N° d'Ordre : 1114
EDSPIC : 194

UNIVERSITÉ BLAISE PASCAL - CLERMONT II

ECOLE DOCTORALE

SCIENCES POUR L'INGÉNIEUR DE CLERMONT-FERRAND

Formation Doctorale :

Electronique et systèmes

Thèse

présentée par

François BERRY

pour obtenir le grade de

DOCTEUR D'UNIVERSITÉ

(SPÉCIALITÉ : Vision pour la Robotique)

Contournement d'objet par asservissement visuel

Soutenue publiquement le 2 février 1999 devant le jury :

Monsieur	M. Michel DHOME	Président
Monsieur	M. François CHAUMETTE	Rapporteur
Monsieur	M. Dominique MEIZEL	Rapporteur
Monsieur	M. Jean GALLICE	Examinateur
Monsieur	M. Philippe MARTINET	Examinateur

A mes parents,
mon frère,
Lolo.

Remerciements

LES travaux présentés dans cette thèse ont été effectués au sein du GRoupe Automatique VIsion et Robotique (GRAVIR) du LAboratoire des Sciences et Matériaux pour l'Electronique, et d'Automatique (LASMEA) de l'Université Blaise Pascal de Clermont-Ferrand, Unité Mixte de Recherche 6602 du Centre National de la Recherche Scientifique (CNRS).

Tout d'abord, je tiens à remercier Monsieur M. RICHETIN, Professeur à l'Université Blaise Pascal, directeur du LASMEA pour son accueil au sein du laboratoire.

J'exprime ma sincère reconnaissance à Monsieur M. DHOME, Directeur de Recherche à l'université Blaise Pascal pour l'honneur qu'il m'a fait en acceptant de juger ce travail et de présider le jury de soutenance.

Je remercie également Monsieur D. MEIZEL, Professeur à l'Université Technologique de Compiègne, ainsi que Monsieur F. CHAUMETTE, Chargé de recherche à l'IRISA de Rennes pour l'intérêt qu'ils ont porté à ce travail en acceptant d'en être rapporteurs.

Enfin, un vif merci à Monsieur J. GALLICE et Monsieur P. MARTINET respectivement Professeur et Maître de conférence au Centre Universitaire des Sciences et Techniques (C.U.S.T.) pour l'attention avec laquelle ils ont suivi ces travaux.

Bien entendu, une pensée toute particulière à mes parents qui ont su me soutenir et m'encourager quand la dérivée du moral devenait négative, ainsi qu'à Lolo© (c'est une marque déposée) à qui j'ai fait revivre les hautes heures de la rédaction. Soyez ici remercier pour votre patience, votre tolérance et bien plus encore...

Enfin, je ne terminerai pas sans remercier toutes les personnes, de mon entourage qui ont contribué à me faire penser à autre chose qu'à l'asservissement visuel. Domi, le dernier et (probablement seul) utilisateur de transputer T9000 au monde! Que ta brillante carrière, au pays de la moutarde, soit aussi haute en couleurs que nos con....ies au labo. En second lieu, Marmot, DOCTEUR [1] de son état, qui a régulièrement fait les frais de nos bêtises tout en réalisant une très belle thèse!

Merci à tous les autres, Eric, Fred, Alex, Djamel, Jocelyn, DD, Popol[2], Yoyo, les secrétaires JJ, Pascale, Eliane, Françoise et Christine, tous les autres doctorants et permanents que je ne peux citer individuellement, clients assidus du bar du premier étage, ainsi que le quinze montferrandais pour m'avoir fait prendre l'air 80 minutes par samedi de rédaction!

