe (*i.e.*, $P(x_1), \ldots, P(x_n)$). La propagation de l'influence de l'instanciation d'une de ces variables peut alors être réalisée.

Les réseaux Bayésiens ont l'avantage de permettre de représenter les connaissances que l'on a *a priori* de l'application. Cette connaissance est reflétée à deux niveaux différents :

- dans la structure même du réseau à travers la nature et le nombre des nœuds (variables) qui le compose, les différents états que peuvent prendre ces variables et enfin les relations d'indépendance entre les variables (caractérisées par les arcs qui relient les différent nœuds entre eux).

- dans les tables de probabilités conditionnelles associées aux différentes variables du réseau et qui reflètent les choix de l'expert. Ajoutons que ces tables permettent aussi de modéliser l'incertitude associée aux observations.

Enfin, les mécanismes de propagation associés au réseaux Bayesiens permettent de prendre en compte toute nouvelle observation. Son influence est propagée à toutes les autres variables du réseau compte tenu des relations de dépendances fixées.

Réseaux Bayesiens en vision par ordinateur. Les réseaux Bayesiens ont déjà été de nombreuses fois utilisés en analyse mono-image : pour établir des groupements perceptuels [Sarkar 93], la segmentation en niveaux de gris [Kumar 96], la reconnaissance d'objets [Agosta 88], etc. L'utilisation de ces réseaux en vision active est plus récente. Les travaux les plus discriminants ont été proposés par Rimey et Brown [Rimey 94] avec le système TEA-1, Buxton et Gong [Buxton 95], ou encore Djian et Rives [Djian 96]. Les finalités de ces systèmes sont différentes, ils ont cependant un point commun : la réalisation de la tâche nominale nécessite l'exécution d'actions de perception qui sont générées en utilisant les connaissances présentes dans le ou les réseaux Bayesiens utilisés.

Rimey et Brown [Rimey 93], [Rimey 94] ont développé le concept de vision sélective basée sur la collecte séquentielle d'informations pour répondre à des questions spécifiques.

Ces questions concernent l'interprétation d'une scène de manière générale ou bien encore la détection de la présence d'un objet donné. Le système proposé n'exécute que les actions nécessaires pour pouvoir répondre à la question posée, et utilise une base de connaissance sur le modèle général de la scène (organisation) et sur les modèles particuliers des objets. L'ensemble de ces connaissances est regroupé dans des réseaux Bayesiens. Certains de ces réseaux regroupent la connaissance sur un domaine spécifique (*composite nets*), d'autres regroupent la connaissance relative à une tâche précise (*task net*). Le *composite net* est lui-même décomposé en trois réseaux Bayesiens : l'un d'eux regroupe la structure physique de la scène (*PART-OF net*) indiquant la présence ou l'absence de tel ou tel objet dans la scène, un autre regroupe les relations géométriques (*expected area net*), c'est-à-dire où l'objet devrait se situer dans le plan observé, le dernier, enfin, exprime les relations hiérarchiques entre objets ou groupes d'objets (réseau en forme d'arbre *IS-A tree*). Les actions sont séparées en deux catégories en fonction de leur coût : d'une part les actions visuelles traitant les données extraites des images avec pour objectif l'extraction d'indices visuels nécessaires pour répondre aux questions posées, d'autre part les actions de déplacement de la caméra (calcul de points de vue) dont le coût est plus élevé. Le système décrit utilise les informations contenues dans l'ensemble des réseaux Bayesiens et les fonctions de coût pour déterminer la meilleure séquence d'actions à exécuter. Deux solutions de planification des tâches sont comparées. La première est basée sur le calcul d'une fonction de qualité qui décide de la nécessité de déplacer la caméra ou d'effectuer une action visuelle. La seconde consiste à énumérer l'ensemble des séquences d'actions réalisables et d'extraire celle dont le coût est le plus faible. Le méthode employée ci-dessus reste dédiée à des tâches très spécifiques qui nécessitent de définir précisément les *a priori* contenus dans les différents réseaux Bayésiens. Une approche très similaire est proposée dans [Jensen 91].

Buxton et Gong [Buxton 95] cherchent à construire un système de surveillance du trafic routier. Ils ne souhaitent pas uniquement suivre les véhicules dans une séquence d'images mais ils désirent aussi analyser, interpréter et classifier leur comportement. Pour cela, il convient de disposer de connaissances *a priori* sur les comportements possibles des différents objets mis en jeu. De plus, il est nécessaire de concevoir un système permettant de définir les différentes actions de perception utiles à l'accomplissement de la tâche nominale. Les réseaux Bayesiens fournissent un cadre unifié permettant de disposer à la fois de la connaissance *a priori* et de la modélisation du contrôle de tâche. Enfin, ils autorisent une gestion dynamique (grâce au processus de propagation) des dépendances entre les acteurs impliqués dans le processus d'interprétation visuelle. L'approche proposée est décomposée en trois points : fournir une interprétation possible (la plus probable) des données disponibles, puis mettre à jour cette interprétation en fonction des nouvelles données et de l'interprétation courante. Dans cet article, Buxton et Gong proposent un cadre méthodologique général intéressant pour la conception et la mise en œuvre de systèmes de vision intentionnelle.

Djian, Probert et Rives utilisent les réseaux Bayesiens dans un cadre de reconnaissance d'objets [Djian 96]. Ils cherchent à reconnaitre des modèles géométriques simples (des portes et des fenêtres) dans une séquence d'images. Pour cela, des observateurs dynamiques cherchent à détecter l'apparition de motifs élémentaires dans la scène. Les réseaux Bayesiens servent ici à guider le raisonnement en prenant une décision sur l'action à exé-

cuter (positionnement dans l'image des observateurs) et en émettant une hypothèse sur le motif que l'on doit détecter dans ces observateurs. Un deuxième réseau permet de vérifier si l'hypothèse émise est correcte.

Dans tous ces exemples, l'utilisation des réseaux Bayesien implique une connaissance parfaite de la tâche à effectuer. La définition du réseau permet de coder les connaissances *a priori* dont on dispose ainsi que les stratégies d'interprétation et d'analyse (action) de scène. Par conséquent, l'utilisation de réseaux Bayesiens implique que l'on se place résolument dans le cadre de la **vision intentionnelle**.

3.2.1.2 Approche générale retenue

Le principe de base de notre approche est décrit sur la Figure 3.14. La partie perception / reconstruction / modélisation correspond à l'algorithme de reconstruction incrémentale présenté dans la première partie de ce chapitre (la modélisation dans ce cas correspond uniquement à la création de la carte 3D de l'environnement et à sa mise à jour après chaque estimation active).

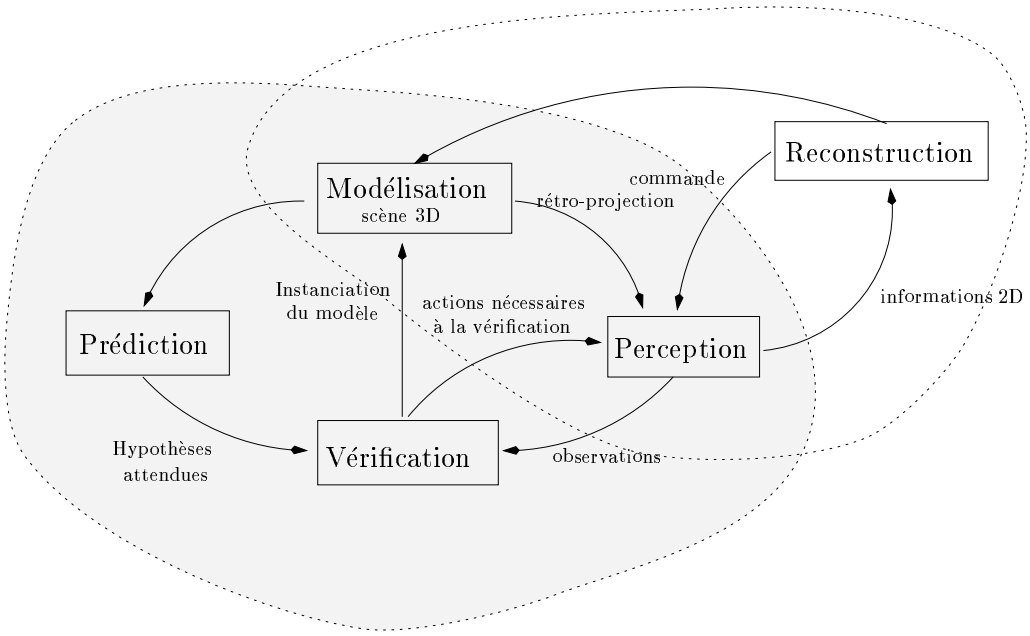


FIG. 3.14 – *Principe général*

L'objectif principal est d'obtenir une représentation de la scène en termes d'objets simples : segments, polygones. L'approche proposée pour parvenir à cet objectif repose sur des techniques de prédiction / vérification d'hypothèses. L'information sur la scène dont nous disposons est composée, initialement, uniquement d'un ensemble $\mathcal{S}(\mathcal{T}_0^{t-1})$ de segments 3D. $\mathcal{S}(\mathcal{T}_0^{t-1})$ est un sous-ensemble de l'ensemble $\mathcal{O}(\mathcal{T}_0^{t-1})$ qui représente l'ensemble des objets de la scène (et qui contient des segments 3D mais aussi des cylindres, des jonctions, des polygones, etc.). L'objectif est de regrouper ces segments afin de former des

objets plus complexes. La reconstruction se faisant de façon incrémentale, ce module doit permettre de déterminer les conséquences induites par l'intégration de chaque nouveau segment S_t. Il est donc utilisé dès qu'un segment est rajouté à l'ensemble $\mathcal{S}$ des segments 3D composant la scène.

Pour cela, nous allons procéder en trois étapes. Pour chaque couple de segments $(S_{t'}, S_t), t' \in [0, t-1]$, nous allons déterminer les informations 3D supplémentaires que ce couple de segments peut apporter. Puis nous vérifierons que les hypothèses émises sont conformes à la réalité. Nous considérons trois étapes successives qui seront détaillées dans la suite de ce chapitre :

- **Génération d'hypothèses.** Pour cela, nous formulons quatre hypothèses :

  1. les segments S_t et $S_{t'}$ sont identiques ;
  2. une jonction relie S_t et $S_{t'}$;
  3. un segment relie les extrémités de S_t et $S_{t'}$;
  4. il n'y a pas de relation (ou il y a d'autres relations plus complexes que nous ne prenons pas en compte) entre S_t et $S_{t'}$.

  Il est possible d'émettre plusieurs de ces hypothèses en parallèle. À chacune d'entre elles, nous associons la confiance que nous lui accordons sur la base des connaissances relatives à la position 3D des segments considérés et des connaissances introduites dans le réseau Bayesien gérant la génération d'hypothèses.

- **Vérification des hypothèses émises.** Chacune de ces hypothèses doit ensuite être vérifiée. Cette vérification se fait en utilisant des informations 2D. Deux cas peuvent se présenter. L'information 2D permettant de vérifier ou d'infirmer cette hypothèse est présente dans la séquence d'images acquises sur la trajectoire $\mathcal{T}_0^t$; dans ce cas la décision peut se prendre sans acquisition d'informations supplémentaires. Dans le cas contraire, il faut acquérir explicitement cette information ; des mouvements de la caméra ainsi que l'acquisition et le traitement des images supplémentaires sont alors nécessaires pour prendre cette décision.

- **Instanciation des modèles de la scène.** Enfin, cette modélisation locale de la scène (puisqu'elle ne concerne que deux segments) doit être intégrée dans une modélisation de plus haut niveau de la scène considérée. Cette information supplémentaire que l'on vient d'acquérir permet d'inférer l'existence d'objets plus complexes (chaîne de segments coplanaires, polygones, faces,...). Là encore, les hypothèses émises à ce niveau doivent être vérifiées et les conclusions propagées au reste du système.

La génération d'une hypothèse se fait en utilisant bien sûr le modèle 3D courant, mais aussi en se basant sur un certain nombre d'*a priori* très généraux sur les caractéristiques d'une scène polyèdrique. Notons qu'en aucun cas, ces *a priori* ne s'opposent aux hypothèses qui ont été initialement réalisées sur le type de scènes que nous avons décidé d'étudier (à savoir aucune connaissance préalable sur la localisation des primitives non reconstruites de la scène). Nous allons maintenant décrire les différents réseaux Bayesiens utilisés ainsi que les stratégies de perception mises en œuvre.

3.2.2 Description de la méthode

3.2.2.1 Prédiction : génération des hypothèses

Les différentes actions possibles que nous pouvons exécuter afin de former de nouveaux objets à l'aide des deux segments $S_{t'}$ et S_t sont les suivantes :

- fusionner les segments $S_{t'}$ et S_t ;
- joindre les segments $S_{t'}$ et S_t en une jonction $J_{t't}$;
- ajouter un ou deux segments entre $S_{t'}$ et S_t ;

L'objectif de cette première étape est donc d'émettre une ou plusieurs hypothèses menant à la réalisation d'une ou plusieurs de ces actions. Nous allons donc décrire le processus permettant, à partir d'un couple $(S_{t'}, S_t)$ de segments 3D, d'émettre une ou plusieurs hypothèses sur les relations existant entre les deux segments.

Les hypothèses à émettre sont directement reliées aux actions que l'on souhaite réaliser, à savoir :

- H_1 : il existe une jonction entre $S_{t'}$ et S_t ;
- H_2 : il existe un ou deux segments entre $S_{t'}$ et S_t, ce qui implique qu'il existe une chaîne $(S_{t'}, S_k, S_t)$ ou $(S_t, S_j, S_{t'})$ ou $(S_{t'}, S_k, S_t, S_j)$ (voir Figure 3.15) ;

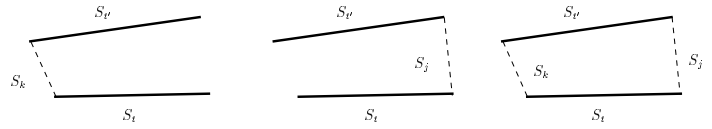


FIG. 3.15 – *Hypothèse H_2 : ajout de segments*

- H_3 : $S_{t'}$ et S_t sont le même segment ;
- H_4 : il n'existe pas ou il existe d'autres relations entre $S_{t'}$ et S_t.

Deux séries d'hypothèses seront émises, l'une concernant les relations entre les deux extrémités les plus proches des segments et l'autre concernant les relations entre les deux extrémités les plus éloignées. Notre objectif est d'associer à chacune de ces hypothèses la confiance que nous avons en elle, en fonction des observations et des connaissances déjà acquises sur la scène.

Pour cela, nous procédons en plusieurs étapes : dans un premiers temps, nous recherchons des relations topologiques simples (voisinage, coplanarité, colinéarité) entre les deux segments $S_{t'}$ et S_t, puis nous définissons cinq classes distinctes auxquelles peuvent appartenir ces segments, chacune de ces classes représentant une combinaison particulière des différentes relations élémentaires précédemment définies. Enfin, à partir de ces classes, nous émettons une hypothèse sur la nature de la relation associée aux segments. Ce schéma de raisonnement est codé dans la structure du réseau Bayesien de la Figure 3.16. Celui-ci, relativement simple, comporte six nœuds principaux, chacun étant associé à une étape du

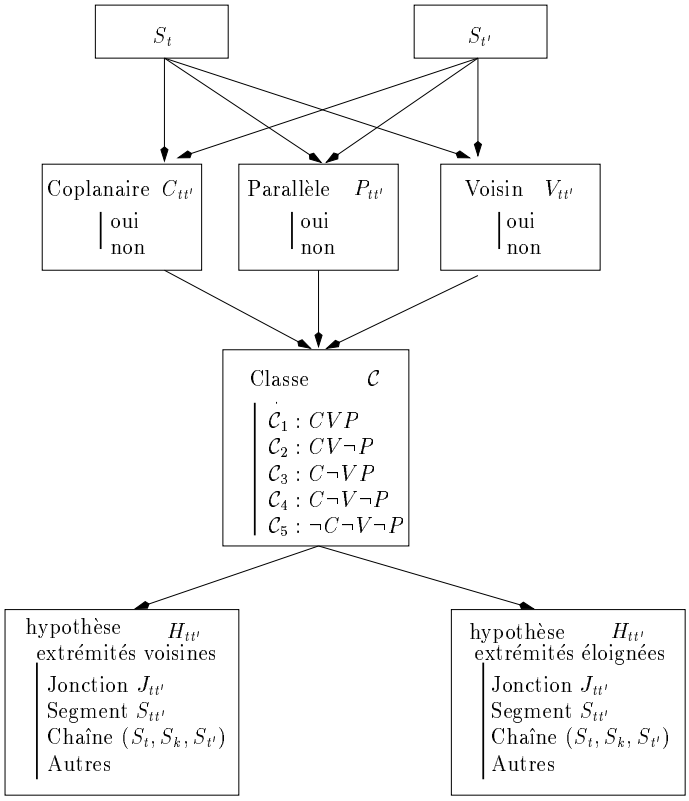


FIG. 3.16 – *Réseaux de génération d'hypothèses*

raisonnement. Un nœud est associé à chacune des relations topologiques 3D élémentaires : *voisinage, coplanarité* et *parallélisme*. Un autre nœud est associé à la classe à laquelle appartient le couple de segments. Enfin, deux nœuds sont associés aux hypothèses proposées (H_1, H_2, H_3, H_4). Les liens entre ces différents nœuds définissent les relations de causalité entre les différentes étapes du raisonnement et donc l'ordre dans lequel celui-ci doit s'effectuer.

Dans un premier temps, il faut déterminer les relations de voisinage, coplanarité et colinéarité entre les segments ainsi que la confiance que l'on accorde à cette relation (c'est-à-dire par exemple la probabilité que deux segments soient voisins). Les erreurs intervenant dans l'estimation des paramètres des segment 3D sont en général faibles. Elles ne sont cependant pas totalement négligeables. On considérera donc ces mesures comme des variables aléatoires qui suivent une loi Gaussienne. Cette probabilité doit être maximale quand les deux segments vérifient la propriété considérée. Par exemple dans le cas de la propriété de voisinage[3], on doit avoir $p(S_{t'}$ voisin de $S_t) = 1$ quand $d(S_{t'}, S_t) = 0$ où d est la distance Euclidienne des deux points extrémités les plus proches. Cette probabilité doit décroitre quand la distance augmente. On peut aussi modéliser la probabilité que deux segments soit parallèles (en fonction de la valeur du produit scalaire de leur vecteur

3. Nous considérons que deux segments sont voisins si les extrémités les plus proches sont voisines

directeur) ou que deux segments soit coplanaires (en fonction de la distance maximale d'un des points extrémités au plan formé par les trois autres).

Compte tenu de la confiance que nous accordons ainsi à ces trois relations (coplanarité $p(C_{tt'})$, voisinage $p(V_{tt'})$ et parallèlisme $p(P_{tt'})$), il est alors possible de classer ces couples de segments en cinq classes. Un couple de segments peut appartenir à l'une des classes élémentaires $\mathcal{C}$ suivantes (voir la première colonne du tableau de la Figure 3.17) :

- $\mathcal{C}_1$: les segments sont coplanaires, voisins et parallèles CVP ;
- $\mathcal{C}_2$: les segments sont coplanaires, voisins et non parallèles $CV\neg P$;
- $\mathcal{C}_3$: les segments sont coplanaires, non voisins et parallèles $C\neg VP$;
- $\mathcal{C}_4$: les segments sont coplanaires, non voisins et non parallèles $C\neg V\neg P$;
- $\mathcal{C}_5$: les segments sont non coplanaires, non voisins et non parallèles $\neg C\neg V\neg P$

  Remarque : Dans l'absolu, huit classes sont possibles mais seules les cinq ci-dessus sont compatibles avec une réalité physique (par exemple une classe qui définirait deux segments non coplanaires, voisins et/ou parallèles ne peut évidemment pas exister puisque deux segments voisins ou parallèles sont obligatoirement coplanaires).

Enfin, à partir de la classe à laquelle appartient ce couple de segments, il convient de déterminer l'hypothèse à adopter concernant la nature des relations entre ces deux segments. Nous allons donc définir des stratégies de décisions qui détermineront la meilleure hypothèse possible, compte tenu des connaissances disponibles. Ces stratégies sont codées dans les tables de probabilités conditionnelles $P(\mathcal{H}|C)$. Deux séries d'hypothèses sont émises en même temps, l'une concerne les relations établies entre les extrémités les plus proches des segments S_t et $S_{t'}$ (voir Figure 3.17) et l'autre les relations établies entre les extrémités les plus éloignées (voir Figure 3.18). Dans les deux cas, le même type d'hypothèse peut être émis, les probabilités conditionnelles associées sont cependant très différentes (ainsi si on considère les relations entre les extrémités éloignées, la probabilité d'une jonction sera presque nulle).

Les hypothèses que nous pouvons faire sur la nature des relations entre deux segments dépendent de la classe à laquelle ils appartiennent. Afin d'émettre cette hypothèse, il nous apparaît nécessaire de proposer quelques considérations élémentaires sur les relations topologiques que l'ont retrouve généralement dans un groupe de segments. Ces considérations partent du constat que, dans la plupart des cas, il y a une certaine organisation qui régit la modélisation des objets polyédriques. Notons qu'en aucun cas, elles ne s'opposent aux hypothèses qui ont été initialement réalisées sur le type de scènes que nous avons décidé d'étudier. Ces considérations sont les suivantes :

- il est rare qu'un segment se retrouve isolé car il appartient généralement à un objet plus complexe ;
- il est courant que deux segments voisins forment une jonction et appartiennent, par conséquent, au même objet.

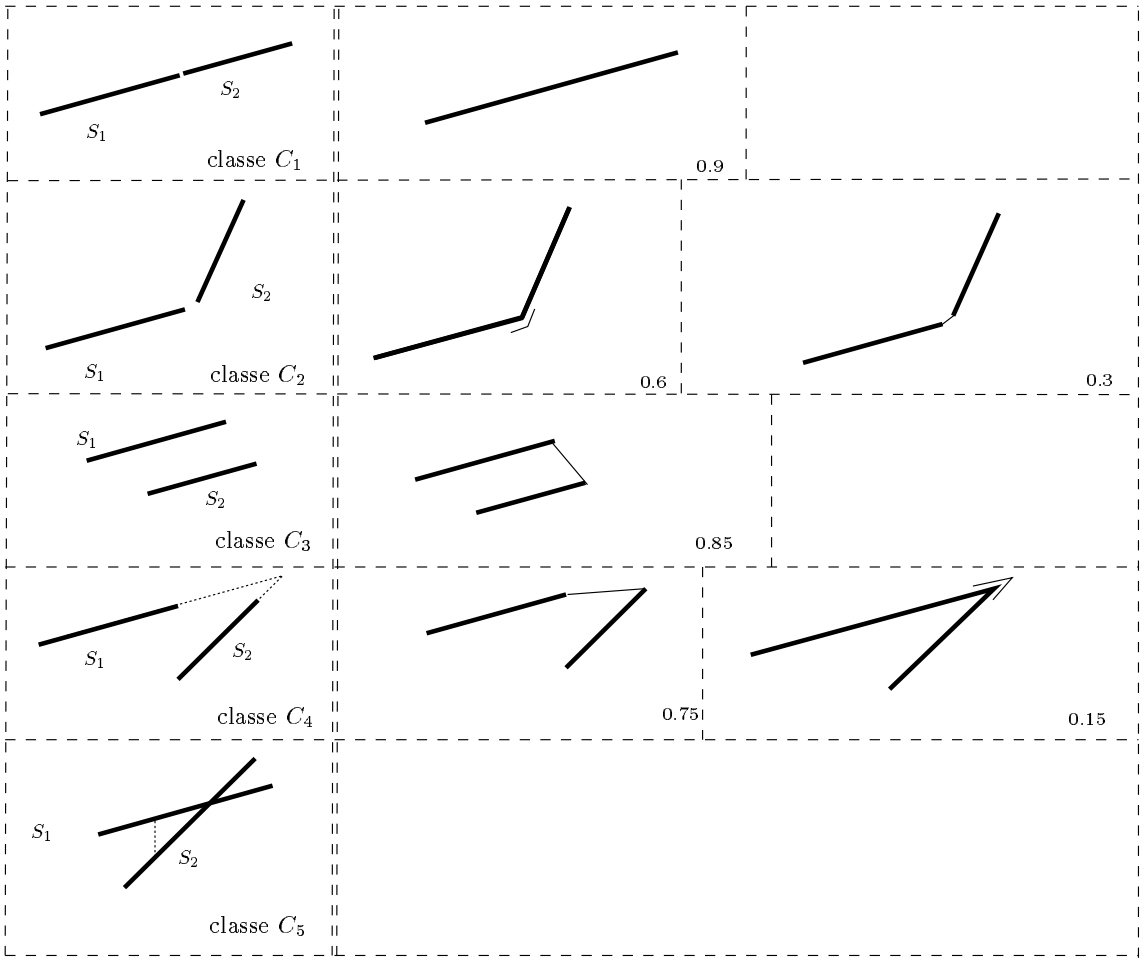


FIG. 3.17 – *Classes élémentaires et hypothèses associées: extrémités proches*

- de même, il est courant de voir un ou deux segments reliant les extrémités de deux segments coplanaires et/ou parallèles, formant ainsi un polygone.
- par contre, il est rare de voir deux segments n'ayant aucune relation de voisinage, coplanarité ou de parallèlisme entre eux appartenir au même objet.

Ces considérations reflètent dans la plupart des cas la réalité. Cependant, elles ne sont, bien entendu, pas toujours vérifiées. Elles peuvent cependant servir de base pour la définition des hypothèses que nous souhaitons pouvoir émettre. Les hypothèses principales choisies sont représentées sur les Figures 3.17 et 3.18. Compte tenu de l'incertitude, d'autres hypothèses peuvent être associées à chaque classe, mais avec une probabilité très faible.

Pour illustrer ce point, prenons l'exemple de deux segments voisins, coplanaires et non parallèles (ces deux segments appartiennent donc à la classe $\mathcal{C}_2$ et les hypothèses associées sont décrites sur la deuxième ligne de la Figure 3.17). L'hypothèse la plus vraisemblable dans ce cas est de supposer la présence d'une jonction entre ces deux segments. Cependant,

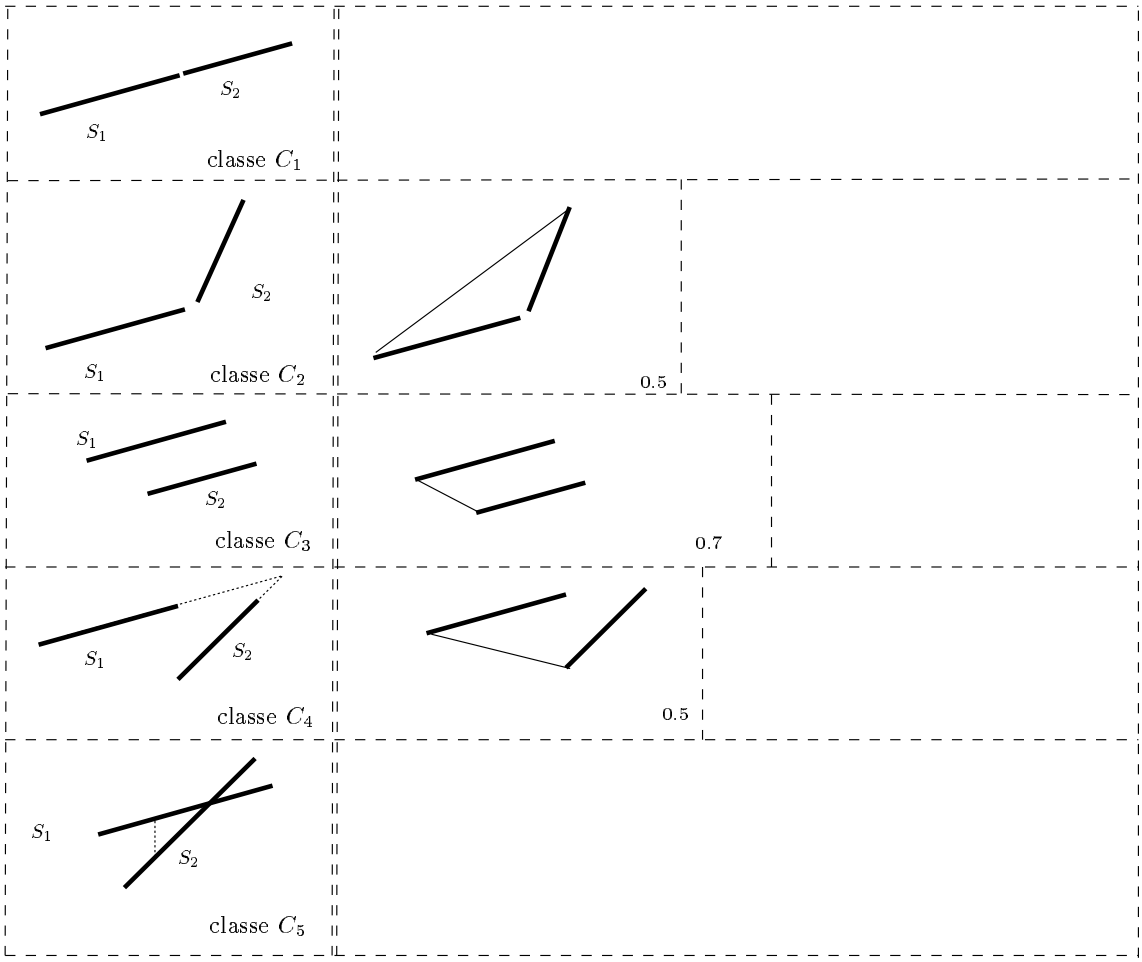


FIG. 3.18 – *Classes élémentaires et hypothèses associées: extrémités éloignées*

compte tenu de l'incertitude associée aux calculs des paramètres des deux segments (et en particulier de leur longueur), il est aussi possible de supposer qu'il existe un segment reliant leur extrémité voisine. Les autres hypothèses (fusion des segments, etc) sont envisageables mais très peu probables. Ce type de raisonnement nous permet de définir la table de probabilités conditionnelles associées à cette classe :

$$\begin{aligned} P(H = fusion \mid \mathcal{C}_2) &= 0.025 \\ P(H = jonction \mid \mathcal{C}_2) &= 0.6 \\ P(H = chaine \mid \mathcal{C}_2) &= 0.3 \\ P(H = autre \mid \mathcal{C}_2) &= 0.075 \end{aligned}$$

Ces tables de probabilités conditionnelles sont définies de manière empirique. Cependant, fixer précisément les valeurs de chaque probabilité conditionnelle n'est pas fondamental. Elle doivent juste refléter de manière grossière la connaissance que l'on souhaite transmettre au système. Si l'on considère par exemple les trois segments reconstruits de

la scène présentée sur la Figure 3.19, la classe $\mathcal{C}_2$ a été retenue pour les deux couples de segments (S_1, S_2) et (S_2, S_3). Si l'on se contente de prédire une jonction entre les deux segments, cette hypothèse se révélera vérifiée dans le cas (S_1, S_2), mais fausse dans le cas (S_2, S_3). C'est pourquoi les deux possibilités doivent apparaître dans les tables de probabilités conditionnelles, leurs valeurs résultant du risque que l'on estime prendre en choisissant l'une ou l'autre des hypothèses. Ainsi, prédire une jonction est ce qui parait être la meilleure hypothèse possible. Par contre, prédire un chaîne est une hypothèse acceptable mais moins probable, les autres cas étant très improbables (une analyse plus poussée d'un cas similaire mais plus complet sera faite dans la section résultat à la fin de ce chapitre).

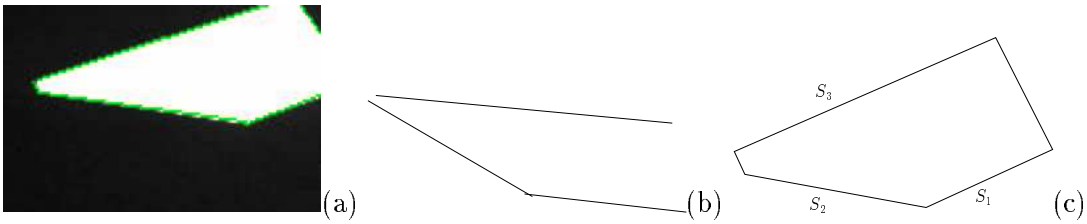


FIG. 3.19 – *Cas de figure les plus probables pour des segments voisins (a) zoom sur une partie de l'image (b) scène reconstruite (c) numérotation des segments*

Les tableaux 3.1 et 3.2 montrent les différentes probabilités conditionnelles associées aux hypothèses. Comme nous l'avons déjà indiqué, l'hypothèse la plus vraisemblable n'est pas toujours celle qui est correcte. C'est pourquoi nous retenons, dans chacune des deux séries d'hypothèses émises, les deux hypothèses auxquelles les plus fortes confiances ont été accordées. Une approche similaire dans un contexte de reconnaissance d'objet est proposée par [Djian 96]. Les deux hypothèses sélectionnées font l'objet d'une vérification, étape que nous décrivons à présent.

	Classe				
hypothèse	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$
H_1 Segment	0.90	0.025	0.05	0.05	0.025
H_2 Jonction	0.025	0.60	0.05	0.15	0.025
H_3 Chaîne	0.025	0.30	0.85	0.75	0.025
H_4 Autres	0.05	0.075	0.05	0.05	0.925

TAB. 3.1 – *Table de probabilité conditionnelle P(hypothèse | classe) pour les extrémités proches (Figure 3.17)*

	Classe				
hypothèse	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$
H_1 Segment	0.025	0.025	0.025	0.025	0.025
H_2 Jonction	0.025	0.025	0.025	0.025	0.025
H_3 Chaîne	0.025	0.5	0.7	0.5	0.025
H_4 Autres	0.925	0.45	0.25	0.45	0.925

TAB. 3.2 – *Table de probabilité conditionnelle P(hypothèse | classe) pour les extrémités éloignées (Figure 3.18)*

3.2.2.2 Vérification des hypothèses émises

L'étape de prédiction repose essentiellement, comme nous l'avons vu, sur les connaissances 3D déjà acquises et sur la connaissance d'un "expert" qui est codée dans un réseau Bayesien. Celui-ci, présenté sur la Figure 3.20, permet d'inférer la nature de l'objet associé aux deux segments considérés en confirmant ou infirmant les hypothèses émises dans l'étape précédente.

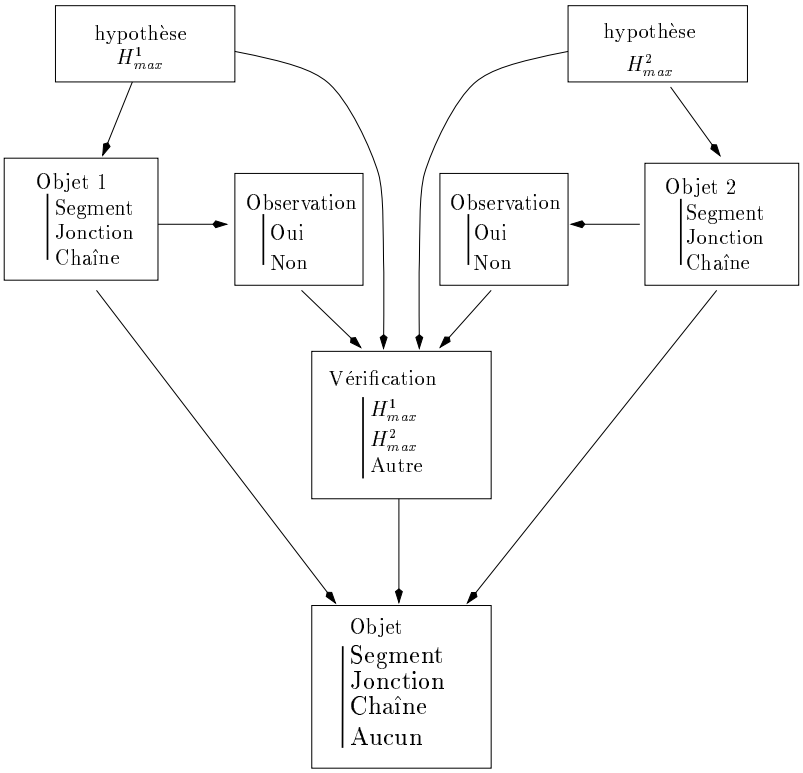


FIG. 3.20 – *Réseau de vérification d'hypothèses*

Dans un premier temps, en considérant les deux hypothèses retenues, nous définissons, la nature (segment, jonction, chaîne) et la position de l'objet qui sont associées à cette hypothèse. Puis, en utilisant des informations extraites des images, il est possible de déterminer si cet objet existe ou non. Cette étape nous paraît la plus importante, car elle fait directement intervenir les stratégies de perception. En effet, si dans certains cas l'hypothèse peut être directement confirmée ou infirmée en se basant uniquement sur les informations déjà acquises, dans d'autres cas, cette information peut s'avérer insuffisante. Il est alors nécessaire de déplacer la caméra afin d'acquérir (ou non) le complément d'information attendu. Ainsi, connaissant la confiance accordée à chacune des deux hypothèses, ainsi que la vraisemblance des observations associées, il est possible de déterminer laquelle des deux hypothèses était valide (ou éventuellement de rejeter les deux hypothèses). Enfin, compte tenu de la validation de l'une de ces deux hypothèses, nous en déduisons l'objet associé au couple de segments considéré.

Observations - Actions de vérification : La partie la plus importante du réseau correspond aux nœuds d'*observation*. C'est en effet à ce niveau qu'interviennent des stratégies de perception au sens où des mouvements complémentaires de la caméra sont parfois nécessaires pour valider ou infirmer une hypothèse. À chaque hypothèse émise, le réseau Bayesien associe un objet 3D à détecter dans la scène. Cet objet 3D peut être une jonction entre deux segments ou un autre segment 3D.

Cette validation se fait en utilisant l'information 3D déjà disponible et surtout les observations 2D réalisées. La vérification s'effectue en plusieurs étapes. Dans un premier temps, il convient de calculer, pour toutes les positions précédemment occupées par la caméra (c'est-à-dire appartenant à la trajectoire $\mathcal{T}_0^t$), la position dans le plan image de l'objet considéré. Pour cela, on procède par rétro-projection du modèle 3D supposé dans le plan image. Nous cherchons ensuite à mettre en correspondance les segments 2D observés dans plusieurs images (afin d'éviter de faux appariements) avec la projection de l'objet 3D. Pour chaque appariement possible, une mesure de la confiance accordée à cet appariement est calculée.

Si l'objet recherché est une jonction entre deux segments, et dans l'hypothèse où cette jonction existe, elle a forcément déjà été observée. En effet, l'hypothèse de la présence d'une jonction découle inéluctablement de la présence de deux segments dont l'existence a déjà été validée.

Le cas où l'objet supposé est un segment est plus intéressant. Dans une chaîne composée de trois segments, l'existence de deux d'entre eux a déjà été confirmée (ce sont eux qui ont servi à prédire la présence de l'autre). Ce dernier, par contre, n'a pas encore été reconstruit et sa présence doit être validée. Dans le cas où aucun appariement n'est trouvé en utilisant les images déjà acquises, il convient d'essayer de déterminer la raison pour laquelle le segment attendu n'a pas été observé. Il existe deux raisons possibles : la première est bien entendu que l'objet considéré n'existe pas, et la seconde est qu'il ait été occulté pour l'ensemble des points de vue de la trajectoire $\mathcal{T}_0^t$. Il convient alors de déplacer la caméra vers un nouveau point de vue afin de permettre l'observation de l'objet attendu. Le paragraphe suivant présente la méthode retenue pour atteindre ce nouveau point de vue.

Calcul d'un nouveau point de vue : Plusieurs techniques sont envisageables pour calculer ce nouveau point de vue (citons par exemple les techniques mises au point par Cowan et Kovesi [Cowan 88], Tarabanis [Tarabanis 95b] ou Seales et Dyer [Seales 92]).

La méthode que nous proposons est proche de celle décrite par Cowan [Cowan 88]. Celle-ci consiste à déterminer le lieu géométrique des points depuis lesquels la primitive est visible tout en respectant un certain nombre de contraintes. Pour des polygones convexes, la région d'occlusion est obtenue en faisant pivoter des plans autour des segments composant le polygone responsable de l'occlusion, jusqu'au contact du polygone en occlusion (ici le plan contenant le segment supposé). On obtient ainsi des plans de séparation définissant une région de l'espace. Pour des objets polyédriques, on ne garde que les plans séparant l'espace en deux demi-espaces, l'un contenant l'objet réalisant l'occlusion, et l'autre l'objet

en occlusion.[4] Cette méthode a l'avantage d'être simple, mais les régions de l'espace qui sont solutions de ce problème de calcul de points de vue peuvent être très vastes et le choix d'un point de vue parmi l'ensemble des solutions possibles nécessite l'introduction de contraintes supplémentaires qui rendent le calcul de point de vue long et coûteux.

Plutôt que de calculer explicitement ce point de vue, puis de rechercher *off-line* le segment supposé dans l'image, nous avons décidé de faire pivoter la caméra autour d'un des segments appartenant à la fois au polygone d'occlusion et au plan dans lequel se situe le segment recherché. Pendant ce mouvement de rotation, généré par asservissement visuel, nous recherchons *on-line* l'apparition dans l'image du segment supposé. Comme nous l'avons déjà dit dans la première partie de ce chapitre, il n'est pas possible de réaliser une acquisition complète de l'ensemble des segments 2D présents dans l'image pendant les phases d'asservissement visuel en raison des contraintes de temps réel. La détection de l'apparition éventuelle de ce segment pendant le mouvement de la caméra est cependant impérative. Pour résoudre ce problème, nous disposons des observateurs à des positions adéquates dans l'image [Djian 96]. En l'occurrence, nous cherchons à détecter, comme dans le cas de l'évitement des occlusions (voir paragraphe 2.2.4), un contour en mouvement ayant une certaine orientation et une certaine position déterminées ici encore par rétro-projection. La position de l'observateur dépend donc de ces paramètres. La détection éventuelle du segment, s'il existe, se fera en détectant un contour en mouvement ayant les caractéristiques attendues. Pour cela, on utilise l'algorithme des ECM (éléments de contours en mouvement) [Bouthemy 89] adapté aux droites [Boukir 93b].

L'exemple décrit sur les Figures 3.21 et 3.22 illustre la stratégie retenue. On considère ici un objet polyédrique dont deux faces avec les arêtes associées ont déjà été reconstruites. Lors de la génération des hypothèses, la présence d'un segment a été prédite dans le plan formé par S_1 et S_2. La stratégie retenue consiste à se focaliser sur l'un de ces deux segments puis à effectuer un mouvement de rotation autour de ce segment jusqu'à ce qu'un nouveau segment S apparaisse dans l'image. Si une butée articulaire du manipulateur est rencontrée, l'hypothèse ne sera pas validée. Ici S_1 a été retenu car S_2 n'est pas assez long pour servir de base à la génération d'un mouvement de rotation par asservissement visuel.

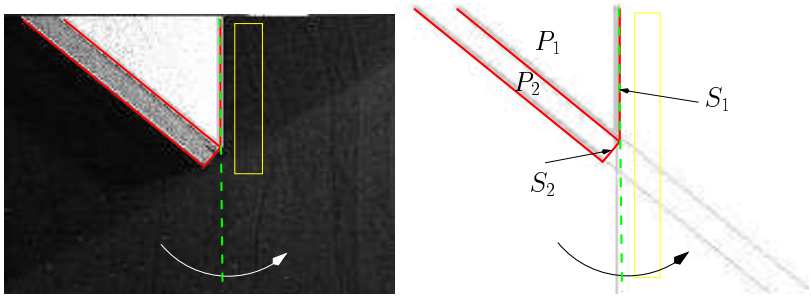


FIG. 3.21 – *La fenêtre sur la droite de l'image représente l'observateur dans lequel est exécuté le traitement d'image (détection de contours en mouvement formant un segment).*

4. Le calcul explicite de point de vue faisant partie des techniques d'exploration globale, une analyse plus complète de l'article de Cowan et Kovesi est proposée dans le chapitre 4.1.

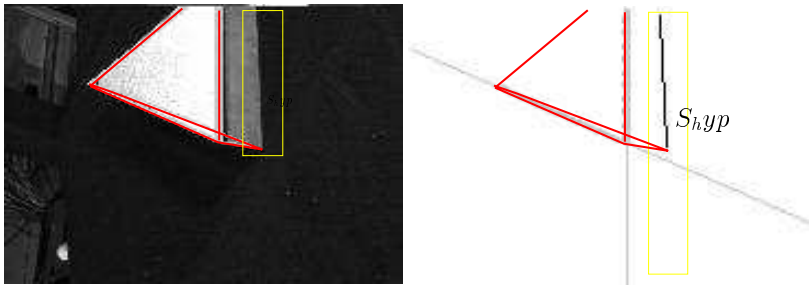


FIG. 3.22 – *Détection du segment prédit après un mouvement de pivot autour de l'axe du segment*

3.2.2.3 Modélisation de la scène

L'utilisation des techniques de prédiction / vérification d'hypothèses couplées à une approche Bayesienne nous permet d'obtenir une représentation plus complète de la scène considérée. La représentation de la scène dont nous disposons est, à ce stade de la reconstruction, une modélisation en termes de segments et jonctions 3D. L'étape suivante consiste, dans la mesure du possible, à passer de cette représentation à une représentation en termes de polygones 3D. Pour cela, nous utilisons les informations obtenues sur les jonctions 3D ainsi que les informations de coplanarité déjà utilisées lors de la génération d'hypothèses.

Xie [Xie 88], [Xie 89a] a proposé une méthode reposant sur la transformée de Hough généralisée pour regrouper les segments 3D appartenant à un même plan. Les paramètres de cette surface sont ensuites estimés aux moindres carrés avant l'extraction de chaînes fermées de segments voisins appartenant au même plan, formant ainsi un polygone.

Dans notre cas, nous utilisons, dans un premier temps, un réseau Bayesien afin d'inférer la présence de chaînes de segments coplanaires et de polygones. Dans ce cas, le réseau Bayesien sert à représenter les connaissances que nous avons sur la scène et à guider le raisonnement permettant d'aboutir à la conclusion. Notre objectif est de savoir si deux objets O_1 et O_2 font en fait partie du même objet (et si oui quel est cet objet). O_1 et O_2 peuvent être soit des segments, soit des chaînes de segments précédemment construites en utilisant cette même approche. Le nœud action du réseau présenté sur la Figure 3.23, permet, le cas échéant, de chaîner deux objets connaissant la probabilité d'existence de ces objets, la probabilité qu'ils soient coplanaires et la probabilité qu'il y ait une jonction entre ces deux objets[5]. Ensuite, connaissant la confiance que l'on accorde au chaînage des deux objets et la probabilité de savoir si la chaîne est fermée (fermeture définie par la jonction (S_i, S_n)), il est possible de déduire la probabilité que l'objet final soit formé de deux chaînes indépendantes, une seule chaîne, ou un polygone.

Quand l'objet ainsi détecté est un polygone, il convient de calculer le plus exactement possible les caractéristiques du polygone considéré. En effet, à ce stade, nous ne disposons pas encore réellement d'un polygone mais plus exactement d'un chaîne fermée de

5. Dans le réseau de la Figure 3.23, une seule jonction (S_j, S_k) est représentée. En fait, si l'objet considéré est une chaîne, il existe plusieurs autres jonctions possibles (*i.e.*, (S_i, S_k), (S_j, S_n), (S_i, S_n), etc).

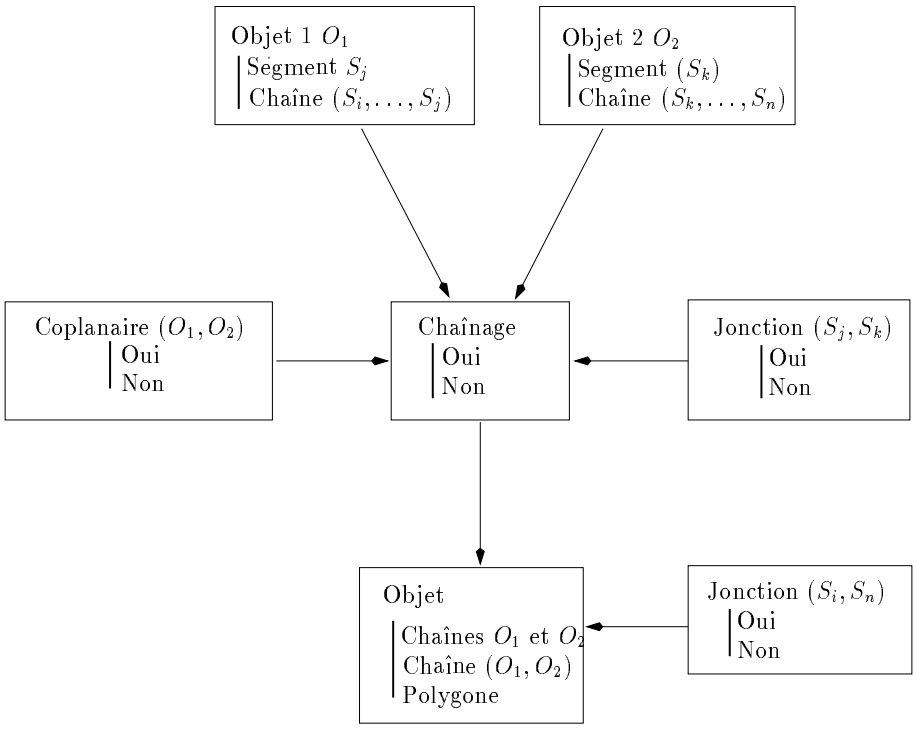


FIG. 3.23 – *Réseau de modélisation*

segments coplanaires. Il n'en demeure pas moins en pratique que tous les segments de la chaîne ne sont ni parfaitement coplanaires, ni parfaitement connexes. C'est pourquoi un traitement supplémentaire est effectué. L'algorithme que nous proposons pour transformer cette chaîne en polygone peut être divisé en plusieurs étapes :

- La première étape consiste à calculer les caractéristiques du plan porteur du polygone. Pour cela, nous calculons les paramètres du plan approximant au mieux les $2n$ points extrémités des arêtes de ce polygone (n étant le nombre d'arêtes). L'utilisation de ces $2n$ points permet d'écrire un système de $2n$ équations à trois inconnues (les trois paramètres indépendants représentant la configuration d'un plan dans l'espace). Le système étant surdimensionné, nous utilisons la technique des moindres carrés pour le résoudre et estimer les paramètres du plan.

- La deuxième étape consiste à projeter les points extrémités de chaque segment sur ce plan. Cette projection est exécutée suivant la normale au segment.

- Enfin, en considérant les segments ainsi projetés, il convient de calculer l'intersection exacte P_{ij} de deux segments voisins S'_i et S'_j (voir sur ce dernier point [Xie 88] pour de plus amples détails).

Cette dernière étape de modélisation nous permet d'obtenir une représentation de plus haut niveau de la scène considérée. La représentation initiale était uniquement composée

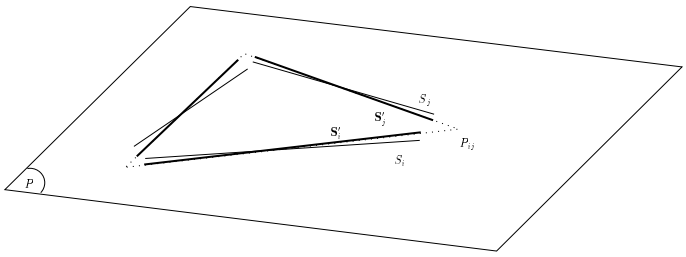
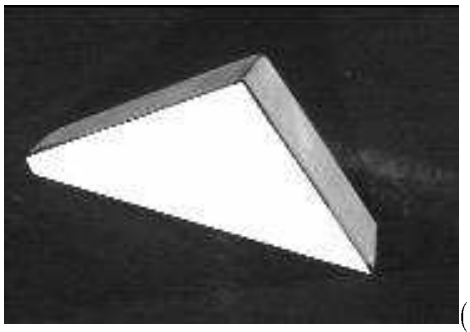


FIG. 3.24 – *Création de polygones à partir de segments 3D*

de segments 3D. La représentation finale est composée de ces même segments, mais aussi d'informations complémentaires comme les jonctions 3D, les polygones et les faces.

3.3 Résultats expérimentaux

Nous présentons ici les résultats issus de la scène polyèdre (voir Figure 3.25). Cette scène permet d'illustrer les intérêts de la méthode proposée. Comme nous l'avons vu dans la première partie de ce chapitre, la reconstruction de cet objet posait un grand nombre de difficultés : segments trop petits pour être reconstruits, présence d'occlusions, caractéristiques géométriques peu visibles (l'angle aigu est en effet coupé (voir Figure 3.25.b), ce qui induit la présence d'un segment d'une longueur de 1 cm environ).


(a)

(b)

FIG. 3.25 – *Scène "polyèdre" (a) Vue de la scène à reconstruire et (b) présence d'un "petit" segment*

Une numérotation des segments dans l'ordre de leur reconstruction est donnée sur la Figure 3.26. La Figure 3.27 montre les différentes étapes de la reconstruction du premier polygone. Quand les deux premiers segments S_0 et S_1 ont été reconstruits (voir Figure 3.27.c), les observations réalisées sont les suivantes : la probabilité que S_0 et S_1 soient coplanaires est de 99%, voisins de 91% et parallèles de 1%.

La classe d'observation à laquelle appartiennent ces deux segments est donc la classe $\mathcal{C}_2$ $(CV\neg P)$. L'utilisation du réseau Bayesien générateur d'hypothèses permet d'émettre les hypothèses suivantes :

– La probabilité que S_0 et S_1 soient identiques est de 3% ;

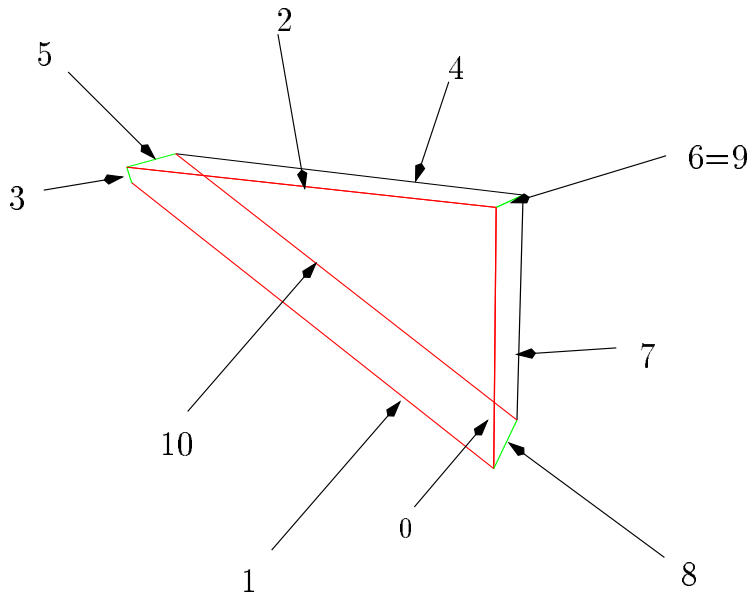


FIG. 3.26 – *Scène reconstruite et numérotation des segments dans l'ordre de leur introduction dans la carte de l'environnement.*

- La probabilité qu'il y ait une jonction entre S_0 et S_1 est de 55%
- La probabilité qu'il y ait un segment reliant S_0 et S_1 entre les deux plus proches extrémités est de 35%
- Enfin, la probabilité qu'il n'y ait aucune relation entre S_0 et S_1 est de 7%.

Ces probabilités dépendant bien sûr de la probabilité pour S_0 et S_1 d'apppartenir à telle ou telle classe, elles sont différentes de celles que l'on trouve dans le Tableau 3.1 car elles dépendent des observations.

Conformément à la stratégie codée dans les tables de probabilités conditionnelles, l'hypothèse d'une jonction est favorisée de même que, mais dans une moindre mesure, l'hypothèse de la présence d'un segment reliant S_0 et S_1. Pour vérifier ces hypothèses, le deuxième réseau Bayesien est utilisé.

Les observations faites dans la séquence d'images précédemment acquise permettent de valider l'hypothèse principale "jonction" avec une probabilité de 98%, alors que l'hypothèse secondaire de la présence d'un segment est rejetée, celui-ci n'étant pas observé (on a pris soin de vérifier que ce segment, s'il doit exister, est observable depuis les positions déjà occupées par la caméra). Enfin, compte tenu de la probabilité de chacune des deux hypothèses réalisées et des observations, on en déduit la probabilité de la présence d'un objet jonction à 85% et de la présence d'une chaîne de segments à 9%. La probabilité de l'absence de toute relation est elle égale à 6%.

Au niveau de la modélisation de la scène, la présence d'une jonction entre deux segments coplanaires permet ensuite de créer un chaîne de segments ayant une probabilité d'existence de 98%.

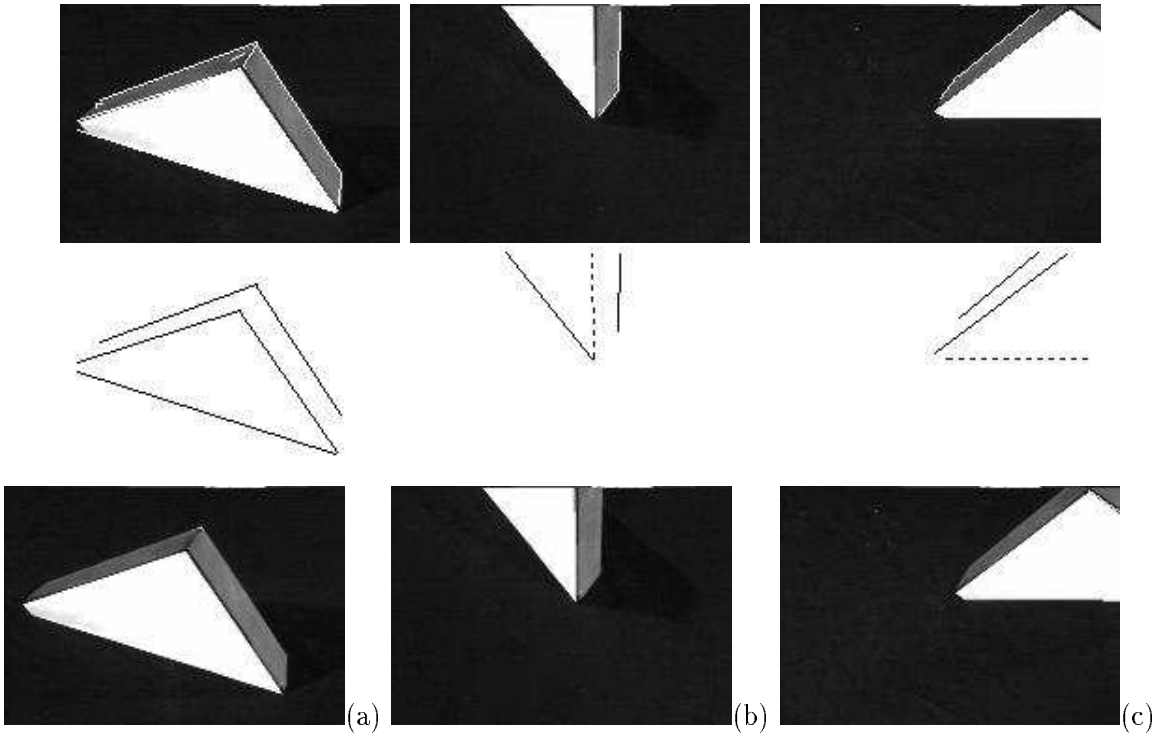


FIG. 3.27 – *Scène "polyèdre" : Reconstruction des deux premiers segments du premier polygone (la ligne du haut montre l'ensemble des segments observés, la ligne centrale montre l'ensemble des segments pouvant être reconstruits par la méthode de reconstruction présentée dans le chapitre précédent, les segments en traits pointillés ayant déjà été reconstruits, enfin la ligne du bas montre le modèle 3D reprojeté sur l'image)*

L'introduction dans $\mathcal{S}$, après sa reconstruction, du segment S_2 est plus intéressante. Il convient à ce niveau de considérer les relations entre le segment S_0 et le segment S_2, mais aussi entre les segments S_1 et S_2. L'hypothèse la plus probable associée au couple (S_0,S_2) est comme dans le cas précédent une jonction ; cette hypothèse est très facilement vérifiée en utilisant les informations présentes dans l'image. Le comportement du système est similaire à celui présenté dans l'exemple précédent. Nous ne le détaillerons donc pas. Précisons simplement que l'ajout de cette jonction permet, au niveau de la modélisation de la scène, la création d'une chaîne de segments coplanaires ayant une probabilité d'existence de 97%.

Le cas du couple (S_1,S_2) est plus intéressant. Ces deux segments sont dans l'espace 3D très proches l'un de l'autre (environ 1cm sépare leur extrémité voisine). La probabilité qu'ils soient voisins est de 61%. La probabilité qu'ils soient coplanaires est également forte (99%) ; la classe la plus probable à laquelle le système associe ces deux segments est, comme dans le cas précédent, la classe $\mathcal{C}_2$. Les hypothèses générées sont donc dans ce cas :

- La probabilité que les deux segments soient identiques est de 3% ;
- La probabilité qu'il y ait une jonction entre S_1 et S_2 est de 46% ;
- La probabilité qu'il y ait un segment reliant les deux segments considérés est de 41% ;
- Enfin, la probabilité qu'il n'y ait aucune relation entre ces deux segments est de 10%.

Lors de la vérification de ces hypothèses, la première (jonction) est vérifiée avec une probabilité de 60%. Cette probabilité relativement élevée résulte du fait que l'observation des différentes images montre que les projections des extrémités de ces deux segments sont proches l'une de l'autre dans l'espace image (environ 5 pixels), ce qui a tendance à renforcer (avec une certaine "prudence") la première hypothèse. Cependant, la seconde hypothèse s'avère vérifiée avec une probabilité de 95%. Un segment 2D, correspondant à la projection du segment prédit, apparaît en effet à la position attendue dans plusieurs images. Finalement, compte tenu de la confiance accordée aux deux hypothèses et de celle accordée aux observations, un nouveau segment S_3 est introduit dans la carte de la scène (avec une confiance de 53%, alors que la confiance accordée à l'ajout d'une jonction n'est que de 37%). On constate ici tout l'intérêt de considérer plusieurs hypothèses en parallèle puisqu'une technique classique de prédiction-vérification aurait vraisemblablement validé la première hypothèse.

Le quatrième segment S_3 est donc introduit dans la carte de la scène. Contrairement aux trois précédents, celui-ci ne résulte pas d'une reconstruction explicite en utilisant la méthode décrite dans le chapitre précédent. Il convient cependant de regarder quelles sont les relations entre ce segment et les segments précédemment reconstruits. Ce cas est assez simple, et les jonctions (S_1,S_3) et (S_3,S_2) sont détectées avec une très forte probabilité. Au niveau du réseau de modélisation, la probabilité d'un chaînage de ce segment avec la chaîne existante (S_1,S_0,S_2) est donc très forte (grâce à la jonction (S_3,S_1) et au fait que le segment S_3 soit coplanaire à la chaîne (S_1,S_0,S_2)). L'existence très probable d'une

jonction entre S_3 et S_2 permet finalement de créer un polygone (S_3, S_1, S_0, S_2) avec une confiance élevée de 98%.

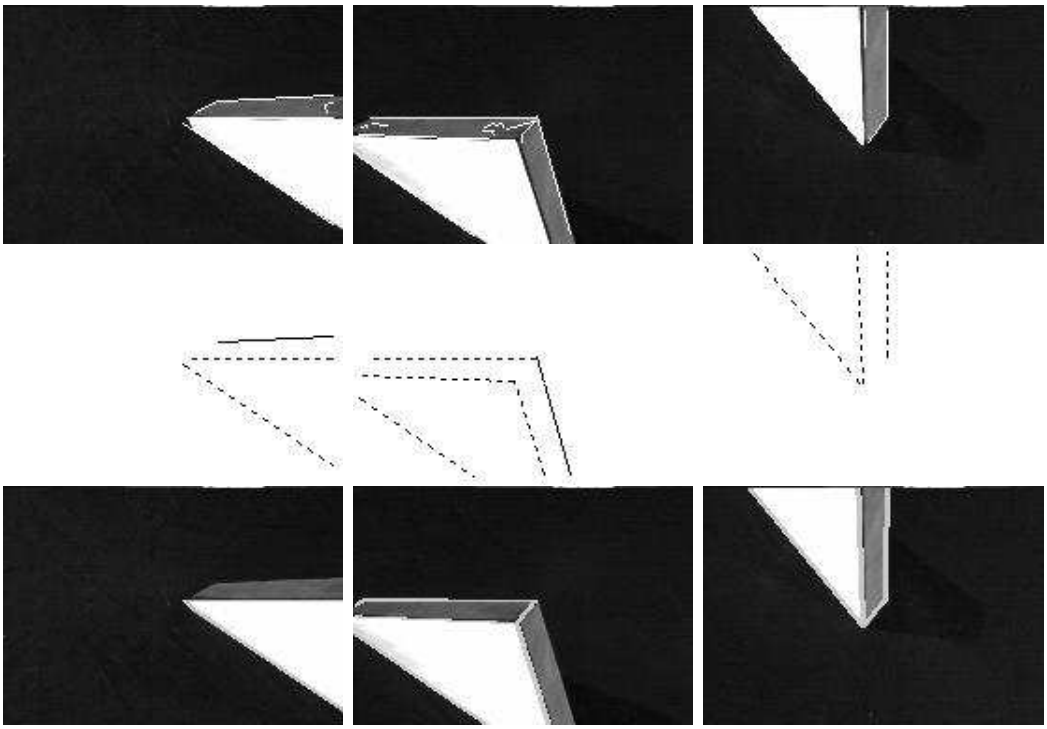


FIG. 3.28 – *Scène "polyèdre" : Reconstruction des deux polygones suivant*

La reconstruction des arêtes suivantes du polyèdre s'effectue de manière similaire et aboutit à la reconstruction de deux polygones supplémentaires (voir Figure 3.26). On peut noter que la reconstruction du segment S_4 a eu pour conséquence la création automatique des segments S_5 et S_6 (après vérification de leur présence). De manière similaire, la reconstruction du segment S_7 entraîne la génération d'une hypothèse portant sur l'ajout de deux segments : S_8 et un éventuel segment "S_9" dont la position est identique à celle du segment S_6. La présence de ces deux segments est vérifiée et ils sont donc introduits dans la carte de l'environnement. La recherche des relations entre S_9 et les huit autres segments amène le système à émettre l'hypothèse que les segments S_6 et S_9 sont identiques et doivent être fusionnés. Après vérification, cette fusion est réalisée et le segment S_9 est supprimé de la carte.

La reconstruction du segment S_7 a aussi pour conséquence la création d'une jonction avec S_4. Comme le montre la Figure 3.18, dans le cas de l'observation d'un couple de segments similaires au couple (S_7,S_4), quand l'hypothèse jonction est réalisée, une seconde hypothèse correspondant à la prédiction d'un segment reliant les deux autres extrémités des segments est aussi émise[6]. Cette hypothèse doit être vérifiée. Ce segment n'apparaît dans aucune image acquise par la caméra pendant le processus d'exploration locale. Cependant,

6. Notons que cette hypothèse a déjà été effectuée pour d'autres couples de segments, mais sa vérification s'étant avérée négative, nous ne l'avons donc pas décrite.

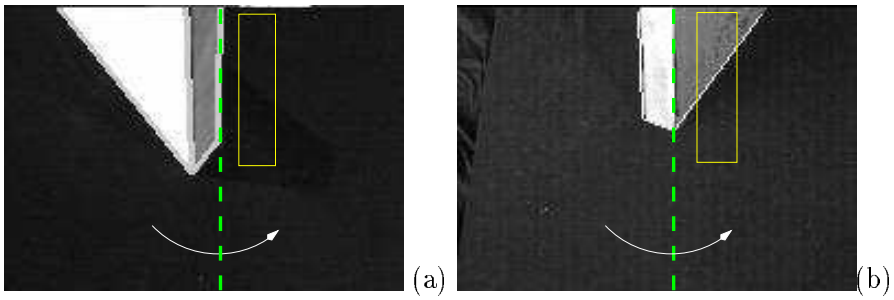
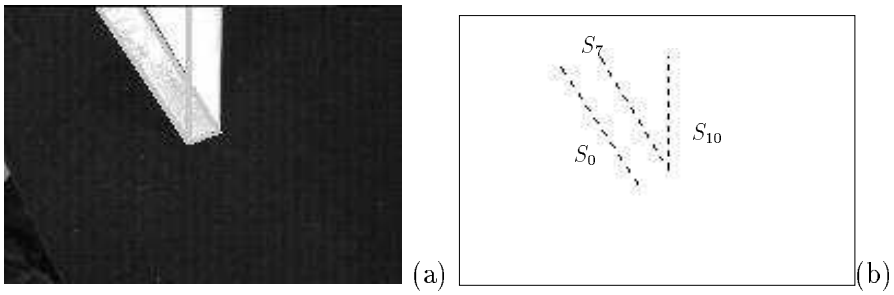

(a) (b)

FIG. 3.29 – *Vérification d'une hypothèse : rotation autour de S_7*



(a) (b)

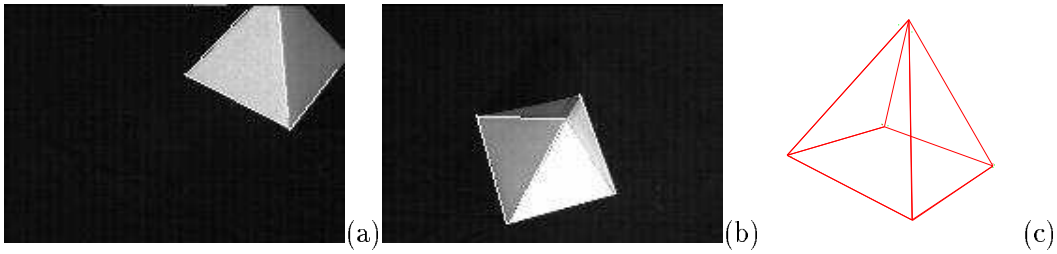
FIG. 3.30 – *Reconstruction de S_{10} (a) Après focalisation (b) ω_ϕ après la reconstruction*

connaissant la position des polygones reconstruits ainsi que les positions successivement occupées par la caméra, il est possible d'affirmer que ce segment n'a jamais pu être observé, les faces du polyèdre ne permettant pas son observation. Afin de vérifier l'hypothèse, un mouvement du capteur est donc réalisé. Comme nous l'avons indiqué dans le paragraphe précédent, la caméra se focalise sur S_7 et effectue un mouvement de rotation autour de ce segment. Pendant ce mouvement, généré par asservissement visuel, les observateurs recherchent un contour en mouvement à la position attendue du segment dans l'image (voir Figure 3.29).

Le segment découvert est ensuite reconstruit (voir Figure 3.30) et inséré dans la carte de l'environnement (notons que contrairement aux segments S_3, S_5 et S_8, S_{10} a une longueur qui permet sa reconstruction en utilisant la méthode présentée au chapitre précédent, de fait, et afin de disposé d'une estimation optimale de ces paramètres, une reconstruction est réalisée). On peut finalement noter l'absence d'un segment reliant S_{10} et S_1. L'hypothèse de sa présence a été faite avec une probabilité de 83% lors de l'ajout de S_{10}, mais sa présence n'a pu être vérifiée en raison de sa position par rapport aux autres polygones et en raison des contraintes mécaniques du manipulateur empêchant sont observation par la caméra.

Signalons que 2400 images environ ont été acquises et traitées à raison d'une image toutes les 120 ms pour reconstruire cette scène. Compte tenu des calculs réalisés hors ligne, il faut donc environ 5 minutes pour reconstruire une scène de ce type.

D'autres scènes ont été traitées en utilisant la même méthode : la scène "Cylindre et polygones" présenté dans le paragraphe précédent (qui ne présente pas de difficultés particulières), ou la scène "Pyramide" (voir Figure 3.31).



(a) (b) (c)

FIG. 3.31 – *Scène Pyramide (a) image initiale (b) image intermédiaire (c) Résultat final*

Cependant, la réussite d'une telle stratégie n'est pas toujours assurée. Ainsi, si on considère le cas de la pyramide présenté sur la Figure 3.32 (contrairement au cas précédent (Figure 3.31), la caméra observe initialement la base de la pyramide), la reconstruction incrémentale de la scène permet de reconstruire cette base, mais à aucun moment les autres arêtes de la pyramide n'ont été observées. Conformément à la stratégie décrite, aucune hypothèse sur la présence de ces arêtes n'a été émise. Pour assurer la reconstruction complète de cet objet, nous devons donc utiliser une autre stratégie[7]. C'est le rôle de l'exploration globale qui sera décrite dans le prochain chapitre.

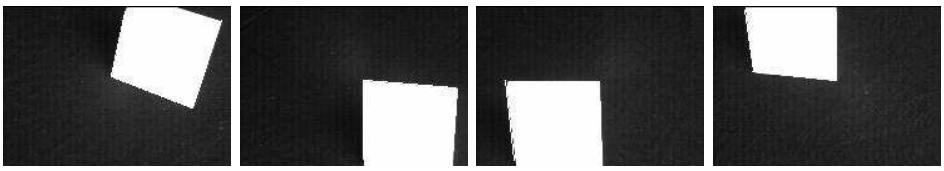


FIG. 3.32 – *Scène "Pyramide 2" : reconstruction de la base, les autres arêtes ne sont jamais observées*

3.4 Conclusion

Dans ce chapitre, nous avons tout d'abord présenté un premier algorithme, simple mais efficace, qui permet de reconstruire de facon incrémentale toutes les primitives qui ont été observées par la caméra. Cet algorithme repose sur le fait que les projections dans l'image des primitives qui nous intéressent sont des segments. Tous les segments observés dans l'image font l'objet d'une phase de reconstruction (à savoir reconnaissance et reconstruction de la primitive 3D associée). L'algorithme permet d'assurer que chaque primitive sera reconstruite une fois et une seule. On obtient ainsi une modélisation 3D d'une partie de la scène.

Cette modélisation reste cependant une modélisation de bas niveau et est généralement incomplète. Concernant les segments, nous souhaitions passer à une représentation en terme de segments 3D, jonctions 3D, polygones. La méthode que nous avons développée pour parvenir à cet objectif vient donc se greffer sur l'algorithme de reconstruction incrémentale et repose sur des techniques de prédiction / vérification d'hypothèses. Du fait des

7. Notons que d'autres hypothèses pourraient également être ajoutées comme par exemple : "une jonction a une forte probabilité d'être triple".

incertitudes dans les mesures et les observations, nous avons utilisé une approche probabiliste. La génération d'une hypothèse se fait en utilisant le modèle 3D courant, mais aussi en se basant sur un certain nombre d'*a priori* très généraux sur les caractéristiques d'une scène polyédrique. Notons qu'en aucun cas, ces *a priori* ne s'opposent aux hypothèses qui ont été initialement réalisées sur le type de scènes que nous avons décidé d'étudier (à savoir aucune connaissance préalable sur le nombre et la localisation des primitives et objets de la scène). Les connaissances que nous avons introduites sont codées dans des réseaux Bayesiens puisque ceux-ci se prêtent parfaitement au raisonnement et à la prise de décision en présence d'incertitude. Dans notre cas, ces réseaux permettent d'émettre des hypothèses sur l'existence et la localisation de nouveaux objets, puis de proposer l'exécution d'une action conduisant à vérifier ou à infirmer cette hypothèse, et enfin, en fonction du résultat de l'étape de vérification, de compléter le modèle 3D de la scène. L'étape de vérification s'appuie à la fois sur les observations déjà réalisées sur la scène, mais aussi sur une acquisition d'information nouvelle nécessitant un déplacement du capteur (déplacement qui est alors automatiquement réalisé par asservissement visuel). Enfin, la modélisation de la scène et la création de nouveaux objets (jonctions, polygones, ...) reposent sur les informations 3D et 2D provenant de l'ensemble du processus de reconstruction déjà réalisé et sur l'apport d'informations introduit par les hypothèses validées. La Figure 3.33 récapitule le processus de reconstruction incrémentale de la scène.

L'utilisation de cette approche nous permet de disposer d'une modélisation de la scène en terme d'objets et non plus en terme de segments 3D. Cette modélisation beaucoup plus riche sert de base à l'exploration globale de la scène que nous allons maintenant décrire.

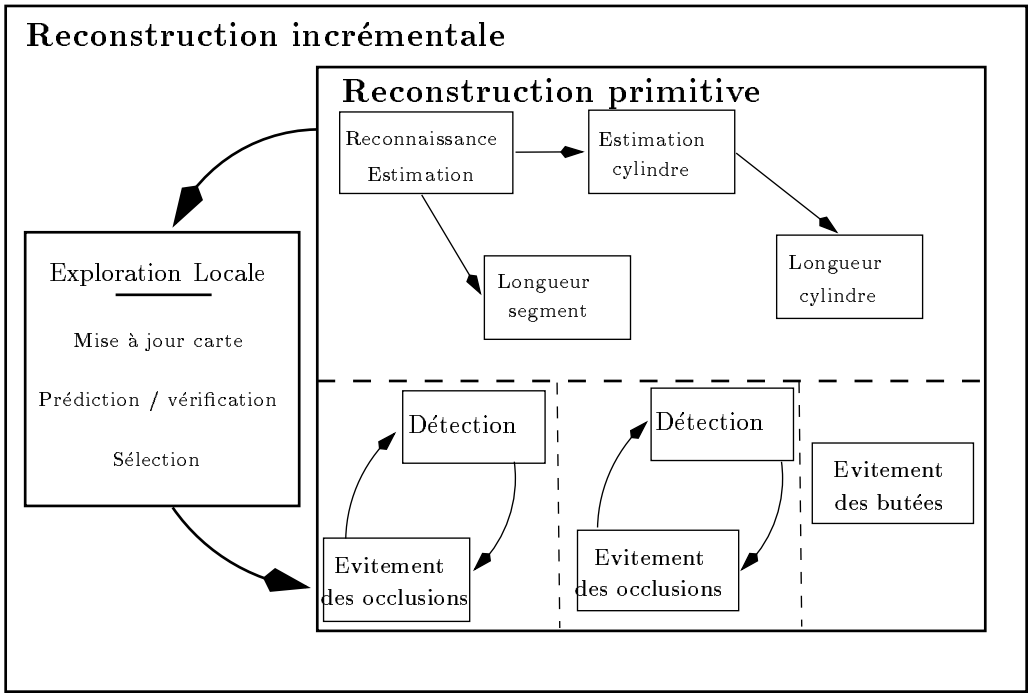


FIG. 3.33 – *Automate hiérarchique assurant la reconstruction incrémentale de la scène*

Chapitre 4

Exploration de l'environnement. Complétude de la reconstruction

> À la fin du quinzième siècle, la découverte du monde et la prise de possession géographique de la terre entière étaient devenues des événements nécessaires, inévitables. [...] Les cartographes prenaient les devants en construisant des globes planétaires sur lesquels ils dessinaient les limites supposées des terres et des mers. Ils prenaient possession de la Terre, même avant de la connaître et prescrivaient d'avance aux explorateurs le travail qu'il leur restait à faire. [...] Il devint nécessaire à la navigation de suivre les grands chemins océaniques partant des ports peu éloignés du détroit de Gibraltar pour se ramifier au Nord, à l'Ouest, au Sud, vers le Nouveau Monde et vers les Indes.
>
> Élisée Reclus, *L'homme et la terre*,
> Librairie Universelle, Paris, 1890

Nous présentons dans ce chapitre une méthode d'exploration de scènes permettant, dans la mesure du possible, d'assurer la complétude de la reconstruction. Cette exploration passe par le calcul d'un certain nombre de points de vue qui permettent l'observation de nouveaux objets ou au contraire qui permettent d'affirmer que telle ou telle partie de la scène est vide.

Après un état de l'art sur le problème, la première partie de ce chapitre décrit le principe général de l'algorithme permettant d'assurer la reconstruction complète de la scène considérée. Le problème du calcul de points de vue sera plus particulièrement abordé dans une deuxième partie. Nous décrirons ensuite la modélisation de l'espace du robot, nécessaire à la mise en œuvre de tels algorithmes. Différents résultats viendront finalement compléter et valider les algorithmes proposés. Nous discuterons ensuite de l'intérêt de produire, à un instant donné, une séquence optimisée de points de vue permettant de minimiser, soit le nombre de points de vue, soit la distance parcourue par la caméra.

4.1 Introduction

Le problème qui nous intéresse ici est celui de l'exploration d'un environnement partiellement inconnu par une caméra embarquée sur l'effecteur d'un robot. Les connaissances

dont on dispose proviennent des phases d'exploration locale précédemment réalisées. Si le problème de l'exploration de scènes se retrouve sous plusieurs formes dans la littérature, le cas de l'exploration complète d'un environnement inconnu est rarement traité. Ajoutons de plus que dans la plupart des cas le capteur utilisé est un capteur laser et non une caméra.

Une classification des différents travaux effectués dans le domaine prend en compte différents paramètres :

- le type de capteur utilisé : caméra monoculaire ou stéréoscopique, laser ;
- la nature de la tâche à réaliser qui dépend des caractéristiques du processus d'exploration. Doit-on réaliser une exploration exhaustive ou une exploration partielle de l'environnement ?
- le degré de connaissance sur l'environnement considéré, allant de l'absence totale d'information à la connaissance complète des caractéristiques de la scène (modèle CAO).

Plusieurs approches ont été proposées afin de résoudre le problème du calcul automatique de point de vue et de l'exploration de scènes (*next best view problem*). Un état de l'art a récemment été effectué par Tarabanis, Allen et Tsai [Tarabanis 95a]. Le cas où une modélisation géométrique complète de la scène est disponible est traité dans [Cowan 88], [Tarabanis 95b], [Al Chami 94], [Triggs 95]. Les tâches que les systèmes décrits par ces auteurs ont à réaliser sont relativement différentes de celle que nous nous sommes donnée. Schématiquement, elles peuvent se formuler de la manière suivante : "*étant donné un objet, où positionner la caméra pour observer telle ou telle partie de cet objet ?* ". Le problème revient donc à choisir un point de vue (position et orientation de la caméra) à partir duquel une primitive géométrique, une surface ou un objet est observé dans les meilleures conditions.

Cowan et Kovesi [Cowan 88] ont proposé de sélectionner automatiquement les points de vue permettant d'observer une ou plusieurs surfaces d'un objet. Ils définissent une série de contraintes sur les paramètres de la caméra qui permettent la réalisation de la tâche visuelle. Les auteurs explicitent quatre de ces contraintes qui permettent chacune de définir une région de l'espace où placer la caméra, l'intersection de ces régions définissant alors un ensemble de points de vue acceptables. Les quatre contraintes analysées portent sur la résolution, la netteté des informations acquises, l'étendue du champ de vision, ainsi que la visibilité des surfaces considérées. L'extraction de certaines caractéristiques d'une surface nécessite une résolution minimum permettant leur visualisation dans l'image ; la *contrainte de résolution* fixe donc cette résolution minimum qui définit une distance maximale entre la surface de l'objet observé et la position de la caméra. Pour le réglage de la profondeur de champ, la *contrainte de netteté* fournit une distance minimale entre la caméra et les surfaces, distance minimale qui permet d'obtenir une netteté satisfaisante pour tous les points de la surface. Toutes les surfaces doivent être contenues dans le champ de vision de la caméra. Cette *contrainte sur le champ de vision* fournit, pour chaque orientation de la caméra, une distance minimale entre la position de la caméra et les surfaces. Les

points de vue qui se situent au-delà de cette distance vérifient la contrainte sur le champ de vision. Enfin, toute surface observée doit être complètement observable par le capteur, ce qui est représenté par la *contrainte de visibilité*. Pour une surface donnée, on calcule les régions de l'espace des positions pour lesquelles un autre objet réalise une occlusion de cette surface. L'ensemble des points de vue satisfaisant la contrainte de visibilité est le complémentaire de l'union de ces régions. L'intersection des régions ainsi définies permet de définir les points de vue satisfaisant l'ensemble des contraintes. La méthode employée traite les contraintes nécessaires à la réalisation de la tâche visuelle de manière indépendante et, de fait, l'intersection de l'ensemble de ces contraintes ne s'effectue qu'*a posteriori* au niveau géométrique. L'existence d'un lieu géométrique assurant simultanémenent l'ensemble de ces contraintes n'est donc pas assurée.

Le problème traité par Tarabanis et Tsai dans [Tarabanis 91], [Tarabanis 95a] rejoint celui abordé précédemment : la détermination des points de vue permettant de satisfaire les contraintes imposées par l'observation d'une ou plusieurs caractéristiques particulières d'un objet dont le modèle est *a priori* connu. Les contraintes visuelles prises en considération sont proches de celles définies par Cowan et Kovesi, c'est-à-dire la visibilité, le champ de vision, la netteté et la résolution. Mais le système optique est plus général et permet d'obtenir un point de vue généralisé dans un espace à 8 dimensions (à savoir cinq paramètres géométriques : les trois de degré de liberté en position du capteur et l'orientation de celui-ci en *pan* et *tilt*, ainsi que des paramètres optiques : la distance du plan image au point nodal, la distance focale et l'ouverture). Chacune des contraintes imposées au système est modélisée par une relation analytique équivalente ($g_i \geq 0, i = 1, 2, 3, 4$) qui est satisfaite dans un domaine de valeurs admissibles sur l'espace des paramètres. Les domaines obtenus pour chaque contrainte doivent être combinés pour déterminer les valeurs des paramètres qui satisfont toutes les contraintes simultanément. Cette intersection se ramène à la recherche d'un point de vue globalement admissible. Une fonction d'objectif à maximiser permet de caractériser la qualité de la solution calculée en se basant sur les mesures fournies par les poids g_i liées aux contraintes. Une importante valeur indique que la contrainte est satisfaite confortablement, une faible valeur indique une satisfaction faible de la contrainte, alors que les valeurs négatives correspondent aux solutions inadmissibles. À partir d'un point initial, un point de vue généralisé globalement admissible et localement optimal est généré en optimisant la fonction d'objectif par une méthode d'optimisation non linéaire contrainte. L'intérêt de ces travaux par rapport à ceux de Cowan et Kovesi réside principalement dans la réunion de l'ensemble des contraintes au sein d'une unique fonction à optimiser et dans l'importance plus ou moins grande accordée à certaines contraintes.

La méthode proposée par Al Chami et Laugier dans [Al Chami 94] consiste à déterminer automatiquement les positions d'une caméra embarquée sur un robot afin de réaliser un ensemble de tâches qui seront utilisées par la suite à des fins de manipulation. Comme pour la méthode précédente, cette approche consiste à rechercher un positionnement de la caméra qui minimise une fonction d'énergie dont les termes représentent les contraintes imposées par la tâche et par l'environnement. Cependant, celui-ci n'est que partiellement connu (imprécision des positions ou des modèles d'objets). Les contraintes qui sont de type optique, géométrique ou mécanique, permettent ici encore de définir des expressions

analytiques dont les valeurs dépendent du degré de satisfaction de la contrainte représentée. La contrainte optique est basée sur la qualité de l'image en vue d'obtenir le minimum de flou. Les contraintes géométriques correspondent à des contraintes sur le champ de vision et des contraintes de non-occlusion (un taux d'occlusion est obtenu au moyen du calcul d'une distance minimale de pénétration entre un objet et l'axe optique). L'accessibilité est une contrainte binaire précisant si le capteur peut atteindre la position spécifiée. Toutes les fonctions définies par ces contraintes sont réunies au sein d'une *fonction de coût* qui doit être minimale pour une situation idéale. Chaque composante de la fonction de coût est pondérée et les poids associés sont hiérarchisés de façon à accorder plus de priorité à certaines contraintes. La méthode étant très dépendante du point initial choisi, les auteurs ont utilisé une technique de "recuit simulé" pour permettre d'obtenir un optimum global de la fonction considérée. L'intérêt principal de ces travaux réside dans la définition d'une fonction de coût et des priorités accordées aux différentes contraintes sous la forme de poids hiérarchisés. La méthode d'optimisation employée, c'est-à-dire le "recuit simulé", permet, en théorie, d'obtenir le meilleur résultat mais sa mise en œuvre est cependant très coûteuse en temps de calcul. Cette méthode est reprise par Triggs et Laugier dans [Triggs 95] qui ont proposé d'optimiser la fonction d'énergie à l'aide d'une méthode de recherche probabiliste.

Cependant, contrairement au cas qui nous intéresse, les approches précédentes ont pour objectif la détermination d'**un** seul point de vue permettant d'observer une partie d'un objet, et non pas de déterminer une trajectoire (séquence de points de vue) permettant l'observation de l'objet considéré ou de la scène dans son ensemble.

Les travaux de Tarbox et Gottschlich [Tarbox 95] tentent de répondre à ce problème. Leur objectif est de planifier la trajectoire d'un capteur laser pour réaliser l'inspection complète d'objets complexes dont le modèle CAO est disponible *a priori*. Le modèle créé à partir des données acquises par le capteur est ensuite comparé avec le modèle CAO. Trois algorithmes réalisant la planification de cette trajectoire sont proposés. Les deux premiers reposent sur la recherche du point de vue qui maximise le nombre de points observés par le capteur. Le processus est réitéré jusqu'à ce que l'ensemble des points aient été observés. Le calcul de points de vue se fait par optimisation d'une fonction d'objectif qui intègre, outre les occlusions, des contraintes propres au système de capteur considéré. Dans ces algorithmes, le point de vue retenu est celui qui maximise le nombre de points de l'objet observé de manière efficace (efficacité qui dépend entre autre de l'angle entre le laser et la surface). La différence entre les deux algorithmes réside dans la formulation de cette fonction d'objectif. Ces deux algorithmes sont donc des algorithmes de"recherche en profondeur d'abord", le choix d'un point de vue fait à l'étape t n'est plus jamais remis en cause. Le troisième algorithme propose une méthode permettant de réajuster les plans de manière à optimiser les performances du système (en l'occurrence à minimiser le nombre de points de vue nécessaire à une observation complète de l'objet). Cet algorithme repose sur le constat suivant : il n'est pas possible, ou plutôt il n'est pas réaliste, de déterminer le plan optimal (le problème appartient en effet à la classe des problèmes NP-complets). Par contre, il est possible, sous certaines conditions, de déterminer si un plan composé de n points de vue existe et de déterminer, par une série d'ajustements locaux, ce plan. Si un tel plan existe, le problème est en partie résolu (il conviendra cependant de regarder si un plan de longueur $n-1$ existe). Dans le cas contraire, il est nécessaire de chercher un plan

de longueur $n+1$. Malheureusement, et contrairement aux deux algorithmes précédents, les auteurs affirment qu'une méthode d'optimisation reposant sur une approche de type descente de gradient n'est pas utilisable pour cet algorithme (à cause du grand nombre de minima locaux de la fonction à minimiser) et un recuit simulé doit donc être utilisé. L'utilisation de cette méthode d'optimisation stochastique a pour conséquence une lenteur d'exécution, mais les plans générés sont beaucoup plus courts.

Une autre classe de méthodes se rapportant à la planification de points de vue repose sur l'utilisation des graphes d'aspect (un débat sur l'utilisation des graphes d'aspect en vision est proposé dans [Faugeras 92b]). Dans un graphe d'aspect, chaque nœud représente un aspect de l'objet considéré et les arcs connectent deux aspects voisins de l'objet. Un aspect peut être défini comme un ensemble de points de vue à partir desquels les parties visibles de l'objet sont qualitativement les mêmes. Le graphe d'aspect est en fait une segmentation de l'espace des points de vue possibles en un petit nombre d'aspects distincts. Un algorithme de planification doit ensuite être mis en œuvre pour déterminer l'ensemble des points de vue nécessaires à l'observation complète de l'objet. De telles approches ont été proposées par exemple par Hutchinson et Kak [Hutchinson 89]. Il reste que le coût calculatoire nécessaire à la construction de ces graphes d'aspect est prohibitif et leur utilisation apparaît donc difficilement envisageable de manière réaliste.

Le problème de l'**exploration autonome** est posé dans le cas où la caméra évolue dans un environnement totalement ou partiellement inconnu. Quelques auteurs ont précédemment traité ce problème sous des angles différents [Connoly 85], [Wixson 94b], [Kutulakos 94a], [Whaite 94] ou encore [Moutarlier 91].

Connoly [Connoly 85] est le premier à s'être posé le problème de l'observation d'une scène inconnue. Il propose de reconstruire un modèle complet de la scène en utilisant un capteur laser à partir d'un ensemble de points de vue. La scène reconstruite est modélisée par une structure de données en *octree*. Chaque nœud de cet arbre peut avoir la valeur "observé" (vide ou occupé) ou "non vu". Les nœuds de type "non vu" correspondent à des volumes de l'espace qui n'ont pas encore été observés depuis l'ensemble des points de vue occupés précédemment par la caméra et les nœuds de type "vu et occupé" correspondent à un objet. Le premier algorithme décrit est l'*algorithme du planétarium*. Le capteur se déplace sur une sphère régulièrement échantillonnée circonscrite à la scène. À partir de chaque point de la sphère et en ne tenant compte que des nœuds occupés ou non vus de l'arbre, la zone de l'image correspondant aux nœuds non vus est déterminée. Le point de vue permettant de dévoiler la zone non visible la plus importante est choisi comme prochain point de vue. Notons que la recherche se fait de manière exhaustive parmi l'ensemble des points de vue possibles. La trajectoire permettant d'obtenir un modèle complet de la scène est déterminée en calculant successivement plusieurs points de vue et en mettant à jour le modèle de la scène acquis depuis chacun d'entre eux. Le second algorithme proposé, l'*algorithme normal*, est très similaire au premier mais tire partie de la structure interne de l'arbre pour optimiser la recherche de la position du prochain point de vue. Il s'agit donc, là aussi, d'algorithme en profondeur d'abord, et à aucun moment le choix d'un point de vue, une fois effectué, ne sera remis en question. Le problème principal de la première méthode est le coût en temps de calcul qu'il nécessite puisqu'il faut estimer les zones non visibles à partir de *chaque* point de vue échantillonné sur la sphère. Le second

algorithme est plus rapide mais il ne traite pas les occlusions aussi efficacement. Enfin, les deux algorithmes permettent de calculer la position du capteur mais pas son orientation (qui est supposée fixée, par hypothèse, vers le centre de la scène). Il reste que l'algorithme du planétarium est à la base de presque toutes les méthodes proposées pour résoudre le problème. Il faut reconnaitre que, si la structure de données choisie pour représenter la scène, le critère de qualité d'un nouveau point de vue, ou la méthode d'optimisation choisie peuvent changer, fondamentalement quand la scène est *a priori* totalement inconnue, le schéma de base proposé dans cet algorithme est presque indispensable.

Maver et Bajcsy [Maver 93] ont défini une stratégie pour acquérir des informations 3D dans un environnement inconnu. Elles utilisent les occlusions pour déterminer les différents points de vue nécessaires pour acquérir les informations 3D des parties cachées de l'environnement. La méthode définie par Maver et Bajcsy est très liée au type de capteur utilisé. Ce système de vision active composé d'une caméra et d'un laser permet d'obtenir directement l'information 3D, mais est très rigide, ce qui limite l'exploration de certaines régions.

Wixson décrit des stratégies de perception pour la recherche d'un objet connu dans une zone encombrée [Wixson 94b]. L'objectif initial de Wixson est différent de celui des auteurs précédents puisqu'il s'agit de la recherche d'un objet et non pas de la reconstruction de l'environnement. Cependant, certains des algorithmes retenus par Wixson passent par une approche reconstructioniste et, dans ce cas, la recherche d'un objet dans l'environnement est équivalente à l'exploration et à la reconstruction de celui-ci. Déterminer qu'un objet n'est pas dans l'environnement doit en effet passer par l'exploration exhaustive de la scène considérée. Wixson propose et compare différentes stratégies pour réaliser cette exploration. Dans cette optique, il se place volontairement dans des situations simples et considère un monde 2D dans lequel sont placés un certain nombre d'obstacles. Concernant le capteur, il considère un capteur 1D pouvant se déplacer sur un cercle situé autour de la sphère. L'évaluation de l'efficacité d'une méthode par rapport à une autre repose sur une mesure de l'angle visuel total ($\Sigma\alpha_i$) ainsi que sur une mesure de la somme des angles de rotation autour de la scène ($\Sigma\theta_{t,t+1}$) (voir Figure 4.1). Wixson propose quatre types de stratégies différents dont deux reposent sur une approche reconstructioniste. Les deux premiers types de stratégie sont fondés sur la construction d'un modèle de la scène et reposent sur un algorithme très inspiré de l'algorithme du *planétarium* de Connoly. Deux variantes sont proposées : la première repose sur la recherche du point de vue offrant le maximum de zones découvertes (voir Figure 4.2), la deuxième favorisant le point de vue qui maximise le bénéfice déterminé en effectuant un rapport gain/coût, le gain étant représenté par la partie inexplorée qui deviendra visible, et le coût par l'angle de rotation $\theta_{t,t+1}$ entre deux positions successives.

La troisième méthode est plus "brutale" puisqu'elle consiste simplement à faire tourner la caméra d'un angle $\Delta\theta$ fixé *a priori*. La dernière méthode enfin est basée sur l'utilisation des arêtes d'occlusions suivant un algorithme proche de celui developpé par Maver et Bajcsy [Maver 93]. L'intérêt principal de la méthode proposée est l'évaluation des coûts et des bénéfices des opérations de perceptions effectuées. Malheureusement, l'auteur se place dans un contexte 2D, ce qui simplifie les stratégies utilisées (la dernière approche semble en effet peu adaptée à des scènes 3D peu structurées). Enfin, les résultats obtenus ne sont

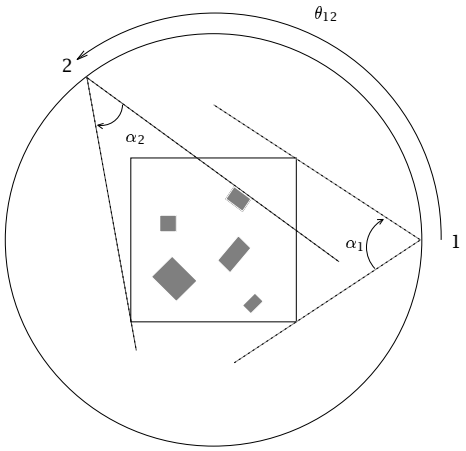


FIG. 4.1 – *Type de scène considérée et mesures utilisées pour l'évaluation des algorithmes de Wixson* [Wixson 94b]

que des simulations.

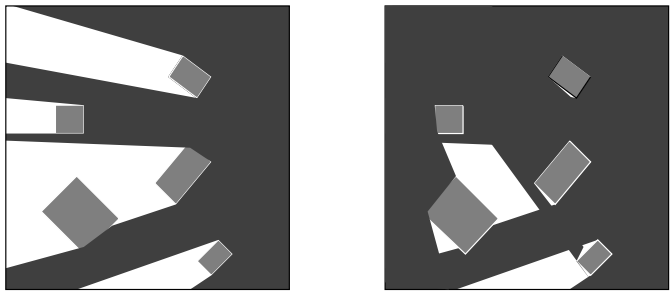


FIG. 4.2 – *Zones observées depuis deux points de vue successifs*

Des travaux très intéressants ont été menés par K. Kutulakos sur l'exploration d'un objet en vue de sa reconstruction exhaustive [Kutulakos 94b]. La méthode de reconstruction n'est pas ici ce qui nous intéresse le plus, soulignons juste qu'une méthode permettant d'obtenir une modélisation 3D de la surface de l'objet peut être réalisée à partir de l'observation dans la séquence d'images des contours d'occultation [Kutulakos 94a]. Kutulakos propose des stratégies de déplacement d'un capteur de manière à obtenir une modélisation 3D de la surface de l'objet considéré tout en assurant une reconstruction prouvée complète de celui-ci. L'objectif est donc de déterminer une trajectoire sans collision permettant l'observation et la reconstruction de tous les points situés à la surface de l'objet considéré (qui est supposé avoir un volume fini et borné par des surfaces fermées). Le capteur considéré peut être soit un système de laser (*range sensor*), soit une caméra. Afin d'explorer la surface de l'objet, Kutulakos propose une stratégie itérative qui calcule une séquence de mouvements élémentaires de façon à faire augmenter incrémentalement le nombre de points visibles à la surface de l'objet. Ces mouvements élémentaires sont construits afin d'offrir les garanties suivantes : chacun de ces mouvements ne sera executé qu'un nombre

fini de fois et, quand le processus d'exploration se terminera, la surface de l'objet sera complètement observée. La nature de ces mouvements élémentaires dépend des possibilités de mouvement du capteur ainsi que de la nature de celui-ci. De même, la nature de ces mouvements caractérise le type d'objet observable. Si l'on considère un laser et que le capteur est capable de se déplacer très près de l'objet, alors les auteurs montrent qu'un objet quelconque peut être exploré et observé. Dans le cas d'une caméra, celle-ci est capable de reconstruire la forme de l'objet à partir de la projection de limbes (contours d'occlusion) dans l'image. Les mouvements élémentaires sont des mouvements situés dans le plan défini par la normale aux limbes en un point considéré et par la position du centre optique de la caméra. Dans ce cas, le capteur n'a pas nécessairement besoin de se déplacer près de l'objet. Cependant, l'exploration ne peut s'effectuer que sur des objets non-concaves ou sur les régions non-concaves d'un objet quelconque.

Ces travaux ne tiennent pas compte du bruit associé aux observations acquises par le ou les capteurs. Des travaux intégrant cette incertitude dans le processus de perception ont cependant été menés. Whaite et Ferrie [Whaite 94] proposent une méthode permettant la création d'un modèle tridimensionnel d'un environnement inconnu en utilisant les observations provenant d'un capteur laser (*laser range finder*). Afin de minimiser l'incertitude associée à la description paramétrique de la scène (modélisée par des superquadriques), un nombre de points de vue important est nécessaire. Ceux-ci sont déterminés en utilisant un processus de perception-action en boucle fermée reposant sur la minimisation de l'incertitude associée au modèle. D'autres auteurs tiennent compte de l'incertitude associée aux données qui sont souvent modélisées par des fonctions aléatoires [DurrantWhyte 88], [Cameron 90], [Hager 91].

Le cas de la recherche d'information dans le contexte de la robotique mobile à été traité par [Moutarlier 91] et [Grandjean 91]. Dans ce cas, le processus de reconstruction incrémentale ou de reconnaissance nécessite la mise en œuvre de processus de fusion de données à cause de l'imprécision associée aux données acquises par les capteurs (télémétriques [Moutarlier 91] ou télémétriques/stéréoscopiques [Grandjean 91]).

Dans notre cas, le problème à résoudre consiste à choisir les positions et orientations de la caméra permettant d'obtenir le maximum d'informations possibles sur les zones encore inconnues de la scène. L'ensemble des points de vue choisis doit conduire à une modélisation complète de la scène. Le nombre et la position des objets constituant la scène étant inconnus, les approches reposant sur des modèles connus *a priori* ne sont donc pas applicables. Par ailleurs, l'approche de Kutulakos, si elle est adaptée à des volumes fermés non polyédriques, ne semble pas adaptée pour des scènes composées de plusieurs objets dont la plupart sont polyédriques.

Comme l'a montré Wixson [Wixson 94b], différentes stratégies peuvent cependant être employées : des stratégies basées sur le contour des objets déjà reconstruits (comme le font Maver et Bajcsy [Maver 93]), des stratégies reposant sur des mouvements réguliers du capteur autour de la scène, ou encore des stratégies basées sur une mesure du gain potentiel apporté par une nouvelle position. Le premier type de stratégies reposant sur les contours d'occultation des objets ne nous apparaît pas très adapté à ce niveau d'exploration globale. Des solutions reposant sur de telles stratégies ont été proposées dans le chapitre précédent avec pour objectif d'augmenter la perception locale de l'environne-

ment. Les stratégies reposant sur des mouvements réguliers du capteurs sont sans doute utilisables. Il faut cependant remarquer que l'utilisation d'une telle stratégie ne peut pas assurer la complétude de l'exploration. Il faut en effet considérer que le capteur évolue ici dans un espace à six dimensions et qu'un unique mouvement de la caméra (ou même quelques mouvements différents) ne sera vraisemblablement pas à même de permettre la couverture complète d'une scène complexe inconnue. Il reste donc à considérer le troisième type de stratégies, inspiré initialement de l'algorithme du *planétarium* de Connoly [Connoly 85]. Comme nous l'avons déja indiqué, cet algorithme se propose de découvrir le point de vue qui apportera un maximum d'informations à l'explorateur. La difficulté est de trouver une mesure adéquate de cet apport d'informations.

Nous proposons donc une méthode permettant de calculer de nouveaux point de vue en utilisant uniquement la connaissance courante sur la géométrie spatiale de la scène et une représentation des zones de l'espace déjà observées. L'algorithme proposé, combiné avec les stratégies locales de perception proposées au chapitre précédent, assurent la reconstruction aussi complète que possible de la scène en un temps borné. Il repose sur l'optimisation d'une fonction de coût modélisant un certain nombre de contraintes. C'est un algorithme de recherche en "profondeur d'abord" qui ne remet pas en cause le point de vue calculé. Une solution optimale serait de déterminer globalement une trajectoire minimisant le nombre de points de vue et/ou la distance parcourue par la caméra. Nous verrons cependant que cet objectif n'est pas réaliste sous les hypothèses que nous nous donnons. Nous proposerons cependant une méthode permettant d'obtenir, si ce n'est la trajectoire optimale, du moins une trajectoire satisfaisante.

4.2 Stratégies d'exploration globale

4.2.1 Le problème et les hypothèses

Le problème peut être formulé de façon relativement simple. Considérons une scène quelconque composée d'un ensemble $\mathcal{O}$ d'objets inconnus. À l'issue de l'étape d'exploration locale présentée dans le chapitre précédent, un sous ensemble $o \subseteq \mathcal{O}$ des objets de la scène a été reconstruit (notons que cet ensemble o d'objets peut être vide si aucun objet n'était visible depuis la position initiale de la caméra). Le problème consiste alors à déterminer la position de la caméra qui permettra d'obtenir une information complémentaire sur la structure de la scène. Ici, le mot *information* est employé au sens large, car il peut aussi bien signifier la découverte de nouveaux objets que la certitude qu'une zone donnée de la scène est vide. Dès qu'un objet est trouvé dans l'image, le processus d'exploration locale permettant sa reconstruction est à nouveau activé.

L'acquisition d'informations n'est cependant pas suffisante en soit, il convient de plus de s'assurer que la trajectoire que réalisera le robot aura une distance bornée (*i.e.*, que le processus d'exploration terminera) et que la reconstruction de la scène sera complète à l'issue de cette exploration (*i.e.*, tous les objets de la scène auront été reconstruits).

4.2.2 Définition de stratégies d'exploration

À la fin d'un processus d'exploration locale, une reconstruction des objets observés pendant cette phase et depuis le début du processus a été effectuée. Un modèle 3D de la scène visible depuis la position courante de la caméra est donc disponible. Cette position courante ϕ_t est caractérisée par un vecteur à 6 paramètres $\phi_t = (x, y, z, \alpha, \beta, \gamma)$ qui représente la position et l'orientation de la caméra dans l'espace 3D.

Connaissant l'ensemble des positions $\mathcal{T}_0^t = \{\phi_0, \phi_1, \ldots, \phi_t\}$ de la caméra depuis le début du processus de reconstruction, il est possible de maintenir à jour une carte des zones déjà connues et des zones restant à explorer. Les connaissances sur la scène sont de deux types :

- les objets déjà reconstruits (notés $\mathcal{O}(\mathcal{T}_0^t)$).
- l'espace libre déjà repéré (espace qui sera noté $\mathcal{V}(\mathcal{T}_0^t)$).

La zone de l'espace observée depuis une position ϕ est notée $\mathcal{V}(\phi)$ et représente l'espace associé à une perception [1]. Connaissant la trajectoire $\mathcal{T}_0^t$, il est bien sûr possible de déterminer la zone de l'espace observée $\mathcal{V}(\mathcal{T}_0^t)$ depuis le début de la reconstruction :

$$\mathcal{V}(\mathcal{T}_0^t) = \bigcup_{i=0}^{t} \mathcal{V}(\phi_i) \tag{4.1}$$

Cet espace libre peut être aussi construit de manière incrémentale en considérant uniquement le supplément d'informations apporté par le dernier point de vue considéré :

$$\begin{cases} \mathcal{V}(\mathcal{T}_0^t) &= \mathcal{V}(\mathcal{T}_0^{t-1}) \cup \mathcal{V}(\phi_t) \\ \mathcal{V}(\mathcal{T}_0^0) &= \mathcal{V}(\phi_0) \end{cases} \tag{4.2}$$

Les stratégies que nous proposons pour l'exploration globale de la scène reposent principalement sur la seule information disponible à ce stade : la zone inexplorée. Cette zone inexplorée à l'instant t peut être déduite des connaissances courantes dont on dispose sur l'environnement à cet instant. En effet, la zone inexplorée, que l'on notera $\mathcal{I}(\mathcal{T}_0^t)$, peut être définie comme le complémentaire des zones représentant les objets et l'espace libre :

$$\mathcal{I}(\mathcal{T}_0^t) = \overline{\mathcal{V}(\mathcal{T}_0^t) \cup \mathcal{O}(\mathcal{T}_0^t)} \tag{4.3}$$

La Figure 4.3 montre une représentation planaire de ces différents ensembles pour une trajectoire composée de deux points de vue ϕ_0 et ϕ_1 (nous avons choisi ici une représentation planaire pour des raisons de clarté, il va de soi que dans le cadre de notre application, toutes ces zones sont des volumes).