1. Debugeur Officiel et Confirmé de Transputer Enervant Uniquement Ruineux
2. Et en plus, il est docteur !!!

Table des matières

Table des figures

Liste des tableaux

Introduction

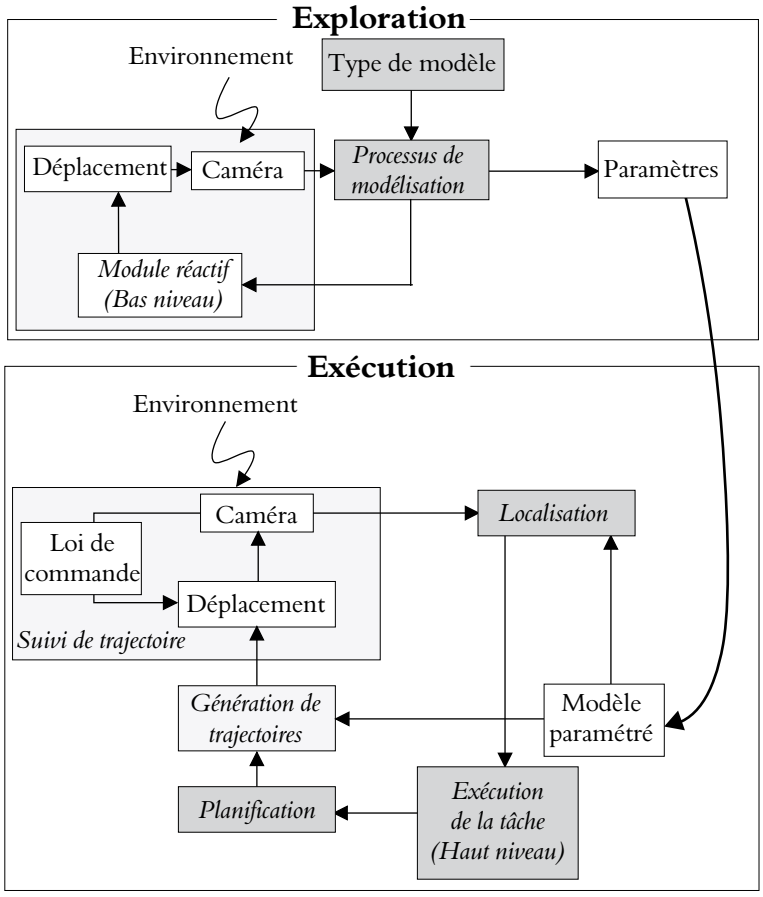
VERS la fin des années 60, quelques grands centres de recherche comme le MIT (Massachusetts Institute of Technology), le SRI (Stanford Research Institute) ou l'université d'Edinbourg ont lancé de vastes projets de recherche dont les maîtres mots étaient *Robots intelligents*. De là à penser qu'auparavant, tous les robots étaient idiots?

Il est en réalité difficile et ambigu de parler véritablement de l'intelligence d'un robot. Un terme plus approprié serait autonomie, dans le sens où les motivations des roboticiens de l'époque (et d'aujourd'hui) ne sont pas de fabriquer un futur *Prix Nobel*, mais plutôt une machine versatile pouvant s'adapter aux modifications de l'environnement. Une des clefs de cette adaptation repose sur l'utilisation de capteurs d'environnement permettant au robot de prendre en compte ces "conditions de travail". Très rapidement la caméra vidéo s'est avérée être un des capteurs extéroceptifs les plus performants et fournissant une information particulièrement riche. Toutefois, l'évolution de la vision pour la robotique a été (et est encore) largement conditionnée par les progrès des technologies périphériques comme la microélectronique ou les systèmes dédiés (architecture parallèle par exemple).
Cependant, malgré cette explosion technologique le problème de *perception pour l'action* se trouve toujours d'actualité et dans la plupart des cas de figures, on se trouve encore devant le dilemme *Complexité des calculs/Temps d'exécution.* Ce dilemme est particulièrement pénalisant dans le cadre de la vision pour la robotique où il souhaitable d'avoir un rafraîchissement et un traitement élevé des informations afin de pouvoir commander les actionneurs du robot dans de bonnes conditions.

Le sujet abordé dans ce manuscrit traite du contournement d'objet par robot manipulateur. Cette approche s'intègre typiquement dans un contexte de navigation, dans le sens où la méconnaissance de l'objet implique un processus incrémental d'actions de perception et de déplacements. Toutefois dans le cas d'objets inconnus, le problème peut être scindé en deux phases distinctes:

- **l'exploration,**
- **l'exécution de tâches.**

Ces deux phases peuvent être schématisées comme suit:



Sur cette figure, apparaîssent différents modules (notés en italique) qui interviennent dans chacune des deux phases.

L'objectif de la phase d'exploration est d'évaluer les paramètres d'un modèle générique d'objet (ellipsoïde d'inertie). Pour cela, il est nécessaire de pouvoir se déplacer autour de l'objet inconnu en toute sécurité (éviter les collisions). Ceci est assuré par le *module réactif* qui permet de traduire les informations visuelles et les déplacements requis par la *modélisation* en actions sur l'effecteur.

Dans la phase exécutive, on utilise les paramètres calculés durant l'exploration pour réaliser différentes actions autour de l'objet (planification, inspection, préhension, usinage, ...). Les mouvements ne sont alors plus appliqués en "boucle ouverte[1]" comme précédemment, mais sont contrôlés par une boucle d'asservissement visuel assurant le suivi de trajectoire. Les paramètres du modèle sont alors nécessaires à

1. Le module réactif permet de contrôler uniquement l'interaction caméra/objet.

deux niveaux: *Génération de trajectoire* et *localisation*. Ce dernier module permet d'informer le système sur sa situation par rapport à l'objet et intervient directement sur les modules haut niveau (*Exécution de tâche* et *planification*).

Durant ce travail de thèse, nous nous sommes plus particulièrement intéressés aux problèmes de commande à partir d'informations visuelles. Pour cette raison, nos recherches se sont focalisées sur les modules (coloriés en gris clair sur la figure précédente) où interviennent des boucles d'asservissement visuel.

Partant de cet état de fait, ce rapport se décompose en trois parties présentées comme suit:

Le premier chapitre situe **notre problématique** par rapport au vaste domaine de la navigation.
Aprés avoir présenté les différents types d'architecture robotique mis en œuvre à ce jour, nous mettons en évidence la nécessité d'une architecture hybride pour résoudre le problème du contournement d'objet. Dans un second temps, nous abordons les techniques d'asservissement visuel qui peuvent être utilisées dans ce type d'architecture.

Dans un second chapitre, nous nous sommes intéressés aux problèmes de **la génération et du suivi de trajectoire par asservissement visuel**.
Pour cela, nous avons mis en œuvre une loi de commande référencée vision dont l'originalité repose sur l'utilisation d'une consigne visuelle variable dans le temps. Cette variation de consigne nous permet, sous certaines conditions, de contrôler le déplacement de la caméra autour d'un objet (modélisé). La fin de ce chapitre propose différents résultats obtenus en simulation et sur notre plate-forme expérimentale.

Le troisième chapitre traite le problème du module réactif en abordant le problème **des déplacements autour d'un objet inconnu**.
Après avoir défini les conditions de navigation, différentes primitives utilisables sont proposées pour la boucle de commande référencée vision. Pour chacune d'entre elles, nous établissons la tâche robotique associée, indépendamment des spécificités de la scène observée. Nous réalisons ainsi différents mouvements autour de l'objet, en gardant une distance fixe entre la cible et le capteur. Toutefois cette approche permet de se déplacer autour de la scène mais ne considère pas la situation entre la caméra et l'objet. Dans une dernière partie nous présentons les différents résultats obtenus sur le site expérimental du laboratoire nous permettant de valider la méthodologie utilisée.

En dernier lieu, nous concluons ce mémoire en évoquant quelques applications potentielles de l'étude présentée. Aussi suggérons-nous en guise de perspectives quelques idées parachevant la méthode mise en œuvre.

Chapitre 1

Navigation et asservissement visuel

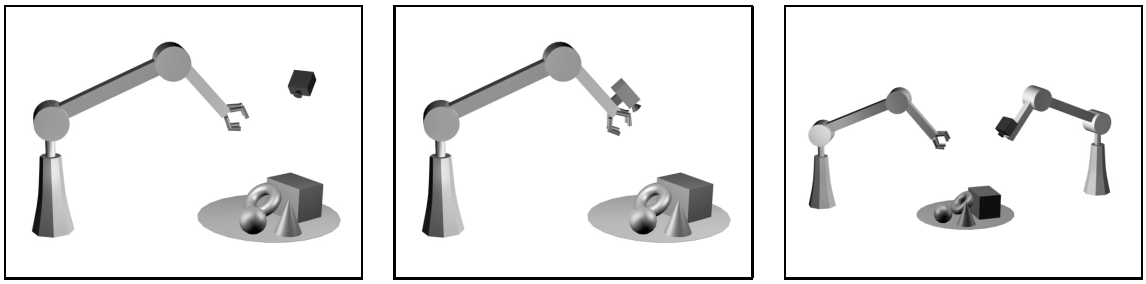
1.1 Introduction

La plupart des manipulateurs industriels sont des machines ne possédant aucune capacité de perception intrinsèque. En environnement manufacturier, un programme de commande fait l'hypothèse que les pièces à manipuler arrivent à l'endroit désiré avec une orientation correcte selon un ordre donné et en un temps voulu. C'est ce qui se passe encore dans la majorité des usines où la plupart des applications fonctionnent avec succès sans aucune référence sensorielle extéroceptive.