Une stratégie simple consisterait à déterminer la position ϕ_{t+1} qui maximise la zone de l'espace passant de l'état inconnu à l'état connu. Cependant, si cette stratégie permet

1. Le problème de la représentation et du calcul de ces zones connues ou inconnues n'est pas un problème trivial et sera abordé au cours de ce chapitre (dans le paragraphe 4.3.3). À ce niveau on supposera qu'une telle représentation existe.

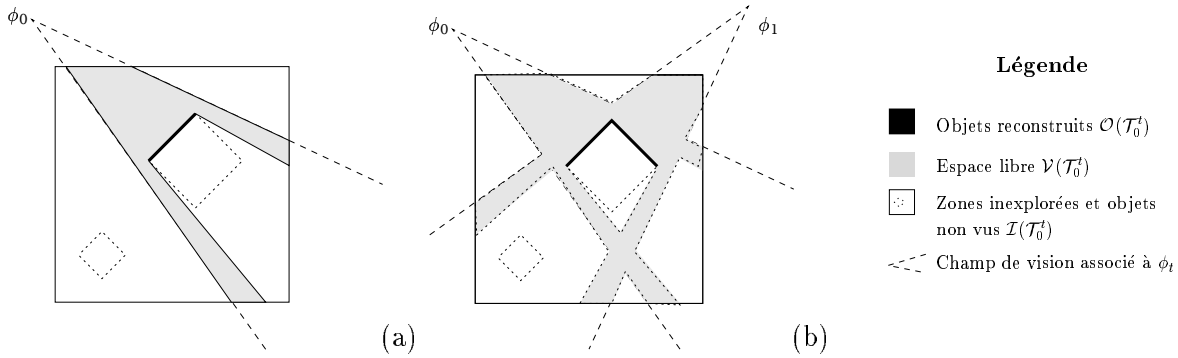


FIG. 4.3 – *Représentation des objets $\mathcal{O}(\mathcal{T}_0^t)$ et de l'espace libre $\mathcal{V}(\mathcal{T}_0^t)$ associés à la trajectoire (a) $\mathcal{T}_0^0 = \{\phi_0\}$ et (b) $\mathcal{T}_0^1 = \{\phi_0, \phi_1\}$*

dans l'absolu de répondre au problème posé, elle ne tient pas compte d'un certain nombre d'autres facteurs :

- la trajectoire finale risque d'être relativement inefficace vis-à-vis de la distance parcourue par le robot ;
- une telle stratégie ne tient pas compte des problèmes physiques comme l'accessiblité à la fois cinématique (y a t-il une configuration articulaire du robot qui permette d'atteindre cette position ?) et géométrique (la position que l'on souhaite atteindre est-elle libre de tout objet ?).

Il convient donc de rajouter à cette stratégie initiale des contraintes supplémentaires représentant au mieux ces aspects annexes du processus d'exploration.

Ce calcul de point de vue s'effectue, comme pour [Al Chami 94] et [Tarabanis 95b], en minimisant une fonction d'objectif qui doit respecter les contraintes imposées au système et représenter le niveau de satisfaction apporté par un point de vue. Si la formulation de la fonction d'objectif est importante, il convient aussi de se définir un espace de recherche adéquat et de choisir une méthode d'optimisation à la fois rapide et efficace. L'application d'une telle méthode dans le cadre d'un environnement inconnu a en effet pour conséquence une charge de calcul relativement lourde si l'on ne restreint pas la tâche en réduisant l'espace de recherche.

Pour résumer, la reconstruction globale de la scène repose sur les étapes suivantes :

1. reconstruction des primitives observables si elles existent (exploration locale) ;
2. calcul des zones libres, des zones comportant des objets et des zones inexplorées ;
3. calcul d'un nouveau point de vue apportant de l'information supplémentaire.

Le processus est itéré jusqu'à la fin de la reconstruction complète de la scène (voir Section 4.4). Notre approche repose donc sur l'alternance de phases de reconstruction des primitives observées et de phases d'exploration ou recherche d'informations supplémentaires. Les différents aspects de ce problème de calcul de points de vue sont à présent abordés.

4.3 Calcul de points de vue

La stratégie de calcul de points de vue que nous proposons repose sur la définition et l'optimisation d'une fonction d'objectif représentant au mieux la tâche que nous souhaitons effectuer. Nous allons donc dans un premier temps définir cette fonction d'objectif avant de présenter une méthode d'optimisation efficace vis-à-vis du problème à traiter.

4.3.1 Définition d'une fonction de coût

La recherche du point de vue s'effectue en minimisant une fonction d'objectif $\mathcal{F}(\phi)$ qui représente la qualité du point de vue ϕ. Nous avons retenu quatre critères, associés au point de vue, qui seront intégrés dans cette fonction :

- le gain apporté par la nouvelle position. L'objectif de l'exploration est l'acquisition d'informations supplémentaires sur la scène. Il est donc impératif de modéliser cet apport potentiel d'information dans la fonction d'objectif.
- le coût de déplacement vers la nouvelle position. Un tel critère se justifie par le fait que nous souhaitons avoir une trajectoire de longueur minimale en évitant des mouvements d'amplitude importante.
- des contraintes mécaniques dûes au robot justifieront l'introduction d'une contrainte éloignant le robot de ses butées articulaires ;
- enfin, les connaissances déjà acquises sur la scène permettent de définir un critère binaire représentant l'accessibilité d'un point de vue.

À chacun de ces critères, on associe une mesure à valeur dans $[0,1] \cup \infty$. Une valeur proche de 0 signifie que la satisfaction apportée par le point de vue vis-à-vis du critère associé est maximale, une valeur proche de 1 signifie que la position n'a aucun intéret vis-à-vis du critère associé (satisfaction minimale), enfin, une valeur à l'infini signifie que la position est automatiquement à rejeter.

4.3.1.1 Gain apporté par une nouvelle position

Le gain apporté par une nouvelle position ϕ_{t+1} est défini par le volume de la zone inexplorée qui apparaît dans le cône de vision de la caméra quand celle-ci se déplace de ϕ_t à ϕ_{t+1}. La zone découverte depuis cette position correspond à la zone $\mathcal{G}(\{\phi_{t+1}\})$ définie par (voir Figure 4.4) :

$$\mathcal{G}(\phi_{t+1}) = \mathcal{V}(\phi_{t+1}) - \mathcal{V}(\phi_{t+1}) \cap \mathcal{V}(\mathcal{T}_0^t) \tag{4.4}$$

La mesure du gain associé à la position ϕ_{t+1} peut donc être définie à partir du rapport entre le volume de la zone découverte $\mathcal{G}(\phi_{t+1})$ et le volume de la zone réellement observée $\mathcal{V}(\phi_{t+1})$:

$$g(\phi_{t+1}) = \begin{cases} \infty & \text{si } \mathcal{G}(\phi_{t+1}) = \emptyset \\ 1 - \frac{volume(\mathcal{G}(\phi_{t+1}))}{volume(\mathcal{V}(\phi_{t+1}))} & \text{sinon} \end{cases} \tag{4.5}$$

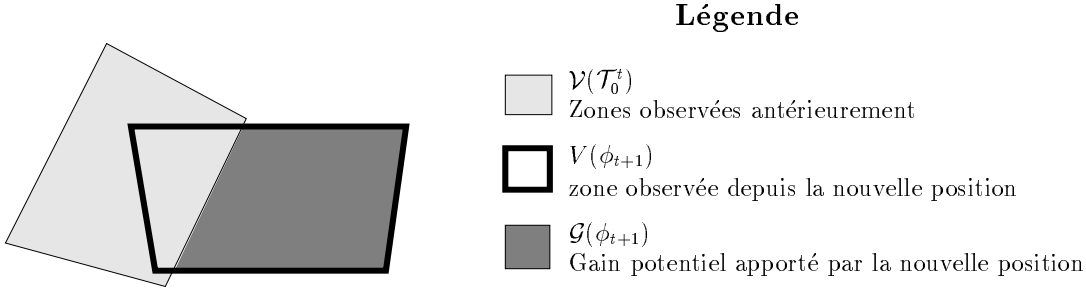


FIG. 4.4 – *Calcul de la fonction de gain : mesure du gain (projection 2D)*

L'utilisation de ce rapport permet de normaliser entre 0 et 1 la valeur du gain attendu. Ainsi si la zone observée par la caméra est entièrement nouvelle, la valeur de la mesure du gain $g(\phi_{t+1})$ sera égale à 0 et donc optimale. La zone $\mathcal{G}(\phi_t)$ correspond en fait à une zone potentiellement découverte. En effet, dans le cas où un nouvel objet apparaît dans le champ de vision, les occlusions dûes à cet objet font que la zone réellement observée est de taille inférieure ($\mathcal{G}'(\phi_t) \subseteq \mathcal{G}(\phi_t)$). Cependant, l'important dans notre cas est qu'il y ait un apport d'information se caractérisant soit par la découverte d'un nouvel objet, soit par celle d'une zone vide.

Enfin notons que si l'apport en informations d'une nouvelle position est nul (c'est-à-dire $\mathcal{G}(\phi_{t+1}) = \emptyset$) la mesure de la fonction g associée à cette position est fixée à l'infini. Il convient en effet de rejeter totalement ces positions si l'on souhaite assurer la terminaison du processus d'exploration. Si une telle contrainte n'est pas imposée sur le choix de ϕ_{t+1}, l'optimisation de la fonction d'objectif finale pourrait amener à choisir une telle position.

4.3.1.2 Coût de déplacement entre deux positions

Idéalement, la mesure du coût du déplacement entre deux positions ϕ_t et ϕ_{t+1} devrait être définie en calculant la distance entre ϕ_t et ϕ_{t+1}. Traditionnellement, les positions sont définies sur l'ensemble noté SE_3, isomorphe à $\mathbb{R}^3 \times SO_3$ où SO_3 est le groupe des rotations. Aucune distance ne peut être définie sur SE_3 [Le Borgne 87], il n'est donc pas possible de considérer une distance comme fonction de coût. Nous définissons alors la fonction de coût par la relation suivante (voir Figure 4.5) :

$$\mathcal{C}(\phi_t, \phi_{t+1}) = \frac{1}{N_{ddl}} \sum_{i=1}^{N_{ddl}} \beta_i \frac{\mid q_{i_t} - q_{i_{t+1}} \mid}{\mid Q_{i_{Max}} - Q_{i_{Min}} \mid} \tag{4.6}$$

où

- N_{ddl} représente le nombre de degrés de liberté du robot ;
- q_i représente la coordonnée articulaire de l'axe i (ϕ est représentée par $(q_0, q_1, \ldots, q_{N_{ddl}})$) ;
- $\mid Q_{i_{Max}} - Q_{i_{Min}} \mid$ est la distance entre les deux butées articulaires sur l'axe i ;

– β_i est un poids qui permet de relativiser l'importance d'un déplacement sur tel ou tel axe du robot ($\beta_i \in [0,1]$).

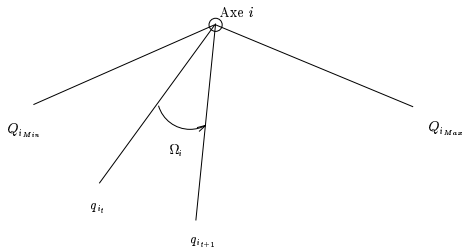


FIG. 4.5 – *Calcul de la fonction de coût sur un axe de rotation du robot*

Un telle formulation présente un certain nombre d'avantages. Elle répond à deux (mais deux seulement) des critères définissant une distance (à savoir $\mathcal{C}(\phi,\phi) = 0$ et $\mathcal{C}(\phi_1,\phi_2) = \mathcal{C}(\phi_2,\phi_1)$). De ces propriétés, on peut immédiatement déduire qu'une absence de déplacement se traduit par un coût de déplacement nul, ce qui correspond au rôle régulateur que l'on souhaite faire jouer à ce critère (*i.e.*, diminuer l'amplitude des déplacements). De plus, le coefficient β_i joue un rôle important en permettant de favoriser les déplacements sur certains axes du robot. Ainsi, on peut par exemple favoriser des mouvements de rotation ou des mouvements de translation en fonction des caractéristiques mécaniques du manipulateur.

4.3.1.3 Contraintes supplémentaires

Des contraintes supplémentaires sont associées aux deux critères principaux que l'on vient de définir. En effet, certaines positions, ou la proximité de certaines positions, sont indésirables quant à l'exécution du processus d'exploration ou du processus de reconstruction. Ces contraintes tendent à :

– éviter les positions inatteignables pour la caméra en raison des butées articulaires du robot ou des caractéristiques géométriques de la scène déjà reconstruite. À cet effet, une pénalité infinie est donnée à une position si celle-ci n'est pas accessible (occupée par un objet ou inatteignable) ou inconnue :

$$\mathcal{A}(\phi) = \begin{cases} 0 & \text{si } \phi \text{ est accessible} \\ \infty & \text{sinon} \end{cases} \tag{4.7}$$

– éviter des positions trop proches des butées articulaires afin que les mouvemenents nécessaires à la reconstruction d'une primitive apparaissant depuis ϕ_{t+1} soient possibles :

$$\mathcal{B}(\phi) = \frac{1}{N_{ddl}} \sum_{i=1}^{N_{ddl}} \frac{4(q_i - \frac{Q_{i_{Max}} + Q_{i_{Min}}}{2})^2}{(Q_{i_{Max}} - Q_{i_{Min}})^2} \tag{4.8}$$

La mesure associée à cette contrainte sera optimale (égale à 0) si les coordonnées articulaires q_i se trouvent au milieu de l'extension de chacun des axes du robot.

Une telle contrainte est motivée par le fait que les tâches d'asservissement visuel nécessaires à la reconstruction peuvent mal se comporter à proximité de ces butées. Des stratégies permettant de s'éloigner des butées articulaires pendant ces phases ont été développées pour pallier à ce problème ([4][5], voir aussi l'annexe D). Cependant, la réussite de telles stratégies n'est pas garanti si l'asservissement est déclenché trop près d'une butée.

4.3.1.4 Fonction de coût globale

La fonction d'énergie $\mathcal{F}(\phi_{t+1})$ reflétant le degré de satisfaction apporté par le point de vue ϕ_{t+1} est définie comme une somme pondérée des différentes mesures proposées :

$$\mathcal{F}(\phi_{t+1}) = \mathcal{A}(\phi) + \alpha_1 g(\phi_{t+1}) + \alpha_2 \mathcal{C}(\phi_t, \phi_{t+1}) + \alpha_3 \mathcal{B}(\phi_{t+1}) \tag{4.9}$$

La solution recherchée au problème d'optimisation ainsi défini sera donc donnée par la position ϕ_{t+1} telle que :

$$\phi_{t+1} = \arg\min_{\phi \in \mathcal{S}} \mathcal{F}(\phi) \tag{4.10}$$

où $\mathcal{S}$ est l'espace de recherche possible dont la détermination est décrite dans le prochain paragraphe.

La détermination des coefficients α_i dans un problème d'optimisation de ce type est un problème non trivial. Des méthodes permettant de calculer automatiquement ces coefficients ont été proposées (par exemple dans [Gennert 88]). Pour notre part, nous nous sommes contentés de choisir ces coefficients de manière empirique. Cependant, leur valeur fixe l'ordre de priorité associé à chacun des critères et détermine ainsi le comportement du processus d'optimisation. L'accessibilité est bien sûr prioritaire (le caractère "binaire" de son résultat rend inutile une quelconque pondération). La découverte de nouvelles zones à explorer étant notre objectif principal, nous avons choisi $\alpha_1 > \alpha_2 > \alpha_3$ avec $\alpha_1 + \alpha_2 + \alpha_3 = 1$. Plus précisément, nous avons fixé : $\alpha_1 = 0.6$, $\alpha_2 = 0.3$ et $\alpha_3 = 0.1$ (des valeurs similaires ne modifient pas outre mesure la position des points de vue trouvés).

4.3.2 Optimisation de la fonction de coût

4.3.2.1 Espace de recherche des solutions

Chaque position ϕ peut *a priori* être solution de ce problème d'optimisation. Même limité à l'espace opérationnel du robot, considérer cet espace comme l'espace de recherche des solutions $\mathcal{S}$ n'est pas réaliste et ce pour différentes raisons.

Le critère principal sur lequel repose le choix de la position ϕ_{t+1} est l'apport d'un maximum d'information supplémentaire. Utiliser l'ensemble de l'espace opérationnel aurait donc pour conséquence de favoriser un éloignement maximal par rapport à la scène. En effet, plus la caméra s'éloigne de la scène et plus l'apport d'information est potentiellement

important. Une telle augmentation de la distance entre le point de fixation et la position de la caméra pose cependant des problèmes optiques et géométriques importants :

- **problèmes optiques**. La profondeur de champ étant fixe sur la caméra, un éloignement trop important entrainerait un flou dans l'image. D'autre part, la résolution du capteur étant également fixe, les détails de la scène ne seraient plus perceptibles. Des méthodes permettant de résoudre ces problèmes optiques liés à la profondeur de champ ont été proposées par Cowan et Kovesi [Cowan 88], Tarabanis et al. [Tarabanis 95a] et Al Chami et Laugier [Al Chami 94]. Comme nous l'avons vu au début de ce chapitre, les solutions proposées reposent généralement soit sur la recherche des lieux géométriques assurant une adéquation entre les contraintes optiques et la position de la caméra [Cowan 88][Al Chami 94], soit sur le réglage explicite des paramètres optiques de la caméra quand celui-ci est possible [Tarabanis 95a]. Ce type de solution repose cependant sur une connaissance *a priori* de la position des objets dans la scène, connaissance dont nous ne disposons pas ici. Il reste que de telles contraintes pourraient être introduites dans la fonction de coût en ajoutant des *a priori* sur la scène.

- **problèmes géométriques** : ces problèmes sont en fait étroitement reliés aux précédents. La qualité de l'estimation des paramètres d'une primitive et par conséquence la qualité de la reconstruction de la scène dans son ensemble, dépend de l'éloignement relatif entre la caméra et la primitive considérée. On peut en effet facilement se convaincre que cette distance influe sur la qualité de l'estimation du mouvement apparent de la primitive dans l'image, qui influe à son tour sur la qualité de l'estimation de la primitive considérée. Là encore, des contraintes pourraient être rajoutées à la fonction à optimiser afin de limiter l'éloignement entre la scène et la caméra.

Déplacement *sur* une demi-sphère circonscrite à la scène : De manière à contraindre le problème, nous autorisons la caméra à se déplacer sur la surface d'une demi-sphère circonscrite à la scène. De fait, nous supposons connu un volume englobant celle-ci (voir Figure 4.6). La position de la caméra peut alors être décrite par un vecteur à 5 paramètres $(\theta, \varphi, \alpha, \beta, \gamma)$ où θ et φ représentent la latitude et la longitude de la caméra sur la sphère et où α, β et γ représentent l'orientation de la caméra.

Contraindre la position recherchée de la caméra à appartenir à cette demi-sphère permet de prendre en compte les contraintes optiques et géométriques. D'autres solutions sont envisageables et ont été testées, comme par exemple limiter les positions désirées entre deux demi-sphères circonscrites à la scène (les résultats sont similaires mais les temps de calcul sont plus longs), ou les contraindre à appartenir à un plan ($Z = cte$). Dans ce dernier cas, les temps de calculs sont légèrement réduits mais l'exploration est moins efficace.

Déplacement *à l'intérieur* d'une demi-sphère circonscrite à la scène : Il apparaît cependant réducteur de limiter les mouvements de la caméra à un positionnement sur cette demi-sphère. En effet, il est souhaitable que la caméra puisse se déplacer à l'intérieur de cette demi-sphère pour se focaliser si besoin sur certaines zones particulières de la scène.

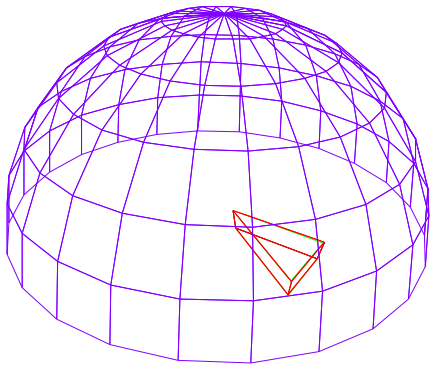


FIG. 4.6 – *Espace de recherche des solutions : demi-sphère positionnée autour de la scène*

La position de la caméra est alors décrite par un vecteur à 6 paramètres $(x, y, z, \alpha, \beta, \gamma)$ où x, y et z représentent la position de la caméra dans la sphère et où α, β et γ représentent l'orientation de la caméra.

Cependant, cet espace ne peut être considéré dès le départ comme l'espace des positions possibles de la caméra. En effet, il faut s'assurer que les positions possibles sont accessibles. Les connaissances sur la scène étant initialement limitées à un volume englobant la scène plus l'espace libre calculé après la première phase d'exploration locale, les mouvements de la caméra sont tout d'abord limités à ce volume. Dès lors que les connaissances sur la scène augmentent, les zones libres d'obstacles augmentent également et la caméra peut s'engager dans ces espaces libres. La contrainte $\mathcal{A}(\phi)$ (équation (4.7)) permet de gérer cette accessibilité. L'espace de recherche des points de vue acceptables est donc limité aux zones de l'espace connu comme étant libres d'obstacles (et de fait, cet espace augmente avec la connaissance des espaces libres), et à la portion de l'espace située sur la demi-sphère définie dans le paragraphe précedent [2].

4.3.2.2 Technique d'optimisation

Pour minimiser la fonction de coût $\mathcal{F}(\phi)$ et trouver la solution ϕ_{t+1} de l'équation (4.10) plusieurs techniques sont envisageables :

- d'une part des techniques déterministes (ou itératives) de type descente de gradient, qui ont l'inconvénient de produire des résultats correspondant à des minima locaux de la fonction de coût mais qui ont l'avantage d'être peu coûteux en temps de calcul ;
- et d'autre part, des techniques stochastiques de type recuit simulé [Kirkpatrick 83], [Aarts 87], qui permettent théoriquement de converger vers l'optimum global de la fonction de coût, mais qui sont très coûteux en temps de calcul.

2. Pour gérer les obstacles, nous n'avons pris en compte dans la fonction d'accessibilité que la position de la caméra. Dans l'absolu, il faudrait considérer son volume mais également la géométrie du manipulateur sur lequel elle est montée.

Dans notre cas, nous avons choisi d'utiliser une méthode déterministe classique de type gradients conjugués avec un pas de descente à plusieurs niveaux : nous utilisons tout d'abord des incréments importants afin de déterminer la région de l'espace des paramètres où l'optimum de la fonction $\mathcal{F}(\phi)$ est probablement situé. Puis, nous itérons le processus depuis cette nouvelle position avec un incrément plus faible. Ce comportement est décrit sur la Figure 4.7 dans le cas planaire (seuls deux paramètres sont modifiés). Un premier calcul d'optimum est réalisé en considérant un pas de discrétisation Δ_1 menant de D à O_1. Puis une deuxième minimisation de la fonction est effectuée en prenant O_1 comme position initiale et en considérant un pas de discrétisation $\Delta_2 < \Delta_1$. Cela permet de trouver un nouveau minimum local O_2. Le processus est itéré tant que le pas de discrétisation Δ est supérieur à un seuil donné[3].

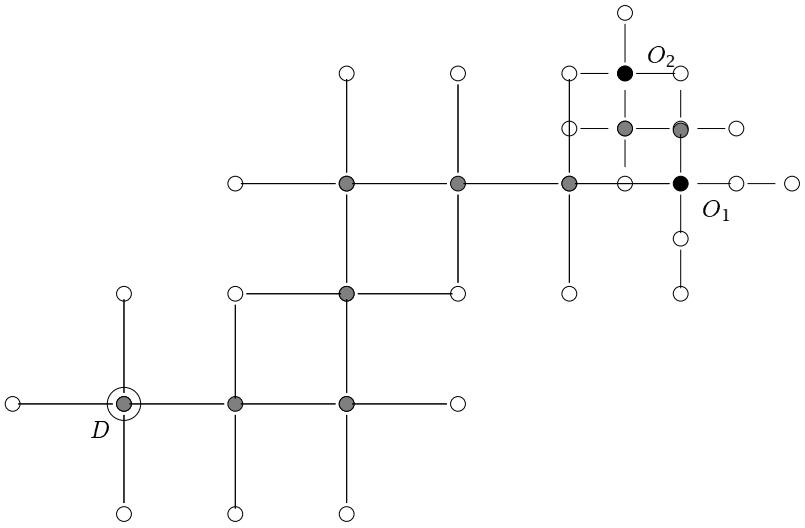


FIG. 4.7 – *Descente de gradient "multi-échelle"*

Avec cette méthode, nous ne pouvons assurer que la convergence s'effectue vers le minimum global de la fonction. Cependant, le gain en temps de calcul est très important et les expériences ont montré qu'un optimum correct est toujours atteint en un faible nombre d'itérations. Comme le montre la Figure 4.8, les fonctions de coût manipulées sont relativement continues et le nombre de minima locaux est très limité (sur ces figures, une zone sombre correspond à une faible valeur de la fonction d'objectif, et donc à un "bon point de vue", et une zone claire correspond à une forte valeur de la fonction de coût). De plus, l'intérêt de trouver un minimum global de la fonction d'énergie ne nous a pas paru fondamental dans la mesure où une position apportant un complément d'informations est trouvée. Finalement, le processus final de focalisation sur les zones résiduelles, qui sera présenté dans le paragraphe 4.5, permet de pallier à un éventuel échec du processus d'optimisation (*i.e.* lorsque celui-ci fournit un minimum local n'apportant pas d'information supplémentaire).

3. Des techniques similaires ont été utilisées pour la mise en correspondance d'images par corrélation [Jain 81]

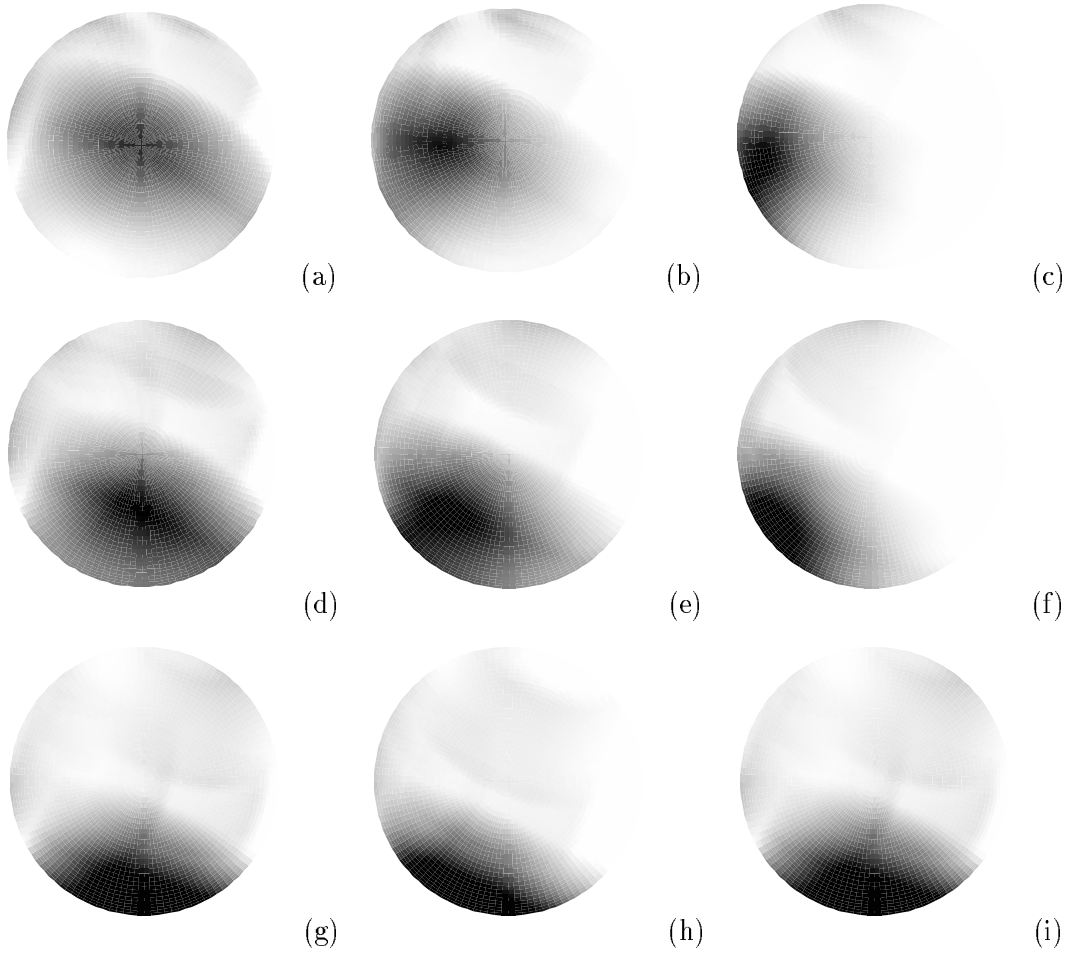


FIG. 4.8 – *Variation de la fonction d'objectif en fonction de la position en latitude et longitude de la caméra sur la sphère pour la scène de la Figure 4.12 : cas de l'espace de recherche à 5 paramètres* $(\theta, \varphi, \alpha, \beta, \gamma)$. *Valeur de la fonction d'objectif en fonction de* φ *et* θ *avec une variation* $\Delta\alpha$ *et* $\Delta\beta$ *des paramètres* α *et* β *autour de leur valeur initiale (donnéee en (e)) : (a)* $\Delta\alpha = -10$ *et* $\Delta\beta = -10$ *(b)* $\Delta\alpha = 0$ *et* $\Delta\beta = -10$ *(c)* $\Delta\alpha = 10$ *et* $\Delta\beta = -10$ *(d)* $\Delta\alpha = -10$ *et* $\Delta\beta = 0$ *(e)* $\Delta\alpha = 0$ *et* $\Delta\beta = 0$ *(f)* $\Delta\alpha = 10$ *et* $\Delta\beta = 0$ *(g)* $\Delta\alpha = -10$ *et* $\Delta\beta = 10$ *(h)* $\Delta\alpha = 0$ *et* $\Delta\beta = 10$ *(i)* $\Delta\alpha = 10$ *et* $\Delta\beta = 10$

4.3.3 Modélisation et mise à jour de l'environnement

Construire puis maintenir à jour une carte de la scène est une étape indispensable du processus de reconstruction et d'exploration. La carte de la scène associée aux primitives reconstruites ne pose pas de problèmes particuliers. Chaque primitive est en effet décrite par une équation et peut donc être représentée dans une base de données de manière exacte (aux erreurs d'estimation près).

Il est également important de pouvoir disposer d'une représentation des zones libres

et des zones inexplorées. Cette représentation doit avoir les propriétés suivantes :

- pouvoir représenter le plus fidèlement possible les différentes zones de la scène : objets $\mathcal{O}$, espace libre $\mathcal{V}$ et zones inconnues $\mathcal{I}$.
- pouvoir être mise à jour rapidement ;
- servir de base au calcul des points de vue : le gain potentiel d'une position $g(\phi_{t+1})$ est en effet calculé à partir de cette représentation.

Comme nous l'avons vu dans le chapitre précédent, le processus d'exploration incrémentale qui a été proposé permet d'assurer que toutes les primitives qui ont été observées ont été reconstruites. Sans les phases de retour arrière qu'il comporte, des primitives pourraient ne pas être reconstruites, ce qui impliquerait à tort que les zones dans lesquelles elles sont situées seraient marquées comme ayant été observées. C'est donc cette propriété qui permet la construction *a posteriori* des zones réellement libres.

Obtenir une représentation exacte de ces zones sous la forme d'équations semble être une tâche relativement délicate (même si elle est sans doute réalisable dans le cas qui nous intéresse puisque les primitives considérées sont relativement simples). Si cette représentation présente l'avantage d'être la plus fidèle, sa mise à jour est complexe et le calcul du gain potentiel d'un point de vue qui résultera de l'intersection des zones inconnues avec un cône ne nous paraît pas être une tâche aisément implémentable.

La méthode qui serait sans doute la plus adaptée serait de représenter les zones inconnues à l'aide d'une triangulation de l'espace, en l'occurrence une triangulation de Delaunay. Cette triangulation est couramment utilisée en vision par ordinateur et en robotique [Boissonnat 90], [Buffa 93]. Cependant, la création d'une telle triangulation est coûteuse et si des algorithmes de triangulation incrémentale sont apparus récemment dans la littérature, une triangulation de Delaunay 3D dynamique et contrainte (c'est-à-dire telle que les arêtes des objets soient des segments de la triangulation) n'est pas à notre connaissance disponible. Il reste que l'utilisation d'une triangulation de Delaunay pour une application de ce genre est possible. Buffa [Buffa 93] a ainsi traité le problème de l'exploration de l'environnement d'un robot mobile en utilisant une triangulation de Delaunay dans un espace 2D.

Un autre type de représentation classiquement utilisé est l'octree [Zell 95]. Cette représentation a l'avantage d'être simple et aisément consultable. Cependant là encore, sa mise à jour est coûteuse en temps de calcul et le calcul du gain potentiel demanderait une modification complète de l'oct-tree en fonction du point de vue considéré, et ce afin de réaliser l'intersection du cône de vision avec l'oct-tree.

Dans notre cas, nous avons choisi deux représentations différentes. La première est basée sur un découpage régulier de l'espace en volumes élémentaires et la deuxième repose sur la projection des différentes zones considérées sur un plan virtuel.

Représentation volumique de l'espace : La première représentation que nous avons retenue repose sur un découpage régulier de la scène en volumes élémentaires (voxels). Les accès à l'information sont dans ce cas simplifiés, ce qui réduit les temps de calculs.

Chaque voxel de cette grille 3D est codé en fonction de la nature de la zone auquel il appartient : inexploré (non vu ou occlusion), zone accessible, objet. Initialement tous les voxels de la scène sont marqués comme appartenant à une zone inexplorée (non vu). Afin de mettre à jour cette structure de données en fonction des nouvelles informations acquises par la caméra, on utilise un algorithme de lancer de rayon [Amanatides 87]. Des rayons sont lancés depuis les voxels du fond de la scène appartenant au cône de vision de la caméra jusqu'à celle-ci (voir Figure 4.9). Si le rayon intersecte un ou plusieurs des objets de la scène, les voxels situés entre le fond de la scène et l'objet le plus proche de la caméra sont marqués comme appartenant à une zone inexplorée (occlusion), et ceux situés entre cet objet et la caméra sont marqués comme appartenant à une zone libre. L'algorithme de lancer de rayon que nous avons utilisé est l'algorithme d'Amanatides [Amanatides 87]. Cet algorithme adapté à un espace 3D partitionné a l'avantage d'être extrêmement rapide (le passage d'un voxel à un autre s'effectuant avec seulement une addition et deux comparaisons).

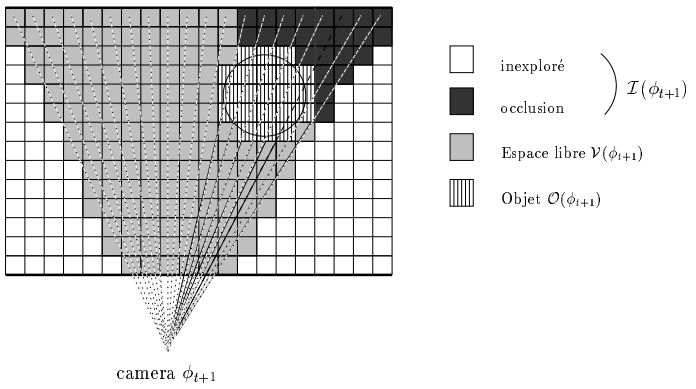


FIG. 4.9 – *Lancer de rayon pour la mise à jour de la représentation de la scène (cas d'une coupe transversale d'un cylindre)*

L'avantage majeur de cette représentation est sa facilité de mise à jour, puisque la fusion des informations venant de plusieurs points de vue se fait grâce à de simples opérations logiques. D'autre part, le degré de précision que l'on souhaite atteindre dépend directement de la taille de chaque voxel. La contrepartie à cette simplicité est l'espace mémoire utilisé qui peut être important. Dans notre cas, une scène d'un mètre de côté (1 m^3) avec des voxels de 5 mm de coté occupe un espace de 7.65 Mo en mémoire.

Représentation surfacique de l'espace : La deuxième représentation que nous avons utilisée repose sur la projection des différentes zones de l'espace sur un plan virtuel. Cette méthode a l'avantage d'être très simple à mettre en œuvre et permet en outre un calcul beaucoup plus rapide des nouveaux points de vue si l'on souhaite considérer dans le gain $g(\phi)$ la surface découverte (et non plus le volume découvert).

Ce plan virtuel est représenté par un *quadtree*. Dans ce cas, le *quadtree* est intéressant car il permet d'accélerer le calcul de point de vue en limitant le nombre de carreaux de surface à considérer. Pour savoir si un point de ce plan est visible depuis la caméra, on

regarde dans un premier temps s'il est dans le cône de vision, puis on lance un rayon entre ce point et la caméra ; si le rayon traverse l'un des objets reconstruits de la scène alors l'objet en question réalise une occlusion de ce point ; dans le cas contraire, le point est visible depuis la caméra.

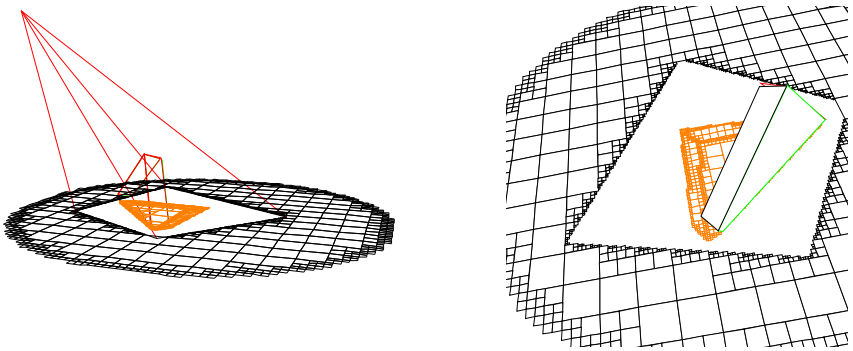


FIG. 4.10 – *Représentation surfacique de l'espace : (a) plan virtuel sur lequel est projetée la scène, cône de vision associée à la caméra et objet déjà reconstruit (b) zoom sur une partie de la Figure 4.10(a)*

Si l'on souhaite utiliser cette représentation dans la fonction de coût, le gain $g(\phi_{t+1})$, associé non plus au volume découvert mais à la surface observée sur ce plan, est à présent donné par :

$$g(\phi_{t+1}) = 1 - \frac{surface(\mathcal{G}(\phi_{t+1}))}{surface(\mathcal{V}(\phi_{t+1}))} \tag{4.11}$$

Le choix de cette représentation entraîne une perte évidente d'informations sur les zones observées par la caméra et sur les zones d'occlusions puisque l'on passe d'informations en trois dimensions à des informations en deux dimensions. Des zones entières de l'espace peuvent ainsi demeurer inexplorées. D'autre part le choix du plan virtuel sur lequel s'effectue la projection n'est pas simple. Cette représentation présente cependant les avantages suivants : la mise à jour comme le calcul du gain potentiel sont très rapides. Elle permet de plus une visualisation aisée des différentes zones (observée, inconnue, ou occlusion). Dans les faits, nous utiliserons peu la fonction $g(\phi)$ définie par l'équation (4.11) pour le calcul des points de vue. Par contre, nous utiliserons cette représentation pour la visualisation des résultats, la visualisation en termes de volumes des différentes zones étant très difficile à interpréter.

4.4 Complétude de la reconstruction

Nous disposons maintenant d'une fonction qui, à partir de la trajectoire $\mathcal{T}_0^t$ et des connaissances déjà acquises sur l'environnement, permet de calculer la position suivante

ϕ_{t+1} susceptible d'apporter un complément d'informations sur la scène. Il reste à s'assurer que la trajectoire finale permettra une reconstruction aussi complète que possible de la scène et que cette trajectoire n'a pas une longueur infinie.

La reconstruction doit théoriquement prendre fin quand la scène ne comporte plus de zones inexplorées, c'est-à-dire si à un instant t donné :

$$\mathcal{I}(\mathcal{T}_0^t) = \emptyset \tag{4.12}$$

Définir la fin du processus d'exploration de cette manière n'est pourtant pas très réaliste en pratique. En effet, il n'est pas toujours possible d'assurer une reconstruction complète de la scène : dans certains cas, des zones particulières peuvent être visibles uniquement depuis des positions inacessibles à la caméra. La reconstruction complète ne peut alors avoir lieu. Cependant, s'il n'est pas possible d'assurer cette complétude, il est possible d'assurer que la reconstruction est aussi complète que possible compte tenu des contraintes imposées au et par le système. En tenant compte de cette remarque, nous proposons donc une deuxième condition d'arrêt à l'exploration :

$$\forall \phi_{t+1}, \begin{cases} \mathcal{V}(\mathcal{T}_0^t) \cup \mathcal{V}(\{\phi_{t+1}\}) = \mathcal{V}(\mathcal{T}_0^t) \\ \\ \mathcal{O}(\mathcal{T}_0^t) \cup \mathcal{O}(\{\phi_{t+1}\}) = \mathcal{O}(\mathcal{T}_0^t) \end{cases} \tag{4.13}$$

Cette condition signifie que quel que soit le point de vue choisi parmi tous les points de vue accessibles, il n'y aura pas d'apport supplémentaire d'information. Remarquons que si l'équation (4.12) est vérifiée à l'issue de l'exploration, alors on pourra affirmer que l'exploration est complète et que toute la scène a été reconstruite.

Finalement, la fonction de coût que nous avons modélisée nous permet d'assurer, sous condition que son minimum global soit atteint, que tant qu'il reste une zone non observée, l'apport d'information en zone découverte ou en objet existera : $\mathcal{G}(\{\phi_{t+1}\}) \neq \emptyset$. Par contre, quand aucun point de vue permettant d'acquérir une quelconque information ne sera trouvé, alors on aura $(\mathcal{V}(\{\phi_{t+1}\}) \cup \mathcal{O}(\{\phi_{t+1}\})) - \left(\mathcal{V}(\mathcal{T}_0^t) \cup \mathcal{O}(\mathcal{T}_0^t)\right) = \emptyset$ et la condition donnée dans l'équation (4.13) sera vérifiée. En pratique, à cause de la méthode de minimisation déterministe retenue, il n'est cependant pas toujours possible d'assurer que l'obtention d'un point de vue n'apportant aucune information implique que l'ensemble de la scène a été observée. C'est pourquoi il est finalement nécessaire de se focaliser sur les éventuelles zones résiduelles inexplorées.

4.5 Focalisation sur des zones d'intérêt

Notre méthode de calcul de points de vue présente deux inconvénients auxquels il convient d'apporter une solution :

- d'un point de vue pratique, de petites zones résiduelles demeurent inexplorées. Nous proposons donc dans ce paragraphe un algorithme permettant de se focaliser sur ces zones, assurant ainsi une reconstruction complète de la scène.

- L'algorithme proposé ne remet jamais en cause le choix d'un point de vue obtenu. De fait, aucune optimisation globale de la trajectoire, que ce soit en ce qui concerne la distance parcourue par la caméra ou le nombre de points de vue, n'est possible. Nous proposons donc à la fin de ce chapitre une méthode qui tente d'apporter une solution partielle à ce problème.

Comme le montreront les résultats expérimentaux du prochain paragraphe, de petites zones résiduelles non observées subsistent généralement à l'issue de l'exploration en raison du critère d'arrêt fixé. Idéalement, l'exploration devrait s'achever quand 100% de la scène est observée (équation (4.12)). Cependant, il n'apparaît pas réaliste d'atteindre ce seuil limite :

- tout d'abord, il n'est pas certain que 100 % de la scène puisse être observé. La topologie de certains objets rend des zones entières inobservables. D'autre part, certaines zones sont inaccessibles à la caméra (en raison, par exemple, des butées articulaires du manipulateur), ne permettant pas l'observation de ces zones de la scène ;

- ensuite, comme on le verra dans les résultats, le gain marginal en volume observé décroît avec le nombre de points de vue. Il en découle que l'observation des dernières zones résiduelles nécessite un nombre très important de points de vue.

Pour ces différentes raisons, nous avons décidé d'arrêter l'exploration quand un sous-ensemble de l'espace observable a été réellement observé (pratiquement, on fixe un seuil situé entre 95 et 99 %). Il convient cependant alors de s'assurer que cet espace inexploré ne contient pas d'objet. Pour cela, nous ne prenons plus la scène dans sa globalité, mais nous la traitons comme un ensemble de scènes plus petites que l'on va considérer de manière indépendante et sur lesquelles est exécutée la même méthode d'exploration.

La Figure 4.11.a montre une représentation planaire d'une scène fictive. Dans ce cas, on constate qu'un petit nombre de zones résiduelles n'a pas été observé (zones marquées 1 à 4 sur la Figure 4.11.a).

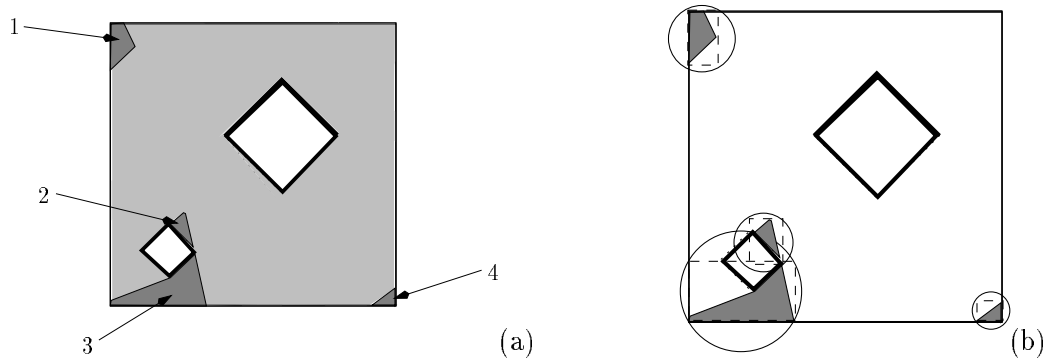


FIG. 4.11 – *Détection et focalisation sur les zones d'intérêt*

Pour se focaliser sur chacune de ces zones, nous devons dans un premier temps calculer leur volume englobant. Pour cela, nous considérons le polyèdre englobant l'ensemble des

voxels connexes non observés. L'objectif étant ici de considérer des zones de petite taille, ce volume englobant est subdivisé en plusieurs sous ensembles s'il s'avère être de volume trop important.

Comme dans le cas de l'algorithme d'exploration globale, nous considérons ensuite que la caméra peut évoluer dans une sphère située autour de ce volume englobant (voir Figure 4.11.b). Les connaissances déjà acquises sur la scène permettent de considérer comme précédemment les conditions d'accessibilité et empêchent tout contact de la caméra avec les objets déjà reconstruits ou avec les zones inexplorées de la scène.

Se focaliser ainsi sur les zones particulières de la scène présente plusieurs avantages :

- ces zones non observées ont nécessairement une petite taille par rapport à la taille initiale de la scène (c'est la raison pour laquelle elles n'ont pas été explorées, le gain n'étant pas assez fort). Désormais, la taille de la scène considérée ayant diminué, le gain devient important et le point de vue calculé exploitable.

- de plus, la taille de la scène considérée étant alors plus petite, il est possible, sans augmenter les temps de calcul des points de vue, de considérer une taille de voxel beaucoup plus faible (on peut descendre à des tailles de l'ordre du millimètre). On peut alors considérer si besoin ces parties de la scène avec un niveau de détail supérieur.

Nous verrons dans le paragraphe suivant que cette méthode s'avère très utile pour résoudre certains problèmes posés par des occlusions.

4.6 Résultats expérimentaux

4.6.1 Exploration d'une scène simple

Afin d'illustrer les différents aspects des algorithmes proposés (exploration de la scène, influence des coefficients de la fonction d'énergie, focalisation sur des zones d'intérêt), nous montrons tout d'abord divers résultats obtenus sur une scène composée d'un cylindre et d'un polygone.

La Figure 4.12 décrit le résultat de la reconstruction à l'issue de l'exploration locale. A ce stade, tous les objet ont déjà été reconstruits. Considérer une scène où tous les objet ont déjà été reconstruits en utilisant le processus d'exploration locale présente l'intérêt de faire ressortir l'influence des différents paramètres entrant en jeu dans l'exploration globale (dans le cas de scènes plus complexes, cette influence est plus difficile à percevoir). Environ 25% de la scène a déjà été observé et reconstruit. L'étape suivante consiste donc à acquérir les 75% d'informations manquantes.

4.6.1.1 Exploration de la scène et influence des paramètres de la fonction d'énergie

Nous analysons ici l'influence des coefficients α_i de l'équation (4.9) sur le parcours de la caméra. En particulier, nous mettons en évidence l'importance de la prise en compte

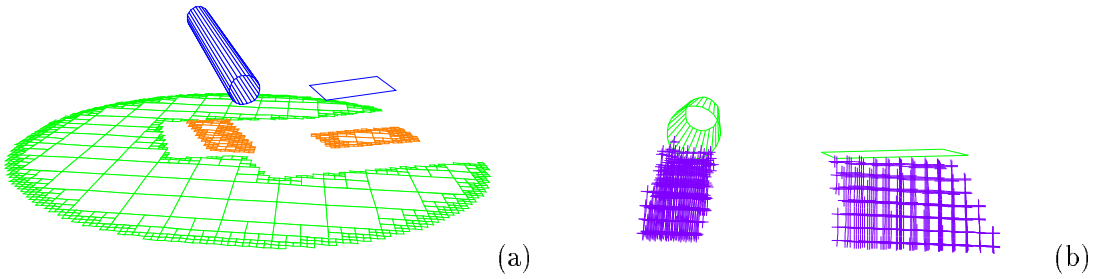


FIG. 4.12 – *Résultat de l'exploration locale de la scène considérée : (a) scène reconstruite et projection sur un plan virtuel de la zone non observée (b) vue de coté représentant la scène reconstruite et les zones en occlusion*

de la distance entre deux points de vue successifs dans la fonction d'énergie. Dans un premier temps, nous limitons les mouvements de la caméra sur la surface de la demi-sphère englobant la scène, puis nous considérons des mouvements de la caméra à l'intérieur de celle-ci.

Déplacement sur la demi-sphère englobante : la première stratégie, dont les résultats sont présentés sur la Figure 4.13.a, ne prend pas en compte la distance parcourue par la caméra et est donc principalement basée sur une maximisation de la zone découverte pour chaque point de vue (le coefficient α_2 dans l'équation (4.9) est nul). La deuxième stratégie (voir Figure 4.13.b) tient compte de cette distance et tend donc à diminuer la distance totale parcourue par la caméra. Les Figures 4.13.a et 4.13.b représentent les différentes trajectoires effectuées menant à une observation de plus de 95% de la scène. On notera que si la distance parcourue n'est pas prise en compte, la caméra effectue une trajectoire en "vol d'abeille" alors que de tels mouvements ne se produisent plus si la distance est introduite dans la fonction d'énergie. La Figure 4.14.a montre le pourcentage de volume de la scène observée après chaque nouvelle position de la caméra. Le fait que le gain marginal apporté par chaque nouvelle position de la caméra soit plus faible si l'on ne prend pas en compte la distance (contrairement à ce que l'on attendrait) resulte de l'utilisation d'un ICM, et non d'un recuit simulé, pour la minimisation de la fonction de coût. De fait, le minimum global n'a pas été atteint lors du calcul du premier point de vue et cette "erreur" se répercute sur l'ensemble de la trajectoire. La Figure 4.14.b représente la distance cumulée parcourue par la caméra pour les deux stratégies. Comme on peut s'y attendre, la distance parcourue par le robot est beaucoup plus faible si l'on prend en compte la distance dans la fonction de coût. D'autre part, on peut noter que la deuxième stratégie a tendance à favoriser les mouvements suivant les trois axes de rotation par rapport aux mouvements suivant les trois axes de translation. Ceci est dû au fait que les rotations étant moins coûteuses, elles sont moins pénalisées (les coefficients β_i dans l'équation (4.8) ont été fixés à 0.4 pour les degrés de libertés en rotation et 0.6 pour ceux en translation).

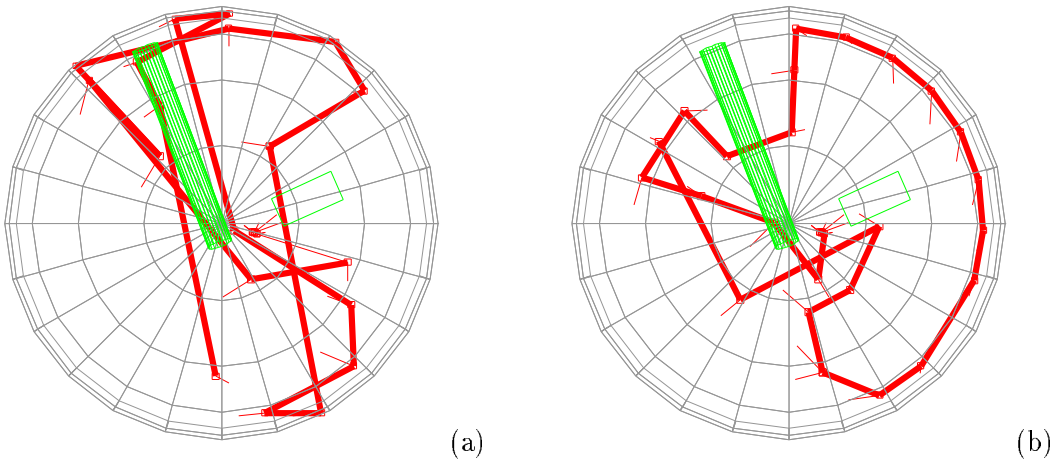


FIG. 4.13 – *Exploration globale de la scène - Déplacement de la caméra sur la sphère - (a) Trajectoire du robot sans tenir compte du coût de déplacement ($\alpha_2 = 0$) (b) en tenant compte du coût de déplacement ($\alpha_2 > 0$)*

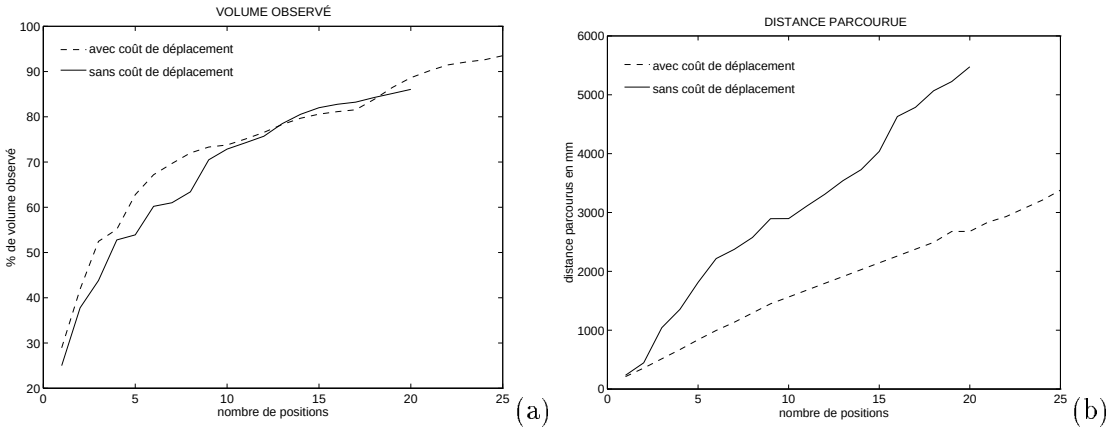


FIG. 4.14 – *(a) Pourcentage de la scène observée (b) distance parcourue par le robot en fonction du nombre de positions occupées par le robot (en mm)*

Déplacement à l'intérieur de la demi-sphère englobante : dans cette deuxième expérience, nous ne limitons plus les mouvements de la caméra à la surface de la demi-sphère, mais nous autorisons la caméra à pénétrer à l'intérieur de celle-ci (voir Figure 4.15 et Figure 4.16). Les mouvements de la caméra étant beaucoup moins contraints, la longueur de la trajectoire est beaucoup plus faible (voir Tableau 4.1).

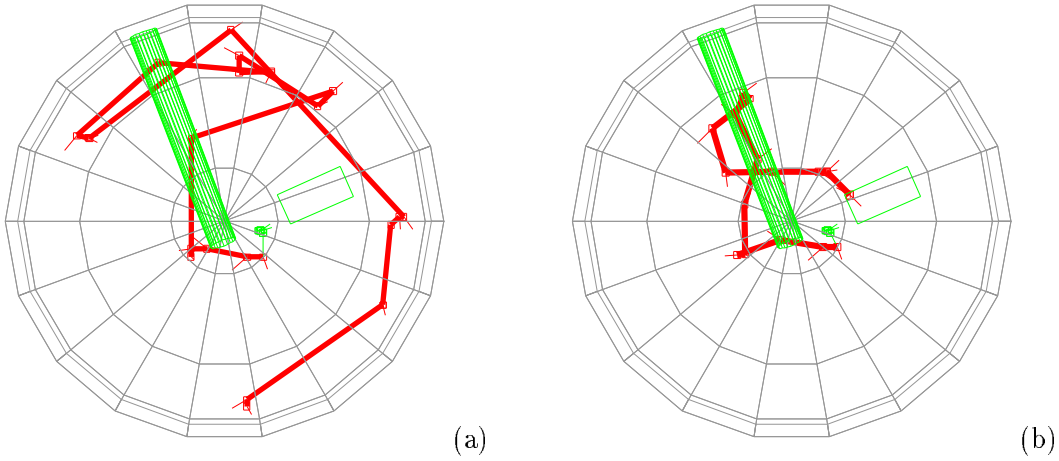


FIG. 4.15 – *Exploration globale de la scène - Déplacement de la caméra à l'intérieur de la sphère - (a) Trajectoire du robot sans tenir compte du coût de déplacement ($\alpha_2 = 0$) (b) en tenant compte du coût de déplacement ($\alpha_2 > 0$)*

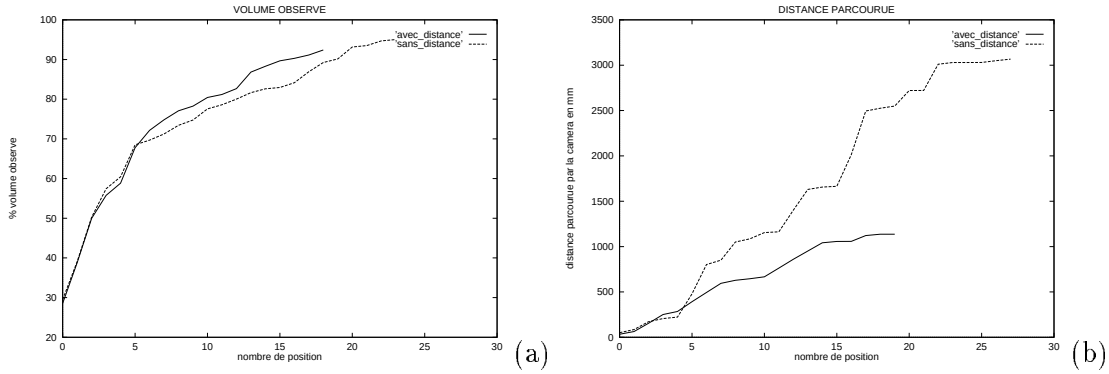


FIG. 4.16 – *(a) Pourcentage de la scène observée (b) distance parcourue par le robot en fonction du nombre de positions occupées par le robot (en mm)*

On pourrait s'attendre à ce que le nombre de points de vue augmente si on introduit la contrainte sur la distance, le mouvement de la caméra étant alors moins libre. Si cette augmentation se produit lors du déplacement sur la sphère (de 20 à 25 points de vue, voir Tableau 4.1), on constate un diminution (de 24 à 19) du nombre de points de vue lors du déplacement à l'intérieur de la sphère. Comme nous l'avons indiqué, l'introduction de cette contrainte a eu pour effet de favoriser les mouvements suivant les axes de rotation par

rapport aux mouvements suivant les axes de translation. Il s'est averé ici que l'utilisation de ces axes combinée à des déplacement à l'intérieur de la sphère a permis de diminuer le nombre total de points de vue. D'autre part, là encore, le fait d'utiliser un algorithme d'optimisation déterministe peut expliquer en partie cette diminution du nombre de points de vue.

	sans la distance		avec la distance	
déplacement	nb pts vue	distance (mm)	nb pts vue	distance (mm)
sur la sphère	20	5476	25	3379
dans la sphère	24	3066	19	1136

TAB. 4.1 – *Comparaison des différentes stratégies utilisées*

Temps de calcul d'un point de vue : Il est difficile de donner un temps de calcul d'un point de vue. En effet, celui-ci dépend fortement de la vitesse de convergence de l'algorithme de minimisation utilisé. Par contre, on peut donner le temps de calcul moyen d'un point de vue en fonction du type de représentation de l'espace choisi (surface ou volume) et du type de déplacement retenu (sur ou à l'intérieur de la sphère). Comme base de comparaison, nous considérons toujours la même scène, le volume à explorer représentant environ 1 m^3. Nous avons volontairement arreté l'exploration quand 90% de la scène a été observée. La taille des voxels était de 10 mm et celle des carreaux de surface les plus petits de 5mm. Les temps de calcul obtenus (avec une Sparc station 20) sont présentés sur le Tableau 4.2.

Représentation de l'espace	Déplacement	Temps moyen (s) pour un pt de vue	Temps maximal (s) pour un pt de vue	nombre de points de vue	Temps total (s)
surface	sur la sphère	4.2	7.6	9	37.9
surface	dans la sphère	4.5	8.9	14	62.9
volume	sur la sphère	8.1	19.9	10	81.2
volume	dans la sphère	10.6	23.9	8	84.3

TAB. 4.2 – *Temps de calcul*

Comme on peut s'y attendre, le temps de calcul d'un point de vue en considérant une représentation volumique de l'espace est plus important que si l'on considère une représentation surfacique (rapport de 1 à 2 environ). Cela est facilement explicable par la complexité des calculs de lancer de rayon mis en jeu. Par contre, le surcoût en temps de calcul engendré par le passage d'un espace de recherche de cinq à six paramètres est très faible.

4.6.1.2 Focalisation sur les zones d'intérêt

Comme on peut le constater sur la Figure 4.17, des zones résiduelles n'ont pas été explorées. Pour cette expérimentation, le critère d'arrêt du processus d'exploration a été

fixé à 95%. Moins de 5% du volume total n'a donc pas été observé. Ces zones résiduelles correspondent soit à des portions de l'espace résultant d'une occlusion (et donc situées sous le cylindre et sous le rectangle), soit à des zones situées en périphérie de la scène. Pour explorer ces zones résiduelles, nous utilisons l'algorithme de focalisation sur des zones d'intérêt. La Figure 4.17.a représente la scène reconstruite avec la trajectoire initiale d'exploration. La Figure 4.17.b montre la segmentation des zones résiduelles en quatre sous ensembles. Deux sont associés à des zones en occlusion (sous le cylindre et sous le rectangle), les deux autres sont associés à des zones résiduelles.

La Figure 4.18 montre la trajectoire de la caméra assurant une exploration plus complète de la scène (supérieure à 99%). La caméra s'est tour-à-tour focalisée sur les quatre régions sélectionnées permettant ainsi l'observation de ces différentes zones.

On constate cependant qu'une partie des zones en occlusion demeure inobservée. Ceci est imputable aux contraintes mécaniques du manipulateur. Pour assurer cette observation, le robot aurait en effet dû dépasser ses butées articulaires, ce qui lui est impossible. Il est donc généralement impossible d'affirmer que l'ensemble de la scène a été traité. La reconstruction est seulement *aussi complète que possible* compte tenu des contraintes imposées par le manipulateur.

4.6.2 Scène cylindre et polygones

Considérons maintenant une scène composée d'un cylindre et de plusieurs polygones disposés dans des plans différents. À la fin de la première étape d'exploration locale, cinq primitives ont été reconstruites (un cylindre et un polygone). La Figure 4.12 rappelle l'état de la reconstruction à ce stade.

La Figure 4.19 montre les différentes étapes nécessaires à l'exploration de la scène. Dans cette expérience, la caméra se déplace sur la sphère circonscrite à la scène. Chacune des figures décrit la scène reconstruite, la trajectoire de la caméra jusqu'à sa position courante, et une visualisation de la zone non observée. La Figure 4.19.a correspond à la position ϕ_6 de la caméra obtenue à l'issue de l'exploration locale décrite dans le chapitre précédent. Les premiers déplacements de la caméra (Figures 4.19.b et 4.19.c) permettent de diminuer la zone de la scène non encore explorée. À la position ϕ_{13} représentée Figure 4.19.d, une nouvelle primitive est détectée marquant le début d'une seconde phase d'exploration locale qui se termine à l'étape 24 (Figure 4.19.f). Les deux polygones en haut de la scène sont alors reconstruits. Une nouvelle exploration globale amène la caméra en ϕ_{25} (Figure 4.19.g) où un segment appartenant au dernière objet (une carte de téléphone) apparaît. Après une dernière exploration locale, permettant la reconstruction des quatre cotés de la carte, la caméra est en position ϕ_{30} (Figure 4.19.h). À ce stade, 99% de l'espace a été observé assurant ainsi une reconstruction complète de la scène. La Figure 4.20 montre une visualisation 3D de la scène telle qu'elle a été reconstruite, ainsi que la trajectoire de la caméra pendant l'exploration globale.

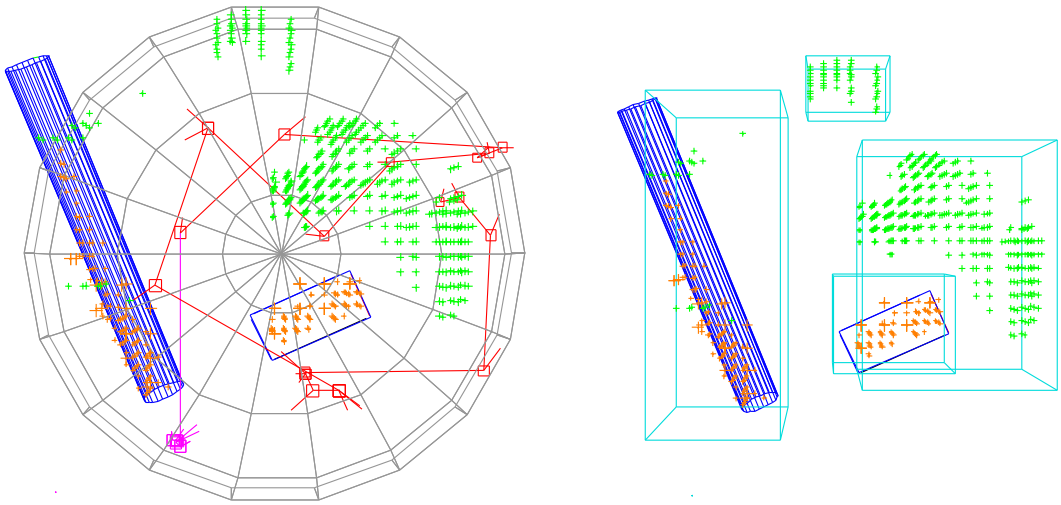


FIG. 4.17 – *Focalisation sur les zones résiduelles (a) Scène observée, trajectoire initiale d'exploration et zones résiduelles (b) segmentation de ces zones*

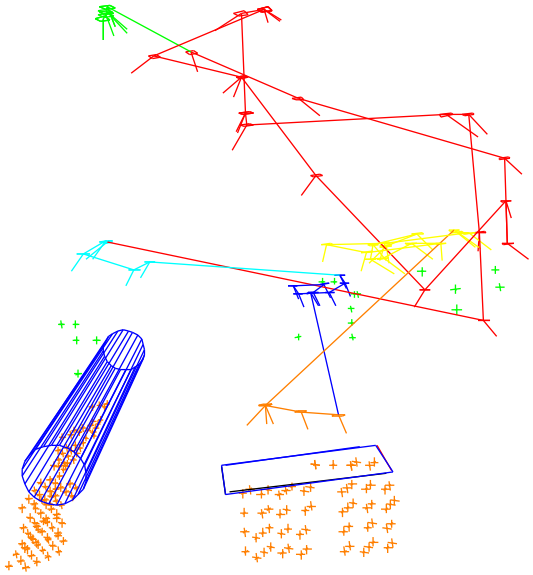


FIG. 4.18 – *Focalisation sur les zones d'intérêt : trajectoire finale permettant l'exploration complète de la scène*

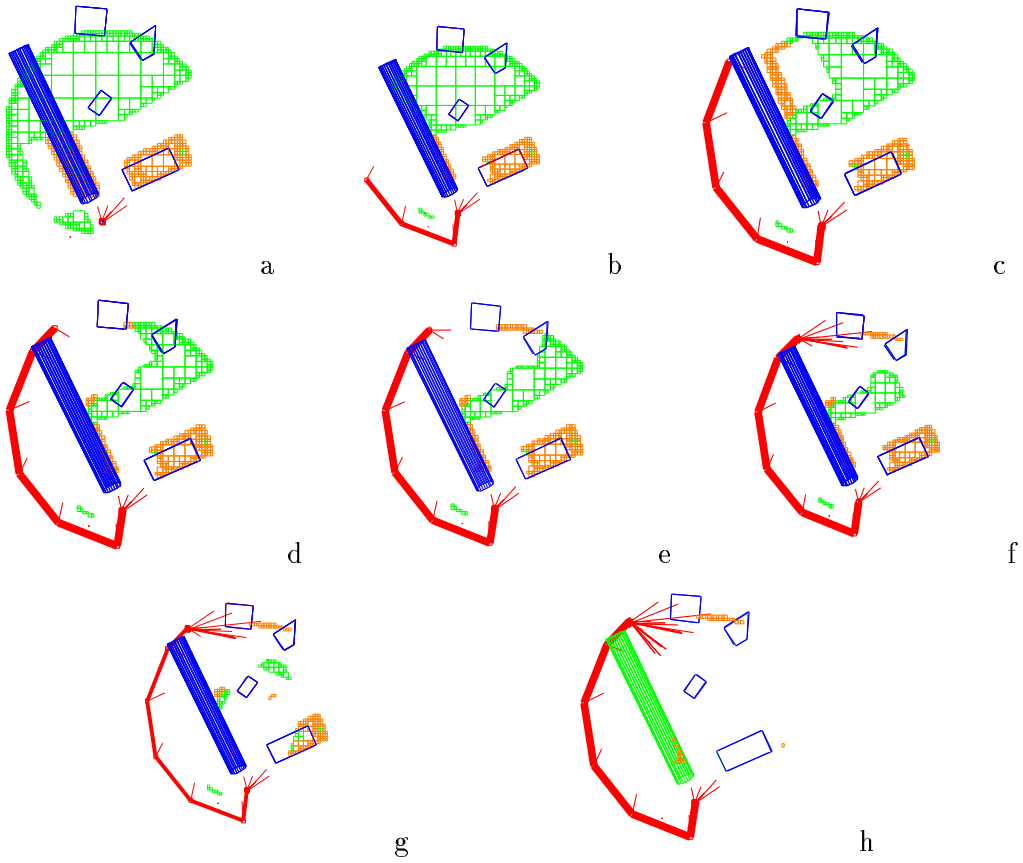


FIG. 4.19 – *Différentes étapes de l'exploration globale (trajectoire de la caméra, scène reconstruite, et projection sur un plan virtuel de la zone non observée)*

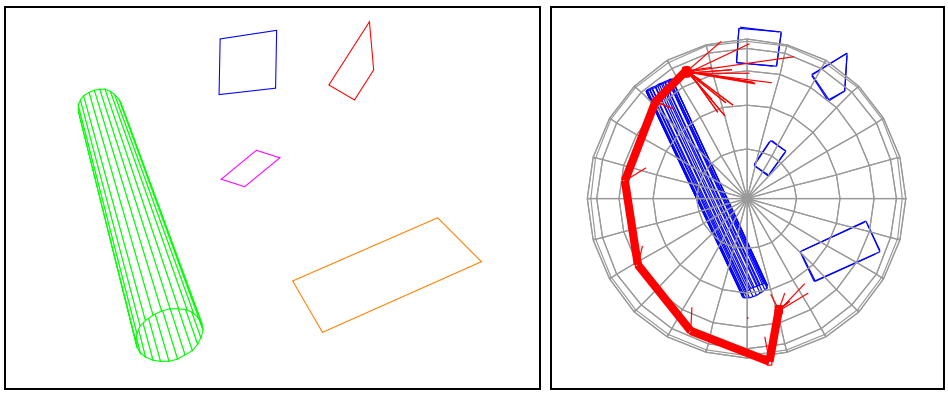


FIG. 4.20 – *Visualisation de la scène reconstruite et vue polaire de la trajectoire finale de la caméra pendant l'exploration globale*

4.6.3 Reconstruction de la scène polyèdre

Considérons maintenant la scène du polyèdre (voir Figure 4.21). Notre objectif est de montrer ici l'intérêt d'une stratégie de focalisation à différentes échelles. La stratégie locale utilisée ici est une version simplifiée de celle qui a été présentée au chapitre précédent (toutes les hypothèses du réseau Bayesien proposé n'ont pas été émises). Elle a permis de reconstruire seulement cinq segments, comme on le constate sur la Figure 4.21.b. Trois segments du polyèdre ne sont jamais apparus dans l'image et n'ont donc pas été reconstruits.

Une première étape d'exploration a permis de s'assurer que la zone autour du polyèdre était vide de tout objet. Cependant, les segments restants n'ont toujours pas été observés car le volume de la zone d'occlusion autour du polyèdre est très faible relativement au volume total de la scène: dans le cadre du processus d'optimisation, ce volume était trop petit pour être pris en compte. Il convient cependant de s'intéresser à cette zone. L'absence maintenant démontrée d'obstacles autour du polyèdre permet de réduire la zone de recherche autour de celui-ci. Un volume englobant l'objet et les zones identifiées comme résultant d'une occlusion par l'un des polygones est donc calculé et une hémisphère située autour de ce volume englobant sert de support au processus de calcul de nouveaux points de vue qui permettent l'observation puis la reconstruction des segments manquants, assurant ainsi la reconstruction complète de cet objet

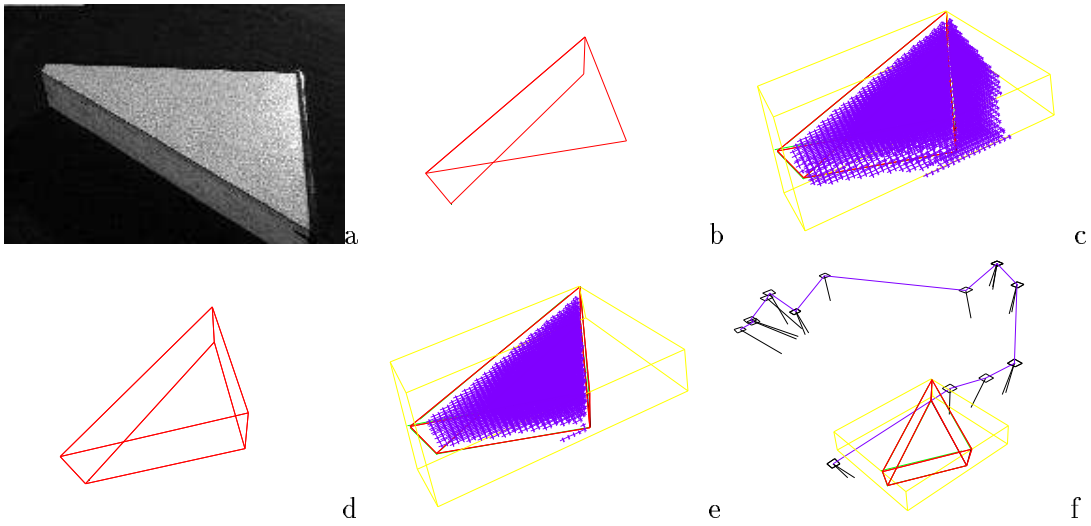


FIG. 4.21 – *Scène polyèdre (a) image initiale (b) modèle de la scène reconstruite après la première phase d'exploration incrémentale (c) zones restant inobservées à l'issue de l'exploration locale (d) modèle de la scène après le processus d'exploration (e) zones restant inobservées à l'issue de ce processus (toutes ces zones sont situées à l'intérieur du polyèdre) (f) trajectoire de la caméra*

Ajoutons que d'autres scènes ont été explorées et reconstruites en utilisant cet algorithme. Un modèle complet de la pyramide présentée dans la Section 3.3 (voir Figure 3.32)

peut ainsi être totalement déterminé.

4.7 Recherche d'une trajectoire plus efficace

Les techniques de calcul de points vue qui ont été présentées dans ce chapitre sont locales en ce sens qu'on ne peut pas remettre en cause la sélection qui est faite (recherche en "profondeur d'abord"). Elles ne permettent donc pas de planifier une trajectoire sur plusieurs points de vue en cherchant à optimiser celle-ci en fonction de la longueur totale à parcourir par le robot ou en fonction du nombre final de points de vue.

4.7.1 Définition du problème

Le problème peut être posé de la façon suivante : ayant à l'instant t effectué la trajectoire $\mathcal{T}_0^t$ et acquis les informations $\mathcal{O}(\mathcal{T}_0^t)$ sur les objets de la scène et $\mathcal{V}(\mathcal{T}_0^t)$ sur l'espace observé, quelle est la trajectoire $\mathcal{T}_{t+1}^{t_{fin}}$ qui permettra d'observer tous les objets de la scène tout en minimisant le nombre de points de vue ? On se rend aisément compte que ce problème n'est pas réaliste pour deux raisons principales.

- La première est de type algorithmique. Même en supposant le modèle déjà totalement acquis, il a été prouvé que ce problème appartient, dans le cas général, à la classe des problèmes NP-Complet [Karp 72], [Tarbox 95].

- La deuxième raison réside dans le manque d'informations disponibles sur la scène. Pour déterminer une telle trajectoire, il faut connaître *a priori* le nombre, la nature et la localisation de tous les objets de la scène. Ce n'est pas notre cas. Si nous cherchons une telle trajectoire, il faudra faire des hypothèses supplémentaires sur la scène.

  La seule supposition qu'il est possible de faire, et qui peut être remise en cause par la suite, est de supposer qu'il n'y a pas d'autres objets dans la scène. Définir cette trajectoire $\mathcal{T}_{t+1}^{t_{fin}}$ permettra donc de vérifier ou d'infirmer cette hypothèse. Dans le cas où elle est confirmée, l'exploration et la reconstruction de la scène sont terminées. Dans le cas contraire (*i.e.*, on a observé depuis une position $\phi_{t'} \in \mathcal{T}_{t+1}^{t_{fin}}$ une nouvelle primitive), alors la primitive devra être reconstruite et la trajectoire $\mathcal{T}_{t'}^{t_{fin}}$ devra être abandonnée et recalculée.

Si trouver la solution optimale à ce problème n'est pas réaliste, nous proposons une méthode permettant de définir une trajectoire $\mathcal{T}_{t+1}^{t_{fin}}$ optimisée, puis au vu des résultats obtenus nous discuterons de l'intérêt d'utiliser une telle approche dans le cadre d'une application comme la nôtre.

4.7.2 Définition d'une trajectoire

On peut aisément obtenir une trajectoire $\mathcal{T}_{t+1}^{fin}$ assurant une exploration de la scène en utilisant la méthode de calcul de points de vue proposée dans les paragraphes pré-

cédents[4]. En calculant cette trajectoire, on constate la chose suivante : de petites zones de la scène demeurent souvent inexplorées entre plusieurs zones connues et observer ces régions "résiduelles" demande le calcul d'un trop grand nombre de nouveaux points de vue supplémentaires. Or le gain marginal apporté par chacun de ces points de vue supplémentaires est souvent très faible. Un ajustement local des positions de la caméra le long de la trajectoire est préférable à l'ajout de ces points de vue. De plus, il arrive souvent que les zones découvertes par un point de vue à la date t le soient aussi par d'autres points de vue situés plus loin dans la trajectoire, ce point de vue devient alors inutile et peut donc être supprimé.

Soit n_0 le nombre de points de vue de la trajectoire initiale. L'algorithme itératif permettant de trouver une trajectoire assurant l'observation complète de la scène en un nombre n de points de vue ($n \leq n_0$) procède donc en deux étapes :

- Dans un premier temps, il convient de rechercher si un ajustement local de la position des différents points de vue permet d'augmenter la quantité d'information visible. Pour cela, nous considérons individuellement l'ensemble des points de vue ϕ_i appartenant à la trajectoire $\mathcal{T}_{t+1}^{fin}$ et testons si une modification de la position d'un de ces points de vue permet un gain d'information. Plus précisément, nous recherchons, à partir de la position $\phi_i \in \mathcal{T}_{t+1}^{fin}$, un point de vue ϕ_i' permettant d'augmenter la quantité d'information perçue.

  Pour cela, on recherche ϕ_i' tel que :

  $$volume(\mathcal{I}(\mathcal{T}_0^{fin})) \quad \leq \quad volume(\mathcal{I}(\mathcal{T'}_0^{fin}))$$

  où

  $$\mathcal{T'}_0^{fin} = \{\phi_0, \ldots, \phi_{i-1}, \phi_i', \phi_{i+1}, \ldots, \phi_{fin}\}$$

  et où $\mathcal{I}(\mathcal{T}_0^t)$ désigne les zones inexplorées (équation (4.3)).

  Pour calculer ϕ_i', l'algorithme de calcul de points de vue est utilisé en considérant dans l'équation (4.4) que $\mathcal{G}(\phi_{i'})$ est égal à :

  $$\mathcal{G}(\phi_{i'}) = \mathcal{V}(\phi_{i'}) - \mathcal{V}(\phi_{i'}) \cap \mathcal{V}(\mathcal{T}_0^{t\,fin} - \phi_i) \tag{4.14}$$

  c'est-à-dire à la zone découverte par le point de vue ϕ_i'.

  Si un tel point de vue existe, ϕ_i' remplace ϕ_i dans la trajectoire. Ce processus est réalisé successivement sur l'ensemble des points de vue de la trajectoire $\mathcal{T}_{t+1}^{fin}$ et est itéré tant qu'un point de la trajectoire est modifié.

  La Figure 4.22 décrit le résultat de cet algorithme d'ajustement local. La Figure 4.22.a représente la projection sur un plan virtuel des cônes de vision associés à chaque point de vue ϕ appartenant à la trajectoire $\mathcal{T}_0^{fin}$. Les points de vue dont les limites apparaissent en trait plein résultent du processus d'exploration locale précédemment réalisé (ils appartiennent à la trajectoire $\mathcal{T}_0^t$), il ne peuvent donc pas être modifiés.

4. En considérant, comme nous l'avons déjà dit que la scène ne comporte plus d'objets "nouveaux" à observer

Afin de montrer l'efficacité de ce type d'algorithme, nous avons volontairement arrêté l'algorithme d'exploration quand 75% de la scène a été observé. La Figure 4.22.b montre l'état des zones observées après une étape d'ajustement local. Plus de 95% de la scène est alors observé, le nombre de points de vue étant resté constant.

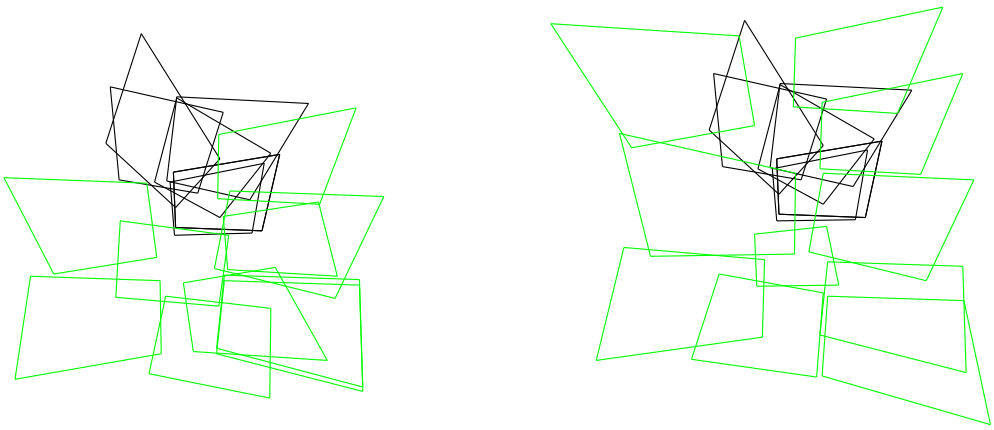


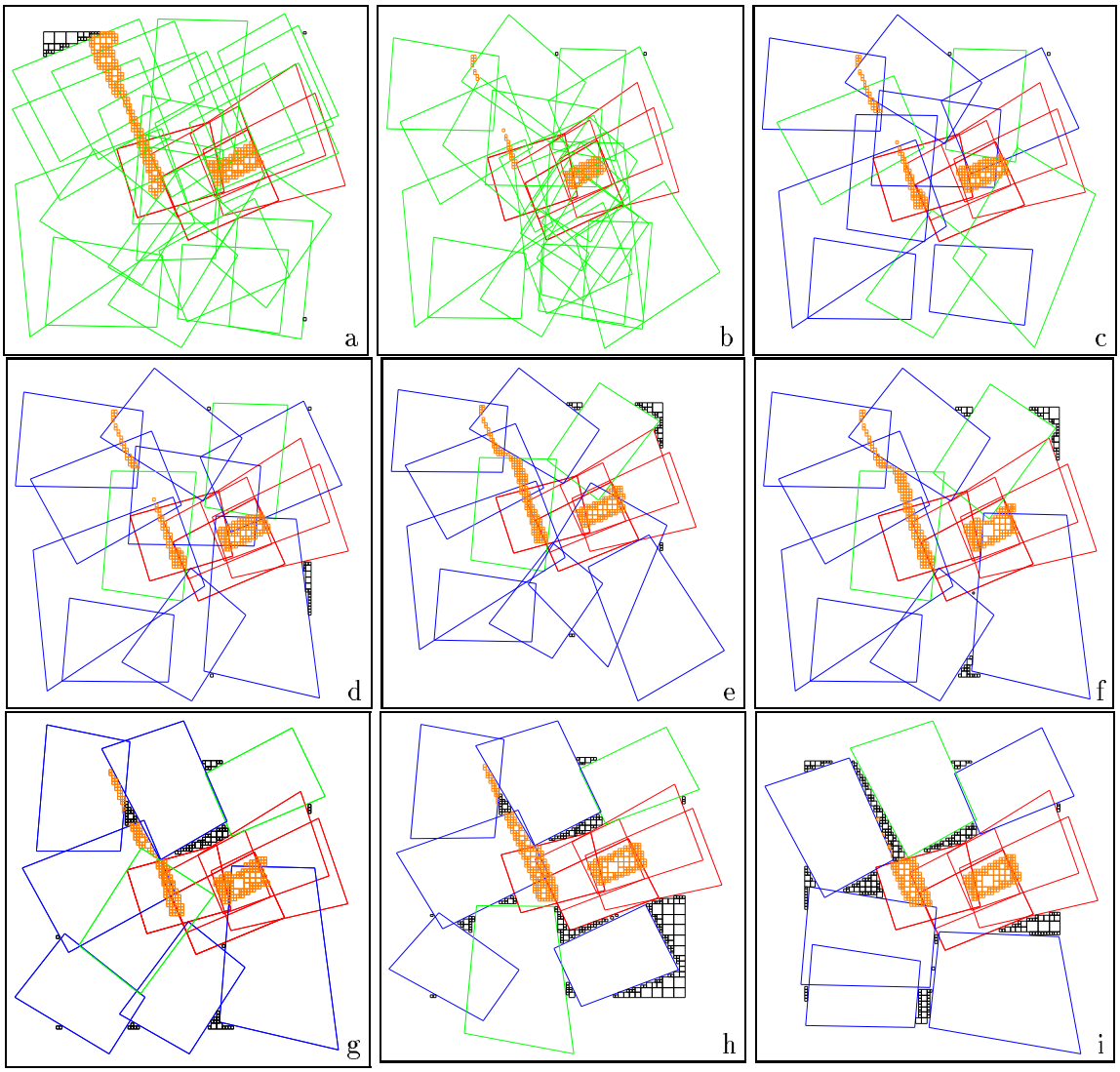
FIG. 4.22 – *Résultat de l'algorithme d'ajustement local des points de vue (a) avant ajustement (b) après ajustement*

– La seconde étape consiste à réduire le nombre de points de vue. En effet, si l'étape précédente à permis, éventuellement, d'augmenter la quantité d'information perçue, le nombre de points de vue est resté constant. Il convient donc de calculer le gain marginal apporté par chacun des points de vue, les autres restant fixes. Cette étape permet de déterminer les points de vue apportant pas ou peu d'information. Le point de vue ϕ_{min} apportant le moins d'information est supprimé de la trajectoire. Le processus d'ajustement local est alors utilisé afin de tenter de compenser, par une modification de la position des autres points de vue de la trajectoire, la perte d'information induite par la suppression de ϕ_{min}. Cette étape est également réitérée jusqu'à atteindre un nombre désiré de points de vue tout en conservant une quantité d'information satisfaisante (si le nombre de points de vue choisi est trop faible, l'information restante peut s'avérer insuffisante).

4.7.3 Résultats

L'expérience suivante illustre le processus d'optimisation de trajectoire proposé ci-dessus. Les Figures 4.23.a à 4.23.o représentent l'état des zones observées en fonction du nombre de points de vue. La Figure 4.23.a montre les zones observées à l'issue de l'exploration globale de la scène. Quelques zones résiduelles n'ont pas été observées mais nous avions seulement requis une couverture à 95% de la scène (ici 95.96% de la scène a

été observée avec 28 points de vue). La Figure 4.23.b montre l'état de ces zones observées après une étape d'ajustement local des positions des points de vue, le nombre de points de vue restant constant. Nous n'avons pas représenté l'état des zones observées pour un nombre de points de vue allant de 27 à 20, car si il y a une modification notable dans la position des points de vue, il n'y a pas de perte d'information. Par contre, sur les Figures 4.23.c à Figure 4.23.n le nombre de points de vue diminue de 20 à 8. On peut noter (voir Figure 4.24), que le pourcentage de scène couverte est supérieur à celui de la trajectoire initiale tant que le nombre de points de vue est supérieur à 15. Avec un nombre de points de vue inférieur, une perte inévitable d'information apparaît.



Notons enfin que les temps de calcul nécessaires à la mise en œuvre de cet algorithme sont prohibitifs (ici, une trajectoire de 15 points de vue est obtenue en 16 minutes sur une Sparc station 20).

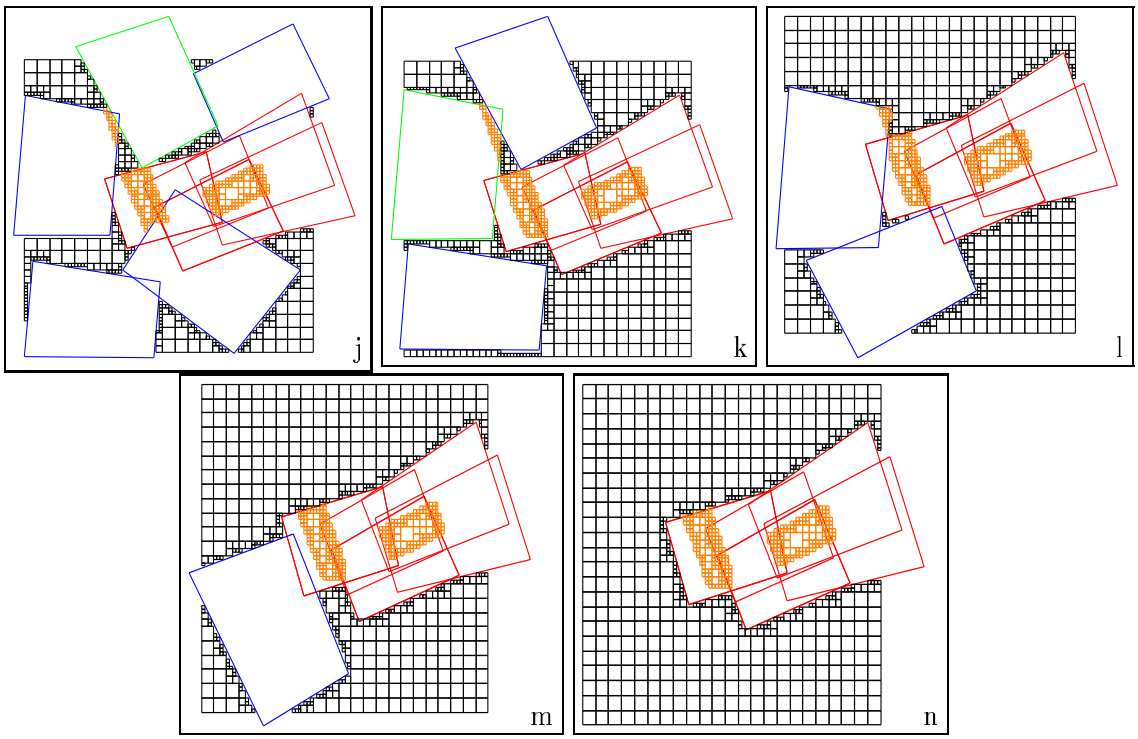


FIG. 4.23 – *État des zones observées en fonction du nombre de points de vue (a) état initial (28 points de vue), (b) ajustement local des positions des points de vue, (c-n) de 20 à 8 points de vue*

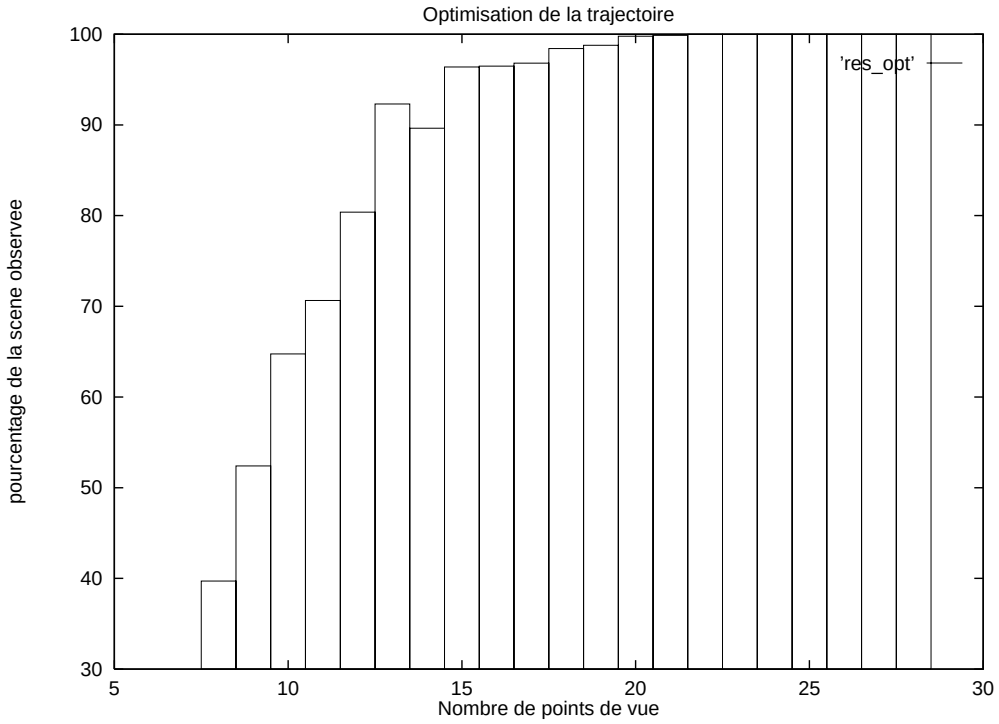


FIG. 4.24 – *Pourcentage de la scène observé en fonction du nombre de points de vue*

4.7.4 Discussion

Comme le montrent les résultats obtenus, l'algorithme proposé permet de réduire de façon significative le nombre de points de vue utilisés. Il convient cependant de rappeler que l'hypothèse qui est faite, à savoir que tous les objets de la scène ont déja été reconstruits, est une hypothèse qui est souvent très forte. Dans le cas de scènes de taille moyenne comme celles que nous reconstruisons, et comportant un nombre relativement important d'objets (rapporté à la taille de la scène), il n'apparaît pas utile d'utiliser ce genre d'algorithme car la trajectoire ainsi calculée sera très rapidement remise en cause par l'apparition de nouveaux objets. D'autre part, le temps de calcul d'une telle trajectoire est très important et s'est avéré, dans les tests que nous avons effectués, toujours supérieur au coût d'exécution de la trajectoire initiale (même si celle-ci comporte beaucoup plus de points de vue). Cependant, dans les cas où le nombre d'objets est très faible relativement au volume observable, alors l'utilisation d'un tel algorithme peut s'avérer bénéfique. De plus, il est très adapté à des tâches d'inspection (recherche d'objet par exemple ou vérification de la position d'objets).

Précisons finalement que, comme l'optimisation repose ici uniquement sur la quantité d'information perçue, il est difficile de faire intervenir le coût séparant deux points de vue. La trajectoire résultant de ce processus d'optimisation n'est alors que très rarement optimale du point de vue de sa longueur. Connaissant l'ensemble des points de vue, il est cependant possible d'optimiser cette trajectoire (il s'agit là-encore, d'un problème NP-

Complet). On utilisera alors le chemin le plus court passant par un nombre "minimal" de points de vue.

En conclusion, il est possible d'améliorer localement la trajectoire de déplacement lors des phases d'exploration. Le gain en nombre de points de vue est dans plupart des cas notable. Cependant, le coût algorithmique d'une telle amélioration est très important et les hypothèses faites sont très fortes (absence de nouveaux objets). Cette hypothèse étant fausse, dans le cas général, la trajectoire devra être remise en question à un moment ou à un autre de son exécution. En tout état de cause, il ne s'agit pas de réaliser une optimisation de la trajectoire, mais d'une amélioration de celle-ci.

4.8 Conclusion

La stratégie de calcul de points de vue que nous avons mise en œuvre repose sur la modélisation et l'optimisation d'une fonction de coût représentant au mieux la tâche que nous souhaitons effectuer. Nous avons retenu quatre critères qui sont intégrés dans cette fonction. En premier lieu, nous nous basons sur le gain apporté par la nouvelle position en volume potentiel découvert (calculé en utilisant des techniques de lancers de rayons). Un critère modélisant le coût de déplacement d'un point de vue au suivant est aussi justifié par le fait que nous souhaitons réduire la distance totale parcourue par la caméra. Les contraintes mécaniques dûes au robot justifient l'introduction d'une contrainte éloignant le robot de ses butées articulaires. Enfin, les connaissances déjà acquises sur la scène permettent de définir un critère binaire représentant l'accessibilité d'un point de vue. La minimisation de la fonction obtenue par un ICM multi échelle (préférable à une minimisation par recuit simulé, notamment pour des raisons de temps de calcul) permet le calcul d'un nouveau point de vue apportant le maximum de nouvelles connaissances sur la scène. Initialement, la caméra se déplace sur une demi-sphère englobant la scène pour éviter les obstacles potentiels (encore inconnus). Dès que des zones accessibles sont reconstruites, la caméra peut se déplacer à l'intérieur de ces zones situées dans la demi-sphère (ce qui permet de traiter de manière adéquate les parties occultées par des objets).

L'exploration de la scène se termine quand, quel que soit le point de vue choisi parmi tous les points de vue accessibles, il n'y a plus d'apport supplémentaire d'informations. Ceci signifie que la reconstruction est alors aussi complète que possible compte tenu des contraintes imposées par le manipulateur et/ou par la scène.

Cette méthode d'exploration de scènes a été développée et testée dans le cadre du processus de reconstruction par vision active et d'exploration locale présenté dans les chapitres précédents. Cependant, on peut signaler qu'elle est indépendante de la méthode de reconstruction choisie et est applicable dès lors que l'on peut disposer d'une représentation dense des zones observées obtenue par exemple par stéréovision dense ou à l'aide d'un capteur laser (dans ce dernier cas, des contraintes augmentant le chevauchement entre les champs de vision pourraient être rajoutées afin d'améliorer la fusion des résultats [Pito 96]).

L'automate hiérarchique parallèle décrivant maintenant l'ensemble du processus de reconstruction est présenté sur la Figure 4.25. Le chapitre suivant décrit comment est réalisée l'exécution séquentielle ou parallèle de ces différentes phases ainsi que leur mise

en œuvre en utilisant le langage synchrone SIGNAL et son extension SIGNAL*GTi*.

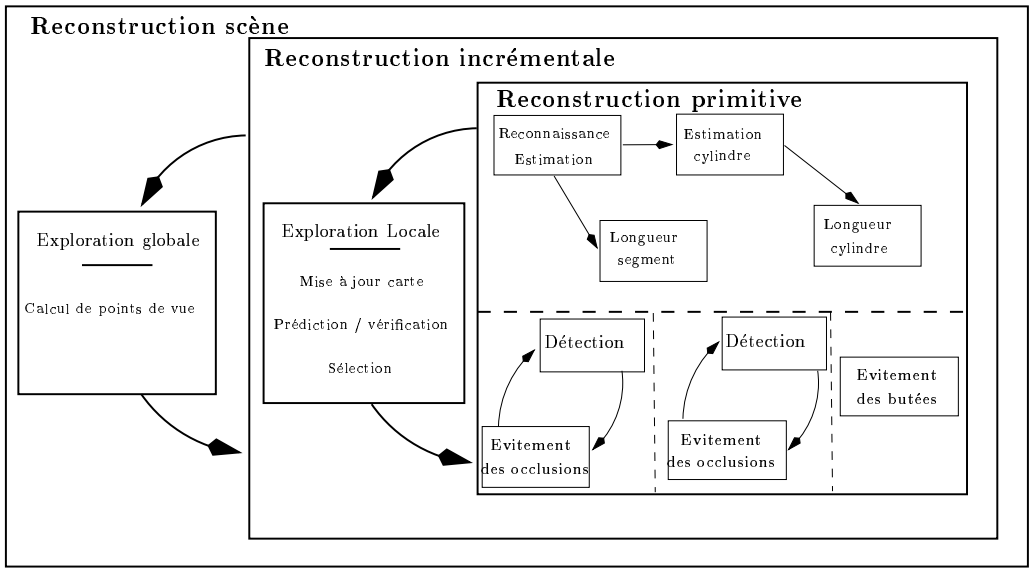


FIG. 4.25 – *Automate hiérarchique parallèle assurant la reconstruction globale de la scène*

Chapitre 5

Une approche synchrone de la dualité continu - événementiel

> La reconstruction était d'ailleurs instantanée. L'opération n'exige donc pas un temps déterminé, et même théoriquement, elle n'exige aucun temps. [...] Mais pour l'artiste qui crée une image en la tirant du fond de son âme, le temps n'est plus un accessoire. Ce n'est pas un intervalle qu'on puisse allonger ou raccourcir sans en modifier le contenu. La durée de son travail fait partie intégrante de son travail. Le temps d'invention ne fait qu'un ici avec l'invention même.
>
> Henri Bergson, *L'Évolution créatrice*, chap. IV.
> Presses Universitaires de France, Paris, 1907.

Ce dernier chapitre traite des aspects implémentation et mise en œuvre d'un système réactif de vision robotique. Il apparaît que l'une des difficultés réside, entre autres, dans la façon de gérer la dualité *continu/événementiel* relative aux mouvements de la caméra et aux stratégies de perception. Dans un tel système, la réactivité intervient à deux niveaux distincts : dans le calcul de la commande en boucle fermée où le système doit réagir à chaque instant en fonction des informations transmises par le capteur, et au niveau du contrôle de tâche où le système doit réagir à des événements extérieurs par des changements d'état. Les méthodes de programmation classiques ne sont pas bien adaptées à la spécification et à l'implémentation de tels algorithmes. Un langage classique impératif requiert une prise en compte explicite des aspects bas niveau de la mise en œuvre ainsi que des problèmes temporels pour lesquels ils ne fournissent pas de support ou de modèle bien établis. Par contre, les langages synchrones sont particulièrement adaptés à la conception des systèmes réactifs. Reposant sur une abstraction du temps réel en un temps discret et logique, ils sont munis d'une forte sémantique. Dans cette optique, le langage SIGNAL met à notre disposition une méthodologie de programmation unifiée permettant d'intégrer l'aspect "continu" des algorithmes de commande et d'estimation avec l'aspect "événementiel" des

stratégies de perception au sein d'un langage temps réel de haut niveau.

Après un rapide état de l'art sur la méthodologie synchrone, nous décrirons la mise en œuvre de notre système de vision à l'aide du langage SIGNAL. Nous insisterons dans un premier temps sur la spécification des lois de commande et de reconstruction 3D par vision active. Puis, dans un deuxième temps, nous décrirons comment l'exécution séquentielle ou parallèle de ces différentes tâches a pu être réalisée efficacement en utilisant SIGNAL*GTi*. Finalement, une discussion présentant les avantages et les inconvénients d'une telle approche sera donnée.

5.1 Systèmes réactifs temps-réel : asynchrone ou synchrone ?

5.1.1 Les systèmes réactifs temps-réel

Le système de vision active qui est proposé dans cette thèse est un système réactif, si l'on se réfère à la définition proposée par Harel et Pnueli [Harel 85]. En effet, il doit uniquement répondre aux sollicitations du monde extérieur et aucun processus de planification n'entre en jeu dans sa conception. De plus, c'est un système temps-réel. En effet, les informations sur lesquelles est fondé son comportement sont extraites de l'environnement, et dans notre cas précis, des images acquises par la caméra. Si le système proposé était un système passif, le concept de système temps-réel n'entrerait pas en jeu. Mais nous nous plaçons dans un contexte de vision active et de ce fait, les informations extraites des images acquises par le capteur doivent être traitées et analysées afin de donner une nouvelle commande à la caméra et ce, avant qu'une nouvelle image ne soit acquise par celle-ci. Ainsi, le concept de temps-réel intervient à ce niveau : les informations doivent être traitées "*le plus rapidement possible*", mais, surtout, elles doivent l'être avant que l'information suivante ne soit disponible. Si l'on admet qu'une telle classification des différents systèmes a un sens, les systèmes réactifs appartiennent, la plupart du temps, à la catégorie des systèmes temps-réel. Ils ont un certain nombre de spécificités et leur mise en œuvre doit (ou devrait) être à même de répondre à certains critères.

On peut rappeler succinctement les principales caractéristiques des systèmes réactifs. Une analyse complète de ce type de système est proposée dans [Harel 85]. Schématiquement, un système réactif est composé de deux parties principales [Gautier 94]. La première partie du système est l'**environnement**. L'autre correspond à un sous-système : la partie "automatisée" du système. Le comportement de cette dernière est issu directement de l'environnement (voir Figure 5.1). En fonction des caractéristiques du système, les infor-

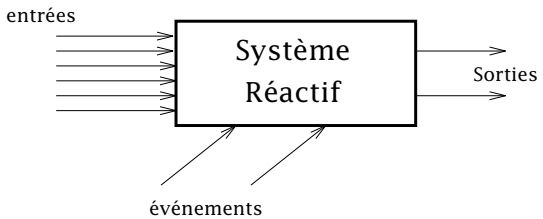


FIG. 5.1 – *Entrée/sortie d'un système réactif*

mations en provenance de l'environnement passent dans un certain nombre de processus. Généralement on retrouve :

- un processus d'observation qui acquiert les données provenant de l'environnement et qui redirige ces informations vers les autres modules du système ;
- un processus réalisant l'analyse des informations et calculant les actions (commandes) à effectuer en réaction à ces informations ;
- un processus modélisant la connaissance sur l'environnement. Cette modélisation doit tenir compte des dernières observations réalisées ainsi que des modifications de l'état de l'environnement induites par les actions effectuées par le système.

En accord avec ce schéma de base, le sous-système automatisé, et en particulier le processus réalisant les actions, peut être mis en œuvre par [Gautier 94] :

- des fonctions : les sorties du système dépendent dans ce cas uniquement des informations disponibles en entrée ;
- des automates déterministes : les sorties du système dépendent alors non seulement des informations disponibles en entrée, mais aussi de l'état courant de l'automate définissant le comportement du processus ;
- des relations : les sorties du système dépendent alors des entrées, de l'état courant et des relations entre entrées et sorties.