Alors quel intérêt peut-on trouver à l'utilisation de capteurs et en particulier à la vision artificielle?

Cette question naïve ne trouve pas sa réponse uniquement dans les laboratoires de recherches, mais aussi dans le monde économique de la robotique. En effet, les industriels sont de plus en plus intéressés pour des systèmes "versatiles" permettant de réunir en une machine, un instrument relativement polyvalent et évitant l'intervention humaine. Cependant l'abstraction de l'homme dans le processus (hormis à un niveau supérieur, telle que la programmation des tâches, ...), nécessite indubitablement l'emploi de capteurs extéroceptifs permettant de renseigner le robot sur son environnement. La caméra vidéo semble être un des moyens le mieux adapté pour répondre à ce problème.

Toutefois dans la plupart des applications alliant caméra et robot on distingue généralement trois types de configurations entre le manipulateur et le système de vision (Fig.1.1).



(a) (b) (c)

FIG. 1.1 – *Configurations possibles entre un robot manipulateur et un système de vision.*

- La première consiste à utiliser une **caméra fixe qui observe la scène** dans sa globalité. La relation entre la vision et la commande est alors unilatérale dans le sens où les informations visuelles permettent de commander le robot mais le déplacement du robot n'interagit pas sur la position de la caméra. Ce type de configuration mis en œuvre par de nombreux chercheurs (Salganicoff, Metta, Oddera and Sandini, 1993) (Grosso, Metta, Oderra and Sandini, 1995) (Allen, Yoshimi and Timcenko, 1991) (Yoshimi and Allen, 1996) (Jagersand and Nelson, 1994) (Jägersand, Fuentes and Nelson, 1997), présente l'avantage de voir la scène de "l'extérieur", mais a l'inconvénient de ne proposer qu'une vue unique de la scène considérée.

- La seconde possibilité consiste à **embarquer la caméra sur le manipulateur** à commander. La caméra évolue alors dans la scène avec l'organe qu'elle contrôle. C'est ce type de configuration que nous emploierons dans la suite de ce manuscrit. Du point de vue de la commande, l'effecteur paraît être l'endroit idéal pour la caméra. En effet, c'est cet organe qui interagit avec l'environnement et c'est son mouvement qui définit l'exécution d'une tâche. On peut ainsi espérer extraire de la caméra les données nécessaires à la régulation de la tâche envisagée.

- Enfin, une troisième possibilité rassemble les avantages des deux précédentes possibilités en embarquant **la caméra sur un second manipulateur**. La caméra observe alors la scène de "l'extérieur" tout en ayant la possibilité de se mouvoir. Contrairement au premier cas, on admet ainsi des possibilités de *vision locale* et de *vision globale*. Ce genre d'approche est largement employé dans les problématiques de placement optimal de caméra (Zheng, Chen and Tsuji, 1991) (Abrams, Allen and Tarabanis, 1993).

Ce chapitre s'articule autour de quatre parties.

Dans un premier temps, après avoir défini la **notion de tâche robotique**, nous en présentons trois grands aspects :

- le positionnement par rapport à une cible fixe,
- le suivi de cible mobile,
- la navigation.

Dans une seconde partie, nous rappelons les principes de la **navigation en robotique** en situant notre problématique par rapport à ce vaste sujet. Nous montrons en particulier, que notre approche traitant du contournement d'objet est typiquement une tâche de navigation pouvant se décomposer en plusieurs modules. Nous mettons ainsi en évidence la nécessité d'une partie réactive que nous réaliserons avec une boucle d'asservissement visuel.

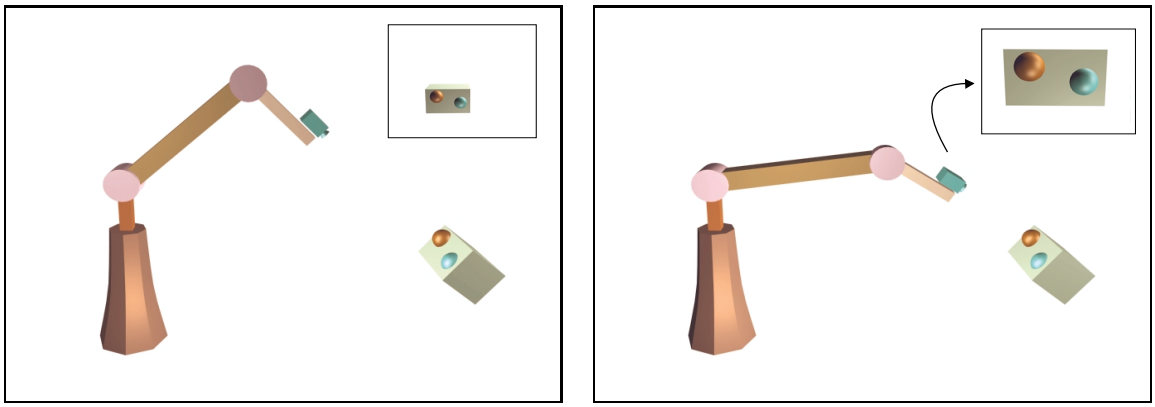
La troisième partie est consacrée aux **techniques d'asservissement visuel**. Après avoir brièvement rappelé le principe de ces techniques, nous proposons un rapide tour d'horizon des travaux effectués dans le domaine. Cette approche nécessite toutefois une perception de l'environnement performante évitant ainsi toute ambiguité et permettant une bonne réalisation de la commande. A travers différentes recherches, nous exposons quelques méthodes d'extraction de primitives visuelles utilisées dans le cadre de l'asservissement visuel.

1.2 Types de tâches robotiques

Avant d'effectuer la commande de robot, il est généralement nécessaire de déterminer les tâches que l'on désire lui faire accomplir. C'est ce que l'on nomme "tâches robotiques". Dans notre approche, nous avons choisi de classer ces tâches suivant trois grandes catégories: le positionnement, le suivi et la navigation.
Dans ces différents cas de figure, le but recherché est le contrôle de la position ou de la trajectoire du robot à partir d'informations fournies par un capteur. Dans le cas de la caméra vidéo, il a fallu attendre les années 70 pour véritablement définir le concept de *vision pour la robotique*. Les progrès techniques, en particulier au niveau des caméras solides [1], ont alors permis la mise en œuvre de cette approche à travers des applications essentiellement industrielles (Saisie d'objets, Suivi de soudure, Positionnement d'outils,...). Nous en présentons ici quelques facettes à travers différents travaux effectués durant ces vingt dernières années.

1.2.1 Positionnement

Le but d'une tâche de positionnement est d'amener le robot dans une situation d'équilibre donnée (Fig. 1.2). Ce type de tâche trouve de nombreuses applications en particulier dans le secteur industriel. Cependant les premières tâches de positionnement couplant un robot à une caméra se révélaient plus être des études de faisabilité plutôt que de réelles recherches.



(a) (b)

FIG. 1.2 – *Exemple de positionnement: à l'instant initial (a) et à l'instant final (b).*

1. Par opposition aux caméras à tube, les caméra à transfert de charges encore notées CCD (Charge Coupled Device) sont constituées d'un substrat semi-conducteur sensible aux variations lumineuses.

Chronologiquement, on trouve Shiraï et Inoue (Shirai and Inoue, 1973) qui ont été parmi les premiers à utiliser les données d'une caméra fixe pour améliorer le positionnement d'un préhenseur. Le système fonctionnait à une cadence de 0.1 Hz et devait saisir un objet cubique afin de le déposer dans une boîte (Fig. 1.3). Agin (Agin, 1977) propose un système pour visser automatiquement les têtes de compresseur. Une caméra et le dispositif de vissage étaient embarqués sur l'effecteur d'un bras Unimate à 6 d.d.l et effectuaient des tâches de positionnement afin d'aligner la vis à fixer en face de l'écrou. Cependant, le traitement d'image se révélait être encore une étape délicate et se résumait bien souvent à de simples traitements sur une image binaire. En 1983, Coulon et Nougaret (P.Y.Coulon and M.Nougaret, 1983) utilisent des fenêtres d'intérêt afin de réduire la quantité d'information à traiter. Leur manipulation consistait à localiser un objet cible dans l'image et à réaliser un asservissement en "XY" afin de saisir l'objet visé.