De ce schéma de base, il est possible de déduire les principales caractéristiques que doivent posséder ces systèmes. Ils sont déterministes (pour une même suite de signaux en entrée, ils doivent fournir une même suite de signaux de sortie) ; ils subissent des contraintes temporelles (afin de réagir au rythme imposé par l'environnement) ; ils doivent supporter le parallélisme (arrivée simultanée d'informations) ; enfin il doivent être fiables (on doit être à même de prouver qu'ils réagissent de manière conforme aux spécifications).

Un système réactif est intrinsèquement un système parallèle. En effet, les informations venant du monde extérieur et qui définissent son comportement peuvent arriver simultanément et intéressent, potentiellement, diverses parties du système. Ainsi plusieurs parties (processus) doivent être à même de recevoir différents stimuli provenant de capteurs hétérogènes et doivent réagir à toutes ces sollicitations en suivant des évolutions parallèles. Ces différents processus ayant chacun une vie indépendante doivent cependant être à même de communiquer les informations acquises ou les signaux de réponses à d'autres processus. Un schéma de synchronisation entre ces différents processus doit donc être envisagé.

5.1.2 Spécification et mise en œuvre classique : l'approche asynchrone

Une étude critique des systèmes classiquement utilisés pour la mise en œuvre des systèmes temps-réel est proposée dans [Berry 89]. Lors de l'intégration de tels systèmes, la manipulation du temps se fait généralement de manière asynchrone. Plusieurs approches différentes sont basées sur ce concept d'asynchronisme : la première est basée sur l'utilisation des systèmes de transitions tels que les automates et les réseaux de Pétri et repose

donc sur des fondements formels et bien maîtrisés. La deuxième approche est basée sur l'utilisation d'outils aptes à exprimer le parallélisme : langages ou systèmes d'exploitation spécialisés.

Examinons dans un premier temps les systèmes à base de transitions. Les **automates d'état finis** sont des outils bien connus et bien formalisés. Leur utilisation pour l'intégration de systèmes réactifs présente de nombreux avantages : il sont déterministes, leur mise en œuvre est efficace et comme ils reposent sur des bases formelles solides, des outils permettant de vérifier leur comportement sont disponibles. Cependant, l'utilisation d'automates soulève rapidement des problèmes : la composition de petits automates mène le plus souvent à la création d'automates de taille très importante, souvent impossibles à comprendre et à interpréter. Il est aussi à noter qu'une modification, même infime dans les spécifications du système conduit généralement à effectuer des modifications très importantes dans l'automate. Enfin, ce formalisme ne peut gérer ni le parallélisme, ni la préemption de tâches. Les **réseaux de Pétri**, quant à eux, sont souvent utilisés pour l'intégration de petites applications. Comme les automates, ils reposent sur des outils formels connus, ce qui autorise la vérification de certaines propriétés. À la différence des automates, il est possible d'exprimer le parallélisme. Cependant, la spécification de la hiérarchisation de tâches reste hors de leur portée et leur déterminisme n'est pas garanti.

La deuxième approche repose sur l'expression du parallélisme. Les **langages de programmation parallèle** comme OCCAM (langage inspiré par CSP [Hoare 85]) ou ADA [Baker 91] ont de nombreux avantages : ils sont bien structurés, et autorisent un très bon niveau de modularité. Cependant, les communications se font de manière asynchrone, ce qui introduit dans ces langages un indéterminisme très fort. Les synchronisations entre les différents processus se font à l'exécution, rendant le comportement de tels systèmes imprévisible. L'approche la plus classique consiste à réaliser l'intégration de plusieurs programmes mis en œuvre de manière classique à l'aide des primitives de communication disponibles dans un **systèmes d'exploitation temps-réel** (VX-WORKS par exemple). Dans ce cas, le problème principal repose sur la multiplicité des programmes à étudier, vérifier et connecter. Le diagnostic et la maintenance de tels systèmes sont problématiques. Les contraintes temporelles sur lesquelles reposent le bon fonctionnement ne sont pas explicitement exprimées dans les différentes tâches (programmes) mais sont satisfaites en utilisant les primitives de synchronisation et de communication du système d'exploitation. Ceci amène généralement à des systèmes non déterministes, sur lesquels aucune propriété de sécurité, vivacité, ou autres, ne peut être vérifiée.

5.1.3 La méthodologie synchrone : une approche originale

De manière à concilier concurrence, déterminisme et sécurité de programmation, nous avons décidé d'utiliser l'approche synchrone [Benveniste 91], [Benveniste 94], [Halbwachs 93] et en particulier le langage SIGNAL. La caractéristique principale des systèmes réactifs est que leur rythme (*pace*) est déterminé uniquement par leur environnement. Une interprétation de l'hypothèse synchrone repose sur l'idée que les signaux impliqués dans un calcul (signaux interne, d'entrée et de sortie) sont présents simultanément. De fait, par hypothèse, le système doit réagir instantanément aux sollicitations extérieures. Cette hypothèse, qui

peut paraître exigente, est valide si l'on peut prouver que la durée maximale d'une action est inférieure à la période de réaction de l'environnement. Cette forme de synchronisme est une abstraction réaliste dans bien des domaines d'application : la synthèse de circuit (*zero delay*), le traitement du signal, la conception de boucle de commande, etc. Dans le cas d'un système réactif, un système basé sur l'hypothèse synchrone peut être vu comme une abstraction de la boucle classique perception-action (acquisition des données, calcul des commandes des actionneurs, envoi des commandes).

L'hypothèse synchrone permet aussi de simplifier les raisonnements sur le temps. Les instants considérés dans de tels systèmes sont des instants logiques. Le comportement d'un système dans le temps est alors considéré comme une suite ordonnée d'instants logiques. Pour chacun de ces instants logiques, la valeur d'un signal se rapportant au système peut être présente ou non. La manipulation du temps est naturelle et exacte, et les propriétés temporelles du système sont vérifiables, toutes les parties du programme communiquant par diffusion instantanée de l'information.

L'intérêt principal de cette hypothèse est qu'elle permet de définir des bases sémantiques solides facilitant la manipulation des programmes. Les langages reposant sur cette hypothèse ont l'avantage d'avoir un comportement déterministe. Les systèmes réactifs sont, par définition, purement déterministes ; une mise en œuvre synchrone autorise donc une spécification "réelle" de ces systèmes, ce qui n'est pas le cas lorsque leur conception repose sur une approche asynchrone.

Enfin, la sémantique synchrone procure à l'utilisateur un ensemble d'outils d'aide à la conception de systèmes temps-réel. Les bases formelles sur lesquelles reposent ces langages autorisent une analyse des différents niveaux d'abstraction constituant le programme, depuis les spécifications jusqu'à la génération de code éxécutable en passant par l'évaluation des performances et l'optimisation du code. Les techniques d'analyse et de vérification reposent sur une étude du temps logique, ce qui permet de prouver le comportement du système dans le temps. Ces garanties de comportement étant disponibles, il devient possible de réaliser des estimations quantitatives très précises sur le comportement temporel d'une application étant donnée une architecture cible particulière (qui peut être multiprocesseur).

Il existe une famille complète de langages synchrones qui ont été développés, principalement en France, depuis le début des années 80. Ils ont chacun des caractéristiques qui leur sont propres mais ont aussi des caractéristiques communes. À la base de leur conception, il y a, bien sûr, l'hypothèse synchrone. Ils reposent sur des bases sémantiques solides et sont donc tous fournis avec des outils de développement puissants. Ainsi, ils supportent tous des outils de spécification, production et vérification de code, ce qui fait d'eux les outils idéaux pour la programmation des systèmes où la sécurité est un problème critique, en particulier ceux portant sur des applications de traitement du signal ou sur le contrôle de processus. L'objectif n'est pas de faire ici une analyse des avantages et des inconvénients de chacun de ces langages ; pour une analyse comparée, le lecteur pourra se reporter à [Halbwachs 93]. Alors qu'ESTEREL est un langage de type impératif destiné principalement à la programmation des systèmes réactifs, LUSTRE et SIGNAL sont tous deux des langages déclaratifs et flot de données destinés plutôt au traitement du signal (bien qu'ils soient de plus en plus utilisés pour d'autres types d'applications). Un quatrième

langage, STATECHARTS, est plus à considérer comme un formalisme graphique que comme un langage synchrone à part entière, car sa sémantique n'est pas totalement conforme à l'hypothèse synchrone (c'est-à-dire qu'il peut être non déterministe). ARGOS permet en partie de combiner une représentation graphique proche de celle des STATECHARTS avec une sémantique synchrone précise. Les paragraphes qui suivent font une rapide présentation de ces différents langages.

Le langage ESTEREL. ESTEREL est un langage temps-réel synchrone de type impératif [Berry 87], [Boussinot 91]. L'aspect modulaire et impératif d'ESTEREL fait de lui un langage principalement adapté à la programmation d'applications décrivant des automatismes séquentiels qui communiquent entre eux par l'intermédiaire de signaux. Deux des éléments de base de ce langage sont les signaux capteurs (utilisés comme entrées du système) et les signaux (utilisés à la fois comme entrées et comme sorties). À chaque signal sont associées deux informations : une valeur et des *tops* qui indiquent les instants auxquels le signal est émis. La valeur d'un signal ne peut changer que si un *top* est émis. Par contre un signal capteur n'a pas de *top* mais uniquement une valeur. Le principe de fonctionnement d'ESTEREL est de faire communiquer une série de modules par l'intermédiaire de ces signaux. Un programme ESTEREL étant conçu selon l'hypothèse synchrone, tous les signaux de sortie sont émis de façon synchrone avec les signaux d'entrée et les signaux capteurs (qui ne correspondent pas à un *top* particulier). Ainsi il y aura émission d'un signal de sortie si et seulement si il y a occurence d'un signal d'entrée. Cette propriété fait d'ESTEREL un langage adapté à la mise en œuvre des systèmes réactifs. Cette programmation modulaire autorise une mise en œuvre indépendante des différentes parties d'un programme, simplifiant ainsi l'implémentation du programme dans sa globalité. Le compilateur d'ESTEREL traduit le programme en un automate d'état fini, élaboré à l'aide de la sémantique formelle du langage. Le compilateur réalise une première étape de vérification des programmes. Ainsi des propriétés statiques portant entre autres sur le caractère déterministe du programme ESTEREL peuvent être vérifiées dès cette phase. De plus, les propriétés des automates d'états finis étant bien connues, la compilation de programmes ESTEREL sous cette forme permet la mise en place d'un système de vérification formelle de programme (AUTO, AUTOGRAPH).

Le langage LUSTRE. Le langage LUSTRE [Halbwachs 91] est à la différence d'ESTEREL un langage de type flot de données (c'est-à-dire déclaratif). Contrairement à ESTEREL, LUSTRE est plus orienté vers des applications de type calcul et traitement du signal que vers des applications décrivant des systèmes séquentiels. Chaque programme est un réseau d'opérateurs fonctionnant en parallèle et activé sur occurrence d'un signal d'entrée. Ce réseau est un ensemble d'équations reliant entre eux des signaux. Chaque équation définit la manière d'obtenir chaque signal de sortie en fonction des signaux d'entrée.

Le langage SIGNAL. Comme LUSTRE, SIGNAL [Le Guernic 91] est un langage de type déclaratif et flot de données. Les objets de base manipulés par ce langage sont les signaux. Les opérateurs du langage permettent de spécifier dans un style équationnel les relations

entre les signaux, c'est-à-dire entre leur valeur et entre leur horloge. Un système d'équations est alors construit en réalisant la composition des différentes équations : le programme SIGNAL. Le compilateur se livre à une analyse de consistance du système d'équations ainsi défini et détermine si les contraintes de synchronisation sont vérifiées. Si c'est le cas et si le programme est contraint de façon à calculer une solution unique, alors un code éxécutable est produit. Cependant, si SIGNAL est très adapté à la mise en œuvre de systèmes réactifs "continus", il apparaît, sous sa forme initiale, peu adapté à la spécification d'applications nécessitant l'enchaînement de différents modes de fonctionnement. Une extension de SIGNAL, SIGNAL*GTi* [Rutten 94], [Rutten 95a], offre la possibilité de spécifier des tâches ayant une durée et autorise la préemption de tâches sur des intervalles de temps. De plus, le système SIGALI autorise la vérification formelle de spécification en SIGNAL et les preuves de propriétés dynamiques. Une présentation plus détaillée de SIGNAL sera donnée dans les sections suivantes.

Les STATECHARTS. Les STATECHARTS [Harel 87] sont plus à considérer comme un formalisme graphique, que comme un langage de programmation synchrone à part entière. La sémantique des STATECHARTS n'est pas entièrement synchrone, mais leur originalité réside dans le fait qu'ils permettent une spécification graphique et hiérarchisée des automates. Chaque état d'un STATECHARTS est un état à part entière ou un automate plus ou moins complexe. Chaque état simple ou complexe peut être relié à d'autres par des transitions valuées par des signaux.

Le langage ARGOS. ARGOS est un langage synchrone d'inspiration proche de celle des STATECHARTS [Jourdan 94]. À la différence de ces derniers, la sémantique synchrone d'ARGOS est conforme à l'hypothèse synchrone. C'est un langage de type impératif basé sur des automates hiérarchiques parallèles. Chaque système élémentaire est décrit directement par l'intermédiaire d'un automate en donnant explicitement la liste des états ainsi que les transitions d'un état à l'autre. Quand le système devient plus complexe, il peut être décrit comme la combinaison de plusieurs systèmes élémentaires. Un opérateur de composition parallèle définit la manière de connecter ces systèmes afin de les faire communiquer ainsi que la façon dont ils participent au comportement global du système.

5.1.4 Spécification de systèmes de vision robotique

Concernant la spécification et l'implémentation de systèmes de vision artificielle, la plupart résulte de l'intégration de différents programmes exécutés sur des cartes différentes, souvent connectées entre elles par des systèmes d'exploitation temps réel (comme PSOS ou VX WORKS). Le séquencement de tâches est classiquement réalisé en utilisant des méthodes reposant sur des automates d'états finis ou des réseaux de Petri. De fait, l'intégration de tels systèmes est rarement décrite (un rapide état de l'art est réalisé pour les systèmes d'inspection temps-réel dans [Thomas 95]).

Cependant, la mise en œuvre de ces systèmes ne repose pas toujours sur des techniques "classiques". De plus en plus, on constate l'apparition de techniques plus formelles reposant la plupart du temps sur le formalisme des systèmes dynamiques à événements

discrets (ou DEDS dont la formalisation est en grande partie due à Ramadge et Wonham [Ramadge 89]). L'utilisation en vision d'un tel formalisme [Aloimonos 93], [Sobh 92], [Košeká 95] permet la synthèse de comportements complexes des différents agents visuels impliqués. Kŏseká [Košeká 95] propose d'utiliser les DEDS pour gérer le comportement d'un robot mobile. Des comportements élémentaires (navigation, évitement d'obstacle,...) sont modélisés par des automates très simples. Puis, ces automates sont ensuites composés, en utilisant le formalisme proposé par Ramadge et Wonham, synthétisant le comportement complexe du système. Notons que l'utilisation des DEDS autorise la vérification d'un grand nombre de propriétés dynamiques sur ces systèmes (vivacité, atteignabilité, etc...). De telles vérifications ont seulement été réalisées dans le cadre de systèmes de vision où seule la conception du superviseur est réalisée en utilisant ce formalisme. De plus, la conception des lois de commande n'est pas prise en compte.

Le système ORCCAD. L'utilisation la plus significative de l'approche synchrone pour des applications robotiques semble être le système ORCCAD [Simon 93]. ORCCAD est un système d'aide à la conception et à la programmation des systèmes robotiques dont les domaines d'application sont très vastes (citons par exemple la commande de robot mobile par asservissement visuel [Pissard-Gibollet 93], [Coste-Manière 92] ou la commande de robots sous-marins [Simon 93]).

L'objet de base du système ORCCAD est la tâche robot (*Robot-Task*) dont la fonction est de spécifier et de mettre en œuvre des actions robotiques simples. Le niveau application (spécifiant des actions complexes) est alors obtenu en composant les tâches robots à l'aide de différents modes de synchronisation. L'objet final ainsi obtenu est appelé Procédure-Robot (*"Robot-Procedure"*). Chaque tâche robot est caractérisée par un aspect fonctionnel relatif aux lois de commande (*"Module-Task"*) et un aspect logique relatif au comportement du robot en réaction aux événements venant du monde extérieur pendant l'exécution des boucles de contrôle. L'exécution d'une loi de commande s'effectue si un ensemble de préconditions est vérifié et se termine sur l'occurrence d'une exception ("problème") ou si un ensemble de postconditions est vérifié. La spécification du système réalisée avec ORCCAD est ensuite automatiquement traduite dans le langage synchrone ESTEREL [Boussinot 91]. Cette traduction en ESTEREL permet d'obtenir un unique automate pour l'ensemble de l'application à partir duquel des propriétés sur le comportement du système peuvent être démontrées [Kapellos 95].

Cependant, seul le niveau application (*"Robot-Procedure"*) et la partie comportementale des tâches robot sont traduits dans ce langage. La partie loi de commande des tâches robot (*"Module-Task"*) n'est pas implémentée à l'aide de langages synchrones et leur spécification est réalisée en utilisant un autre formalisme. La plupart des boucles de commande proposées dans [Coste-Manière 92], [Kapellos 95], [Pissard-Gibollet 93], [Simon 93] reposent sur l'approche fonction de tâche [Samson 91]. De fait, les algorithmes mis en jeu sont de type flot de données et devraient tirer avantage d'une spécification et d'une mise en œuvre avec des langages synchrones flot de données comme LUSTRE ou SIGNAL. Le paragraphe 5.2 montrera comment de telles tâches peuvent être spécifiées avec SIGNAL. Ajoutons qu'une extension d'ORCCAD à la planification est proposée dans [Alami 93].

5.1.5 Motivation pour l'utilisation de SIGNAL

Ce chapitre a pour ambition de montrer comment un système de vision active peut être mis en œuvre efficacement avec le langage SIGNAL. Plus précisément, notre application de reconstruction et d'exploration par vision active est traitée ici à trois niveaux différents :

- le niveau le plus bas concerne le *contrôle du mouvement de la caméra* par asservissement visuel (voir Annexe D). À ce niveau, une tâche robotique est vue comme une simple fonction flot de données qui calcule un "flot de commandes" à transmettre à l'effecteur, à partir d'un "flot d'informations" en provenance du capteur. L'hypothèse synchrone s'applique dans ce cas parfaitement aux équations définissant la loi de commande référencée capteur ;

- le second niveau concerne les aspects de *reconstruction 3D par vision active* (voir Chapitre 2). Le mouvement de la caméra étant contrôlé, la reconstruction se réalise en même temps que la commande de la caméra. La spécification d'un tel processus tire donc avantage du parallélisme implicite offert par SIGNAL. D'autre part, la possibilité d'accéder aux valeurs passées d'un signal est ici indispensable, on profitera donc pleinement de l'aspect dynamique du langage ;

- enfin, le niveau le plus élevé traite des *stratégies de perception* (voir Chapitres 3 et 4). Comme nous l'avons vu, les différentes primitives de la scène doivent être reconstruites de manière indépendante. Une méthode permettant un enchaînement correct des différentes tâches de vision est donc nécessaire. Cet enchaînement sera spécifié à l'aide d'un automate hiérarchique parallèle et mis en œuvre avec SIGNAL*GTi*, l'extension de SIGNAL à la notion de tâche et d'intervalle de temps.

Ces différentes phases seront successivement analysées et leur mise en œuvre sera décrite.

5.2 Spécification de tâches de vision

5.2.1 Langage déclaratif et asservissement visuel

Le langage SIGNAL : primitives de base. Comme indiqué précédemment, SIGNAL est un langage synchrone de type déclaratif et flot de données. Les objets de base manipulés par ce langage sont les signaux. Un signal est une suite non bornée de valeurs typées à laquelle est associée une horloge qui détermine l'ensemble des instants où le signal est présent. Par exemple, un signal `X` dénote la séquence $(\mathtt{x}_t)_{t\in T}$ de données indexées par le temps t dans un domaine T. Des signaux d'un type particulier appelés `event` sont caractérisés seulement par leur horloge, c'est-à-dire leur présence (ils ont la valeur Booléenne `true` à chaque occurrence). Étant donné un signal `X`, son horloge est donnée par l'expression `event X`, qui donne l'événement présent simultanément à `X`.

SIGNAL est construit autour d'un petit nombre d'opérateurs de base qui permettent de spécifier dans un style équationnel les relations entre les signaux. Chaque équation issue d'un programme SIGNAL peut être vue comme un processus élémentaire. Ces processus SIGNAL décrivent donc à la fois les relations fonctionnelles et temporelles entre les signaux.

Ils peuvent communiquer, par l'intermédiaire de signaux constituant leurs ports d'entrée et de sortie, avec le monde extérieur ou avec d'autres processus. Enfin, la composition d'un ensemble de processus produit le programme SIGNAL.

Le système d'équations issu de la composition des différents processus est ensuite analysé par le compilateur qui se livre à une vérification de sa consistance et détermine si les contraintes de synchronisation sont vérifiées. Si c'est le cas et si le programme est contraint de façon à calculer une solution unique, alors un code exécutable en C ou en Fortran est produit.

En SIGNAL, les opérateurs de base définissent des processus élémentaires, chacun correspondant à une équation :

- **Les opérateurs fonctionnels.** Ils sont définis sur les types du langage. Par exemple la négation Booléenne du signal `E` est écrite `not E` et le signal (Y_t), défini par la fonction f dans $Y_t = f(X_{1_t}, X_{2_t}, ..., X_{n_t})$, est écrit:

  ```
  Y := f{X1, X2, ... ,Xn}
  ```

  Les expressions fonctionnelles sont monochrones, ce qui signifie que les signaux `Y`, `X1`, ..., `Xn` sont dits synchrones : ils partagent la même horloge. En d'autres termes, pour calculer la valeur de Y_t, tous les X_i doivent être disponibles à l'instant t. Pour cette raison, ils sont contraints à avoir la même horloge : celle de Y.

- **Le filtre.** Le sous-échantillonnage d'un signal `X` selon une condition `C` est écrit :

  ```
  Y := X when C
  ```

  Cet opérateur est polychrone : les opérandes et le résultat n'ont pas la même horloge. Le signal `Y` est présent si et seulement si `X` et `C` sont présents au même instant et `C` a la valeur `true`. Ainsi, `Y` est moins fréquent que `X` et que `C` à la fois : l'intersection des horloges de `X` et de `C` (c'est-à-dire les instants où l'expression peut être évaluée) inclut l'horloge de `Y` (qui ne comporte que les instants où `C` s'évalue à `true`). Quand `Y` est présent, sa valeur est celle de `X`.

- **La fusion.** On définit la fusion de deux signaux `X` et `Y` par :

  ```
  Z := X default Y
  ```

  Cet opérateur est également polychrone : l'horloge de `Z` est l'union de celles de `X` et `Y`, elle est donc plus fréquente que chacune d'elles.

  La valeur de `Z` est celle de `X` quand il est présent, sinon celle de `Y` quand il est présent.

- **La composition de processus.** Les processus élémentaires peuvent être composés par l'opérateur commutatif et associatif "`|`" qui dénote l'union des systèmes d'équations. En SIGNAL, pour des processus P_1 et P_2, on écrit :

  ```
  (| P1 | P2 |)
  ```

D'autres opérateurs sont disponibles dans le langage, certains comme le retard seront décrits dans la suite de ce chapitre. Une description complète du langage SIGNAL et des

différents opérateurs est donnée dans [Bournai 93a]. Si le langage est construit autour de ce noyau élementaire, il comporte cependant des opérateurs dérivés pour les tableaux ou les variables par exemple. On peut ainsi définir des sortes de macro-instructions : des schémas de processus, qui ont un nom, des paramètres et des signaux d'entrée et de sortie typés, un corps et des déclarations locales. Les instances de schémas de processus rencontrées dans un programme sont expansées par un préprocesseur du compilateur.

Application à l'asservissement visuel. Rappelons brièvement que l'asservissement visuel (ou commande référencée vision [Chaumette 90], [Espiau 92]) consiste à utiliser les informations fournies par une caméra mobile commandable afin de réaliser des tâches élémentaires (positionnement, suivi d'objet mobile, suivi de trajectoire) en contrôlant le mouvement de la caméra par rapport à son environnement. L'approche consiste à spécifier le problème en termes de régulation dans l'image, ce qui permet de compenser les imprécisions des modèles de la caméra et du manipulateur.

Dans notre cas (voir Annexe D), l'asservissement est intégré dans l'approche fonction de tâche [Samson 91] et s'exprime comme la régulation à zéro de la fonction de tâche donnée par l'équation (D.5).

Schématiquement, la loi de commande permettant de se positionner par rapport à un objet statique peut s'écrire (voir pour plus de détail les équations (D.5) et (D.6)) :

$$T_c \;=\; -\lambda A_1 C(p2 - p2_d) \;-\; \lambda A_2 e_2 \;-\; A_2 \frac{\partial e_2}{\partial t} \tag{5.1}$$

où

- T_c est la vitesse de la caméra ;
- $p2$ décrit la position courante de la primitive considérée dans l'image[1] ;
- $p2_d$ représente la valeur désirée que $p2$ doit atteindre (position désirée de la primitive dans l'image).
- C est une matrice définie comme étant la pseudo-inverse d'un modèle ou d'une approximation de la matrice d'interaction associé à $p2$. Deux cas de figure peuvent se présenter :
  - $C = C(p2, \widehat{P3})$ si les paramètres $P3$, représentant l'information de profondeur entre la primitive considérée et le repère de la caméra, peuvent être estimés en ligne (voir Chapitre 2) ;
  - $C = C(p2_d, P3_d)$ si la matrice d'interaction ne peut être mise à jour à chaque itération de la boucle de commande. Des hypothèses doivent alors être faites sur la structure 3D de la primitive considérée afin de calculer les valeurs désirées $P3_d$ et $p2_d$.

1. la variable $p2$ correspond au vecteur de paramètres $\underline{p}$ utilisé dans le chapitre 2. On opposera par la suite les paramètres $p2$ de la primitive (position courante de la primitive considérée dans l'image 2D) et les paramètres $P3$ de la primitive qui représentent l'information 3D associée à cette primitive ($\underline{P}$ dans le chapitre 2)

Dans un premier temps, on considère $C = C(p2_d, P3_d)$. On verra dans le paragraphe suivant que l'on peut évidemment considérer aussi $C = C(p2, \widehat{P}3)$.

- e_2 est une tâche secondaire telle qu'un suivi de trajectoire ;
- A_1 et A_2 sont deux opérateurs de projection qui dépendent de C et qui assurent la réalisation de la tâche secondaire sous la contrainte que la tâche principale $(p2 - p2_d)$ soit réalisée.
- enfin, λ est un gain a régler.

En d'autres termes, la loi de commande consiste à calculer la vitesse de la caméra en fonction, d'une part, d'une *tâche primaire* visant à minimiser l'erreur $(p2 - p2_d)$ et, d'autre part, d'une *tâche secondaire* ; ces deux tâches interagissent mutuellement puisque la tâche secondaire est réalisée dans la mesure où elle n'est pas contraire à la tâche principale.

Sur le plan de la programmation, les algorithmes mis en jeu ont deux caractéristiques principales. D'abord, ils ont une nature équationnelle : ils expriment des relations entre différents flots de données et ce de façon déclarative. Ensuite, ils répondent parfaitement à l'hypothèse synchrone puisque les valeurs des différentes variables utilisées dans les équations sont présentes simultanément (même instant logique).

La loi de commande en boucle fermée sur les informations capteurs utilisées en asservissement visuel consiste donc en la régulation d'une fonction de tâche qui peut, de manière schématique, s'écrire $c = f(s)$ où c est la vitesse de l'effecteur exprimée en fonction des signaux capteurs s. Le contrôle de l'effecteur est une fonction continue f plus ou moins complexe. L'implémentation d'une telle loi de commande est réalisée en "discrétisant" le flot d'informations s en provenance de la caméra en un flot de valeurs s_t, qui est ensuite utilisé pour calculer un flot de commandes c_t : $\forall t, c_t = f(s_t)$. Ce type de calcul numérique et flot de données sont le champ d'application privilégié des langages flot de données et de SIGNAL en particulier.

L'utilisation de SIGNAL pour la mise en œuvre de tels algorithmes permet donc de rendre implicite l'itération perception - calcul de la commande - action. Ainsi cette boucle de contrôle, qui s'exprime classiquement de la manière suivante :

```
While() {
        s := reception() ;
        c := f(s) ;
        envoi(c) ;
}
```

se réécrira :

$$\forall t, \ (\ |\ s_t := \text{reception}()\ |\ \text{envoi}(c_t)\ |\ c_t := \text{f}(s_t)\ |\ )$$

Dans cet exemple, plusieurs remarques peuvent être soulignées :

- dans la mise en œuvre en SIGNAL, l'itération (inhérente à ce type d'algorithme dès lors que l'on utilise un langage impératif séquentiel) est devenue totalement implicite. Le raisonnement ne se fait plus sur une des variables, dont les valeurs sont modifiées à chaque passage dans la boucle, mais sur des flots de données.

- dans le cas de la programmation classique, une gestion explicite des aspects bas niveau de l'implémentation est nécessaire. Ainsi le séquencement des différentes instructions, qui est fonction des dépendances entre les données, doit être géré manuellement. Avec SIGNAL, cette gestion explicite des dépendances n'est pas nécessaire, puisqu'elle est réalisée automatiquement.
- comme le montre l'indice t dans les équations schématiques présentées, la présence simultanée des signaux mis en jeu est parfaitement gérée par l'hypothèse synchrone.

De fait, l'utilisation de SIGNAL procure un niveau d'abstraction adéquat pour la spécification ainsi qu'un modèle de temps cohérent. Une description modulaire du processus d'asservissement visuel est donnée sur la Figure 5.2. Le programme SIGNAL réalisé à l'aide

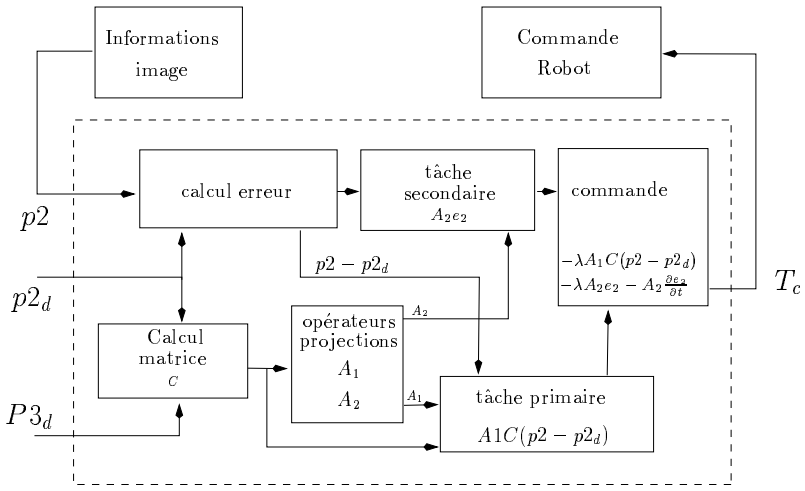


FIG. 5.2 – *Description modulaire de l'asservissement visuel*

de l'interface graphique [Bournai 93b] est présenté sur la Figure 5.3. Ce programme est divisé en trois modules principaux :

- le module d'entrée `CAMERA_OUTPUT` qui recupère les informations image `p2` fournies par le capteur à une cadence proche de la cadence vidéo (dans notre cas 12.5 Hz) ;
- le module de contrôle qui calcule la nouvelle commande du robot en utilisant l'approche fonction de tâche.
- le module de sortie `ROBOT_CONTROL` qui permet de transmettre la commande ainsi calculée au robot.

Le module de contrôle est lui-même décomposé hiérarchiquement en sous-modules. Graphiquement, les sorties d'un module sont reliées aux entrées d'un autre module par des liens qui sont le support des flots de données. Textuellement, la connexion de ces différents modules forme un système d'équations (voir Figure 5.4). Plus précisément, le module `PERFORMING_ERROR` calcule l'erreur entre le motif courant dans l'image et le motif à atteindre $p2 - p2_d$: `error := PERFORMING_ERROR{p2,p2d}`; le processus `C MATRIX`

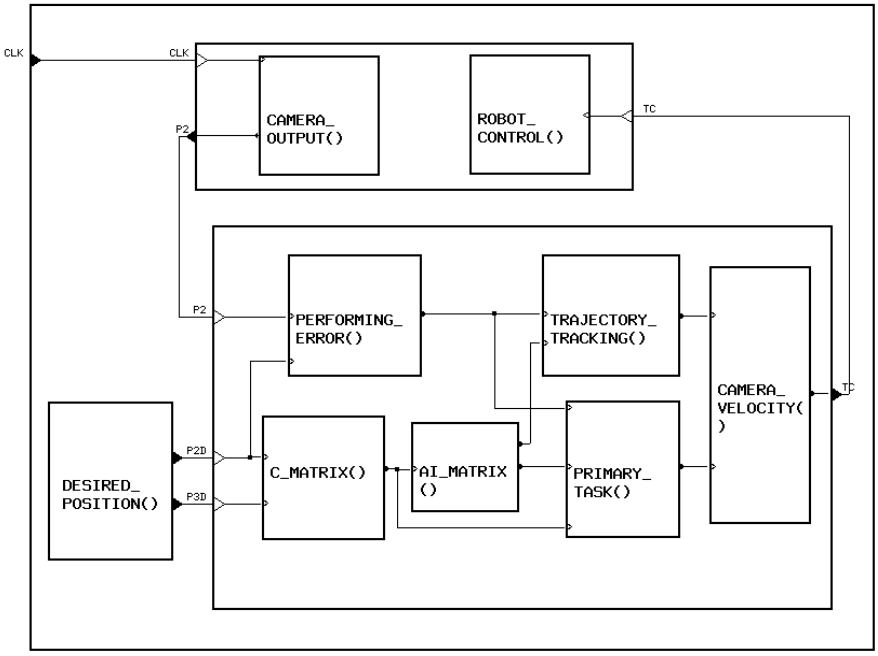


FIG. 5.3 – *Programme* SIGNAL *réalisant l'asservissement visuel (réalisé avec l'interface graphique de* SIGNAL*)*

```
(| p2 := CAMERA_OUTPUT{CLK}
 |(| error := PERFORMING_ERROR{p2,p2d}
   | C := C_MATRIX{P3d,p2d}
   | {A1,A2} := AI_MATRIX {C}
   | tau := PRIMARY_TASK{C,A1,error}
   | traj := TRAJECTORY_TRACKING{A2,error}
   | Tc := CAMERA_VELOCITY{tau,traj}
   |)
 | {P3d,p2d} := DESIRED_POSITION{}
 | ROBOT_CONTROL{Tc}
 |)
```

FIG. 5.4 – *Programme* SIGNAL *réalisant l'asservissement visuel.*

calcule la matrice C ; cette matrice C est ensuite transmise à un module réalisant les calculs de A_1 et A_2 ; A_1 est ensuite utilisée conjointement avec l'erreur `error` et C pour produire la commande de la caméra qui doit réaliser la tâche primaire : `tau := PRIMARY_TASK{C,A_1,error}` ;

Parallèlement, si la convergence est atteinte ($\|p2 - p2_d\| < \varepsilon$), le suivi de trajectoire correspondant à la tâche secondaire est déclenché[2] :

```
process TRAJECTORY_TRACKING =
  {? float A2[6][4], error[4]
   ! float traj[6] }
  (| traj := TRAJECTORY{A2} when Error_min{error} <threshold
                default NULL_MOTION{}
   |)
```

Enfin, la commande finale envoyée `Tc` est calculée en prenant en compte les deux flots de données `tau` et `traj` provenant des modules `PRIMARY_TASK` et `TRAJECTORY_TRACKING`.

Il convient de noter que tous ces flots de données sont synchrones et qu'ils ont la même horloge (imposée par la fréquence d'acquisition des images).

5.2.2 Processus flot de données et reconstruction 3D

Expression du parallélisme et du retard. D'autres opérateurs, dont le rôle est de décrire des comportements dynamiques et des comportements parallèles, sont disponibles dans le langage SIGNAL : le *retard* permet l'accès aux valeurs passées d'un signal et l'opérateur de *composition* peut être vu comme un opérateur de composition parallèle.

- **Le retard** donne la valeur passée d'un signal, ce qui est noté $Y_t = X_{t-d}$, avec la valeur initiale $Y_i = V_i$, pour $0 < i \leq d$; en SIGNAL, pour le cas simple où $d = 1$, on écrit:

  `ZX := X$1` avec l'initialisation `ZX init V0`

  Le retard est un opérateur monochrone, c'est-à-dire que `X` et `ZX` ont la même horloge. Ainsi, par exemple, un filtre moyennant défini par l'équation $y_t = (x_t + x_{t-1} + x_{t-2})/3$ s'écrit en SIGNAL :

  ```
  (| Y := (X + ZX + ZZX)/3 | ZX := X$1 | ZZX := X$2 |).
  ```

- **Le parallélisme.** L'opérateur de composition | que l'on a défini précédemment comme un opérateur réalisant l'union de deux systèmes d'équations peut de plus être considéré comme un opérateur de parallélisme entre plusieurs processus et est noté de la même manière :

  `(| P1 | P2 |)`

  Les communications entre ces processus se font de manière instantanée par l'intermédiaire de signaux.

2. Dans le cas présent, la tâche secondaire n'est déclenchée que si la tâche principale est réalisée. Ceci est dû à l'application traitée. Il va de soi que, de manière générale, cette tâche peut être activée à n'importe quel moment.

Application à la reconstruction 3D. Comme nous l'avons vu dans le chapitre 2, la méthode de reconstruction des primitives 3D que nous avons utilisée est basée sur la mesure du mouvement apparent de la primitive dans l'image et sur la mesure du mouvement de la caméra. Rappelons que, si le mouvement de la caméra est quelconque, l'estimation des paramètres de la primitive est biaisée et très sensible au bruit. C'est pourquoi la caméra est commandée par asservissement visuel. Au niveau de la mise en œuvre, deux besoins fondamentaux se font donc sentir :

- accéder aux valeurs passées d'un signal (vitesse de la caméra et position de la primitive dans l'image) à la date $t - n$: nous utilisons pour cela l'opérateur de retard.
- effectuer l'estimation de la primitive en parallèle au calcul de la commande de la caméra : l'opérateur de composition sera utilisé à cette effet.

Comme le montre l'équation (2.8) : $\underline{P} = \underline{P}(\underline{p}, \underline{\dot{p}}, T_c)$, l'estimation de la structure 3D d'une primitive est basée sur la mesure de $\underline{\dot{p}}$, la vitesse apparente de la primitive dans l'image, qui ne peut pas être calculée de manière explicite. De fait, $\underline{\dot{p}}$ est estimée en utilisant deux positions successives de la primitive (*i.e.*, $\underline{p}_t$ et $\underline{p}_{t-1}$). De même, la vitesse de la caméra $T_{c_{t-1}}$ entre ces positions successives doit être connue. Les valeurs passées de ces deux signaux $\underline{p}$ et T_c peuvent facilement être exprimées en utilisant l'opérateur de retard. Si `p2` est le flot de données portant la position de la primitive dans l'image et `Tc` celui portant la vitesse de la caméra alors schématiquement, le programme de la Figure 5.5 permet d'obtenir les paramètres 3D estimés de la primitive. Le processus `Estimation` comprend une série de transformations (appels de fonction) d'un flot de donnée en un autre. L'utilisation d'un langage flot de données prend ici toute son importance ; il n'est en effet pas nécessaire de créer explicitement des mémoires, comme ce serait le cas avec un langage impératif[3], pour accéder aux valeurs passées des signaux. Ici, une simple référence au flot de données portant cette variable est suffisante, charge au compilateur de transcrire ensuite cette référence.

Signalons de plus qu'un filtre glissant a été mis en œuvre à la sortie du processus d'estimation. Ce filtre calcule simplement la moyenne entre la valeur courante des paramètres et leurs valeurs à l'instant $t - 1$ et $t - 2$. Comme nous l'avons déjà vu, la mise en œuvre d'un tel filtre s'effectue de manière très simple en utilisant le retard.

```
(| P3_est := ESTIMATION{p2,Zp2,ZTc}
 | Zp2 := p2$1
 | ZTc := Tc$1
 |)
```

FIG. 5.5 – *Utilisation de l'opérateur de retard pour l'estimation*

L'autre aspect important dans la mise en œuvre de l'estimation concerne le contrôle du mouvement de la caméra. Comme on l'a déjà dit, le processus d'estimation par vision

3. Le retard est la forme que prend la mémorisation dans les langages flot de données. L'approche est donc différente, mais n'est pas opposée à celle des langages impératifs.

active utilise un mouvement de la caméra réalisé par asservissement visuel. La commande est donc effectuée en parallèle de l'estimation. La figure 5.6 montre le programme SIGNAL correspondant construit à l'aide de l'interface graphique du langage; la version textuelle est donnée sur le listing de la figure 5.7.

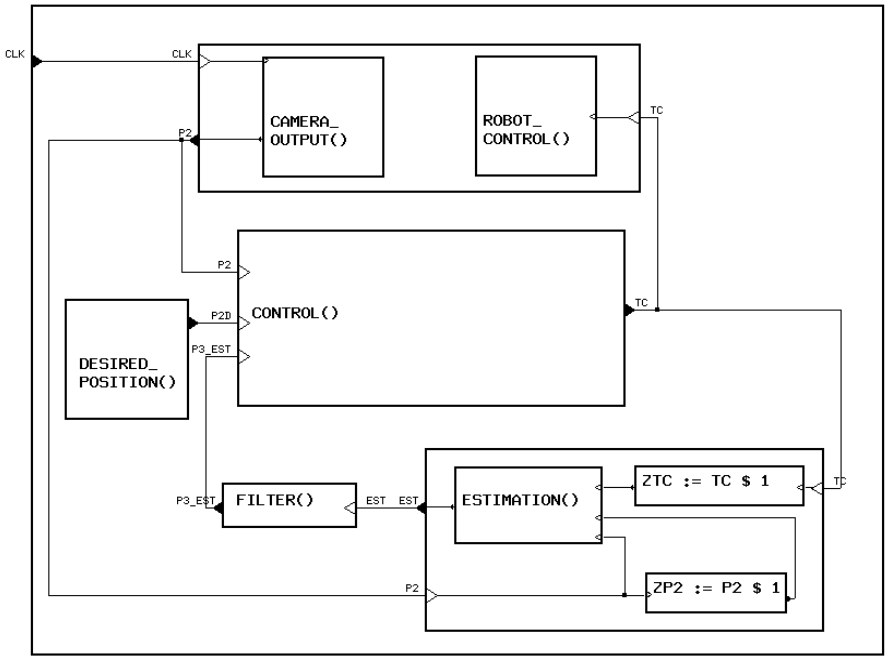


FIG. 5.6 – *Programme* SIGNAL *réalisant l'estimation des paramètres d'une primitive (réalisé avec l'interface graphique de* SIGNAL*).*

```
(| Tc := CONTROL{p2,p2d,P3_est}
 | p2d := DESIRED_POSITION{}
 | ZTc := Tc$1
 | P3_est := FILTER{est}
 | est := ESTIMATION{p2,Zp2,ZTc}
 | Zp2 := p2$1
 |)
```

FIG. 5.7 – *Programme* SIGNAL *réalisant l'estimation des paramètres d'une primitive.*

L'opérateur de composition est ici utilisé au sens du parallélisme. Ceci est réalisable car les signaux d'entrée et de sortie des processus de contrôle et d'estimation sont synchrones. Le processus `CONTROL` est identique à celui présenté dans la section précédente, excepté le fait que les paramètres 2D mesurés et les paramètres 3D estimés des primitives sont réinjectés dans le processus d'asservissement visuel qui met à jour, à chaque top d'horloge (ou "itération"), la matrice d'interaction.

5.3 Préemption et séquencement de tâches de vision

Divers domaines d'application, tels que le traitement du signal ou le contrôle de processus physiques, requièrent la possibilité de spécifier des comportements composés de l'enchaînement de différents modes d'interaction avec leur environnement. À cette fin, la notion d'*intervalle de temps* définie par des instants de début et de fin, et désignant la suite de ces occurrences, a été introduite dans le langage [Rutten 94]. Associer un intervalle de temps à un processus flot de données spécifie alors une tâche, c'est-à-dire une activité non-instantanée, et son intervalle d'exécution. Cette extension se présente sous la forme d'un préprocesseur appelé SIGNAL*GTi* [Rutten 95b]. Les domaines d'application de SIGNAL*GTi* comprennent le séquencement discret de tâches continues flot de données, mais aussi la préemption et la hiérarchisation de tâches.

5.3.1 Préemption et séquencement de tâches : SIGNAL*GTi*

Cette section présente les caractéristiques générales de SIGNAL*GTi*. Dans une application flot de données, la terminaison n'est pas spécifiée de manière explicite. De fait, elle peut se terminer uniquement en réaction à un événement extérieur ou en atteignant une certaine valeur par un de ses signaux internes. La terminaison de ce type de processus peut donc venir, soit de l'extérieur (interruption sur un événement particulier), soit de l'intérieur (le processus peut alors décider de sa propre terminaison). Cet instant final d'exécution du processus est noté ω. L'instant marquant le début de l'exécution de ce processus est noté α et est extérieur à celle-ci. En d'autres termes, le processus s'exécutera sur un intervalle de temps ouvert à gauche, fermé à droite $]\alpha, \omega]$. Les autres formes d'intervalles $[\alpha, \omega[$, $[\alpha, \omega]$, et $]\alpha, \omega[$ peuvent aussi être envisagées, mais nous choisissons $]\alpha, \omega]$ comme standard. En effet, dans un automate, la transition depuis un état de départ E_d vers un état d'arrivée E_a sur l'occurrence e d'un événement signifie que e arrive dans E_d, et ne sera dans E_a qu'après la transition. En d'autres termes, causalement, la fin d'un intervalle se décide à l'intérieur de l'intervalle, alors que son début est externe.

Les intervalles de temps ont été introduits de manière à permettre une décomposition de l'intervalle de temps nominal $]\alpha, \omega]$ (sur lequel l'ensemble de l'application est actif) en sous-intervalles (comme le montre la Figure 5.8) ainsi que l'association de ces sous-intervalles avec un processus. Un sous-intervalle `I`, semi-ouvert à gauche, et borné par les événements de début `B` et de fin `E`, est noté :

`I := ]B, E]`.

Il dénote la suite de ses occurrences : il a la valeur `inside` depuis la première occurrence de `B` jusqu'à l'occurence suivante de `E`; puis il prend la valeur `outside` jusqu'à l'occurrence suivante de `B`. On considérera par défaut, c'est-à-dire sauf mention contraire, que les intervalles sont initialement `outside`.

On peut définir sur les intervalles des opérations comme l'union et l'intersection. Le complémentaire `compl I` est défini comme l'intervalle dont la valeur est `outside` si `I` est

`inside` et réciproquement. Le filtrage des occurrences d'un signal `X` se trouvant à l'intérieur d'un intervalle `I` se note `X in I` ; celles à l'extérieur sont obtenues par `X out I`. Par ailleurs, on peut aussi désigner leur bornes effectives, c'est-à-dire les occurrences des événements qui font changer l'état de l'intervalle : les événements d'entrée sont dénotés par `open I` (définis par `B out I`), et les événements de fin `close I` (définis par `E in I`).

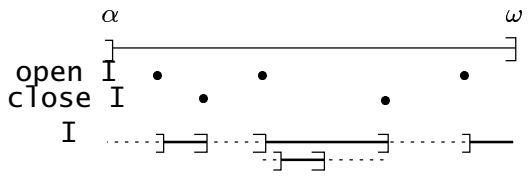


FIG. 5.8 – *Intervalles de temps subdivisant]α,ω].*

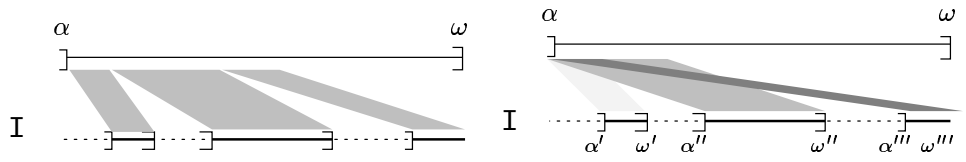
Les tâches. Associer un intervalle de temps à un processus flot de données spécifie une *tâche*, c'est-à-dire une activité non-instantanée **et** son intervalle d'exécution. Les tâches définies et actives sur l'intervalle]α,ω] représentent le cas par défaut : elle sont rémanentes pendant tout le temps d'exécution de l'application. À l'intérieur de son intervalle, le processus associé à la tâche est actif, c'est-à-dire qu'il est présent et s'exécute normalement. Hors de son intervalle, le processus est inexistant, il est donc absent et les valeurs des signaux internes à ce processus ne sont plus disponibles (d'une certaine manière, son horloge est coupée). Quand elle entre dans une occurrence de son intervalle d'exécution, une tâche peut être :

- démarrée dans son état courant, telle que conservée quand elle a été suspendue : pour un processus `P` sur un intervalle `I` on note `P on I`. La Figure 5.9.a illustre le comportement de la tâche `P on I`. On a le même comportement que sur l'intervalle `]`α`,`ω`]`, si ce n'est que son exécution est limitée aux occurrences successives de l'intervalle `I`.

- reprise à son état initial, si elle a été interrompue, ce qu'on écrit pour un processus `P` sur un intervalle `I` : `P each I`. Dans ce cas, comme le montre la Figure 5.9.b, le comportement de la tâche `P each I` est similaire au comportement attendu sur l'intervalle `]`α`,`ω`]` et ce, à chaque occurrence de l'intervalle `I`. Dans ce sens, chacune des occurrences successives de `I` est assimilable à un nouvel intervalle `]`α'`,`ω'`]`, `]`α''`,`ω'`]`, ..., pour `P`.

Les processus associés à un intervalle peuvent eux-mêmes être décomposés en plusieurs sous-tâches permettant ainsi la spécification d'une hiérarchie de comportements complexes.

Le contrôle de tâches. Le contrôle de tâches s'effectue en contraignant les intervalles et leurs événements d'entrée et de sortie et en leur associant la tâche correspondante. Il est ainsi possible de décrire le *parallélisme* ou bien le *séquencement* de tâches. Le parallélisme entre tâches est naturellement obtenu quand elles partagent le même intervalle ou quand ces intervalles se chevauchent. Le séquencement s'effectue en obligeant deux intervalles à

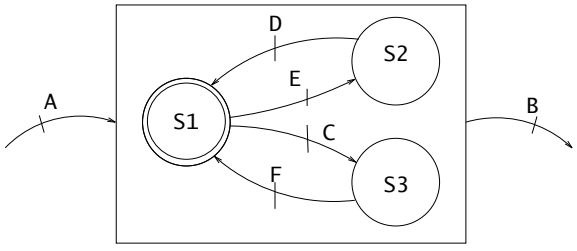


(a) `P on I` (b) `P each I`

FIG. 5.9 – *Tâche associant un processus* `P` *à un intervalle de temps* `I`.

se suivre (l'événement de fermeture du premier correspondant à l'événement d'ouverture du second). L'utilisation des opérateurs `on` et `each`, tels qu'ils ont été définis dans le paragraphe précédent permet de contrôler aisément les activités associées à ces intervalles et donc d'élaborer des comportements complexes. En utilisant soit le parallélisme, soit le séquencement, il est possible de spécifier des automates hiérarchiques parallèles ou encore des systèmes à places et transitions.

La Figure 5.10.a décrit à l'aide d'un automate hiérarchique parallèle un comportement simple. Une transition permet de passer de la place `S1` à la place `S2` sur occurrence de l'événement `E`, ou à la place `S3`, sur occurrence de l'événement `C`. Cependant, si `C` et `E` se produisent simultanément (*i.e.*, s'il existe des occurrences simultanées de `C` et `E`), le système entrera à la fois dans les places `S2` et `S3` et les processus associés s'exécuteront donc en parallèle. Ceci décrit en fait un "sous-comportement" qui est associé à une place particulière où l'on entre sur occurrence de l'événement `A` et d'où l'on sort sur occurrence de `B`. Ce système à places et transitions pourra être simplement codé en SIGNAL*GTi* comme indiqué sur la Figure 5.10.b.




```
(| S1 := ](D in S2) default (F in S3),
           E default C] init inside
 | S2 := ]E in S1, D] init outside
 | S3 := ]C in S1, F] init outside
 |) each ]A, B] init outside
```

(a) Automate hiérarchique (b) Spécification en SIGNAL*GTi*

FIG. 5.10 – *Séquencement, parallélisme et préemption de tâche en* SIGNAL*GTi*.

Le préprocesseur SIGNAL*GTi*. Afin de permettre l'utilisation des primitives de SIGNAL*GTi*, un préprocesseur permettant la réécriture d'un code SIGNAL*GTi* à l'aide des primitives de base du noyau de SIGNAL a été réalisé [Rutten 95b][Martinez 94]. Le fait que les primitives de SIGNAL*GTi* soient transcrites en SIGNAL présente un avantage important.

Le caractère flot de données d'une tâche ainsi que le contrôle de tâches sont ainsi mis en œuvre par le même langage, les modèles sur lesquels ils reposent sont ainsi identiques, ce qui permet de rester dans un formalisme synchrone unifié. La preuve et la vérification du programme dans sa globalité sont donc possibles.

5.3.2 Application aux stratégies de perception

Hiérarchisation des tâches. Comme nous l'avons vu dans les chapitres 3 et 4, un séquencement de sous-tâches d'asservissement visuel et/ou de reconstruction est nécessaire pour passer de la reconstruction d'une primitive à une autre, et ainsi obtenir, par construction incrémentale, une représentation complète de la scène. Afin d'éviter des requêtes conflictuelles, il faut concevoir un séquenceur donnant alternativement à une tâche le contrôle du système de vision. Ce séquenceur peut souvent être réduit à un simple automate (c'est souvent le cas dans les systèmes réactifs). Pour notre part, nous nous sommes intéressés à la conception d'un automate hiérarchique parallèle capable de gérer l'ensemble du processus de reconstruction d'une scène complexe (voir Figure 5.11 qui reprend le système global donné à la fin du chapitre 4). Chaque tâche est contrainte sur un certain intervalle de temps. Cet automate sélectionne et gère les actions à effectuer en fonction des événements 2D perçus dans l'image, des connaissances acquises et répertoriées au fur et à mesure de la reconstruction, ainsi que de la détection de la fin d'une tâche d'asservissement visuel.

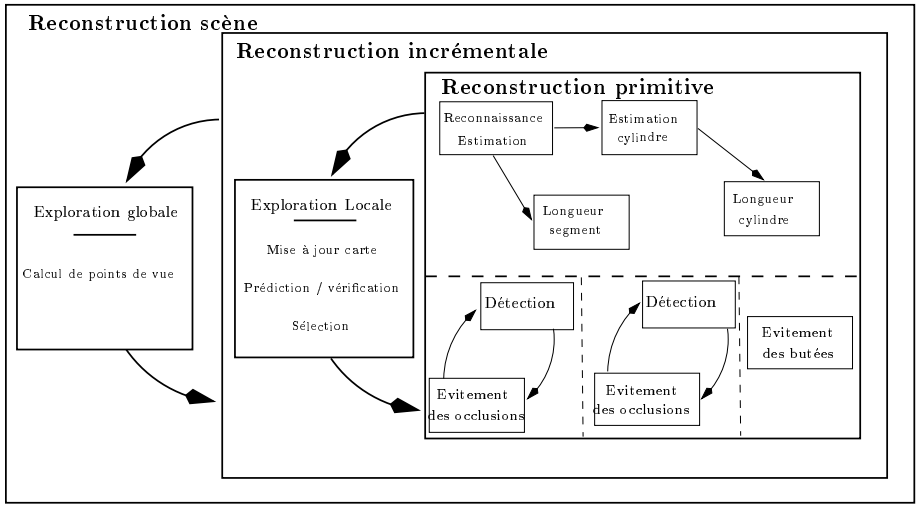


FIG. 5.11 – *Automate hiérarchique parallèle permettant un enchaînement des différentes tâches de vision.*

De manière plus précise, à chaque état de l'automate hiérarchique est associée une tâche, comme par exemple une tâche d'asservissement visuel, et un intervalle de temps pendant lequel cette tâche est active. En contraignant les tâches sur un intervalle de temps, l'utilisation de SIGNAL*GTi* autorise donc la combinaison de comportements (parallélisme entre tâches), la préemption ainsi que le séquencement de tâches. Il permet d'autre part le

```
Exploration :
(| IE := ] when SoSempty, when not SoSempty] init inside
 | IREC := comp IE
 | exploration each IE
 | Incremental_reconstruction each IREC
 |)
```

```
Incremental_reconstruction:
(| IC := ] close Icl default close Isl, SoSempty] init inside
 | IR := comp IC
 | Choice each IC
 | Primitive_estimation each IR
 |)
```

```
Primitive_estimation:
(| Optimal_estimation
 |(| Occlusion_avoidance_1 | Occlusion_avoidance_2 |)
 | Joint_limits_avoidance |)
```

```
Occlusion_avoidance_i:
(| Ir_i := ] New_Segment, Segment_Lost ] init outside
 | occlusion_avoidance each Ir_i |)
```

```
Optimal estimation:
(| Inat := ] close Isl default close Icl, Cyl] init inside
 | Nature each Inat
 | Icv := ] when Cyl, when (| prec |< ε_cv)] init outside
 | Cylinder_estimation each Icv
 | Icl := ] close Icv, when (| prec |< ε_cl)] init outside
 | Cylinder_length each Icl
 | Isl := ] when not Cyl, when (| prec |< ε_sl)] init outside
 | Segment_length each Isl |)
```

FIG. 5.12 – *Spécification de la hiérarchie de tâches décrite avec* SIGNAL*GTi*

traitement de la terminaison des tâches flot de données (dont on a défini le comportement mais pas la terminaison).

Une spécification simplifiée de notre application en SIGNAL*GTi* en gardant uniquement les aspects essentiels est donnée sur la Figure 5.12. Une description complète en est donnée dans [Martinez 94].

On retrouve dans l'automate de la Figure 5.11, comme dans la spécification de la Figure 5.12, les différentes étapes de la reconstruction d'une scène. Au niveau le plus bas de la hiérarchie, on retrouve le processus de reconstruction (`Optimal_estimation`) où sont effectuées la reconnaissance puis la reconstruction de la primitive. Ce niveau correspond à un séquencement de tâches d'asservissement visuel et d'estimation. Chacune de ces tâches est contrainte sur un intervalle de temps qui définit son début et sa terminaison. En parallèle de ce processus d'estimation, un certain nombre de tâches annexes sont réalisées, à savoir, l'évitement des butées articulaires et des occlusions pendant l'exécution de la

tâche secondaire (`Primitive_estimation`). Au niveau suivant, on retrouve le processsus de reconstruction incrémentale de la scène et la gestion des listes de segments 2D observés (`Incremental_reconstruction`). Enfin au niveau le plus haut, on retrouve le processus d'exploration globale de la scène et le calcul de points de vue (`Exploration`). La Figure 5.13 représente une trace possible du séquencement des différents intervalles de temps intervenant dans le processus de reconstruction de la scène.

Nous soulignons ici deux aspects importants de la spécification en termes d'intervalles de temps.

- **Terminaison - Séquencement.** Un processus flot de données, comme les tâches de vision, porte en lui la spécification complète de son comportement mais pas celle de sa terminaison. Cet aspect doit alors être défini séparément. Une façon de décider de la terminaison d'une tâche est de spécifier un critère dépendant des données capteurs ou des calculs effectués par le processus lui-même (par exemple la fin de la tâche de reconnaissance). L'évaluation de ce critère doit se faire à chaque instant ; cette évaluation devient alors un autre processus flot de données. L'instant où la condition est vérifiée peut être marqué par un événement discret, qui, causant la terminaison d'une tâche, peut aussi être la cause d'une transition vers une autre tâche au même niveau ou à un niveau plus élevé dans l'automate hiérarchique. Dans cette optique, ce type d'événement peut être utilisé pour marquer la fin de l'exécution interne d'une tâche (par exemple la fin d'une tâche d'évitement des occlusions provoquée par la perte du segment suivi ou par la fin de la tâche de reconstruction).

- **Parallélisme.** En utilisant l'opérateur de composition, le parallélisme entre deux tâches devient transparent. C'est le cas par exemple des processus d'évitement des butées articulaires ou des occlusions qui sont réalisées en parallèle de l'estimation des paramètres d'une primitive. Les flots d'informations en entrée de ces processus provenant du même capteur sont synchrones ; de fait, il peuvent donc être utilisés au même instant logique. En réalité, nous avons ici un parallélisme de spécification, le travail de synchronisation et de communication entre les différentes tâches étant laissé au compilateur.

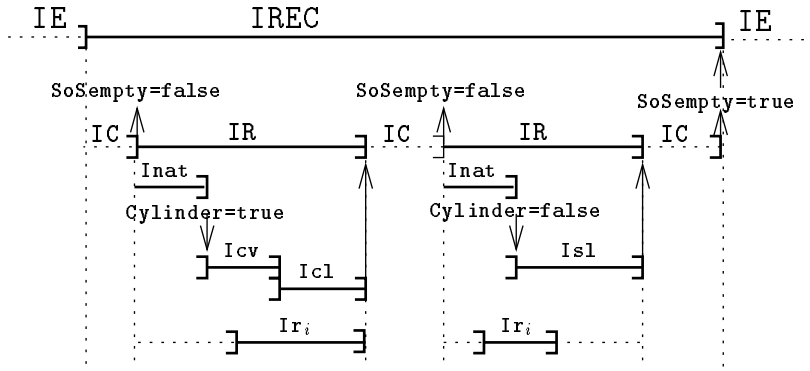


FIG. 5.13 – *Spécification du séquencement en terme d'intervalles : une trace possible.*

5.4 Résultats

Environnement de simulation. Les différentes tâches décrites dans les chapitres précédents ont été réalisées en utilisant l'approche synchrone : des tâches d'asservissement visuel tout d'abord (positionnement par rapport à une droite, un cylindre et avec des tâches secondaires correspondant à des suivis de trajectoire), puis des tâches de reconstruction 3D (d'une droite, d'un cylindre). Dans un premier temps, afin de valider l'approche retenue,

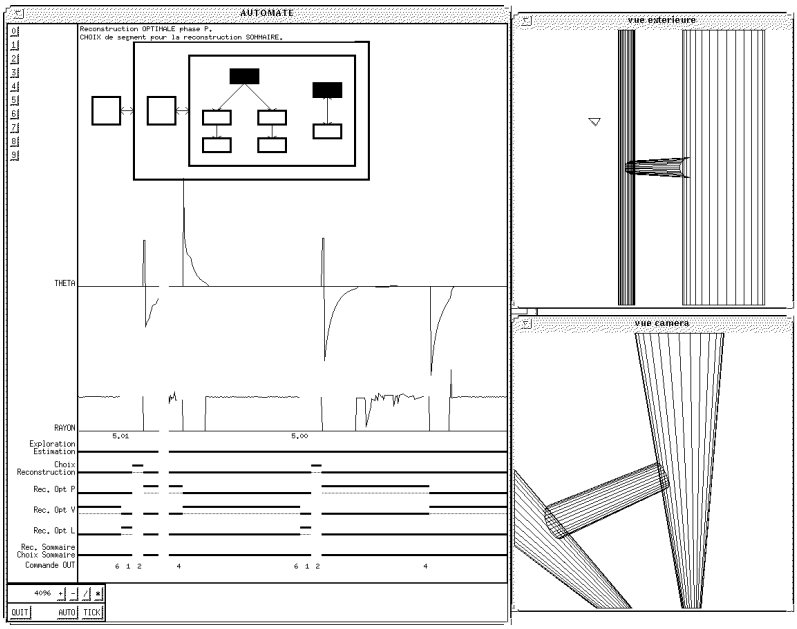


FIG. 5.14 – *Environnement d'exécution du programme* SIGNAL *en simulation*

la mise en œuvre à été effectuée en simulation. Tout l'environnement de simulation a été réalisé avec le langage SIGNAL (voir Figure 5.14). L'introduction de bruits éventuels sur les données capteurs et la position de la caméra a été effectuée afin de rendre les simulations plus réalistes.

Sur la Figure 5.14, on peut voir à la fois l'aspect continu des processus d'estimation et d'asservissement visuel (lignes marquées `theta` (erreur sur la position du cylindre dans l'image par rapport à la position optimale pour son estimation) et `rayon` (estimation du rayon du cylindre) au centre de la Figure, et l'aspect discret avec l'activation et la désactivation des intervalles de temps contraignant les différentes tâches (en bas). Des images synthétiques de la scène vue de la caméra et d'un observateur extérieur sont représentées en bas et haut à droite de la Figure.

Mise en œuvre sur site expérimental. Ces mêmes algorithmes ont ensuite été portés sur la cellule expérimentale de l'IRISA. Les résultats obtenus à l'exécution sont bien sûr similaires à ceux obtenus avec une mise en œuvre classique (en C). Concernant la reconstruction 3D de droites et de cylindres, signalons seulement quelques différences, dans la

mise en œuvre par rapport à ce qui a été décrit dans la section 5.2.2. Comme nous l'avons souligné dans le chapitre 2, afin d'avoir une estimation plus robuste, le mouvement de la primitive dans l'image $\underline{p}$ est calculé en utilisant la formule suivante : $\dot{\underline{p}}_t = (\underline{p}_t - \underline{p}_{t-n})/(n\Delta t)$ où Δt est la durée entre deux estimations successives et $n \geq 1$. Dans le cas de la mise en œuvre sur site expérimental, la formule décrite dans le programme 5.5 n'est pas celle qui est réellement utilisée. En effet, dans ce cas, l'estimation a besoin de la position de la primitive dans l'image à la date $t - n$: `Zp2 := p2$n` ainsi que de la vitesse mesurée de la caméra entre les dates t et $t - n$: `T := f(Tc$1, Tc$2, ..., Tc$n)`. Enfin, sur site expérimental, on ne peut plus considérer que la commande transmise au robot est pafaitement exécutée. Il convient donc d'utiliser dans le processus d'estimation non plus la consigne calculée mais la vitesse mesurée *a posteriori*. Le processus d'estimation qui découle de ces modifications s'écrit schématiquement comme indiqué sur la Figure 5.15.

```
(| P3_est := ESTIMATION{p2,Zp2,T};
 | Zp2 := p2$n;
 | T := f(Tc$1, Tc$2, ..., Tc$n);
 | Tc := mesure_vitesse();
 |)
```

FIG. 5.15 – *Processus d'estimation*

La figure 5.16 montre l'environnement d'exécution de l'application créé avec le langage SIGNAL. Cet environnement est similaire à l'environnement de simulation décrit sur la Figure 5.14.

5.5 Discussion

Notre système de reconstruction de scènes par vision active a aussi été implémentée en utilisant les langages plus classiques C et C++. Donner une description de cette dernière mise en œuvre n'est pas l'objectif de ce chapitre et ne présenterait en outre que peu d'intérêt. Cependant, cette double mise en œuvre donne l'opportunité de proposer une discussion sur les avantages et les inconvénients de la méthodologie synchrone de SIGNAL pour ce type d'application.

- Tout d'abord, il faut reconnaitre que SIGNAL n'est pas un langage adapté à la spécification de tout algorithme. Le calcul matriciel, le traitement d'images, les algorithmes mis en œuvre pour l'exploration globale de scène (les lancers de rayons en particulier) ne peuvent pas, de manière générale (ou alors avec difficulté), être programmés avec ce langage[4]. Dans ces exemples, les algorithmes mis en jeu ne sont pas de type flot de données et on voit mal en quoi on pourrait tirer avantage à une telle mise en œuvre en SIGNAL. Cependant, l'éventualité de l'occurrence de telles procédures lors de la conception d'une application a été prise en compte par les concepteurs

4. Cependant des travaux ont été réalisés pour améliorer la manipulation des tableaux et matrices (par exemple en traitement d'images [Le Guernic 92])

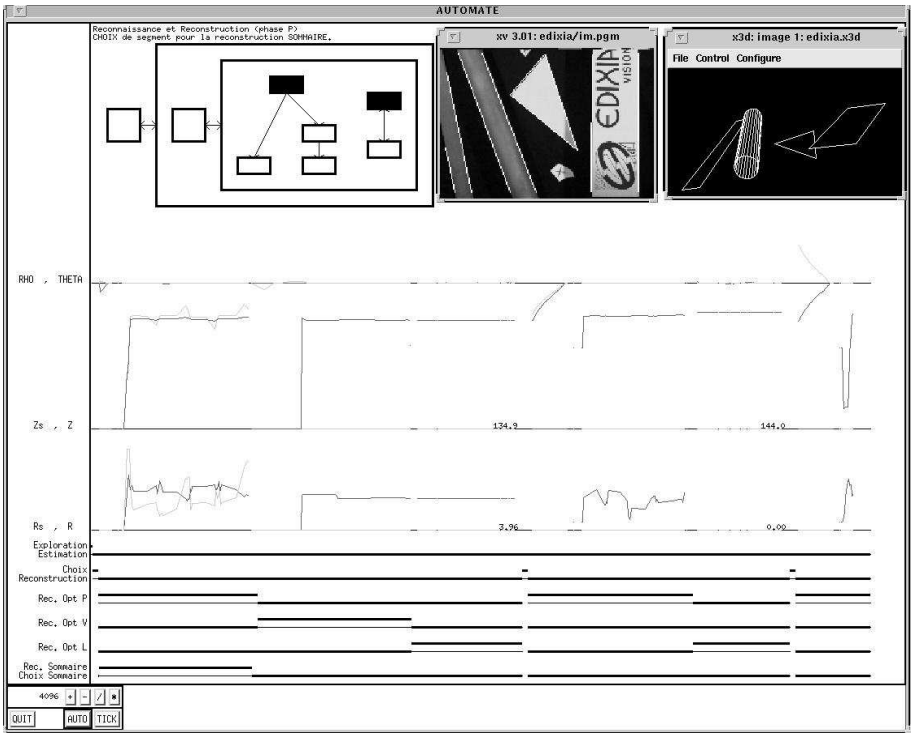


FIG. 5.16 – *Environnement d'exécution du programme* SIGNAL *sur le site expérimental*

du langage. Il est en effet possible depuis un programme SIGNAL de faire appel à des fonctions externes, écrites avec d'autres langages (C, Fortran ou autres). Cette méthode est utilisée dans notre application pour la réalisation des calculs matriciels (inversion de matrice, pseudo-inverse), ainsi que pour la gestion des listes de segment. Cependant, si la mise en œuvre de telles fonctions est réalisée avec un langage asynchrone, l'appel depuis SIGNAL à ces fonctions est réalisé de manière synchrone. Tant que plusieurs fonctions de ce type ne communiquent pas entre elles, le cadre synchrone est préservé (à condition cependant que la terminaison en temps borné de l'exécution de ces fonctions soit garantie).

– La gestion des entrées asynchrones et/ou des interruptions au cours d'une boucle de commande n'est pas supportée. Cependant, dans notre cas, le cycle est périodique et régulier, et ce type d'intervention n'est donc pas nécessaire. Une interruption de la boucle de commande reste cependant possible au niveau application à l'aide de SIGNAL*GTi* (permettant ainsi de gérer d'éventuelles pannes du système).

Si ces inconvénients existent et peuvent, dans certains cas, poser problèmes, les avantages d'une implémentation synchrone et flot de données sont eux très nombreux et apparaîssent fondamentaux dans le cadre de la mise en œuvre de systèmes réactifs temps-réel. Ces mérites d'un tel langage apparaissent à deux niveaux : au niveau du style de programmation et au niveau de l'environnement de programmation.

Si l'on considère dans un premier temps le style de programmation, SIGNAL semble avantageux à la fois au niveau tâche (mise en œuvre des lois de commande) et au niveau application (gestion de tâches) :

- l'aspect flot de données du langage est particulièrement adapté pour la spécification et la mise en œuvre des lois de commande en boucle fermée, et donc à l'asservissement visuel. SIGNAL est en effet adapté à la nature équationnelle et flot de données des algorithmes mis en jeu. On arrive ainsi à un style de programmation où l'aspect séquentiel et itératif des algorithmes classiquement utilisés pour ce type d'application disparaît totalement. Ainsi, le programme final est très proche des spécifications initiales ;

- la possibilité de spécifier explicitement ou implicitement des comportements parallèles s'est aussi averée très utile (pour la reconstruction 3D réalisée en parallèle de l'asservissement visuel par exemple). Une gestion explicite et manuelle du parallélisme dans les langages classiques s'avère souvent problématique ;

- l'intégration des différentes tâches flot de données au sein d'une application nécessitant des mécanismes de séquencement et de préemption de tâches a été réalisée à l'aide de SIGNAL*GTi*. Cette extension du langage autorise la définition d'intervalles de temps ainsi que l'association d'un processus flot de données à cet intervalle. SIGNAL*GTi* procure les opérateurs nécessaires à la spécification d'automates hiérarchiques parallèles. On dispose alors à la fois des avantages d'une programmation par automates (déterminisme et séquencement) et ceux d'une programmation à l'aide de langages parallèles (parallélisme entre tâches) sans en avoir les inconvénients. On obtient ainsi un langage combinant à la fois les aspects flot de données pour les tâches de bas niveau et les aspect préemption et séquencement de tâches. La mise en œuvre de comportements hybrides continu/évenementiel est alors possible et facilement réalisable (ce qui n'est pas fait de façon aussi intégrée dans le système ORCCAD).

Si la programmation de ce type d'application est largement simplifiée par l'utilisation de la combinaison de SIGNAL et SIGNAL*GTi*, les mérites principaux du langage se situent sans doute au niveau de l'environnement de programmation. La sémantique synchrone de SIGNAL est définie à l'aide d'un modèle mathématique reposant sur le calcul des horloges associées à chaque signal ou événement. Les programmes SIGNAL peuvent donc être considérés comme un ensemble d'équations de contraintes entre les différents signaux. Ce système d'équations est la base d'une série d'outils offerts par le compilateur :

- le compilateur synthétise automatiquement un ordonnancement des calculs à l'intérieur de la boucle de contrôle. Cet ordonnancement, qui est une source importante d'erreur quand il doit être réalisé à la main, est correct vis-à-vis des dépendances de données. Ceci implique que le code SIGNAL sera facile à modifier dès lors que la re-synthèse d'un tel ordonnancement est automatique.

- lors de la compilation, un graphe de dépendances entre les différents flots de données est déduit du système d'équations. Une analyse automatique des dépendances entre

les données est réalisée, ce qui permet de détecter les cycles de causalité ainsi que les inconsistances temporelles du point de vue des indices temporels des équations mises en jeu.

- le compilateur SIGNAL optimise globalement et automatiquement le graphe de dépendance (entre autre, une élimination de code mort est réalisée).

Ces outils qui réalisent une vérification statique du code SIGNAL ainsi qu'une optimisation de celui-ci ne sont pas les seuls disponibles. L'environnement de SIGNAL propose également un système d'analyse des comportements dynamiques des programmes. L'utilisation de cet outil nommé SIGALI [Dutertre 92], [Le Borgne 91], [Le Borgne 93], [Marchand 96], permet de vérifier des propriétés comme la vivacité (absence de deadlock), l'attractivité (prouver que le système finira par arriver dans tel état) ou au contraire d'inaccessibilité (le système n'ira pas dans un état). D'autres propriétés spécifiques à l'application peuvent aussi être vérifiées. Ces outils de vérification (qu'ils réalisent une vérification statique ou dynamique) sont importants à deux niveaux : du point de vue du développeur, il est important de vérifier que le système mis en œuvre est conforme aux spécifications et qu'il a le comportement attendu ; d'autre part, sur le plan de la sécurité de fonctionnement, ces outils peuvent aider à la certification de systèmes dits critiques.

Ces outils disponibles dans l'environnement synchrone de SIGNAL sont importants dans un contexte d'ingénierie du logiciel de manière générale. Une analyse beaucoup plus complète des avantages de la méthodologie synchrone de SIGNAL est donnée dans [Gautier 94]. Notons finalement que ces outils sont généralement disponibles avec l'ensemble des langages synchrones comme ESTEREL ou LUSTRE, alors que ce n'est pas le cas pour les autres langages.

5.6 Conclusion

L'objet de ce chapitre était de démontrer que les langages synchrones sont très adaptés à la spécification et à la mise en œuvre des systèmes réactifs mettant en jeu des boucles de commande perception - action.

Nous avons montré que la spécification de notre système de vision active était réalisable avec le langage synchrone SIGNAL. Il est vrai que cela peut être fait avec d'autres langages, et il faut d'ailleurs reconnaitre qu'un langage comme SIGNAL ne peut pas être utilisé pour tous les types de calcul ou d'application. Il apparaît cependant qu'une application réalisée à l'aide d'un langage synchrone est une application correcte au sens où elle sera conforme aux spécifications. En effet, ce langage, réalisant de manière automatique un certain nombre de traitements classiquement réservés au concepteur, lui évite un certain nombre de tâches qui sont souvent source d'erreurs. Ainsi la gestion de l'ordre des calculs, la composition parallèle de processus (et des synchronisations/communications qui en découlent), la préemption et le séquencement de tâches, enfin, et sans être exhaustif, la génération de code exécutable sont réalisés *automatiquement* et *correctement*.

En conclusion, nous pensons qu'une approche synchrone et flot de données de ce type

d'application est avantageuse pour trois raisons principales :

- les langages synchrones reposant sur des bases formelles désormais bien établies, le calcul et l'analyse des programmes sont automatiques et sont implémentés dans un ensemble d'outils, allant du compilateur (qui outre la génération de code, réalise une vérification statique du programme) aux outils permettant de faire des preuves sur le comportement dynamique des programmes ;
- le style de programmation qui est proposé ici dans le cadre de l'application traitée est totalement indépendant de l'aspect séquentiel inhérent à l'architecture classique des ordinateurs. De ce fait, le programme SIGNAL qui en découle est très proche des spécifications issues de l'automatique ;
- enfin, les méthodes proposées permettent de considérer de manière unifiée les différents aspects de l'application, c'est-à-dire depuis l'aspect continu avec la mise en œuvre des tâches flot de données (asservissement visuel et estimation 3D) jusqu'à l'aspect événementiel et discret avec la conception de systèmes autorisant la parallélisation de tâches et la hiérarchisation de tâches préemptives. Si la première partie a déjà fait l'objet de démonstrations en SIGNAL, notons que c'est la première fois que SIGNALGTi a été utilisé pour une application de taille significative.

Conclusion générale et perspectives

> C'est là une histoire sans fin. Car le monde entier est encore une Amérique. Et les mots *terra incognita* – territoire icconnu – sont bien les plus prometteurs que l'on ait jamais écrits sur les cartes de la connaissance humaine.
>
> Daniel Boorstin, *Les Découvreurs*,
> Random House, New York 1993

Le travail que nous venons de présenter apporte sa contribution au difficile problème de la reconstruction et de l'exploration de scènes dans un contexte de vision active. L'étude proposée ici s'est limitée à des environnements assez restreints (objets statiques, scènes constituées d'objets polyédriques et de cylindre droits,...) mais avec comme ambition future la reconstruction de scènes de plus en plus complexes.

A la base du processus de reconstruction, nous avons choisi une méthode qui consiste à contraindre les mouvements de la caméra de manière à obtenir une estimation précise et robuste de primitives géométriques paramétrables telles que les droites, les cylindres, etc. À cet aspect *continu* du processus de reconstruction que constitue l'estimation des paramètres des primitives, il est nécessaire de réaliser des stratégies de reconstruction et d'exploration de la scène. Cette reconstruction est de caractère *événementiel* et est pilotée par la découverte de nouvelles primitives dans l'image. L'approche que nous avons définie consiste à sélectionner automatiquement les informations images pertinentes puis à focaliser successivement la caméra sur les différentes primitives de la scène afin de les reconnaître et ensuite de les reconstruire. Outre ces phases de focalisation et de reconstruction des primitives observées, des phases d'exploration ont été introduites afin de considérer l'ensemble des primitives de la scène, même si celles-ci n'étaient pas initialement visibles. Schématiquement, la reconstruction de la scène se fait donc en deux étapes principales :

- La première étape, qui inclut la reconstruction 3D, permet de reconstruire de manière incrémentale l'ensemble des primitives qui apparaissent dans le champ de vision de la caméra. Nous avons appelé cette phase **exploration locale** car elle ne fait appel qu'à des informations disponibles localement.

- Par contre, dans les autres cas, quand toutes les primitives précédemment observées ont été reconstruites, une stratégie différente a été mise en œuvre afin de focaliser la caméra sur des zones de la scène n'ayant pas encore été observées. Cette étape d'**exploration globale** a eu pour objectif d'assurer une reconstruction aussi complète que possible de la scène.

Le système de vision ainsi développé est totalement autonome et permet d'obtenir avec une excellente précision la cartographie complète d'une scène composée de segments et de cylindres, sans *a priori* initial sur leur nombre, leur disposition et leurs dimensions.

Notre contribution intervient ainsi dans quatre domaines complémentaires :

- la finalisation d'une méthode locale de reconstruction 3D par vision active. Une méthode de reconnaissance de primitives, basée sur un test de maximum de vraissemblance [2], ainsi qu'une méthode d'évitement des butées au cours d'un asservissement visuel [4][5] ont notamment été présentées ;

- le développement d'une approche locale de reconstruction de scènes reposant sur une approche de prédiction / vérification d'hypothèses gérées à l'aide de réseaux Bayesiens. Cette approche permet d'obtenir une représentation de plus haut niveau des objets considérés tout en traitant les problèmes locaux d'occlusion ;

- le développement d'une méthode originale et efficace d'exploration 3D, permettant d'assurer la complétude de la reconstruction et prenant en compte les obstacles de la scène. Cette méthode repose sur l'optimisation par ICM multi-échelle d'une fonction de coût adéquatement modélisée [2][6][12];

- la spécification et la mise en œuvre par le langage synchrone SIGNAL des algorithmes développés. En particulier, l'intégration au sein du même formalisme de la dualité continu / événementiel avec SIGNAL/ SIGNAL*GTi* nous parait être un point important [1][3][7][13].

Les méthodes que nous avons développées ont été mises en œuvre sur la cellule de vision robotique de l'IRISA. Elles permettent de reconstruire en temps réel de façon précise, robuste et complète, un environnement 3D composé de plusieurs primitives. Les objets traités (cylindres, polyèdres) sont encore relativement simples. Le passage à la reconstruction d'objets plus complexes constitue bien évidemment l'une des perspectives de ce travail.

Un certain nombre de questions relatives aux travaux effectués restent à explorer. Nous en proposons ici quelques unes qui constituent des voies d'investigation intéressantes dans la suite des travaux présentés. Il serait notamment souhaitable d'étudier les points suivants :

- la prise en compte d'objets plus complexes que les droites et les cylindres. Ceci passe bien sûr par la reconstruction 3D d'objets non polyédriques complexes ;

- les stratégies de perception de scènes complexes avec ou sans reconstruction explicite de l'environnement ;
- enfin, le couplage perception / action pour l'analyse de scènes dans un cadre multi-capteurs.

Concernant le premier point, les travaux qui ont déjà été réalisés et présentés dans le premier chapitre sur le thème de la reconstruction 3D par vision active portent sur des primitives géométriques relativement simples (points, droites, cylindres, sphères,...). Par contre, des études portant sur la reconstruction d'objets plus complexes restent à réaliser. Si la reconstruction de tels objets par des techniques discrètes commence à être classique (par exemple [Boyer 96]), l'utilisation efficace de techniques continues est un problème difficile qui reste largement ouvert (les premier travaux publiés sur ce domaine sont dûs à Cippola [Cipolla 95]). Cette voie de recherche ouvre des perspectives de travail assez importantes. Plusieurs aspects devront être considérés. Dans un premier temps, il faudra considérer la reconstruction en elle-même de l'objet (ou de sa surface) par vision monoculaire en se basant sur le mouvement de l'objet dans l'image. L'observation du mouvement apparent des contours occultants dans l'image associée à l'observation du mouvement de la caméra doit permettre de remonter à une paramétrisation de la surface observée (position d'un point 3D et de la normale à la surface par exemple). Cependant, ces méthodes étant relativement sensibles au bruit, il conviendra ensuite d'optimiser la reconstruction en définissant des mouvements adéquats de la caméra. Comme dans le cas d'une primitive géométrique simple, il doit exister des positions des contours occultants dans l'image ainsi que des mouvements de la caméra capables d'obtenir une estimation non-biaisée et robuste des paramètres de la surface 3D. Nous tenterons de générer ces mouvements en utilisant l'asservissement visuel. De fait, il sera nécessaire de confronter l'asservissement visuel à des formes géométriques complexes (tels que, par exemple, des splines). Des algorithmes portant sur l'extraction et le suivi d'objets de cette nature sont à mettre en œuvre dans le prolongement de ceux étudiés dans l'équipe [Bouthemy 89], [Boukir 93b]. Enfin, il n'apparaît pas réaliste de considérer qu'une reconstruction globale de l'objet puisse se faire sans (autres) difficultés. Dans le cas d'un objet quelconque, nous serons par exemple confrontés à des problèmes tels que des occlusions ou des changements de topologie de l'objet considéré. Le calcul explicite de nouveaux points de vue, reposant sur des algorithmes similaires, dans l'esprit, à ceux proposés dans cette thèse, peut s'avérer utile pour résoudre ces problèmes. De plus, à ce niveau, la reconstruction d'un objet quelconque (ou d'une surface) ne pouvant se faire que localement, des stratégies de perception déterminant les mouvements de la caméra aboutissant à une reconstruction complète de l'objet devront être mises en œuvre. En particulier, l'analyse de la complétude de la reconstruction est un problème rarement pris en compte dans ce type d'étude et qui se situe dans la suite logique de ce travail.

Dans le cadre de l'analyse de scènes complexes pour la réalisation de tâches robotiques particulières dans un univers inconnu, la mise en œuvre de stratégies de perception de haut niveau est nécessaire. L'absence de modèle sur la scène interdit toute planification

(simple) de la tâche à réaliser. Ces stratégies de perception ont donc pour objectif de déterminer quels sont les informations (géométriques ou autres) pertinentes et quelles sont les mouvements de la caméra qui permettront de réaliser la tâche spécifiée et ce, en réaction aux informations acquises par le capteur, sans planification explicite d'une séquence d'actions à effectuer. Dans cette optique, traditionnellement, on oppose les approches reconstructionnistes, visant à une reconstruction de l'environnement (ou d'une partie de cet environnement) préalablement à la réalisation de la tâche, aux approches intentionnelles (*purposive vision*) visant à extraire de la séquence d'images acquises uniquement les informations pertinentes pour la réalisation de la tâche spécifiée. Ces deux approches ne nous paraissent pas toujours antagonistes. À partir du moment où l'acquisition d'un modèle tridimensionnel de la scène est la tâche à effectuer ou est absolument nécessaire à sa réalisation, il parait alors intéressant, comme nous l'avons fait en grande partie dans le cadre de cette thèse, de se placer alors dans le cadre de la vision intentionnelle pour réaliser cette reconstruction.

Il est clair qu'une approche par vision intentionnelle nécessite une connaissance *a priori* du type d'informations que l'on s'attend à trouver dans l'environnement. Une manière élégante de représenter cette connaissance, tout en tenant compte de l'incertitude qui lui est associée, est l'utilisation de réseaux Bayesiens. Le travail réalisé dans le cadre de notre thèse sur ce sujet a montré qu'il était possible d'associer, dans de tels réseaux, à la fois les connaissances acquises sur la scène et les actions à effectuer pour réaliser la tâche spécifiée. Les réseaux Bayesiens permettent également de prendre en compte de nouvelles observations et de propager leur influence au reste du réseau, ainsi que de quantifier les incertitudes liées à la modélisation et aux observations dans le cadre de la théorie des probabilités. L'utilisation conjointe des réseaux Bayesiens et des techniques de prédiction/vérification d'hypothèses nous paraît une voie de recherche intéressante. En particulier, l'intégration au sein même du réseau Bayesien des actions de perception est ce qui fait l'originalité de cette approche. Les résultats que nous avons obtenus dans le cadre de cette thèse sont très encourageants et ont montré l'efficacité d'une telle approche. D'autre travaux récents confirment ce point de vue pour la réalisation de tâches différentes (voir par exemple les travaux récents de Rimey et Brown [Rimey 94], de Djian et Rives [Djian 96], ou encore de Buxton et Gong [Buxton 95]). Les réseaux Bayesiens que nous avons introduits l'ont été dans le but de guider la reconstruction de scènes polyédriques. Or, il semble possible de confronter cette approche à l'étude de scènes plus complexes. On peut dans cette optique considérer soit le cas d'objets simples mais dans un environnement complexe, soit le cas d'un unique objet complexe. Dans la première hypothèse, l'étude de réseaux complexes de cylindres, du type de ceux que l'on trouve dans des environnements nucléaires, est très envisageable dans le prolongement direct de nos travaux de thèse. Les techniques de prédiction/vérification d'hypothèses, couplées à de tels réseaux y trouveraient un terrain d'application intéressant. Dans la deuxième hypothèse, l'utilisation de techniques similaires peut aussi être envisagée pour gérer les problèmes d'occlusions ou de changement de topologie. Ces problèmes sont en effet très difficiles à résoudre pour un système autonome et des réseaux de ce type peuvent porter en eux la connaissance d'un "expert" et proposer des solutions adaptées aux difficultés rencontrées.

Le cadre "reconstructionniste" n'est cependant pas le seul envisageable. En particulier, il serait intéressant de développer l'aspect stratégies de perception/action ainsi que le couplage continu / événementiel sous-jacent à l'approche que nous avons proposée. Pour cela, il serait intéressant de considérer, avec l'utilisation conjointe de plusieurs caméras, une problématique nouvelle et non moins importante dans le cadre de la perception active. Il ne s'agirait plus uniquement d'analyser une séquence d'images fournies par un unique capteur, comme nous l'avons fait durant cette thèse, mais de manipuler les observations d'une même scène par plusieurs capteurs animés chacun d'un mouvement commandable propre. Ce thème est celui du contrôle dynamique, pendant l'exécution d'une tâche donnée, de caméras embarquées sur l'effecteur d'un robot manipulateur. Il peut s'agir de confronter les approches de planification statique de points de vue [Tarabanis 95a], [Abrams 93], [2], [6] avec les approches dynamiques [Nelson 94], [4], [5] reposant sur une commande continue de la caméra en réaction aux informations visuelles. L'objectif final serait de prendre en compte ces deux types d'approches, l'une à un niveau global de perception, et l'autre à un niveau local. Concernant la partie dynamique du processus de planification de points de vue, le principe consisterait à spécifier le mouvement de la caméra sous la forme d'une tâche secondaire, modélisant des contraintes géométriques, compatible avec la tâche visuelle spécifiée en s'aidant des informations fournies par le second capteur. D'autre part, la spécification de tâches complexes, comme la manipulation d'objets, imposera également le développement de stratégies de plus haut niveau nécessitant l'enchaînement d'un certain nombre de sous-tâches visuelles (préhension, assemblage, zoom sur une partie de la scène, etc.). La réalisation de chacune de ces sous-tâches nécessitera le calcul de points de vue permettant d'assurer au mieux leur exécution. Un extension logique serait ensuite de considérer le cas d'un environnement dynamique où les objets manipulés sont en mouvement.

Enfin, signalons que le langage SIGNAL et SIGNAL*GTi* semblent encore très adaptés à la conception de tels systèmes de vision. La mise en œuvre des lois de commande, le séquencement et/ou l'exécution en parallèle des différentes sous-tâches mises en jeu, l'utilisation conjointe des deux manipulateurs pour des tâches d'asservissement visuel pourraient aussi permettre de nous intéresser à de nouveaux types de problèmes liés à l'utilisation du langage SIGNAL. Dans le contexte multi-capteurs en particulier, les aspects de synchronisation ainsi que de programmation synchrone distribuée (partage de code SIGNAL entre différentes plate-formes matériel, distribution d'algorithmes de commandes sur plusieurs machines), ou encore la vérification de propriétés dynamiques pourront être envisagés.

Annexe A

Conditions expérimentales

L'architecture du banc expérimental dont nous disposons à l'IRISA est présenté sur la Figure A.2. Le système est composé de trois éléments principaux :

- une caméra CCD SONY embarquée sur un robot cartésien AFMA à 6 degrés de liberté (voir Figure A.1).

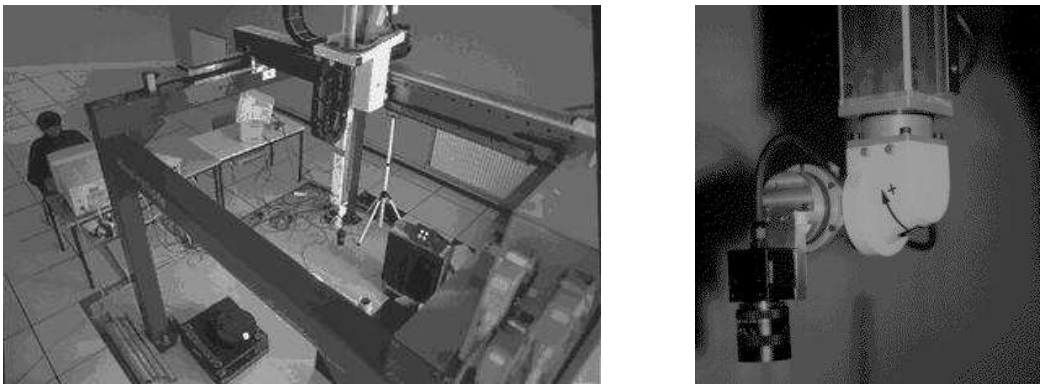


FIG. A.1 – *Cellule de vision dynamique active de l'* IRISA *: caméra CCD montée sur l'effecteur du robot AFMA*

- une carte mono-processeur de traitement d'images EDIXIA IA 1000. Celle-ci permet l'acquisition d'images à la cadence vidéo (20 ms pour l'acquisition d'une trame). Les dimensions de la trame mémorisée sont de 256 lignes et 730 colonnes. C'est sur cette carte que l'algorithme de suivi de segments [Boukir 93a], utile à la reconstruction 3D et à la détection de segments en mouvement (observateurs), est implanté.
- enfin, une station SUN sur laquelle sont calculées les commandes, les estimations des paramètres des primitives (voir Chapitre 2 pour les détails), le traitement d'image permettant de construire les listes de segments observés, ainsi que les algorithmes d'exploration de scènes. Les processus SIGNAL sont implantés sur cette station de travail.

L'architecture informatique de ce système expérimental est présentée sur la Figure A.2.

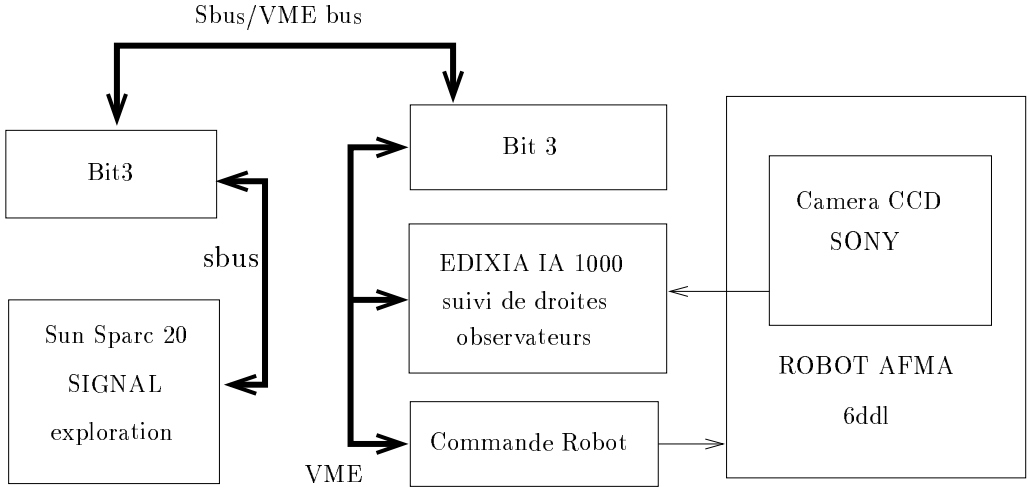


FIG. A.2 – *Architecture du système utilisé*

Annexe B

Suivi de segments

Pour le suivi de segment dans une séquence d'images, nous avons utilisé une méthode reposant sur une approche locale qui en fait ne manipule jamais explicitement de segments de contour. Cette méthode [Boukir 93a] s'appuie sur un algorithme de calcul des éléments de contour en mouvement (ECM) [Bouthemy 89].

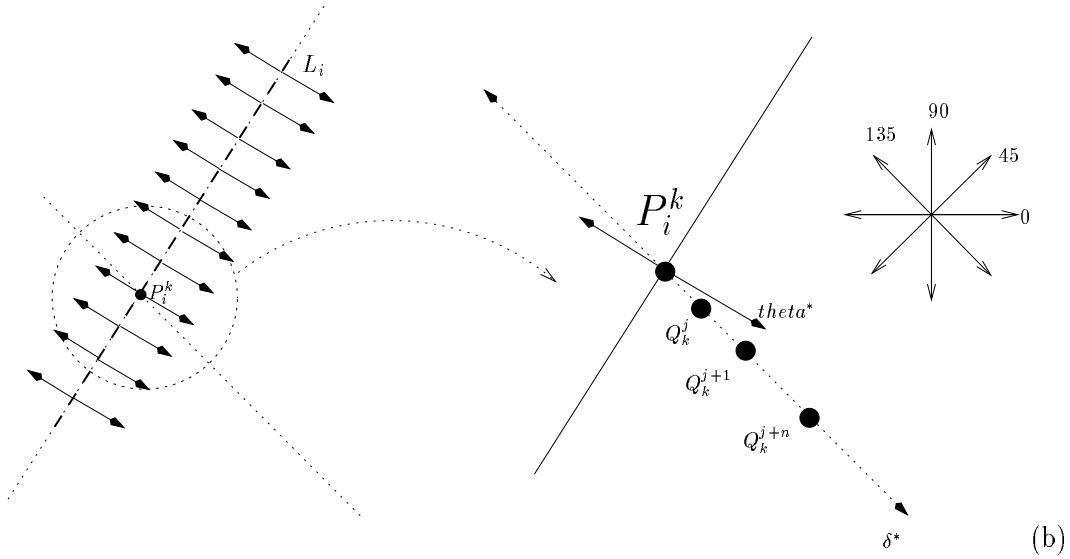


FIG. B.1 – *Suivi de segment (principe élémentaire) (a) recherche des ECM dans la direction normale au contour (b) choix de la direction de recherche*

Cette méthode consiste, non pas à suivre dans une séquence d'images un segment réel, mais à rechercher dans l'image $i+1$ le correspondant d'un segment virtuel S_i présent dans l'image i. Ce segment virtuel S_i correspond en fait à une liste de points $L^i = \{P_k^i\}$. Le suivi revient à chercher dans l'image $i+1$ le correspondant P_k^{i+1} de chacun de ces points dans la direction normale au contour (voir Figure B.1.a). Afin d'assurer des contraintes de temps réel, en pratique uniquement quatre directions δ sont prises en compte $\delta = \{0^o, 45^o, 90^o, 135^o\}$. Parmi ces quatre directions, la direction δ^* la plus proche de la direc-

tion θ^* est retenue. La recherche des correspondants P_k^{i+1} des points P_k^i est effectuée dans cette direction δ^* (voir Figure B.1.b).

Parmi les points possibles Q_k^j, le seul point retenu et inséré dans la liste L_{i+1} est celui dont la vraisemblance ξ d'appartenir à un élément de contour en mouvement (ECM) est maximale (les détails du calcul de cette vraisemblance sont donnés dans [Boukir 93a]). Enfin, les paramètres de la nouvelle droite passant par les points L_{i+1} sont calculés en utilisant une technique de moindres carrés.

Précisons que cet algorithme de suivi de droite est exécuté en 80ms sur la carte de traitement d'image EDIXIA IA 1000. Le suivi de plusieurs droites en parallèle est possible. Il faut alors compter 40 ms supplémentaires par droite. Afin de rester compatible avec les contraintes temps réel imposées par l'asservissement visuel, nous nous sommes toujours limité au suivi d'un nombre maximal de trois droites (deux au plus pour la reconstruction d'un cylindre et les autres pour la détection des occlusions).

Annexe C

Calibration et transformations géométriques

C.1 Calibration de la caméra

La calibration de la caméra et l'identification de la matrice de passage entre le poignet du robot et le repère de la caméra sont effectuées en utilisant respectivement les techniques décrites dans [Chaumette 89] et [Tsai 87]. Ces procédures sont indispensables pour d'une part, effectuer correctement les consignes calculées par la commande et, d'autre part, pour assurer la cohérence entre les données 2D mesurées dans l'image avec les données 3D estimées (passage de données en pixels dans le plan image en données métriques dans le repère caméra).

Concernant la calibration de la caméra, deux classes de paramètres sont déterminés :

- les paramètres intrinsèques qui dépendent de ses caractéristiques électroniques et physiques.
- les paramètres extrinsèques qui dépendent de sa position et de son orientation.

L'origine du repère de la caméra est située au centre de projection O (voir Figure 2.2). L'axe $(O, \vec{X})$ est orienté comme les abscisses du plan image et l'axe $(O, \vec{Y})$ comme les ordonnées. La distance minimale du centre de projection au plan image, appelée distance focale, est notée f. L'intersection de l'axe $(O, \vec{Z})$ avec le plan image a pour coordonnées dans l'image numérisée (x_c, y_c).

Dans le repère caméra, la projection perspective d'un point $M(X, Y, Z)$ sur le plan image est $m(x, y)$ avec :

$$\begin{cases} x = f \,.\, X \,/\, Z \\ \\ y = f \,.\, Y \,/\, Z \end{cases} \tag{C.1}$$

Si l'on néglige les distorsions radiale et tangentielle, la position $m_p\,(x_p, y_p)$ de m dans

l'image numérisée (en pixel) est donnée par :

$$\begin{cases} x_p = x_c + x \,/\, l_x \\ \\ y_p = y_c + y \,/\, l_y \end{cases} \tag{C.2}$$

où l_x et l_y sont des coefficients proportionnels à la taille d'un pixel élémentaire.

A partir de ces équations, on peut écrire le modèle complet de la caméra :

$$\begin{cases} x_p = x_c + F_x.\frac{X}{Z} \\ \\ y_p = y_c + F_y.\frac{Y}{Z} \end{cases} \tag{C.3}$$

où $F_x = f/l_x$, $F_y = f/l_y$.

Les paramètres à identifier, appelés paramètres intrinsèques de la caméra, sont alors x_c, y_c, F_x et F_y.

Les équations (C.3) ne sont pas directement exploitables car l'on ne connaît pas en pratique les coordonnées (X, Y, Z) de M dans le repère de la caméra. Par contre, si l'on se fixe un repère objet dans lequel on connaît les coordonnées $(X_{o_i}, Y_{o_i}, Z_{o_i})$ d'un ensemble de points M_i, il est possible d'identifier les paramètres intrinsèques de la caméra, de même que l'orientation et la position du repère objet par rapport au repère de la caméra, soit la matrice M_c^o. Les paramètres représentant cette matrice sont appelés paramètres extrinsèques.

Soit M un point de coordonnées (X, Y, Z) dans R_c et (X_o, Y_o, Z_o) dans R_o, on a :

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = M_c^o . \begin{pmatrix} X_o \\ Y_o \\ Z_o \\ 1 \end{pmatrix} \text{ avec } M_c^o = \begin{pmatrix} R_c^o & T_c^o \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & T_X \\ a_{21} & a_{22} & a_{23} & T_Y \\ a_{31} & a_{22} & a_{33} & T_Z \end{pmatrix} \tag{C.4}$$

En combinant les équations (C.3) et (C.4), on obtient pour chaque point m de la mire (voir Figure C.1) deux équations reliant les coordonnées d'un point dans le repère objet et les coordonnées de son image dans l'image numérisée.

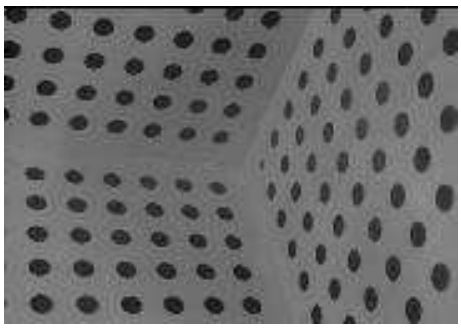


FIG. C.1 – *Mire de calibration à trois faces*

Les paramètres intrinsèques et extrinsèques sont obtenus dans un premier par la méthode de Lagrange [Toscani 87] en résolvant un système linéaire (en supposant donc la distorsion radiale nulle $K_d = 0$). Six points non-coplanaires sont suffisants pour résoudre ce système, mais, pour améliorer la précision des résultats, il est nécessaire d'en utiliser davantage : on a alors un système surdimensionné à (2xNpoints) équations si N > 6. Les paramètres trouvés en utilisant cette méthode servent alors d'initialisation à la résolution du système par une méthode non-linéaire (en supposant donc cette fois la distorsion K_d non nulle). Ajoutons enfin que cette méthode est améliorée en combinant des images de la mire acquises depuis plusieurs points de vue (on alors $(5 + 6 \times m)$ inconnues et $2 \times N \times m$ équations où m est le nombre de points de vue choisi).

C.2 Passage pixel - mètre et mètre - pixel

Une fois que les paramètres intrinsèques de la caméra ont été déterminés, on peut réaliser les transformations de l'espace pixel à l'espace métrique. Dans le cas du point, les mesures de la projection dans l'image du point 3D étant données par (x_p, y_p), exprimées en pixel, le changement de variable suivant est effectué :

$$\left\{ \begin{array}{rcl} x_m & = & (x_p - x_c)/F_x \\ y_m & = & (y_p - y_c)/F_y \end{array} \right. \tag{C.5}$$

où (x_m, y_m) sont les coordonnées du point 2D exprimées en m. Réciproquement on a :

$$\left\{ \begin{array}{rcl} x_p & = & x_c + F_x x_m \\ y_p & = & y_c + F_y y_m \end{array} \right. \tag{C.6}$$

De même, les changements de variables dans le cas d'une droite sont donnés par :

$$\left\{ \begin{array}{ll} \rho_m = \dfrac{\rho_p - x_c \cos\theta_p - y_c \sin\theta_p}{\sqrt{F_x^2 \cos^2\theta_p + F_y^2 \sin^2\theta_p}}, & \theta_m = \arctan(F_y \sin\theta_p, F_x \cos\theta_p) \\ \rho_p = \dfrac{F_x F_y \rho_m + x_c F_y \cos\theta_m + y_c F_x \sin\theta_m}{\sqrt{F_y^2 \cos^2\theta_m + F_x^2 \sin^2\theta_m}}, & \theta_p = \arctan(F_x \sin\theta_m, F_y \cos\theta_m) \end{array} \right. \tag{C.7}$$

où :

- ρ_p (en pixels) et θ_p sont les coordonnées polaires de la projection de la droite dans l'image ;
- ρ_m (en mètre) et θ_m sont les valeurs transformées correspondantes en utilisant les paramètres obtenus par calibrage.

C.3 Transformations géométriques

Nous présentons ici les transformations géométriques permettant d'exprimer dans un repère fixe les paramètres des primitives initialement exprimés dans le repère caméra.