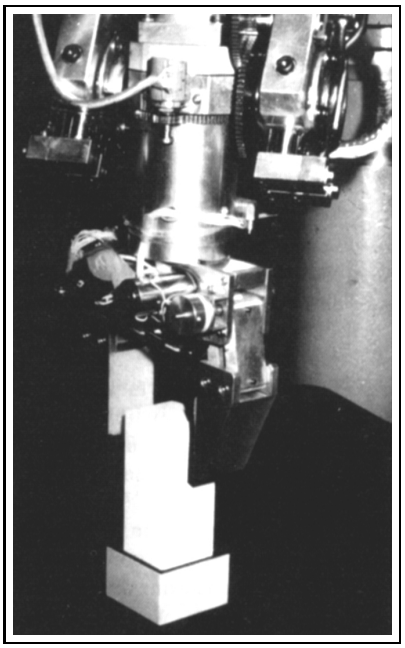


Fig. 1.3 – *Positionnement d'un objet dans une boite (Shirai and Inoue, 1973)*

Toutefois, ce n'est qu'en 1984, avec Lee E. Weiss (Weiss, 1984) que le contrôle de robot par retour visuel est réellement formalisé. Dans sa thèse, Weiss couple une boucle d'asservissement visuel à un processus adaptatif lui permettant d'évaluer "en ligne" les relations non linéaires entre les primitives images et la position de l'effecteur. La particularité de l'approche de Weiss est de ne pas utiliser de boucle interne pour la commande des actionneurs contrairement à Feddema (Feddema and Mitchell, 1989) et Chaumette (Chaumette, 1990). Ce dernier adapte dans sa thèse

l'approche "fonction de tâche" (Samson, Espiau and Borgne, 1991) au cas du capteur visuel et formalise ainsi *la commande référencée vision*. Cette commande relativement performante trouve son originalité en considérant la caméra comme un capteur particulier associé à une tâche donnée et directement inséré dans la boucle de commande.

1.2.2 Suivi de cible

Dans le cadre d'une tâche de suivi, on considère généralement une cible mobile. Le problème ne revient plus simplement à amener le robot dans une situation d'équilibre, mais à estimer le mouvement de la cible de façon à pouvoir maintenir l'interaction entre la cible et le capteur.

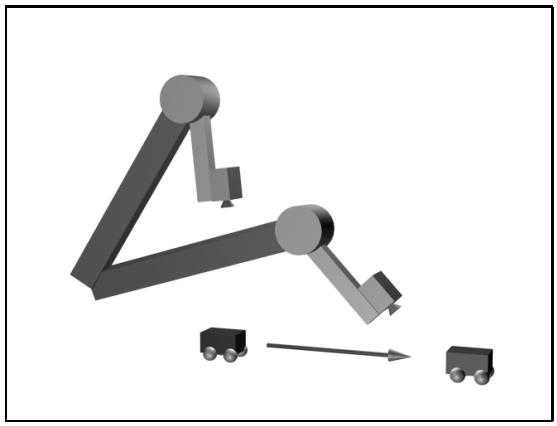


FIG. 1.4 – *Exemple de suivi de cible mobile.*

Cette estimation peut se faire de plusieurs manières. La plupart des recherches utilisent des méthodes prédictives mettant en œuvre différents types de filtrage (Kalman, Rapport de vraisemblance,...). Parmi les nombreux travaux, citons Prajoux (Prajoux, 1979) qui asservit un mécanisme à deux degrés de liberté par retour visuel afin de suivre la course d'un crochet sur un tapis roulant. Le système utilise un filtre prédicteur afin de prévoir la position du crochet à chaque itération.