La situation du repère caméra par rapport au repère fixe est représentée par la matrice homogène M :

$$M = \begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix} \tag{C.8}$$

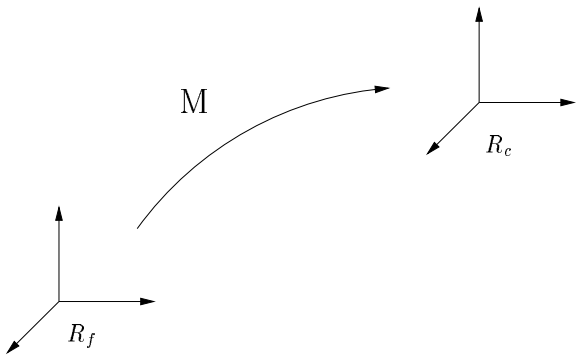


FIG. C.2 – *Changement de repèree*

C.3.1 Le cas du point

Soit $\underline{X}_c$ la position d'un point dans le repère caméra. Les coordonnées $\underline{X}$ de ce point dans le repère fixe sont données par :

$$\underline{X} = R\underline{X}_c + T \tag{C.9}$$

Réciproquement on a :

$$\underline{X}_c = R^T\underline{X} - R^T T \tag{C.10}$$

C.3.2 Le cas de la droite

Soient $\underline{P} = (A_1, B_1, C_1, D_1, A_2, B_2, C_2, D_2)$ les paramètres, exprimés dans le repère caméra, d'une droite définie par :

$$h(\underline{X}_c, \underline{P}) = \begin{cases} A_1X_c + B_1Y_c + C_1Z_c + D_1 = 0 \\ A_2X_c + B_2Y_c + C_2Z_c + D_2 = 0 \end{cases} \tag{C.11}$$

avec les contraintes suivantes :

$$D_1 = 0 \tag{C.12}$$
$$A_1^2 + B_1^2 + C_1^2 = 1 \tag{C.13}$$
$$A_2^2 + B_2^2 + C_2^2 = 1 \tag{C.14}$$
$$A_1A_2 + B_1B_2 + C_1C_2 = 0 \tag{C.15}$$

Posons $\underline{A}_i = (A_i, B_i, C_i)$, le système (C.11) se réécrit alors :

$$\underline{A}_i^T \underline{X}_c + D_i = 0, i = 1, 2 \tag{C.16}$$

On en déduit :

$$\underline{A}_i^T R^T \underline{X} - \underline{A}_i^T R^T T + D_i = 0 \tag{C.17}$$

En posant :

$$\boxed{\begin{array}{l} \underline{A}'_i = R\underline{A}_i \\ D'_i = D_i - \underline{A}_i'^T T \end{array}} \tag{C.18}$$

l'équation de la droite dans le repère fixe qui s'écrit sous la forme :

$$\underline{A}_i'^T \underline{X} + D'_i = 0, i = 1, 2 \tag{C.19}$$

respecte les contraintes (C.13) à (C.15), mais pas (C.12). On pose donc :

$$\boxed{\underline{A}''_1 = \frac{D'_2}{N}\underline{A}'_1 - \frac{D'_1}{N}\underline{A}'_2 \text{ avec } N = \sqrt{D_1'^2 + D_2'^2}} \tag{C.20}$$

On a alors $\|\underline{A}''_1\| = 1$ (soit (C.13)) et $D''_1 = 0$ (C.12), car :

$$\begin{aligned} \underline{A}_1''^T \underline{X} &= \frac{D'_2}{N}\underline{A}_1'^T \underline{X} - \frac{D'_1}{N}\underline{A}_2'^T \underline{X} \\ &= -\frac{D'_1 D'_2}{N} + \frac{D'_1 D'_2}{N} \\ &= 0 \end{aligned}$$

Pour respecter (C.14) et (C.15), on pose finalement :

$$\boxed{\underline{A}''_2 = \frac{D'_2}{N}\underline{A}'_2 + \frac{D'_1}{N}\underline{A}'_1 \text{ et } \underline{D}''_2 = \frac{D_1'^2 + D_2'^2}{N}} \tag{C.21}$$

On a alors d'une part : $\|\underline{A}''_2\| = 1$ et d'autre part $\underline{A}_1''^T \underline{A}''_2 = 0$. En effet :

$$\begin{aligned} \underline{A}_1''^T \underline{A}''_2 &= \left(\frac{D'_2}{N}\underline{A}'_1 - \frac{D'_1}{N}\underline{A}'_2\right)^T \left(\frac{D'_2}{N}\underline{A}'_2 + \frac{D'_1}{N}\underline{A}'_1\right) \\ &= \frac{D_2'^2}{N^2}\underline{A}_1'^T \underline{A}'_2 + \frac{D'_1 D'_2}{N^2}\underline{A}_1'^T \underline{A}'_1 - \frac{D'_1 D'_2}{N^2}\underline{A}_2'^T \underline{A}'_2 - \frac{D_1'^2}{N^2}\underline{A}_2'^T \underline{A}'_1 \\ &= \frac{D'_1 D'_2}{N^2} - \frac{D'_1 D'_2}{N^2} \\ &= 0 \end{aligned} \tag{C.22}$$

Finalement, on a bien :

$$\begin{aligned} \underline{A}''_2 \underline{X} + \underline{D}''_2 &= \frac{D'_2}{N}\left(\underline{A}_2'^T \underline{X} + D'_2\right) + \frac{D'_1}{N}\left(\underline{A}_1'^T \underline{X} + D'_1\right) \\ &= 0 \end{aligned}$$

C.3.3 Le cas du cylindre

Soit l'équation d'un cylindre dans le repère caméra donnée par :

$$\|\underline{X} - \underline{X}_0\|^2 - \left(\underline{U}^T \underline{X}\right)^2 - R_r^2 = 0 \tag{C.23}$$

sous les contraintes

$$\underline{U}^T \underline{X}_0 = 0 \tag{C.24}$$

$$\|\underline{U}\| = 1 \tag{C.25}$$

où $\underline{X}_0$ désigne les coordonnées du point de l'axe du cylindre le plus proche du centre optique, $\underline{U}$ la direction de cet axe et R_r son rayon. En posant (voir Figure C.3) :

$$\boxed{\begin{array}{l} \underline{U}' = R\underline{U} \\ \underline{X}'_0 = R\underline{X}_0 + T - \lambda \underline{U}' \text{ avec } \lambda = \underline{U}'^T T = \underline{U}^T R^T T \end{array}} \tag{C.26}$$

on obtient les paramètres du cylindre (respectant également (C.24) et (C.25)) exprimés dans le repère fixe. En effet, on a alors :

$$\begin{aligned} \underline{U}'^T \underline{X}'_0 &= \underline{U}^T R^T R \underline{X}_0 + \underline{U}^T R^T T - \left(\underline{U}^T R^T T\right) \underline{U}^T R^T R \underline{U} \\ &= \underline{U}^T \underline{X}_0 + \underline{U}^T R^T T - \underline{U}^T R^T T = 0 \end{aligned}$$

et :

$$\begin{aligned} \|\underline{X}' - \underline{X}'_0\|^2 - \left(\underline{U}'^T \underline{X}'\right)^2 &= \|R\underline{X} + T - R\underline{X}_0 - T + \lambda \underline{U}'\|^2 - \left(\underline{U}^T R^T R \underline{X} + U^T R^T T\right)^2 \\ &= \|R\left(\underline{X} - \underline{X}_0\right) + \lambda \underline{U}'\|^2 - \left(\underline{U}^T X - \underline{U}^T R^T T\right)^2 \\ &= \|R\left(\underline{X} - \underline{X}_0\right)\|^2 + \lambda^2 \|\underline{U}'\|^2 + 2\lambda \left(\underline{X} - \underline{X}_0\right)^T R^T R \underline{U} \\ &\quad - \left(\underline{U}^T \underline{X}\right)^2 - \left(\underline{U}^T R^T T\right)^2 - 2\left(\underline{U}^T \underline{X}\right)\left(\underline{U}^T R^T T\right) \\ &= \|\left(\underline{X} - \underline{X}_0\right)\|^2 + \lambda^2 + 2\lambda \left(\underline{X} - \underline{X}_0\right)^T \underline{U} - \\ &\quad - \left(\underline{U}^T \underline{X}\right)^2 - \lambda^2 - 2\lambda \left(\underline{U}^T \underline{X}\right) \\ &= \|\left(\underline{X} - \underline{X}_0\right)\|^2 - \left(\underline{U}^T \underline{X}\right)^2 + 2\lambda \left(\underline{X}^T \underline{U}\right) - 2\lambda \left(\underline{U}^T \underline{X}\right) \\ &= \|\underline{X} - \underline{X}_0\|^2 - \left(\underline{U}^T \underline{X}\right)^2 \end{aligned}$$

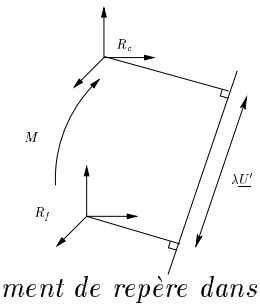


FIG. C.3 – *Changement de repère dans le cas d'un cylindre*

Annexe D

La commande référencée vision

Les différents mouvements satisfaisant les contraintes nécessaires à la reconstruction de la scène peuvent automatiquement être générés par asservissement visuel [Chaumette 90], [Espiau 92], [Hashimoto 93], [Feddema 91]. L'asservissement visuel consiste à introduire directement, et en boucle fermée, les informations extraites de l'image dans une boucle de commande. Par opposition avec l'asservissement en position classique, le problème est ici spécifié en termes de régulation dans l'image et ne nécessite pas la reconstruction tridimensionnelle de la scène à chaque itération de la boucle de commande (voir Figure D.1).

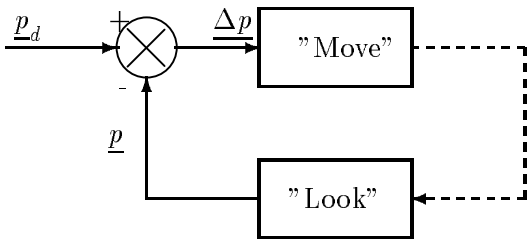


FIG. D.1 – *Asservissement visuel*

Dans cette étude, nous avons utilisé l'approche fonction de tâche, développée par Samson, Le Borgne et Espiau [Samson 91] et appliquée à l'asservissement visuel par Chaumette, Rives et Espiau [Chaumette 90], [Espiau 92].

Ce choix entraîne la recherche de fonctions de tâche appropriées à nos différents objectifs :

- réalisation des trajectoires optimales associées à la reconstruction des droites et des cylindres ;
- réalisation des trajectoires nécessaires au calcul de la longueur de ces primitives ;
- mouvement de rotation autour d'un segment pour la vérification de la présence d'un autre segment (prédite par la génération d'hypothèse de notre réseau Bayesien) ;

Dans le cadre de la reconstruction 3D, nous avons été amenés à constater que les butées articulaires étaient un facteur très handicapant qui limitaient fortement la mobilité du manipulateur et donc la possibilité de reconstruire une primitive. Nous présentons donc également les lois de commande qui tiennent compte de ces limites [Rizzo 94], [4].

D.1 Asservissement visuel

D.1.1 La matrice d'interaction

Ce paragraphe rappelle succinctement quelques notions que nous avons introduites en détail au chapitre 2.

Soit $\mathcal{P}_s$ une primitive géométrique paramétrable décrite par une équation de la forme :

$$h(\underline{X}, \underline{P}) = 0 \ , \ \forall \underline{X} \in \mathcal{P}_S \tag{D.1}$$

où h définit la nature de la primitive et $\underline{P}$ sa configuration. Notons $\mathcal{P}_i$ la projection de la primitive $\mathcal{P}_S$ dans le plan image. Formellement, la configuration de $\mathcal{P}_i$ peut s'écrire par :

$$g(\underline{x}, \underline{p}) = 0 \ , \ \forall \underline{x} \in \mathcal{P}_i \tag{D.2}$$

où g définit la nature de l'image 2D de la primitive et où les valeurs des paramètres $\underline{p}$, fonctions des paramètres $\underline{P}$, définissent sa configuration.

Soit $T = (V(O), \Omega)$ le torseur cinématique de la caméra où $V(O) = (V_x, V_y, V_z)$ représente la vitesse de translation de la caméra et $\Omega = (\Omega_x, \Omega_y, \Omega_z)$ sa vitesse de rotation. La variation de $\underline{p}$ qui relie le mouvement apparent de la primitive dans l'image au mouvement de la caméra T peut être calculée explicitement et s'exprime par :

$$\dot{\underline{p}} = L_{\underline{p}}^T(\underline{p}, \underline{P}_L) \ T \tag{D.3}$$

où $L_{\underline{p}}^T(\underline{p}, \underline{P}_L)$, représente la **matrice d'interaction** qui caractérise les interactions entre le capteur et la primitive considérée. Rappellons que $\underline{P}_L$ est fonction des paramètres $\underline{P}$ (voir Chapitre 2).

D.1.2 Expression de la fonction de tâche et de la commande dans l'espace opérationnel

On peut parfaitement intégrer les techniques d'asservissement visuel dans l'approche fonction de tâche [Samson 91]. En effet, les tâches référencées vision peuvent s'exprimer comme la régulation à zéro de la fonction de tâche suivante :

$$e_1 = C(\underline{p} - \underline{p}_d) \tag{D.4}$$

où $\underline{p}$ est un vecteur représentant l'ensemble des informations visuelles choisies pour réaliser la tâche (*i.e.* la valeur courante des paramètres représentant la primitive considérée,

mesurée dans l'image à chaque itération de la boucle de commande) et $\underline{p}_d$ représente sa valeur désirée. C, appelée matrice de combinaison, peut être définie par :

- $C = W L_{\underline{p}}^{T+}(\underline{p}, \widehat{\underline{P}_L})$ si les paramètres $\underline{P}_L$ utilisés dans la matrice d'interaction peuvent être estimés en ligne (en utilisant par exemple le processus d'estimation 3D présenté dans le Chapitre 2). Dans ce cas, W est définie comme une matrice de rang plein telle que Ker W = Ker $L_{\underline{p}}^{T}(\underline{p}, \widehat{\underline{P}})$.

- $C = W L_{\underline{p}}^{T+}(\underline{p}_d, \underline{P}_{L\,d})$ si la matrice d'interaction ne peut pas être mise à jour à chaque itération de la boucle de commande. Dans ce cas, des hypothèses doivent être faites sur la structure 3D de la primitive considérée afin de calculer les valeurs désirées $\underline{P}_{L\,d}$ et $\underline{p}_d$. Un tel choix permet d'éviter une estimation en ligne des paramètres $\underline{P}$. Dans ce cas, W est définie comme une matrice de rang plein tel que Ker W = Ker $L_{\underline{p}}^{T+}(\underline{p}_d, \underline{P}_{L\,d})$.

Si la tâche de vision spécifiée ne contraint pas les n degrés de liberté du robot, la redondance disponible peut être exploitée pour accomplir une tâche secondaire. Nous obtenons alors la fonction de tâche suivante :

$$\underline{e} = W^{+} C(\underline{p} - \underline{p}_d) + (\mathbb{I}_n - W^{+}W)\underline{g}_s^{T} \tag{D.5}$$

où :

- W^{+} et $\mathbb{I}_n - W^{+}W$ sont deux opérateurs de projection qui garantissent que le mouvement de la caméra dû à la tâche secondaire est compatible avec la régulation de $\underline{p}$ vers $\underline{p}_d$. En effet, en raison du choix de W, $\mathbb{I}_n - W^{+}W$ appartient théoriquement au noyau Ker $L_{\underline{p}}^{T}$, ce qui implique que la réalisation de la tâche secondaire n'aura aucun effet sur la tâche primaire ($L_{\underline{p}}^{T}(\mathbb{I}_n - W^{+}W)\underline{g}_s^{T} = 0$). Cependant, si les paramètres $\widehat{\underline{P}}_L$ sont mal estimés (ou si l'on utilise $\underline{p}_d$ et $\underline{P}_{L\,d}$ alors que la tâche n'est pas parfaitement réalisée), alors l'opérateur de projection $\mathbb{I}_n - W^{+}W$ n'appartiendra pas exactement à Ker $L_{\underline{p}}^{T}$ et la tâche secondaire introduira une perturbation dans la réalisation de la tâche visuelle.

- g_s est une tâche secondaire qui peut s'exprimer comme le gradient d'une fonction h_s à minimiser ($g_s = \dfrac{\partial h_s}{\partial \bar{r}}$). Cette fonction de coût est donc minimisée sous la contrainte que $\underline{e}_1$ soit réalisée.

Le problème de la commande se ramène du point de vue de l'automatique à la régulation de la fonction de tâche $\underline{e}$. Cette dernière est parfaitement réalisée si, à chaque instant t : $\underline{e}(t) = 0$. Une méthode générale permettant de réguler la fonction de tâche $\underline{e}$ est présentée dans [Samson 91]. Une loi de commande simplifiée, qui calcule la vitesse T de la caméra et qui assure une décroissance exponentielle de la fonction de tâche $\underline{e}$ est donnée par [Chaumette 90], [Espiau 92] :

$$T = -\lambda \underline{e} - \frac{W^{+}\widehat{\partial \underline{e}_1}}{\partial t} - (\mathbb{I}_n - W^{+}W)\frac{\partial g_s^{T}}{\partial t} \tag{D.6}$$

où

- λ est un gain réglant la vitesse de convergence exponentielle de $\underline{e}$;
- $\widehat{\frac{\partial \underline{e}_1}{\partial t}}$ représente une estimation d'un éventuel mouvement de la primitive considérée. Si la cible est en mouvement, cette estimation doit être introduite dans la loi de commande afin de supprimer les erreurs de trainage. Elle peut être obtenue en utilisant des techniques classiques de filtrage de Kalman [Chaumette 93], [Bensalah 95]. Dans le cas d'un scène statique, on pourra évidemment supposer que $\frac{\partial \underline{e}_1}{\partial t} = \widehat{\frac{\partial \underline{e}_1}{\partial t}} = 0$, ce que nous ferons par la suite.

D.2 Positionnement et suivi de trajectoires

Nous détaillons dans ce paragraphe certaines lois de commande utilisées dans le processus de reconstruction de scènes.

D.2.1 Loi de commande correspondant à la reconstruction d'un cylindre

Dans le cas du **cylindre**, les effets des erreurs de mesure sont minimisés s'il se projette sous la forme de deux droites symétriques parallèles verticales (ou horizontales) dans l'image avec en outre $V_Y = 0$ (ou $V_X = 0$). La caméra doit alors se déplacer sur un cercle dont le centre est traversé par l'axe du cylindre et perpendiculaire à celui-ci (cf Figure D.2).

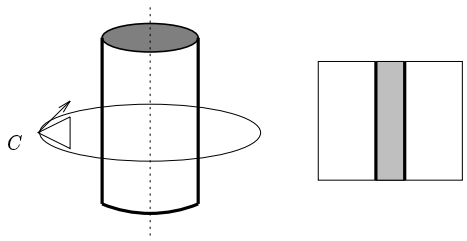


FIG. D.2 – *Mouvements optimaux de la caméra pour la reconstruction d'un cylindre et image associée*

Comme nous l'avons vu au chapitre 2, l'équation du cylindre est donnée par l'équation (2.23) et sa projection dans l'image est caractérisée par deux droites $\mathcal{D}_1(\rho_1, \theta_1)$ et $\mathcal{D}_2(\rho_2, \theta_2)$. Dans le cas vertical, le motif à atteindre dans l'image est donné par le vecteur de paramètres $\underline{p}_d = (\rho_d, 0, \rho_d, \pi)$. Dans cette configuration, l'image du cylindre est donnée par deux droites $\mathcal{D}_1$ et $\mathcal{D}_2$ tels que :

$$\begin{cases} \mathcal{D}_1 : & X - \rho_d = 0 \\ \mathcal{D}_2 : & X + \rho_d = 0 \end{cases} \tag{D.7}$$

Quand la convergence est atteinte (*i.e.*, la tâche primaire $\underline{e}_1$ est réalisée), la matrice d'interaction associée au cylindre s'écrit (voir équation (2.25)) :

$$L^T_{\underline{p}|\underline{p}=\underline{p}_d} = \begin{pmatrix} \lambda_\rho & 0 & -\lambda_\rho\rho_d & 0 & -(1+\rho_d^2) & 0 \\ 0 & 0 & 0 & -\rho_d & 0 & -1 \\ -\lambda_\rho & 0 & -\lambda_\rho\rho_d & 0 & 1+\rho_d^2 & 0 \\ 0 & 0 & 0 & \rho_d & 0 & -1 \end{pmatrix} \tag{D.8}$$

avec $\lambda_\rho = -(\widehat{A}\rho_d + \widehat{C})$ (où $\widehat{A}$ et $\widehat{C}$ sont deux des paramètres estimés du plan des limbes $\widehat{\underline{P}}_L$ du cylindre - voir équation (2.24)). On choisit $W = L^T_{\underline{p}|\underline{p}=\underline{p}_d}$ et on a alors $W L^{T+}_{\underline{p}|\underline{p}=\underline{p}_d} = \mathbb{I}_4$.

La caméra devant effectuer un mouvement de rotation à distance constante de l'axe du cylindre, la tâche secondaire consiste à déplacer la caméra à vitesse constante V_X le long de son axe $\vec{X}$. La fonction de coût secondaire est donc donnée par :

$$h_s = \frac{1}{2}\beta_X (X(t) - X_0 - V_X t)^2 \tag{D.9}$$

où β_X est un scalaire positif. On obtient :

$$\underline{g}_s^T = \begin{pmatrix} \beta_X(X(t) - X_0 - V_X t) \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \text{et} \quad \frac{\partial \underline{g}_s^T}{\partial t} = \begin{pmatrix} -\beta_X V_X \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{D.10}$$

La vitesse de la caméra T est finalement donnée par :

$$T = -\lambda(W^+(\underline{p} - \underline{p}_d) + (\mathbb{I}_n - W^+W)\underline{g}_s^T) - (\mathbb{I}_n - W^+W)\frac{\partial g_s^T}{\partial t}$$

avec

$$W^+ = \begin{pmatrix} \lambda_\rho/2\beta & 0 & -\lambda_\rho/2\beta & 0 \\ 0 & 0 & 0 & 0 \\ -1/2\lambda_\rho\rho_d & 0 & -1/2\lambda_\rho\rho_d & 0 \\ 0 & -1/2\rho_d & 0 & 1/2\rho_d \\ -1/2\beta & 0 & 1/2\beta & 0 \\ 0 & -1/2 & 0 & -1/2 \end{pmatrix}$$

et

$$(\mathbb{I}_6 - W^+W) = \begin{pmatrix} 1/\beta & 0 & 0 & 0 & \lambda_\rho 1/\beta & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \lambda_\rho/\beta & 0 & 0 & 0 & \lambda_\rho^2/\beta & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

où $\beta = 1 + \lambda_\rho^2$.

La consigne T à envoyer à la caméra sera donc donnée par :

$$T = -\lambda \underline{e} + \begin{pmatrix} \beta_X V_X (1+{\rho_d}^2)^2/\beta \\ 0 \\ 0 \\ 0 \\ \beta_X V_X (1+{\rho_d}^2)\lambda_\rho/\beta \\ 0 \end{pmatrix} \tag{D.11}$$

En choisissant $\beta_X = \beta/(1+{\rho_d}^2)^2$, la vitesse translationnelle de la caméra le long de l'axe $\vec{X}$ aura la valeur désirée V_X quand $\underline{e} = 0$.

On peut noter que l'opérateur de projection $\mathbb{I}_6 - W^+W$, qui rend compatibles les tâches primaire et secondaire, permet de définir automatiquement un mouvement de rotation Ω_Y qui compense la translation en V_X.

Loi de commande associée à la droite Dans le cas de la droite, le problème est similaire. Dans ce cas, il convient cependant de noter que la condition $\underline{\dot{p}} = 0$ n'entraine pas nécessairement la condition $\underline{\dot{P}}_L = 0$ (voir équation (2.30)). Il faut donc déterminer la consigne T de la caméra qui assure ces deux contraintes. Ceci passe par le calcul de la matrice d'interaction $L^T_{\underline{P}_L}$ associé aux paramètres $\underline{P}_L$ de la droite (voir équation (2.19)). Le détail du calcul de cette matrice d'interaction, ainsi que de la loi de commande associée est donnée dans [Boukir 93a].

D.2.2 Fonction de tâche correspondant au calcul de la longueur d'une primitive

Pour calculer la longueur d'une primitive, il convient de minimiser la distance entre l'extrémité du segment et le centre de l'image et ce sous la contrainte que la primitive apparaisse toujours verticale et centrée. Quand la tâche primaire est réalisée, les coordonnées de ce point sont données par $\underline{p_{ext}} = (y, 0)$. Quand l'extrémité du segment n'est pas visible on choisit $y = y_{max}$ (où y_{max} correspond à la première ou la dernière ligne de l'image).

Une solution aurait été de rajouter y dans les paramètres $\underline{p}$ à réguler et de modifier en conséquence la matrice d'interaction. Nous avons choisi de définir une tâche secondaire. La fonction de coût associé à cette tâche doit être minimale quand l'extrémité du segment est au centre de l'image ($y = 0$). On choisit donc :

$$h_s = \frac{1}{2}\alpha y^2 \tag{D.12}$$

Dans ce cas, g_s et $\frac{\partial g_s}{\partial t}$ sont données par :

$$g_s^T = \alpha y \begin{pmatrix} 0 \\ -\frac{1}{Z} \\ \frac{y}{Z} \\ 1+y^2 \\ -xy \\ -x \end{pmatrix}, \frac{\partial g_s^T}{\partial t} = 0 \tag{D.13}$$

On ne veut exécuter qu'un mouvement autour de l'axe Ω_x. On pose donc :

$$g_s^T = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \alpha y(1+y^2) \\ 0 \\ 0 \end{pmatrix} \tag{D.14}$$

Quand l'erreur sur y est nulle (c'est-à-dire que l'extrémité du segment est au centre de l'image), on a bien $g_s = 0$. Si on désire une décroissance exponentielle de l'erreur, il suffit de fixer :

$$\alpha = -\frac{\lambda}{(1+y^2)^2}$$

En effet, on a alors :

$$\begin{aligned} \dot{y} &= -(1+y^2)\Omega_x \\ &= -(1+y^2)\frac{\lambda}{(1+y^2)^2} y(1+y^2) \\ &= -\lambda y \end{aligned}$$

D.2.3 Résultats

Mouvements de la caméra nécessaires à l'estimation des paramètres d'un cylindre : Nous présentons dans un premier temps une tâche de positionnement par rapport à un cylindre, celui-ci devant apparaître vertical centré dans l'image. Une tâche secondaire correspondant à un mouvement autour du cylindre et le long de son axe est activée quand la tâche primaire est réalisée. Plus précisément, la tâche secondaire consiste à déplacer la caméra à vitesse constante (soit successivement $V_x = 5$ cm/s (mouvement nécessaire à l'estimation), $V_y = 10$ cm/s (mouvement nécessaire au calcul de la longueur), $-V_x$ et $-V_y$). Les résultats sont reportés sur les Figures D.3 et D.4. La Figure D.3 montre l'erreur entre la position courante du cylindre dans l'image et la position désirée, les Figures D.4.a et D.4.b montrent les composantes de la commande en translation (resp. en rotation) transmises au robot. Comme indiqué, une fois la convergence atteinte (itération 170), la tâche secondaire est déclenchée.

Mouvements de la caméra nécessaires à l'estimation des paramètres d'une droite Les résultats présentés ici décrivent les différentes phases d'asservissement visuel nécessaires à la reconstruction d'une droite : tâche de fixation et de focalisation, puis calcul de sa longueur. La Figure D.5 représente les commandes envoyées à la caméra pendant ces différentes phases. La Figure D.6.a montre l'erreur sur la position et l'orientation de la droite dans l'image.

La phase de focalisation apparait sur les Figures D.5 et D.6.a entre les itérations 0 et 60. Comme on peut le constater sur la Figure D.5, deux degrés de liberté en rotation seulement sont utilisés. Quand la convergence est atteinte, la tâche secondaire correspondant à un

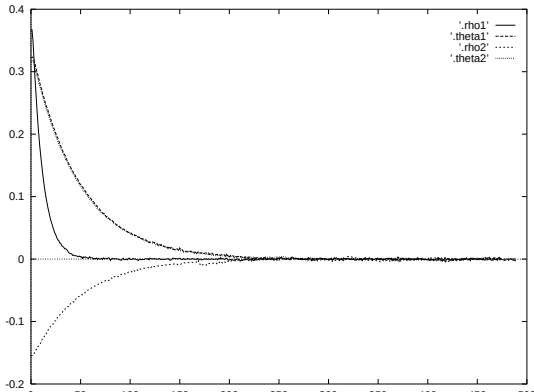


FIG. D.3 – *Positionnement par rapport à un cylindre : Erreur* $\underline{p} - \underline{p}_d$.

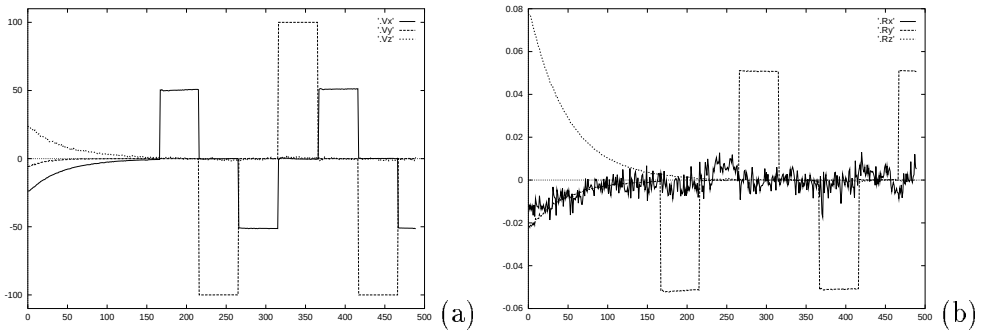


FIG. D.4 – *Positionnement par rapport à un cylindre : vitesse de la caméra.*

mouvement autour de la droite est déclenchée. Plus précisément, la caméra effectue un mouvement de rotation autour de l'axe de la primitive (itérations 60 à 170 pour l'aller et 160 à 260 pour le retour). On observe sur la Figure D.6.a que la primitive s'écarte de la position d'équilibre pendant quelques itérations au début du mouvement de rotation (itération 60 à 110). Cette pertubation est dûe au fait que les paramètres $\underline{P}_L$ de la primitive 3D introduits dans la matrice d'interaction $L^T_{\underline{P}}(\underline{p}, \widehat{\underline{P}}_L)$ sont constants et erronés pour ces itérations de la boucle de commande, car aucune estimation des paramètres $\underline{P}_L$ n'a encore été réalisée. Dans ce cas, $\mathbb{I}_n - W^+W$ n'appartient pas exactement à Ker $L^T_{\underline{P}}(\underline{p}, \underline{P}_L)$, ce qui introduit une pertubation dans la réalisation de la tâche visuelle dûe à la tâche secondaire. Dès que les premières estimations de $\underline{P}_L$ sont réalisées, $\mathbb{I}_n - W^+W$ appartient à Ker $L^T_{\underline{P}}(\underline{p}, \underline{P}_L)$ et les pertubations disparaissent.

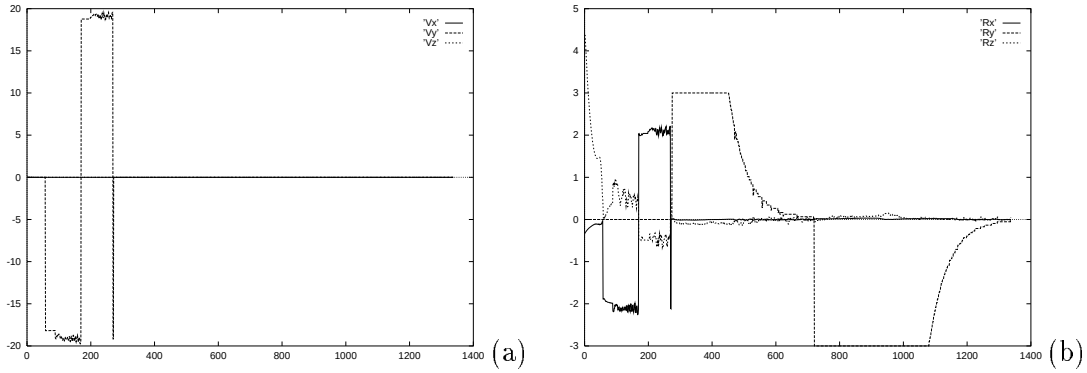


FIG. D.5 – *Vitesse de la caméra pendant la reconstruction (a) V_x, V_y et V_z en m/s et (b) Ω_x, Ω_y et Ω_z en rad/s*

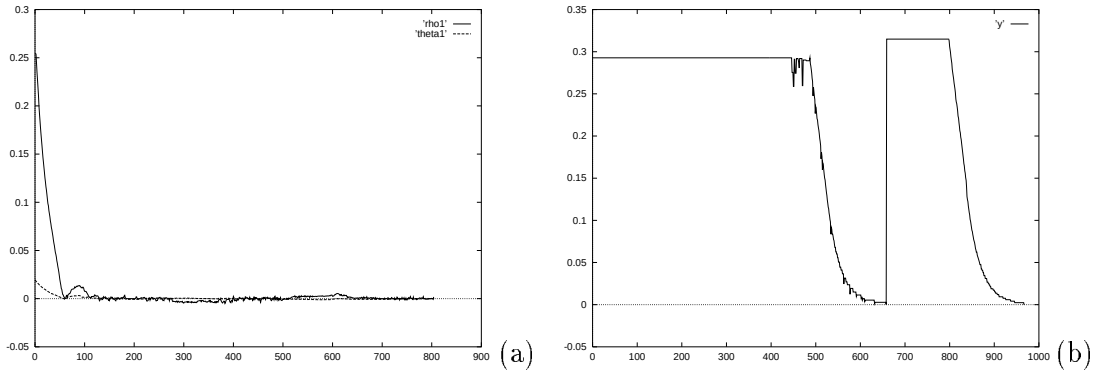


FIG. D.6 – *Erreur sur (a) les paramètres de la droite et (b) sur la distance aux extrémités pendant le calcul de la longueur (en mètre)*

La Figure D.6.b décrit l'erreur y entre l'extrémité du segment et le centre de l'image pendant le calcul de la longueur. Après une phase constante dûe à l'utilisation de y_{max} pendant 400 itérations (l'extrémité du segment n'étant pas encore visible), on observe une décroissance exponentielle de l'erreur. La vitesse de la caméra pendant cette phase est

décrite sur la Figure D.5 entre les itérations 260 à 720 et les itérations 720 à 1350. Nous avons utilisé une commande saturée afin de ne pas avoir de vitesse de la caméra trop importante en rotation (3 deg/s).

D.3 Évitement des butées articulaires

Dans toutes les tâches robotiques, et notamment dans le cas de l'asservissement visuel, un problème très important est d'essayer d'éviter les butées articulaires qui sont des limites physiques à l'extension de l'espace opérationnel du robot. Dans notre cas d'asservissement visuel, aucune planification des trajectoires ne peut être réalisée car la commande du robot est calculée en ligne. Si l'on ne prend pas en compte ces limites, les tâches d'asservissement visuel ne peuvent pas être réalisées lorsque la loi de commande produite amène le robot sur ses butées articulaires. Dans le cadre de la reconstruction 3D, nous avons été amenés à constater que ces butées articulaires étaient un facteur très handicapant qui limitait fortement la mobilité du manipulateur et donc la possibilité de reconstruire une primitive. Nous avons donc élaboré une loi de commande qui tienne compte de ces limites.

Des travaux ont déjà été réalisés dans ce domaine. Chang et Dubey [Chang 95] décrivent une méthode fondée sur une solution à "norme minimale pondérée" pour un robot redondant. Ils ne considèrent que la tâche visuelle et calculent une solution à norme minimale (c'est-à-dire aux moindres carrés) en pénalisant les mouvements articulaires qui amènent le robot près des butées. La pénalisation de ces mouvements est réalisée en multipliant les composantes de la norme de la solution par une matrice de poids. Des résultats satisfaisants ont été obtenus. Cependant, avec cette méthode, la réalisation de la tâche visuelle est perturbée par l'introduction des pénalisations, et des déviations par rapport à une convergence idéale peuvent apparaître. Elle n'est donc plus réalisée de façon indépendante de la tâche d'évitement des butées.

Une autre approche a été suivie par Nelson et Khosla dans [Nelson 93][Nelson 95]. Ils proposent une loi de commande classique pour réaliser la tâche visuelle et/ou le suivi d'une cible (minimisation d'une fonction d'objectif dans l'espace opérationnel du manipulateur). Pour résoudre le problème des butées (et également des singularités du robot), ils définissent une fonction qui pénalise les mouvements amenant le robot près de ces butées. Ils utilisent cette fonction en exploitant la redondance du système, c'est-à-dire les degrés de liberté non utilisés pour réaliser la tâche visuelle. Les mouvements générés peuvent cependant produire d'importantes perturbations dans le processus d'asservissement visuel car ils ne sont généralement pas compatibles avec la régulation à zéro des primitives visuelles sélectionnées.

Dans notre cas, nous devons rechercher une fonction de tâche secondaire appropriée à notre objectif. Afin d'appliquer l'approche fonction de tâche, la fonction de coût à minimiser doit être telle qu'elle atteigne sa valeur maximale quand le manipulateur arrive à une butée articulaire. De plus, il ne faut pas que le gradient de la fonction atteigne zéro quand la fonction atteint sa valeur maximale. En effet, la méthode de minimisation utilisée dans [Samson 91] est une méthode de minimisation du gradient de la fonction de coût. De telles fonctions de coût on déjà été proposées pour assurer une vergence symétrique

dans le cas de l'asservissement visuel avec une tête de stéréovision active [Crowley 95], [Mesrabi 94].

D.3.1 Expression de la fonction de tâche et de la commande dans l'espace articulaire

Les lois de commande en asservissement visuel sont généralement exprimées, comme dans les paragraphes précédents, dans l'espace opérationnel du robot (c'est-à-dire dans le repère caméra). Les commandes sont ensuite calculées dans l'espace articulaire en utilisant le Jacobien inverse du robot [Espiau 92], [Feddema 91], [Nelson 95], [Papanikolopoulos 91]. Cependant, dans certains cas, il paraît plus intéressant d'exprimer les lois de commande directement dans l'espace articulaire. Les butées articulaires du manipulateur sont en effet directement exprimées dans cet espace.

Nous proposons ici, une version modifiée de la loi de commande (D.6), permettant de commander directement le manipulateur dans l'espace articulaire.

Pour cela, il faut définir une liaison entre les variations des informations visuelles et les vitesses articulaires du robot, ce qui entraîne la recherche d'une matrice d'interaction adéquate, telle que :

$$\dot{\underline{p}} = H^T \, \dot{\underline{q}} \tag{D.15}$$

où $\underline{q} = (q_1 \cdots q_n)$ représente la position du robot dans l'espace articulaire. Comme $T = J(\underline{q}) \, \dot{\underline{q}}$, où $J(\underline{q})$ est le jacobien du robot exprimé dans le repère de la caméra, on en déduit :

$$H_{\underline{p}}^T(\underline{p}, \underline{P}_L, \underline{q}) = L_{\underline{p}}^T(\underline{p}, \underline{P}_L) \; J(\underline{q}) \tag{D.16}$$

La fonction de tâche primaire $\underline{e}_1$ associée à la tâche de vision sera alors donnée par :

$$\underline{e}_1 = C(\underline{p} - \underline{p}_d) \tag{D.17}$$

où C, comme précédemment, peut être définie par :

- $C = W H_{\underline{p}}^{T+}(\underline{q}, \underline{p}, \widehat{\underline{P}_L})$ si les paramètres $\underline{P}_L$ utilisés dans la matrice d'interaction sont estimés en ligne.
- $C = W H_{\underline{p}}^{T+}(\underline{P}, \underline{p}_d, \underline{P}_{Ld})$ sinon.

La stratégie de contrôle adoptée qui assure une décroissance exponentielle de la fonction de tâche $\underline{e}$ (équation (D.5)) est alors donnée par :

$$\dot{\underline{q}}_d = -\lambda \underline{e} - W^+ \widehat{\frac{\partial \underline{e}_1}{\partial t}} - (\mathbb{I}_n - W^+ W) \frac{\partial \underline{g}_s^T}{\partial t} \tag{D.18}$$

où $\dot{\underline{q}}_d$ est la vitesse articulaire envoyée au contrôleur du robot.

D.3.2 Tâches secondaires

Comme nous l'avons déjà indiqué, quand la tâche visuelle ne contraint pas tous les degrés de liberté du robot, une tâche secondaire peut être combinée avec $\underline{e}_1$. Nous pouvons ainsi utiliser la redondance pour réaliser l'évitement des butées articulaires. Plusieurs fonctions de coût reflétant le comportement désiré sont maintenant présentées. On peut également signaler que cette approche a également été utilisée pour éviter les singularités du robot [Rizzo 94], [4][5].

D.3.2.1 Spécification en position

Pendant la réalisation des tâches d'asservissement visuel, il est souhaitable de voir le manipulateur aussi éloigné que possible des butées articulaires. Dans un premier temps, nous présentons une fonction de coût secondaire dont l'objectif est d'amener la caméra au milieu de l'extension de chacun des axes du robot tout en réalisant la tâche primaire. Quand la fonction de coût secondaire est parfaitement minimisée, la position finale souhaitée pour chaque axe est :

$$q_{i_{fin}} = \frac{q_{i_{max}} + q_{i_{min}}}{2} \tag{D.19}$$

où $q_{i_{min}}$ et $q_{i_{max}}$ sont les valeurs des butées articulaires de l'axe i. La position finale désirée est donc située au centre de l'extension de chacun des axes. De manière évidente, cette position ne correspond généralement pas à une position compatible avec la tâche de vision. Cependant, il est tout de même possible de définir une fonction de coût qui est minimale quand le manipulateur atteint cette position et maximale dans le voisinage d'une butée articulaire. L'opérateur de projection $\mathbb{I}_n - W^+W$ mènera le robot le plus près possible de cette position à la condition que la tâche visuelle soit réalisée.

Nous pouvons donc choisir comme fonction de coût secondaire la fonction suivante :

$$h_s = \sum_{i=1}^{n} \frac{\beta}{2} \frac{\left(q_i - q_{i_{fin}}\right)^2}{q_{i_{max}} - q_{i_{min}}} \tag{D.20}$$

où β est une constante qui spécifie l'importance de la tâche secondaire dans la loi de commande. Dans ce cas, les composantes de g_s et de $\frac{\partial g_s}{\partial t}$ utilisées dans les équation (D.5) et (D.18) sont données par :

$$g_{s_i} = \frac{\partial h_s}{\partial q_i} = \beta_i \frac{\left(q_i - q_{i_{fin}}\right)}{q_{i_{max}} - q_{i_{min}}} \tag{D.21}$$

et :

$$\frac{\partial g_{s_i}}{\partial t} = 0 \tag{D.22}$$

On peut noter que les valeurs de g_{s_i} appartiennent à l'intervalle $[-\frac{\beta}{2}, \frac{\beta}{2}]$ ce qui autorise un réglage aisé du paramètre β.

D.3.2.2 Spécification en vitesse

Une autre approche pour l'éloignement des butées consiste à spécifier un comportement en termes de vitesse. Ainsi, on pose des contraintes plus fortes sur la performance du système. L'objectif final souhaité sera le même, c'est-à-dire que la position finale est toujours le milieu de l'extension de chacun des axes (voir équation (D.19)). Mais, nous spécifions aussi une décroissance exponentielle de l'erreur entre la position courante et cette position finale désirée :

$$\dot{q}_i^*(t) = -\alpha \frac{q_i^*(t) - q_{i_{fin}}}{q_{i_{max}} - q_{i_{min}}} \tag{D.23}$$

où α est une constante qui permet de régler la rapidité avec laquelle le manipulateur doit s'éloigner des butées et donc se rapprocher de $\underline{q}_{fin}$. La solution $q_i^*(t)$ de cette équation différentielle est :

$$q_i^*(t) = (q_{i_0} - q_{i_{fin}}) e^{-\frac{\alpha t}{q_{i_{max}} - q_{i_{min}}}} + q_{i_{fin}} \tag{D.24}$$

où $q_{i_0} = q_i(t=0)$. Une nouvelle fonction de coût secondaire peut alors être écrite :

$$h_s = \frac{\beta}{2} \sum_{i=1}^{n} [q_i(t) - q_i^*(t)]^2 \tag{D.25}$$

g_{s_i} et $\frac{\partial g_{s_i}}{\partial t}$ sont alors donnés par :

$$g_{s_i} = \frac{\partial h_s}{\partial q_i} = \beta \left(q_i(t) - q_i^*(t) \right) \tag{D.26}$$

et :

$$\frac{\partial g_{s_i}}{\partial t} = -\alpha\beta \, \frac{q_i^*(t) - q_{i_{end}}}{q_{i_{max}} - q_{i_{min}}} \tag{D.27}$$

D.3.2.3 Seuils d'activation

Les deux méthodes précédentes sont basées sur l'obtention de positions idéales. Comme on pourra le voir dans les résultats expérimentaux, la tâche secondaire de convergence vers ces positions idéales ralentit beaucoup la convergence de la tâche primaire en raison des inévitables erreurs dans l'opérateur de projection $\mathbb{I}_n - W^+W$. De plus, dans beaucoup d'applications pratiques, il est souvent inutile d'atteindre cette position idéale dans l'espace articulaire. On souhaite donc plutôt obtenir des convergences rapides en s'éloignant suffisamment des butées pour pouvoir réaliser la tâche visuelle. Cette nouvelle fonction de tâche secondaire ne doit être activée que pour les axes qui se trouvent dans le voisinage des butées articulaires.

On définit donc des seuils d'activation de la fonction tâche secondaire par $\tilde{q}_{i_{min}}$ et $\tilde{q}_{i_{max}}$ tels que :

$$\begin{aligned} \tilde{q}_{i_{min}} &= q_{i_{min}} + \rho(q_{i_{max}} - q_{i_{min}}) \\ \tilde{q}_{i_{max}} &= q_{i_{max}} - \rho(q_{i_{max}} - q_{i_{min}}) \end{aligned} \tag{D.28}$$

où $0 < \rho < 1/2$. La fonction de tâche secondaire sera donnée par :

$$h_s = \frac{\beta}{2} \sum_{i=1}^{n} \frac{s_i^2}{q_{i_{max}} - q_{i_{min}}} \tag{D.29}$$

où :

$$s_i = \begin{cases} q_i - \tilde{q}_{i_{max}} & \text{si } q_i > \tilde{q}_{i_{max}} \\ q_i - \tilde{q}_{i_{min}} & \text{si } q_i < \tilde{q}_{i_{min}} \\ 0 & \text{sinon} \end{cases} \tag{D.30}$$

Le gradient de la fonction de tâche secondaire est alors donné par :

$$g_{s_i} = \begin{cases} \frac{(q_i - \tilde{q}_{i_{max}})}{(q_{i_{max}} - q_{i_{min}})} & \text{si } q_i > \tilde{q}_{i_{max}} \\ \frac{(q_i - \tilde{q}_{i_{min}})}{(q_{i_{max}} - q_{i_{min}})} & \text{si } q_i < \tilde{q}_{i_{min}} \\ 0 & \text{sinon} \end{cases} \tag{D.31}$$

et :

$$\frac{\partial g_{s_i}}{\partial t} = 0 \tag{D.32}$$

Cette dernière fonction de coût visant à éloigner le robot de ses butées articulaires est similaire à la mesure de manipulabilité de Tsai utilisée dans [Nelson 95]. Elle est cependant plus simple puisqu'elle permet de définir les seuils d'activation avec le seul paramètre ρ. Il faut noter finalement que toutes les fonctions $\underline{g}_s$ et $\frac{\partial g_s}{\partial t}$ sont continues, ce qui permet de ne pas avoir de discontinuités dans la commande.

D.3.3 Résultats expérimentaux

D.3.3.1 Positionnement par rapport à un point

Afin de valider et de comparer les différentes fonctions de tâche secondaire, nous présentons dans un premier temps les résultats de positionnement par rapport à un point. L'objectif est de positionner ce point au centre de l'image. Soit m un point 3D décrit par le vecteur de paramètres $\underline{P} = (X, Y, Z)$ et M sa projection dans l'image donnée par le vecteur de paramètre $\underline{p} = (x, y)$. La matrice d'interaction qui relie le mouvement projeté du point dans l'image au mouvement de la caméra est donnée par :

$$L_{\underline{p}}^T = \begin{pmatrix} -1/Z & 0 & x/Z & xy & -(1+x^2) & y \\ 0 & -1/Z & y/Z & 1+y^2 & -xy & -x \end{pmatrix} \tag{D.33}$$

La position désirée du point dans l'image est fixée à $\underline{p} = (0, 0)$ (voir Figure D.7.b).

Comme le montre le tableau D.1 où sont présentés les résultats obtenus pour les trois stratégies proposées dans la section précédente, la position initiale du manipulateur est proche de trois butées articulaires différentes, à savoir q_1, q_3 et q_5. En utilisant une approche d'asservissement visuel classique, le manipulateur rencontre ses butées très rapidement.

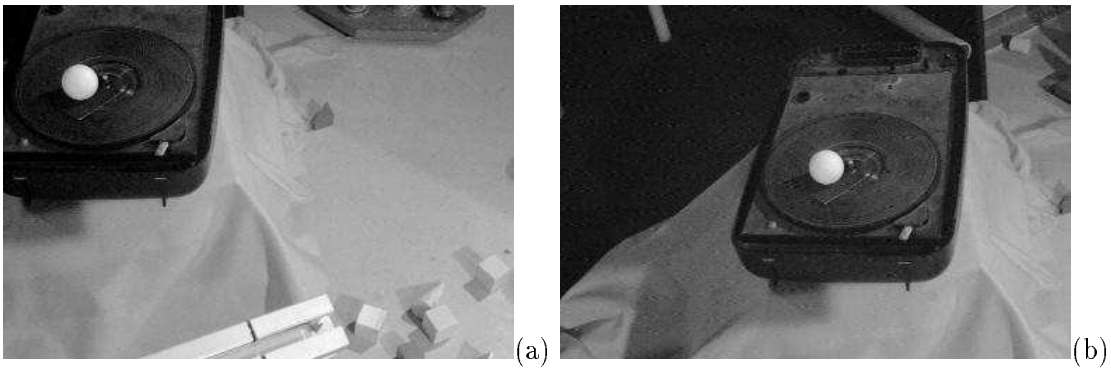


FIG. D.7 – *Positionnement par rapport à un point : (a) image initiale (b) image finale*

	$q_1(mm)$	$q_2(mm)$	$q_3(mm)$	$q_4(dg)$	$q_5(dg)$	$q_6(dg)$
Butée max $q_{i_{max}}$	750	640	500	162	141	90
Butée min $q_{i_{min}}$	-740	-750	-496	-171	-5	-90
Position initiale $q_{i_{init}}$	740	-241	-495	22	3	-45
Position finale (spéc. position)	244	-110	-90	0	19	-41
Position finale (spéc. vitesse)	135	-84	-35	0	22	-38
Seuils activation Max	601	501	400	129	126	72
Seuils activation Min.	-591	-611	-396	-138	9	-72
Position finale	601	-241	-396	127	8	-42

TAB. D.1 – *Éloignement des butées : données significatives*

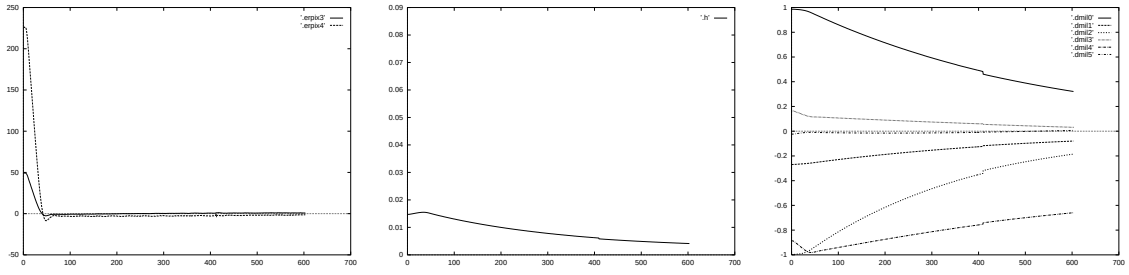
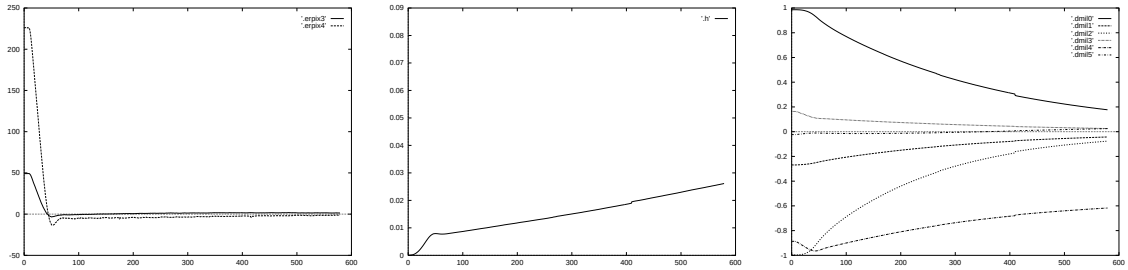
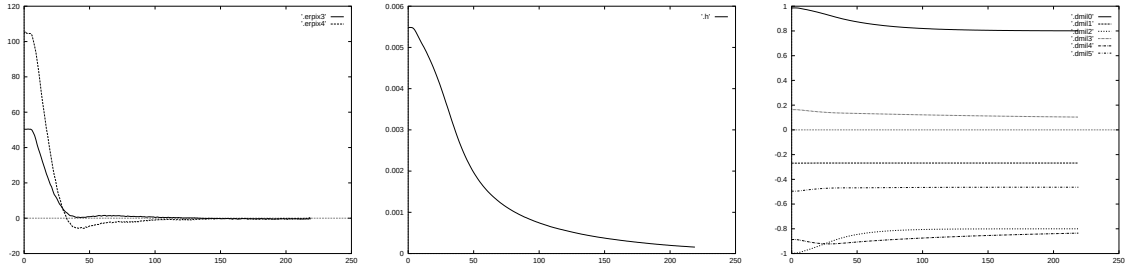


FIG. D.8 – *Éloignement des butées - spécification du comportement en position : (a) erreur* $(\underline{p} - \underline{p}_d)$ *(en pixels) (b) fonction de coût secondaire (c) distances aux butées articulaires* $\frac{\underline{q} - \underline{q}_{end}}{\underline{q}_{max} - \underline{q}_{end}}$*, toutes ces courbes sont exprimées en fonction du temps*



(a) (b) (c)

FIG. D.9 – *Même expérience en utilisant un comportement en vitesse*



(a) (b) (c)

FIG. D.10 – *Même expérience en utilisant les seuils d'activation*

Par contre, comme on peut le voir sur les Figures D.8, D.9 et D.10, la tâche primaire est correctement réalisée tout en évitant les butées articulaires si l'on utilise les tâches secondaires que nous avons proposées.

En comparant les Figures D.8(c) et D.9(c), on peut voir que la méthode relative à la spécification du comportement en vitesse est similaire mais légèrement plus performante que celle en position, du point de vue de l'éloignement des butées. En fait, pour la même expérience de convergence, elle permet de mieux s'approcher des positions idéales, et en outre, plus rapidement. Comme on le voit sur les Figures D.8(a) et D.9(a), la convergence de la tâche visuelle est lente quand celle-ci est presque achevée (l'expérience s'arrête quand l'erreur $|\ \underline{P} - \underline{P}_d\ |$ est inférieure à 1 pixel). Ceci est dû au fait que la profondeur Z, utilisée dans la matrice d'interaction a été fixée de manière arbitraire à 2 m alors que sa valeur réelle est d'approximativement 1 m. Cette mauvaise approximation a pour conséquence que la tâche secondaire n'est pas parfaitement compatible avec la tâche visuelle (en raison des erreurs dans $\mathbb{I}_n - W^+W$). Ceci introduit les petites perturbations observées sur ces deux courbes, perturbations qui prennent fin dès que le robot s'éloigne suffisamment des butées articulaires, ce qui a pour effet de diminuer l'influence de la fonction de tâche secondaire $\underline{g}_s$. On peut aussi noter que, contrairement à ce que l'on pourrait logiquement attendre, le coût secondaire h_s augmente quand on utilise une spécification en vitesse (voir Figure D.9.b). Ceci est dû au fait que le robot s'éloigne de la trajectoire spécifiée $q_i^*(t)$ (voir (D.25)) afin de réaliser les mouvements nécessaires à l'exécution de la tâche primaire.

Par ailleurs, la méthode avec seuils d'activation est la plus performante du point de

vue de la vitesse de convergence car seulement trois axes sont initialement utilisés pour l'évitement des butées (et aucun dès que les seuils d'activation sont atteints). Nous avons choisi comme seuil d'activation 10% du débattement de chaque axe ($\rho = 0.1$). On peut voir sur la Figure D.10.c que le robot s'éloigne du voisinage des butées articulaires pour atteindre les régions situées à l'intérieur des seuils d'activation (qui correspondent sur cette figure à la valeur ± 0.8)

Par la suite, les expériences réalisées ne considèreront que cette dernière méthode.

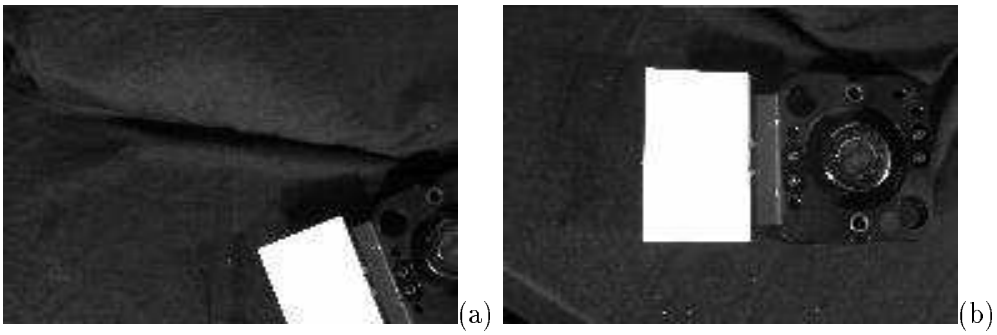
D.3.3.2 Positionnement par rapport à un segment

Nous présentons ici les résultats obtenus pour le positionnement par rapport à un segment. Cette tâche est utilisée lors des phases de focalisation sur la primitive à reconstruire. On peut considérer un segment de deux façons différentes : soit en utilisant les coordonnées de ses extrémités (ce qui revient à un positionnement par rapport à deux points), soit en utilisant seulement les paramètres de la droite porteuse.

Positionnement par rapport à un segment : Considérons un segment 3D ayant pour extrémités les points M_1 et M_2. Ce segment peut être simplement représenté dans l'image par les coordonnées x_1, y_1, x_2 et y_2 de ses extrémités m_1 et m_2. La matrice d'interaction est alors simplement obtenue, comme pour le point, à partir des équations du mouvement projeté et on a :

$$\begin{pmatrix} \dot{x}_1 \\ \dot{y}_1 \\ \dot{x}_2 \\ \dot{y}_2 \end{pmatrix} = \begin{pmatrix} -1/Z_1 & 0 & x/Z_1 & x_1 y_1 & -(1+x_1^2) & y_1 \\ 0 & -1/Z_1 & y/Z_1 & 1+y_1^2 & -x_1 y_1 & -x_1 \\ -1/Z_2 & 0 & x/Z_2 & x_2 y_2 & -(1+x_2^2) & y_2 \\ 0 & -1/Z_2 & y/Z_2 & 1+y_2^2 & -x_2 y_2 & -x_2 \end{pmatrix} T \quad \text{(D.34)}$$

La position désirée du segment dans l'image est $\underline{P} = (0, -Y_d, 0, Y_d)$. Sur la figure D.11, on peut voir un exemple des motifs de départ et d'arrivée relatifs à cette tâche. L'expérience

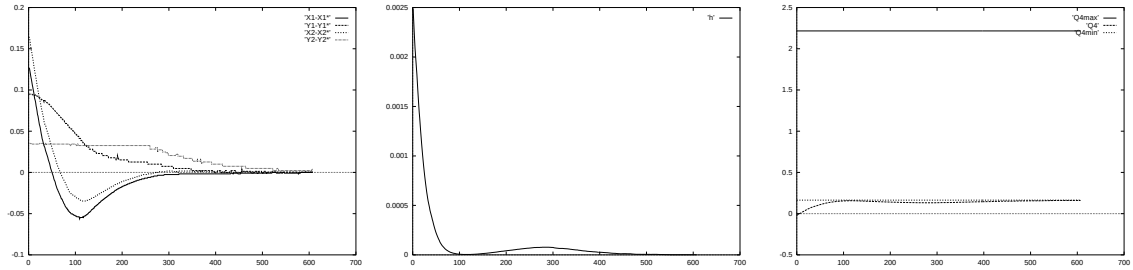


(a) (b)

FIG. D.11 – *Positionnement par rapport à un segment (a) image initiale (b) image finale*

réalisée est similaire aux précédentes, la position initiale est proche d'une butée articulaire (en l'occurrence le deuxième axe de rotation q_4). Là encore, si l'on ne cherche pas à éviter cette butée, la réalisation de la tâche d'asservissement visuel est impossible. On

constate, sur les courbes des Figures D.12.b et D.12.c, que les seuils d'activation sont atteints dès l'itération 100. À ce stade, tous les axes étant suffisamment éloignés des butées, et conformément aux spécifications, la tâche secondaire est désactivée. La tâche visuelle ramenant alors le manipulateur vers la zone critique (itérations 150 à 500), la tâche secondaire est alors réactivée permettant la convergence de la tâche principale. On peut remarquer que les lignes $q_{4_{min}}$ et $q_{4_{max}}$ de la Fig.D.11.c correspondent à la valeur de ces seuils (si $q_4 \geq q_{4_{max}}$ ou $q_4 \leq q_{4_{min}}$, la tâche secondaire est activée).

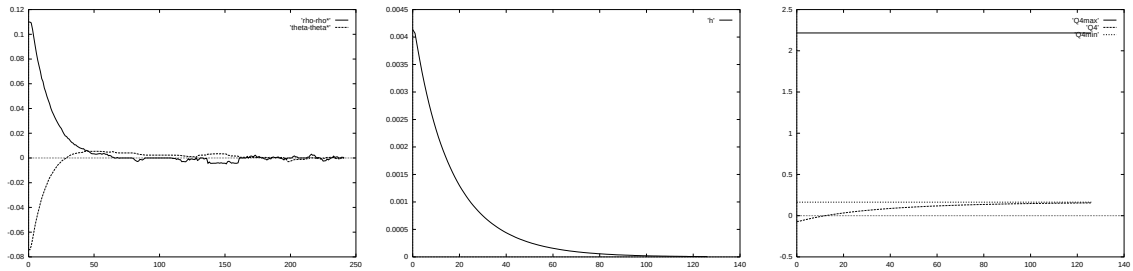


(a) (b) (c)

FIG. D.12 – *Évitement des butées dans le cas d'un segment : (a) erreur $\underline{p} - \underline{p}_d$ (b) fonction de coût secondaire (c) distance au seuil d'activation sur l'axe 4*

On peut finalement noter sur la Figure D.12.a que l'erreur $Y_2 - Y_{d2}$ reste constante pendant 260 itérations, ce qui correspond au nombre d'itérations nécessaires pour que la deuxième extrémité du segment apparaîsse dans l'image (voir Figure D.11).

Positionnement par rapport à une droite : La dernière expérience décrit le positionnement par rapport à une droite. La position finale désirée de la droite dans l'image est $\underline{p} = (\rho_d, \theta_d) = (0, 0)$. En d'autres termes, on souhaite voir la droite verticale centrée dans l'image. Initialement le manipulateur est proche d'une butée articulaire (q_4 ; en fait, la même position initiale et la même scène que dans l'expérience précédente ont été utilisées). Sans tâche secondaire, le robot se déplace vers cette butée et l'asservissement visuel échoue. Les résultats qui sont présentés sur la Figure D.13 montrent le positionnement correct par rapport à la droite. La tâche visuelle est ainsi rapidement réalisée et la tâche secondaire éloigne le robot du voisinage de la butée sur q_4.



(a) (b) (c)

FIG. D.13 – *Évitement des butées dans le cas d'une droite : (a) erreur $\underline{p} - \underline{p}_d$ (b) fonction de coût secondaire (c) distance au seuil d'activation sur l'axe 4*

Bibliographie

[Aarts 87] E.H.L. Aarts, P.J.M. Van Laarhoven. – *Simulated annealing: theory and applications.* – D. Reidel publishing, Dordrecht, 1987.

[Abrams 93] S. Abrams, P.K. Allen, K. Tarabanis. – Dynamic sensor planning. – *IEEE Int. Conf. on Robotics and Automation*, vol. 2, pp. 605–610, Atlanta, Géorgie, Mai 1993.

[Adiv 85] G. Adiv. – Determining three-dimensional motion and structure from optical flow generated by several moving objects. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 7(4):384–401, Juillet 1985.

[Adiv 89] G. Adiv. – Inherent ambiguities in recovering 3D motion and structure from a noisy flow field. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11(5):477–489, Mai 1989.

[Aggarwal 87] J.K. Aggarwal, Y.F. Wang. – Analysis of a sequence of images using point and line correspondences. – *IEEE Int. Conf. on Robotics and Automation*, vol. 2, pp. 1275–1280, Raleigh, Caroline du Nord, Mars 1987.

[Aggarwal 88] J.K. Aggarwal, N. Nandhakumar. – On the computation of motion from sequences of images : a review. *Proc. of the IEEE*, 76(8):917–935, Août 1988.

[Agosta 88] J.M. Agosta. – The structure of bayes nets for vision recognition. – *Proc. of 4th Workshop on Uncertainty in Artificial Intelligence*, Minneapolis, Minnesota, Août 1988.

[Al Chami 94] O. Al Chami, C. Laugier. – Stratégie perceptive pour positionner une caméra. – *Congrès AFCET Reconnaissance des Formes et Intelligence Artificielle, RFIA'94*, vol. 1, pp. 617–622, Paris, France, Janvier 1994.

[Alami 93] R. Alami, R. Chatila, B. Espiau. – Designing an intelligent control architecture for autonomous robots. – *6th Int. Conf. on Advanced Robotics, ICAR'93*, pp. 435–440, Tokyo, Japon, Novembre 1993.

[Aloimonos 87] Y. Aloimonos, I. Weiss, A. Bandopadhay. – Active vision. *International Journal of Computer Vision*, 1(4):333–356, Janvier 1987.

[Aloimonos 88] Y. Aloimonos. – Visual shape computation. *Proc. of the IEEE*, 76(8):899–916, Août 1988.

[Aloimonos 90] Y. Aloimonos. – Purposive and qualitative active vision. – *IAPR Int. Conf. on Pattern Recognition, ICPR'90*, vol. 1, pp. 346–360, Atlantic City, New Jersey, Juin 1990.

[Aloimonos 93] Y. Aloimonos, E. Rivlin, L. Huang. – *Designing Visual Systems: Purposive Navigation,* In Active Perception, *Aloimonos Y. Editor*, pp. 47–102. – Lawrence Erlbaum Assoc. publishers, Hillsdale, New Jersey, 1993.

[Aloimonos 94] Y. Aloimonos. – What I have learned. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 60(1):74–85, Juillet 1994.

[Amanatides 87] J. Amanatides, A. Woo. – A fast voxel transversal algorithm for ray tracing. – *Proc. of Eurographics*, pp. 3–9, Amsterdam, Pays-Bas, Août 1987.

[Arbogast 91] E. Arbogast. – *Modélisation automatique d'objets non polyédriques par observation monoculaire.* – Thèse de Doctorat, Institut National Polytechnique de Grenoble, INPG, Juillet 1991.

[Ayache 83] N. Ayache. – *Un système de vision bidimensionnelle en robotique industrielle.* – Thèse de Doctorat, Université de Paris-Sud, Orsay, Juin 1983.

[Ayache 86] N. Ayache, O. Faugeras. – Hyper: A new approach for the recognition and positioning of two-dimensional objects. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8(2):44–54, Janvier 1986.

[Ayache 89a] N. Ayache. – *Vision stéréoscopique et perception multisensorielle.* – InterEditions, Paris, France, 1989.

[Ayache 89b] N. Ayache, O.D. Faugeras. – Maintaining representation of the environment of a mobile robot. *IEEE Trans. on Robotics and Automation*, 5(6):804–819, Décembre 1989.

[Bajcsy 88] R. Bajcsy. – Active perception. *Proc. of the IEEE*, 76(8):996–1005, Août 1988.

[Baker 91] T.P. Baker, O. Pazy. – Real time features for ADA 9X. – *IEEE Real-Time Systems Symposium*, pp. 172–180, San Antonio, Texas, Décembre 1991.

[Ballard 91] D.H. Ballard. – Animate vision. *Artificial Intelligence*, 48(1):57–86, Février 1991.

[Bayes 63] T. Bayes. – An essay towards solving a problem in the doctrine of chances. *Phil. Trans.*, 3:370–418, 1763. – Reproduced in *Two papers by Bayes*, ed. W.E. Deming, New-York: Hafner. 1963.

[Bensalah 95] F. Bensalah, F. Chaumette. – Compensation of abrupt motion changes in target tracking by visual servoing. – *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'95*, vol. 1, pp. 181–187, Pittsburgh, Pennsylvannie, Août 1995.

[Benveniste 91] A. Benveniste, G. Berry. – Real-time systems designs and programming. *Proc. of the IEEE*, 79(9):1270–1282, Septembre 1991.

[Benveniste 94] A. Benveniste. – Synchronous languages provide safety in reactive systems design. *Control Engineering*, pp. 87–89, Septembre 1994.

[Berry 87] G. Berry, P. Couronné, G. Gonthier. – Programmation synchrone des systèmes réactifs : le langage ESTEREL. *Technique et Science Informatiques, TSI*, 6(4):305–316, Avril 1987.

[Berry 89] G. Berry. – Real time programming: Special purpose languages or general purpose languages. – *11th IFIP World Congress*, pp. 11–17, San Francisco, Californie, Août 1989.

[Blake 90] A. Blake, R. Cipolla. – Robust estimation of surface curvature from deformation of apparent contours. – *1st European Conf. on Computer Vision, ECCV'90*, pp. 465–474, Antibes, France, Avril 1990.

[Bobet 94] P. Bobet. – *Tête stéréoscopique, réflexes oculaires et vision.* – Thèse de Doctorat, Institut National Polytechnique de Grenoble, INPG, Décembre 1994.

[Boissonnat 90] J.D. Boissonnat, O. Faugeras, E. Le Bras-Mehlman. – Representing stereo data with the Delaunay triangulation. *Artificial Intelligence*, 44(1-2):41–87, Juillet 1990.

[Bolles 86] R.C. Bolles, R. Horaud. – 3DPO: A three-dimensional part orientation system. *International Journal of Robotics Research*, 5(3):3–26, Juin 1986.

[Boukarri 89] B. Boukarri. – *Reconstruction 3D récursive de scènes structurées au moyen d'une caméra mobile. Application à la robotique.* – Thèse de Doctorat, Université d'Orsay, France, Octobre 1989.

[Boukir 93a] S. Boukir. – *Reconstruction 3D d'un environnement statique par vision active.* – Thèse de Doctorat, Université de Rennes 1, IRISA, Octobre 1993.

[Boukir 93b] S. Boukir, P. Bouthemy, F. Chaumette, D. Juvin. – Real-time contour matching over time in an active vision context. – *8^{th} Scandinavian Conference on Image Analysis, SCIA'93*, pp. 113–120, Tromsö, Norvège, Mai 1993.

[Bournai 93a] P. Bournai, B. Chéron, T. Gautier, B. Houssais, P. Le Guernic. – *Signal Manual.* – Rapport de Recherche n° 745, IRISA, Juillet 1993.

[Bournai 93b] P. Bournai, P. Le Guernic. – *Un environnement graphique pour le langage* SIGNAL. – Rapport de Recherche n° 741, IRISA, Juillet 1993.

[Boussinot 91] F. Boussinot, R. De Simone. – The ESTEREL language. *Proc. of the IEEE*, 79(9):1293–1304, Septembre 1991.

[Bouthemy 89] P. Bouthemy. – A maximum likelihood framework for determining moving edges. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11(5):499–511, Mai 1989.

[Boyer 96] E. Boyer. – Reconstruction et régularisation de la surface d'objets courbes. – *10ème Congrès Reconnaissance des formes et intelligence artificielle, RFIA'96*, vol. 1, pp. 23–32, Rennes, France, Janvier 1996.

[Brady 84] M. Brady, A. Yuille. – An extremum principle for shape from contour. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 6(3):288–301, Mai 1984.

[Brown 92] C. Brown. – Issues in selective perception. – *IAPR Int. Conf. on Pattern Recognition, ICPR'92*, vol. 1, pp. 21–24, La Haye, Pays Bas, Août 1992.

[Brown 94] C. Brown. – Toward general vision. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 60(1):89–91, Juillet 1994.

[Brunnström 96] K. Brunnström, J.O. Eklundh, T. Uhlin. – Active fixation for scene exploration. *International Journal of Computer Vision*, 17(2):137–162, Février 1996.

[Buffa 93] M. Buffa. – *Navigation d'un robot mobile à l'aide de la stéréovision et de la triangulation de Delaunay.* – Thèse de Doctorat, Université de Nice, INRIA Sophia Antipolis, Juin 1993.

[Buxton 95] H. Buxton, S. Gong. – Visual surveillance in a dynamic and uncertain world. *Artificial Intelligence*, 78(1-2):431–459, Octobre 1995.

[Cameron 90] A. Cameron, H. Durrant-Whyte. – A bayesian approach to optimal sensor placement. *International Journal of Robotics Research*, 9(5):70–88, Octobre 1990.

[Chang 95] T.F. Chang, R.V. Dubey. – A weighted least-norm solution based scheme for avoiding joints limits for redundant manipulators. *IEEE Trans. on Robotics and Automation*, 11(2):286–292, Avril 1995.

[Chaumette 89] F. Chaumette, P. Rives. – Modélisation et calibration d'une caméra. – *7ème congrès AFCET Reconnaissance des Formes et Intelligence Artificielle, RFIA'89*, vol. 1, pp. 527–536, Paris, Décembre 1989.

[Chaumette 90] F. Chaumette. – *La relation vision-commande: théorie et application à des tâches robotiques.* – Thèse de Doctorat, Université de Rennes I, IRISA, Juillet 1990.

[Chaumette 93] F. Chaumette, A. Santos. – Tracking a moving object by visual servoing. – *12th IFAC World Congress*, vol. 3, pp. 643–648, Sidney, Australie, Juillet 1993.

[Chaumette 96] F. Chaumette, S. Boukir, P. Bouthemy, D. Juvin. – Structure from controlled motion. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(5):492–504, Mai 1996.

[Chien 89] C. Chien, J.K. Aggarwal. – Model construction and shape recognition from occluding contour. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11(4):372–389, Février 1989.

[Cipolla 90] R. Cipolla, A. Blake. – The dynamic analysis of apparent contours. – *IEEE Int. Conf. on Computer Vision, ICCV'90*, pp. 616–623, Osaka, Japon, Décembre 1990.

[Cipolla 95] R. Cipolla. – *Active Visual Inference of Surface Shape.* – Lecture Notes in Computer Science 1016, Springer Verlag, Heidelberg, 1995.

[Connoly 85] C. Connoly. – The determination of next best views. – *IEEE Int. Conf. on Robotics and Automation*, pp. 432–435, St Louis, Missouri, Mars 1985.

[Coste-Manière 92] E. Coste-Manière, B. Espiau, D. Simon. – Reactive objects in a task level open controller. – *IEEE Int. Conf. on Robotics and Automation*, vol. 3, pp. 2732–2737, Nice, France, Mai 1992.

[Cowan 88] C.K. Cowan, P.D. Kovesi. – Automatic sensor placement from vision task requirements. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 10(3):407–416, Mai 1988.

[Crowley 92] J.L Crowley, P. Stelmaszyk, P. Puget. – Measurement and integration of 3D structures by tracking edge lines. *International Journal of Computer Vision*, 8(1):29–52, Juillet 1992.

[Crowley 95] J.L Crowley, M. Mesrabi, F. Chaumette. – Comparison of kinematic and visual servoing for fixation. – *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, vol. 1, pp. 335–341, Pittsburgh, Pennsylvannie, Août 1995.

[Djian 96] D. Djian, P. Probert, P. Rives. – Reconnaissance de modèles géométriques simples à l'aide de réseaux bayésiens. – *10ème Congrès AFCET Reconnaissance des Formes et Intelligence Artificielle, RFIA'96*, vol. 1, pp. 396–404, Rennes, France, Janvier 1996.

[DurrantWhyte 88] H.F Durrant-Whyte. – Sensor models and multisensor integration. *International Journal of Robotics Research*, 7(6):97–113, Décembre 1988.

[Dutertre 92] B. Dutertre. – *Spécification et preuves de systèmes dynamiques : Application à* SIGNAL. – Thèse de Doctorat, Université de Rennes I, IRISA, Décembre 1992.

[Espiau 87] B. Espiau, P. Rives. – Closed-loop recursive estimation of 3D features for a mobile vision system. – *IEEE Int. Conf. on Robotics and Automation*, vol. 3, pp. 1436–1443, Raleigh, Caroline du Nord, Avril 1987.

[Espiau 92] B. Espiau, F. Chaumette, P. Rives. – A new approach to visual servoing in robotics. *IEEE Trans. on Robotics and Automation*, 8(3):313–326, Juin 1992.

[Faugeras 87] O. Faugeras, F. Lustman, G. Toscani. – Motion and structure from motion from point and line matches. – *IEEE Int. Conf. on Computer Vision, ICCV'87*, pp. 25–34, Londres, Royaume Uni, Juin 1987.

[Faugeras 92a] O. Faugeras. – What can be seen in three dimensions with an uncalibrated stereo rig. – *2nd European Conf. on Computer Vision, ECCV'92*, pp. 536–578, Gênes, Italie, Mai 1992.

[Faugeras 92b] O. Faugeras, J. Mundy, N. Ahuja, C. Dyer, A. Pentland, R. Jain, K. Ikeuchi. – Why aspect graph are not (yet) practical for computer vision (workshop panel report). *Computer Vision, Graphics, and Image Processing: Image Understanding*, 55(2):212–218, Février 1992.

[Faugeras 93] O. Faugeras. – *Three-Dimensionnal Computer Vision: a Geometric Viewpoint.* – MIT Press, Cambridge, Massachusetts, 1993.

[Feddema 91] J.T Feddema, C.S.G. Lee, O.R. Mitchell. – Weighted selection of image features for resolved rate visual feedback control. *IEEE Trans. on Robotics and Automation*, 7(1):31–47, Février 1991.

[Fermüller 95] C. Fermüller, Y. Aloimonos. – Vision and action. *Image and Vision Computing*, 13(10):725–744, Décembre 1995.

[Gautier 94] T. Gautier, P. Le Guernic, O. Maffeïs. – *For a New Real-Time Methodology.* – Rapport de recherche, IRISA No 2364, Octobre 1994.

[Gennert 88] M.A. Gennert, A.L. Yuille. – Determining the optimal weights in multiple objective function optimization. – *IEEE Int. Conf. on Computer Vision, ICCV'88*, pp. 87–94, Tampa, Floride, 1988.

[Giblin 87] P. Giblin, R. Weiss. – Reconstruction of surfaces from profiles. – *IEEE Int. Conf. on Computer Vision, ICCV'87*, pp. 136–144, Londres, Royaume-Uni, Juin 1987.

[Grandjean 91] P. Grandjean. – *Perception multisensorielle et interprétation de scènes.* – Thèse de Doctorat, Université Paul Sabatier, LAAS, Toulouse, Octobre 1991.

[Hager 91] G. Hager, M. Mintz. – Computational methods for task-directed sensor data fusion and sensor planning. *International Journal of Robotics Research*, 10(4):285–313, Août 1991.

[Halbwachs 91] N. Halbwachs, P. Caspi, P. Raymond, D. Pilaud. – The synchronous data flow programming language LUSTRE. *Proc. of the IEEE*, 79(9):1305–1320, Septembre 1991.

[Halbwachs 93] N. Halbwachs. – *Synchronous programming of reactive systems.* – Kluwer, 1993.

[Harel 85] D. Harel, A. Pnueli. – On the development of reactive systems. – *Logics and Models of Concurrent Systems*, vol. 13 of *NATO ASI Series*, pp. 477–498, New York, 1985.

[Harel 87] D. Harel. – STATECHARTS: a visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, Juin 1987.

[Hashimoto 93] K. Hashimoto (édité par). – *Visual Servoing: Real Time Control of Robot Manipulators Based on Visual Sensory Feedback.* – World Scientific Series in Robotics and Automated Systems, Vol 7, World Scientific Press, Singapour, 1993.

[Herman 86] M. Herman, T. Kanade. – Incremental reconstruction of 3D scenes from multiple, complex images. *Artificial Intelligence*, 30(3):289–341, Décembre 1986.

[Hervé 92] J.Y. Hervé, Y. Aloimonos. – Exploratory active vision : theory. – *IEEE Int. Conf. on Computer Vision and Pattern Recognition, CVPR'92*, pp. 10–15, Champaign, Illinois, Juin 1992.

[Hoare 85] C.A.R. Hoare. – *Communicating Sequential Process.* – Prentise-Hall Int., Englewood Cliffs, 1985.

[Horn 89] B.K.P Horn, M.J Brooks. – *Shape from shading.* – MIT Press, Cambridge, Massachussets, 1989.

[Huang 91] L. Huang, J. Aloimonos. – Relative depth from motion using normal flow : an active and purposive solution. – *IEEE Workshop on Visual Motion*, pp. 196–204, Princeton, New Jersey, Octobre 1991.

[Hutchinson 89] S.A Hutchinson, A.C Kak. – Planning sensing strategies in a robot work cell with multi-sensor capabilities. *IEEE Trans. on Robotics and Automation*, 5(6):765–783, Décembre 1989.

[Jain 81] J.K. Jain, A.K. Jain. – Displacement measurement and its application in interframe image coding. *IEEE Trans. on Communications*, 29(12):1799–1808, Décembre 1981.

[Jensen 91] F.V. Jensen, H. Christensen, J. Nielsen. – *Bayesian methods for interpretation and control in multi-agent vision systems.* – Rapport de Recherche n° LIA 91-5, Aalborg University, Suède, Octobre 1991.

[Jourdan 94] M. Jourdan, F. Lagnier, F. Maraninchi, F. Raymond. – A multiparadigm language for reactive systems. – *IEEE Int. Conf. on Computer Languages*, Toulouse, France, Mai 1994.

[Kanade 95] T. Kanade, H. Kano, S. Kimura, A. Yoshida, K. Oda. – Development of a video-rate stereo machine. – *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'95*, vol. 3, pp. 95–100, Pittsburgh, Pennsylvannie, Août 1995.

[Kapellos 95] K. Kapellos. – *Environnement de programmation des applications robotiques réactives.* – Thèse de Doctorat, École des Mines de Paris, Novembre 1995.

[Karp 72] R.M. Karp. – *Reducibilty among combinatorial.* – In *R.E. Miller and J.W. Thatcher, Eds*, Complexity of computer computations, IBM research symposia series, Plenum, New York, 1972.

[Kirkpatrick 83] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi. – Optimization by simulated annealing. *Science*, 220(4598):671–680, Mai 1983.

[Koenderink 90] J. Koenderink. – *Solid shape.* – MIT Press, Cambridge, Massachusetts, 1990.

[Košeká 95] J. Košeká, H. Christensen, R. Bajcsy. – Discrete event modeling of visually guided behaviors. *International Journal of Computer Vision*, 14(2):179–191, Mars 1995.

[Krause 93] P. Krause, D. Clark. – *Representing uncertain knowledge: artificial intelligence approach.* – Kluwer Academic Publishers, Oxford, Royaume Uni, 1993.

[Kumar 96] V.P. Kumar, U.B. Desai. – Image interpretation using bayesian networks. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(1):74–77, Janvier 1996.

[Kutulakos 94a] K.N. Kutulakos, C.R. Dyer. – Recovering shape by purposive viewpoint adjustment. *International Journal of Computer Vision*, 12(2):113–136, Février 1994.

[Kutulakos 94b] K.N. Kutulakos, C.R. Dyer, V. Lumelsky. – Provable strategies for vision-guided exploration in three dimensions. – *IEEE Int. Conf. on Robotics and Automation*, vol. 2, pp. 1365–1372, San Diego, Californie, Juin 1994.

[Kutulakos 95] K.N. Kutulakos. – Affine surface reconstruction by purposive viewpoint control. – *IEEE Int. Conf. on Computer Vision, ICCV'95*, pp. 894–901, Cambridge, Massachusset, Juin 1995.

[Lauritzen 90] S. Lauritzen, D. Spiegelhalter. – Local computing with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Serie B*, 50(2):157–224, 1990.

[Le Borgne 87] M. Le Borgne. – *Quaternion et contrôle sur l'espace des rotations.* – Rapport de recherche, INRIA No 751, Novembre 1987.

[Le Borgne 91] M. Le Borgne, A. Benveniste, P. Le Guernic. – Dynamical Systems over Galois fields and DEDS control problems. – *33th IEEE Conf. on Decision and Control, CDC'91*, vol. 3, pp. 1505–1509, Brighton, Royaume Uni, Décembre 1991.

[Le Borgne 93] M. Le Borgne. – *Systèmes dynamiques sur des corps finis.* – Thèse de Doctorat, Université de Rennes I, IRISA, Septembre 1993.

[Le Guernic 91] P. Le Guernic, M. Le Borgne, T. Gautier, C. Le Maire. – Programming real time application with SIGNAL. *Proc. of the IEEE*, 79(9):1321–1336, Septembre 1991.

[Le Guernic 92] P. Le Guernic. – The SIGNAL programming environment. *Algorithms an Parallel VLSI Architecture II,* Elsevier Science Publishers, pp. 347–358, Septembre 1992.

[Liu 86] Y.C. Liu, T.S. Huang. – Estimation of rigid body motion using straight line correspondences : Further results. – *IAPR Int. Conf. on Pattern Recognition, ICPR'86*, pp. 306–309, Paris, France, Octobre 1986.

[Marchand 96] H. Marchand, E. Rutten, M. Le Borgne, M. Samaan. – Formal verification of SIGNAL programs: application to a power transformer station controller. – *Proc. of the 5th Int. Conf. on Algebraic Methodology and Software Technology, AMAST'96*, LNCS No 1101, pp. 270–285, Munich, Allemagne, Juillet 1996.

[Marr 82] D. Marr. – *-Vision- A computational investigation into the human representation and processing of visual information.* – W.H. Freeman, San Francisco, Californie, 1982.

[Martinez 94] F. Martinez. – *Intervalles de temps et tâches en langage* SIGNAL. – Rapport de DEA, IFSIC/IRISA, Université de Rennes I, France, Septembre 1994.

[Matthies 89] L. Matthies, T. Kanade, R. Szeliski. – Kalman filter-based algorithms for estimating depth from image sequences. *International Journal of Computer Vision*, 3(3):209–236, Septembre 1989.

[Maver 93] J. Maver, R. Bajcsy. – Occlusions as a guide for planning the next view. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(5):417–433, Mai 1993.

[Mesrabi 94] M. Mesrabi. – *Controle du Regard pour un Système de Vision Active.* – Thèse de Doctorat, Institut National Polytechnique de Grenoble, INPG, Mars 1994.

[Milios 93] E. Milios, M. Jenkin, J. Tsosos. – Design and performance of trish, a binocular robot head with torsional eye movements. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(1):51–68, Février 1993.

[Mitiche 89] A. Mitiche, G. Habelrih. – Interpretation of straight line correspondances using angular relations. *Pattern Recognition*, 22(3):299–308, 1989.

[Mohr 91] R. Mohr, L. Morin, E. Grosso. – Relative positionning with poorly calibrated cameras. – *DARPA-ESPRIT Workshop on Applications of Invariants in Computer Vision*, pp. 7–45, Reykjavik, Islande, Mars 1991.

[Moutarlier 91] P. Moutarlier, R. Chatila. – Incremental free-space modelling from uncertain data by an autonomous mobile robot. – *Workshop on Geometric Reasoning for Perception and Action*, LNCS No 708, Grenoble, France, Septembre 1991.

[Murray 95] D. Murray, K. Bradshaw, P. Mc Lauchlan, P. Sharkey. – Driving saccade to pursuit using image motion. *International Journal of Computer Vision*, 16(3):205–228, Mars 1995.

[Neapolitan 90] R. Neapolitan. – *Probabilistic Reasoning in Expert Systems.* – John Wiley, Chichester, 1990.

[Negahdaripour 87] S. Negahdaripour, B.K.P. Horn. – Direct passive navigation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 9(1):168–176, Janvier 1987.

[Nelson 93] B. Nelson, P.K. Khosla. – Increasing the tracking region of an eye-in-hand system by singularity and joint limit avoidance. – *IEEE Int. Conf. on Robotics and Automation*, vol. 3, pp. 418–423, Atlanta, Géorgie, Mai 1993.

[Nelson 94] B. Nelson, P.K. Khosla. – The resolvability ellipsoid for visual servoing. – *IEEE Int. Conf. on Computer Vision and Pattern Recognition, CVPR'94*, pp. 829–832, Seattle, Washington, Juin 1994.

[Nelson 95] B. Nelson, P.K. Khosla. – Strategies for increasing the tracking region of an eye-in-hand system by singularity and joint limits avoidance. *International Journal of Robotics Research*, 14(3):255–269, Juin 1995.

[Noton 71] D. Noton, L. Stark. – Eye movements and visual perception. *Scientific American*, 224(6):34–43, Juin 1971.

[Okutomi 91] M. Okutomi, T. Kanade. – A multiple-baseline stereo. – *IEEE Int. Conf. on Computer Vision and Pattern Recognition, CVPR'91*, pp. 63–69, Lahaina, Maui, Hawaï, Juin 1991.

[Papanikolopoulos 91] N. Papanikolopoulos, P.K. Khosla, T. Kanade. – Vision and control techniques for robotic visual tracking. – *IEEE Int. Conf. on Robotics and Automation*, vol. 1, pp. 857–864, Sacramento, Californie, Avril 1991.

[Pearl 86] J. Pearl. – Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, 29(3):241–288, Septembre 1986.

[Pearl 88] J. Pearl. – *Probabilistic reasoning in intelligent systems: networks of plausible inference.* – Morgan Kaufmann Publisher Inc., San Mateo, Californie, 1988.

[Pentland 84] A.P. Pentland. – Local shading analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 6(2):170–187, Mars 1984.

[Pissard-Gibollet 93] R. Pissard-Gibollet. – *Conception et commande par asservissement visuel d'un robot mobile.* – Thèse de Doctorat, École des Mines de Paris, Décembre 1993.

[Pito 96] R. Pito. – A sensor based solution to the next best view problem. – *IAPR Int. Conf. on Pattern Recognition, ICPR'96*, vol. A, pp. 941–945, Vienne, Autriche, Août 1996.

[Ramadge 89] P.J. Ramadge, W.M. Wonham. – The control of discrete events systems. *Proc. of the IEEE*, 77(1):81–97, Janvier 1989.

[Rimey 91] R.D. Rimey, C. Brown. – Controlling eye movements with hidden markov models. *International Journal of Computer Vision*, 7(1):47–65, Janvier 1991.

[Rimey 93] R.D. Rimey. – *Control of selective perception using Bayes nets and decision theory.* – PhD. Thesis, University of Rochester, 1993.

[Rimey 94] R.D. Rimey, C. Brown. – Control of selective perception using Bayes nets and decision theory. *International Journal of Computer Vision*, 12(2/3):173–207, Avril 1994.

[Rives 87] P. Rives, B. Espiau. – *Estimation récursive de primitives 3D au moyen d'une caméra mobile.* – Rapport de Recherche n° 652, INRIA-IRISA, Mars 1987.

[Rizzo 94] A. Rizzo. – Évitement des butées articulaires et des singularités internes en asservissement visuel. *Rapport de stage ERASMUS, IRISA*, Juin 1994.

[Robert 95] L. Robert. – Camera calibration without feature extraction. *Computer Vision, Graphics, and Image Processing*, 63(2):314–325, Mars 1995.

[Rutten 94] E. Rutten, P. Le Guernic. – Sequencing of data flow tasks in SIGNAL. – *ACM SIGPLAN Workshop on Language, Compiler and Tool Support for Real-Time Systems*, Orlando, Floride, Juin 1994.

[Rutten 95a] E. Rutten, P. Le Guernic. – Sequencing and preempting data flow tasks. – *Proc. of the 20^{th} IFAC/IFIP Workshop on Real Time Programming, WRTP'95*, Fort Lauderdale, Floride, Novembre 1995.

[Rutten 95b] E. Rutten, F. Martinez. – SIGNAL *GTi*: implementing task preemption and time intervals in the synchronous data flow language SIGNAL. – *7th Euromicro Workshop on Real Time Systems*, pp. 176–183, Odense, Danemark, Juin 1995.

[Safaee-Rad 92] R. Safaee-Rad, I. Tchoukanov, B. Benhabib, K.C. Smith. – 3D-pose estimation from a quadratic-curved feature in two perspective views. – *IAPR Int. Conf. on Pattern Recognition, ICPR'92*, vol. 1, pp. 341–344, La Haye, Pays Bas, Août 1992.

[Samson 91] C. Samson, M. Le Borgne, B. Espiau. – *Robot control: the task function approach.* – Clarendon Press, Oxford, Royaume Uni, 1991.

[Sandini 86] G. Sandini, M. Tistarelli. – Recovery of depth information: camera motion as an integration to stereo. – *IEEE Workshop on Motion*, pp. 39–43, Charleston, Caroline du Nord, Mai 1986.

[Sandini 90] G. Sandini, M. Tistarelli. – Active tracking strategy for monocular depth inference over multiple frames. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12(1):13–27, Janvier 1990.

[Sandini 94] G. Sandini, E. Grosso. – Why purposive vision? *Computer Vision, Graphics, and Image Processing: Image Understanding*, 60(1):109–112, Juillet 1994.

[SantosVictor 95] J. Santos-Victor, G. Sandini. – Docking behaviors via active perception. – *Int. Symposium on Intelligent Robotic Systems*, Pise, Italie, 1995.

[Sarkar 93] S. Sarkar, K. Boyer. – Integration, inference, and management of spatial information using Bayesian networks: perceptual organization. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(3):256–274, Mars 1993.

[Seales 92] W. Seales, C. Dyer. – Viewpoint from occluding contour. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 55(2):198–211, Mars 1992.

[Shafer 76] G. Shafer. – *A Mathematical Theory of Evidence*. – Princeton University Press, Princeton, New Jersey, 1976.

[Shen 92] J. Shen, S. Castan. – An optimal linear operator for step edge detection. *Computer Vision, Graphics, and Image Processing*, 54(2):13–17, Mars 1992.

[Simon 93] D. Simon, B. Espiau, E. Castillo, K. Kapellos. – Computer-aided design of a generic robot controller handling reactivity and real-time controller issues. *IEEE Trans. on Control Systems Technology*, 1(4):213–229, Décembre 1993.

[Sobh 92] M.T. Sobh, R. Bajcsy. – Visual observation under uncertainty as a discrete event processs. – *IAPR Int. Conf. on Pattern Recognition, ICPR'91*, pp. 429–432, La Haye, Pays Bas, Août 1992.

[Subbarao 87] M. Subbarao. – Solution and uniqueness of image flow equations for rigid curved surfaces in motion. – *IEEE Int. Conf. on Computer Vision, ICCV'87*, pp. 687–692, Londres, Royaume Uni, Juin 1987.

[Sundareswaran 94] V. Sundareswaran, P. Bouthemy, F. Chaumette. – Active camera alignment using dynamic image parameters. – *3rd European Conf. on Computer Vision, ECCV'94*, pp. 111–116, Stockholm, Suède, Mai 1994.

[Swain 93] M.J. Swain, M.A. Stricker. – Promising direction in active vision. *International Journal of Computer Vision*, 11(2):109–127, Octobre 1993.

[Tarabanis 91] K. Tarabanis, R. Tsai, P.K. Allen. – Automated sensor planning for robotic vision tasks. – *IEEE Int. Conf. on Robotics and Automation*, vol. 1, pp. 76–82, Sacramento, Californie, Avril 1991.

[Tarabanis 95a] K. Tarabanis, P.K. Allen, R. Tsai. – A survey of sensor planning in computer vision. *IEEE Trans. on Robotics and Automation*, 11(1):86–104, Février 1995.

[Tarabanis 95b] K. Tarabanis, R. Tsai, P.K. Allen. – The MVP sensor planning system for robotic vision tasks. *IEEE Trans. on Robotics and Automation*, 11(1):72–85, Février 1995.

[Tarbox 95] G.H. Tarbox, S.N. Gottschlich. – Planning for complete sensor coverage in inspection. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 61(1):81–111, Janvier 1995.

[Tarr 94] M.J. Tarr, M.J. Black. – A computational and evolutionary perspective on the role of representation in vision. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 60(1):65–73, Juillet 1994.

[Thomas 95] A.D.H. Thomas, M.G. Rodd, J.D. Holt, C.J. Neill. – Real time industrial visual inspection: A review. *Real Time Imaging*, 1(2):139–158, Juin 1995.

[Tistarelli 92] M. Tistarelli, G. Sandini. – Dynamic aspects in active vision. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 56(1):108–129, Juillet 1992.

[Tomasi 92] C. Tomasi, T. Kanade. – Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137–154, Novembre 1992.

[Toscani 87] G. Toscani. – *Système de calibration optique et perception du mouvement en vision artificielle.* – Thèse de Doctorat, Université d'Orsay, Décembre 1987.

[Triggs 95] B. Triggs, C. Laugier. – Automatic camera placement for robot vision. – *IEEE Int. Conf. on Robotics and Automation*, vol. 2, pp. 1732–1738, Nagoya, Japon, Mai 1995.

[Tsai 86] R.Y. Tsai. – An efficient and accurate camera calibration technique for 3D machine vision. – *IEEE Int. Conf. on Computer Vision and Pattern Recognition, CVPR'86*, pp. 364–374, Miami, Floride, Juin 1986.

[Tsai 87] R.Y. Tsai, R. Lenz. – *A new technique for autonomous and efficient 3D robotics hand-eye calibration.* – Rapport de Recherche n° RC12212, Yorktown Heights, New York, IBM T.J. Watson Research Center, Octobre 1987.

[Tsosos 87] J. Tsosos. – A complexity level analysis of vision. – *IEEE Int. Conf. on Computer Vision, ICCV'87*, Londres, Royaume Uni, Juin 1987.

[Tsosos 92] J.K. Tsosos. – On the relative complexity of active v.s. passive visual search. *International Journal of Computer Vision*, 7(2):127–141, Janvier 1992.

[Vaillant 90] R. Vaillant. – Using occluding contours for 3D objects modeling. – *1st European Conf. on Computer Vision, ECCV'90*, pp. 454–464, Antibes, France, Avril 1990.

[Vernon 90] D. Vernon, M. Tistarelli. – Using camera motion to estimate range for robotic part manipulation. *IEEE Trans. on Robotics and Automation*, 6(5):509–521, Octobre 1990.

[Viala 92] M. Viala. – *Contribution à la reconstruction de scènes constituées d'objets cylindriques et polyédriques à partir d'une séquence d'images acquises par une caméra en mouvement.* – Thèse de Doctorat, Université d'Orsay, Novembre 1992.

[Viéville 95] T. Viéville, E. Clergue, R. Enciso, H. Mathieu. – Experimentating with 3D vision on a robotic head. *Robotics and Autonomous Systems*, 14(1):1–27, 1995.

[Waxman 87] A.M. Waxman, B.K. Parsi, M. Subbarao. – Closed-form solutions to image flow equations for 3D structure and motion. *International Journal of Computer Vision*, 1(3):239–258, Octobre 1987.

[Wei 94] G.-Q. Wei, S.D. Ma. – Implicit and explicit camera calibration: Theory and experiments. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(5):469–480, Mai 1994.

[Wells 89] W.M. Wells. – Visual estimation of 3D lines segments from motion. A mobile robot vision system. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 5(6):820–825, Décembre 1989.

[Weng 89] J. Weng, T.S. Huang, N. Ahuja. – Motion and structure from two perspective views : algorithms, error analysis, and error estimation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11(5):451–476, Mai 1989.

[Weng 92a] J. Weng, P. Cohen, N. Rebiho. – Motion and structure estimation from stereo image sequences. *IEEE Trans. on Robotics and Automation*, 8(3):362–382, Juin 1992.

[Weng 92b] J. Weng, T.S. Huang, N. Ahuja. – Motion and structure from line correspondences : closed-form solution, uniqueness, and optimization. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(3):318–336, Mars 1992.

[Whaite 94] P. Whaite, F. Ferrie. – Autonomous exploration: Driven by uncertainty. – *IEEE Int. Conf. on Computer Vision and Pattern Recognition, CVPR'94*, pp. 339–346, Seattle, Washington, Juin 1994.

[Witkin 81] A.P. Witkin. – Recovering surface shape and orientation from texture. *Artificial Intelligence*, 17(1-3):17–47, Août 1981.

[Wixson 94a] L.E. Wixson. – *Gaze Selection for Visual Search.* – PhD. Thesis, University of Rochester, Computer Science Dept., New York, 1994.

[Wixson 94b] L.E. Wixson. – Viewpoint selection for visual search. – *IEEE Int. Conf. on Computer Vision and Pattern Recognition, CVPR'94*, pp. 800–805, Seattle, Washington, Juin 1994.

[Wu 88] J.J. Wu, R.E. Rink, T.M. Caelli, V.G. Gourishankar. – Recovery of 3D location and motion of a rigid object through camera image (an extended Kalman filter approach). *International Journal of Computer Vision*, 2(4):373–384, Avril 1988.

[Xie 88] M. Xie, P. Rives. – *Identification des polygones 3D à partir des segments 3D.* – Rapport de recherche, IRISA-INRIA Rennes, No 427, Septembre 1988.

[Xie 89a] M. Xie. – *Contribution à la vision dynamique : reconstruction d'objets 3D polyédriques par une caméra mobile.* – Thèse de Doctorat, Université de Rennes 1, IRISA, Juin 1989.

[Xie 89b] M. Xie, P. Rives. – Toward dynamic vision. – *IEEE Workshop on Interpretation of 3D Scenes*, pp. 91–99, Austin, Texas, Novembre 1989.

[Yarbus 67] A.L. Yarbus. – *Eye Movements and vision.* – Plenum Press, 1967.

[Zell 95] J.F. Zell. – *Utilisation de silhouettes pour la perception et la modélisation de l'environnement en robotique.* – Thèse de Doctorat, Université / École centrale de Nantes, LAN, Février 1995.

[Zhang 90] Z. Zhang, O. Faugeras. – Building a 3D world model with a mobile robot: 3D line segment representation and integration. – *IAPR Int. Conf. on Pattern Recognition, ICPR'90*, pp. 38–42, Atlantic City, New Jersey, Juin 1990.

[Zhang 92] Z. Zhang, O. Faugeras. – Finding clusters and planes frome 3D line segments with application to 3D motion determination. – *2nd European Conf. on Computer Vision, ECCV'92*, pp. 227–236, Gênes, Italie, Juin 1992.

[Zhang 95] Z. Zhang. – Estimating motion and structure from correspondences of line segments between two perspective images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(12):1129–1139, Décembre 1995.

[Zhao 93] C. Zhao. – *Reconstruction de surfaces tridimensionnelles en vision par ordinateur.* – Thèse de Doctorat, Institut National Polytechnique de Grenoble, INPG, Décembre 1993.

[Zhao 94] C. Zhao, R. Mohr. – Relative 3D regularized B-Splines surface reconstruction through image sequences. – *3rd European Conf. on Computer Vision, ECCV'94*, vol. 2, pp. 417–426, Stockholm, Suède, Mai 1994.

Publications

Revues

[1] E. Marchand, E. Rutten, F. Chaumette. – From Data-Flow Task to Multi Tasking : Applying the Synchronous Approach to Active Vision in Robotics. – *IEEE Transactions on Control Systems Technology*, 5(2), Mars 1997.

[2] E. Marchand, F. Chaumette. – Contrôle actif d'une caméra pour la reconstruction et l'exploration de scènes. – *Technique et Science Informatiques, TSI*, 16(1), Janvier 1997.

[3] E. Rutten, E. Marchand, F. Chaumette. – An experiment with reactive data-flow tasking in active robot vision. – *Software : Practice & Experience*, (À paraitre, 1997).

Conférences internationales

[4] E. Marchand, F. Chaumette, A. Rizzo. – Using the task function approach to avoid robot joint limits and kinematic singularities in visual servoing. – *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'96*, Osaka, Japon, Novembre 1996.

[5] E. Marchand, A. Rizzo, F. Chaumette. – Avoiding robot joint limits and kinematic singularities in visual servoing. – *13th IAPR/IEEE Int. Conf. on Pattern Recognition, ICPR'96*, Vol. A, pp. 297–301, Vienne, Autriche, Août 1996.

[6] E. Marchand, F. Chaumette. – Controlled camera motions for scene reconstruction and exploration. – *IEEE Int. Conf. on Computer Vision and Pattern Recognition, CVPR'96*, pp. 169–176, San Francisco, Californie, Juin 1996.

[7] E. Marchand, F. Chaumette, E. Rutten. – Real time active visual reconstruction using the synchronous paradigm. – *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'95*, Vol. 1, pp. 96–102, Pittsburgh, Pennsylvannie, Août 1995.

[8] E. Marchand, F. Chaumette. – Active visual 3D perception. – *IEEE Workshop on Vision for Robots*, pp. 10–17, Pittsburgh, Pennsylvannie, Août 1995.

[9] E. Marchand, F. Chaumette. – Real time estimation of 3D environment with an active vision system. – *Int. Symposium on Intelligent Robotic Systems*, pp. 311–318, Grenoble, France, Juillet 1994.

[10] E. Rutten, E. Marchand, F. Chaumette. – The sequencing of data flow tasks in SIGNAL : application to active vision in robotics. – *6th Euromicro Workshop on Real Time Systems*, IEEE Computer Society Press, pp. 80–84, University of Maerlardalen, Väesterås, Suède, Juin 1994.

Conférences nationales

[11] E. Marchand, F. Chaumette, A. Rizzo. – Évitement des butées articulaires et des singularités internes en asservissement visuel. – *5ème journées ORASIS*, pp. 37–42, Clermont-Ferrand, Mai 1996.

[12] E. Marchand, F. Chaumette. – Reconstruction et exploration de scènes par vision active. – *10ème Congrès AFCET Reconnaissance des Formes et Intelligence Artificielle, RFIA'96*, Vol. 1, pp. 405–414, Rennes, Janvier 1996.

[13] H. Marchand, E. Marchand, E. Rutten. – Spécification et vérification de systèmes réactifs : expérimentations de la méthodologie synchrone SIGNAL – *Congrès AFCET "Modélisation des systèmes réactifs"*, pp. 3–11, Brest, Mars 1996.

[14] E. Marchand, F. Chaumette. – Réalisation d'un réseau d'automates pour la reconstruction incrémentale d'une scène 3D. – *4ème journées ORASIS*, Mulhouse, pp. 26–30, Octobre 1993.

Rapports de recherche

[15] E. Marchand, F. Chaumette, E. Rutten. – Stratégie perceptive d'un environnement statique dans un contexte de vision active. – Rapport de recherche INRIA No 2092 (IRISA No 775), Janvier 1994.

[16] E. Marchand, E. Rutten, F. Chaumette. – Applying the synchronous approach for real time active visual reconstruction. – Rapport de recherche INRIA No 2383 (IRISA No 875), Octobre 1994.

Résumé

Ce travail apporte sa contribution au problème de la reconstruction et de l'exploration de scènes statiques dans un contexte de **vision active**. À la base du processus de reconstruction, nous avons choisi une méthode qui consiste à contraindre les mouvements de la caméra de manière à obtenir une estimation précise et robuste de primitives géométriques paramétrables telles que les segments et les cylindres. À cet aspect *continu* du processus de reconstruction que constitue l'estimation des paramètres des primitives, il est nécessaire de définir des stratégies de reconstruction et d'exploration de la scène, supposée constituée de segments, polyèdres et cylindres. Cette reconstruction est de caractère *événementiel* et est pilotée par la découverte de nouvelles primitives dans l'image. L'approche que nous avons définie consiste à sélectionner automatiquement les informations images pertinentes puis à focaliser successivement la caméra sur les différentes primitives de la scène afin de les reconnaître et ensuite de les reconstruire.

La première étape de l'exploration, qui inclut la reconstruction 3D, permet de reconstruire de manière incrémentale l'ensemble des primitives qui apparaissent dans le champ de vision de la caméra. Nous avons appelé cette phase **exploration locale** car elle ne fait appel qu'à des informations disponibles localement. Elle repose sur une approche de prédiction / vérification d'hypothèses gérées à l'aide de réseaux Bayesiens. Cette approche permet d'obtenir une représentation de plus haut niveau des objets considérés tout en traitant les problèmes locaux d'occlusion. Par contre, quand toutes les primitives précédemment observées ont été reconstruites, une stratégie différente doit été mise en œuvre afin de focaliser la caméra sur des zones de la scène n'ayant pas encore été observées. Cette étape d'**exploration globale** permet d'assurer la complétude de la reconstruction. Cette méthode repose sur l'optimisation par ICM multi-échelle d'une fonction de coût adéquatement modélisée qui prend en compte les obstacles de la scène.

Finalement, les algorithmes développés ont été spécifiés et mis en œuvre par le langage synchrone SIGNAL permettant de l'intégration au sein du même formalisme, SIGNAL et SIGNAL*GTi*, de la dualité continu / événementiel inhérente à ce type d'algorithme.

Les méthodes que nous avons développées ont été mises en œuvre sur la cellule de vision robotique de l'IRISA. Elles permettent de reconstruire en temps réel de façon précise, robuste, complète et totalement autonome, un environnement 3D composé de plusieurs primitives.

Mots-Clefs : Stratégies de perception, vision active, robotique, reconstruction 3D, exploration de scènes, commande, asservissement visuel, réseaux Bayésiens, langages synchrones, SIGNAL.