Gilbert dans (Gilbert, Giles, Flachs, Rogers and Yee, 1980) réalise un suivi de fusée en commandant l'orientation d'une caméra Pan/tilt. Le suivi permet de garder la cible centrée dans l'image pendant qu'un système de traitement d'image l'identifie. En 1991, Chaumette (Chaumette, Rives and Espiau, 1991) propose un suivi avec estimation de la vitesse de la cible. Cette estimation faite simplement par intégration sera améliorée dans (Chaumette and Santos, 1993) avec l'utilisation d'un filtre de Kalman. Ce dernier est aussi employé par Allen (Allen, Timcenko, Yoshimi and

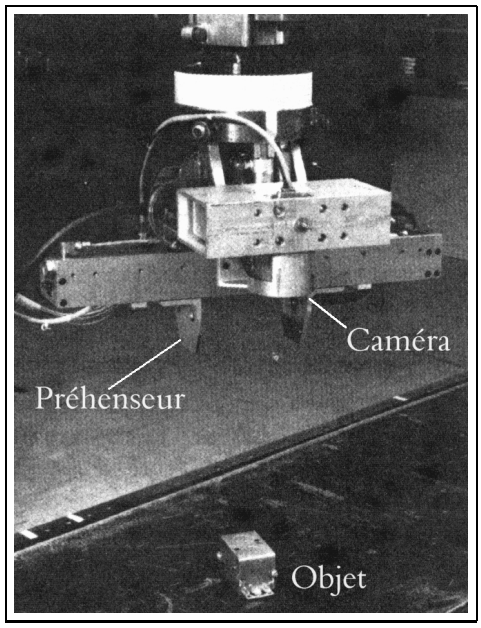


Fig. 1.5 – *Suivi d'objet en manufacture*

Michelman, 1992) pour suivre un train miniature. Allen couple un système stéréoscopique fixe à un détecteur de mouvement basé sur la méthode de Horn et Schunck (Horn and Schunk, 1981). Il en déduit ainsi la position 3D de la cible à chaque instant qu'il injecte dans un filtre de Kalman. Papanikolopoulos dans (Papanikolopoulos, Nelson and Khosla, 1995), couple une loi adaptative avec une mesure de flot optique par corrélation. Ses expérimentations consistent à suivre des objets tels qu'un livre, un bloc-notes,... Dernièrement, des primitives dynamiques de l'image (paramètres affines du mouvement) ont aussi été utilisées par Cretual dans (Cretual and Chaumette, 1998) afin de réaliser un suivi de piéton en mouvement à l'aide d'une caméra Pan/Tilt.

1.2.3 Navigation

L'objectif d'une tâche de navigation est de répondre au problème fondamental :

"A partir d'une situation A, aller dans une situation B suivant un chemin donné"

Pour cela, le robot doit réaliser quatre actions (Lacroix, 1995) :

– Modéliser et interpréter l'environnement,

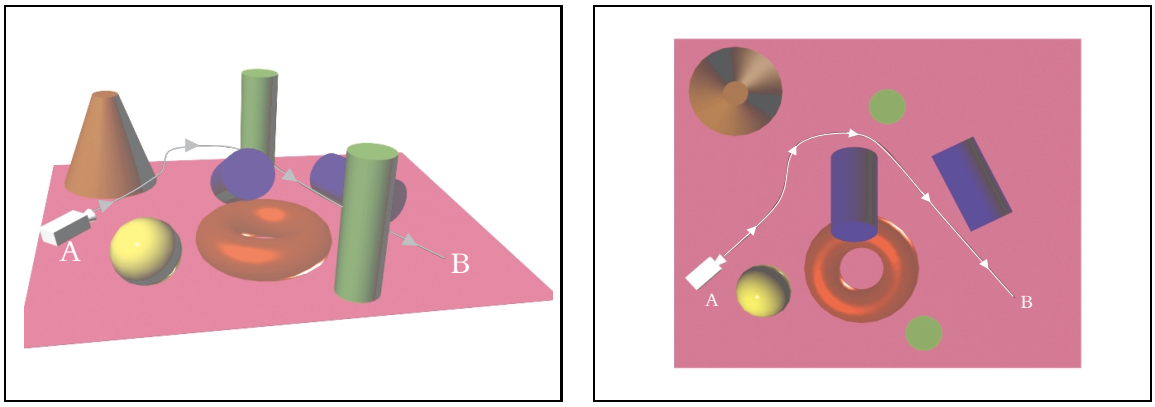


FIG. 1.6 – *Navigation de A vers B*

– Se localiser dans cet environnement,

– Planifier des déplacements,

– Assurer la bonne exécution des déplacements planifiés.

Il est bien évident que le déroulement de ces différentes actions est avant tout contraint par la nature de l'environnement. En ce qui concerne les travaux effectués autour de ce type de tâche, la majeure partie considère des plate-formes mobiles. Plusieurs laboratoires nationaux (LAAS, INRIA, ONERA, Alcatel, ...) et internationaux (CMU, MIT, NASA, ...) ont lancé depuis plusieurs années de vastes projets de recherche permettant de développer de nombreux prototypes (Navlab, Adam, Robby, Rocky, ...). Les objectifs de ce type de projets sont multiples et impliquent dans la plupart des cas une capacité d'autonomie plus ou moins grande.

A titre d'exemples, citons quelques applications envisagées :

↪ ***Environnement naturel:***

– Exploration en environnement contraint : nucléaire, sous-marin, spatial, routier, ...

– Applications militaires : déminage, reconnaissance, ...

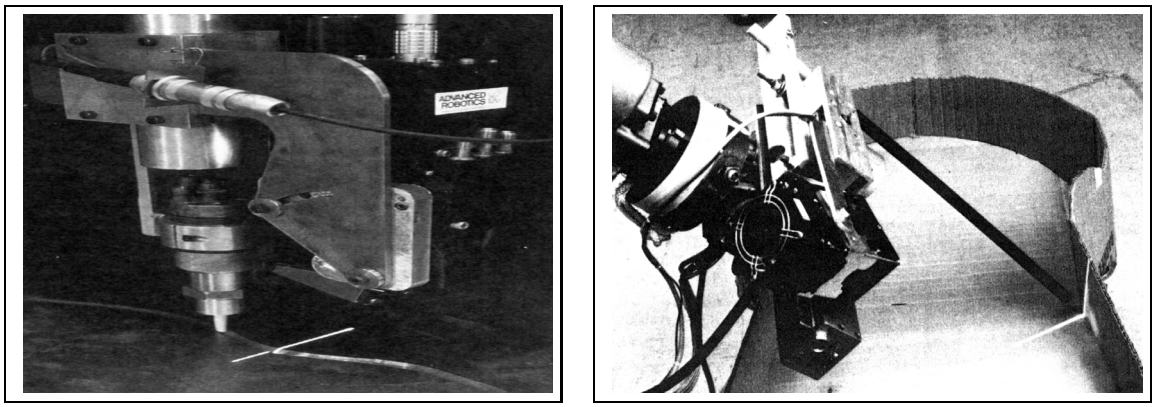
↪ ***Scènes d'intérieur:***

– Aide aux handicapés, ...

– Déplacement de pièces en milieu manufacturier, ...

Par contre, dans le cadre de robots manipulateurs couplés à un système de vision, à notre connaissance, peu de travaux ont été réalisés. Ceci résulte principalement du faible *espace des configurations* dont dispose ce type de robot.

Les principaux travaux réalisés dans le cadre d'applications industrielles concernent le suivi de trajectoires dans des tâches de type *suivi de soudure* ou *découpe de pièces.*



(a) (b)

Fig. 1.7 – *(a) Suivi de soudure et (b) Suivi de chemin dans un cadre manufacturier (Agin, 1977).*

Toutefois cette approche de la navigation est complètement différente de celle présentée précédemment dans le sens où la notion d'autonomie est quasiment inexistante. En effet, dans ce type de tâches, le robot se contente uniquement d'exécuter les mouvements planifiés à l'aide d'une boucle de contrôle faisant intervenir différents capteurs (Sur la Figure 1.7, une raie laser et une caméra) .

Une "véritable" application de la tâche de navigation en robotique manipulatrice est *la vision active* au sens large du terme. En effet cette approche consiste à déplacer le capteur, en l'occurrence une caméra, de façon à explorer, reconstruire ou reconnaître la scène observée. Les principaux concepts de ce type d'études ont été initiés par Aloimonos et Bajcsy[2] (Aloimonos, Weiss and Bandopadhay, 1987) (Bajcsy, 1988) (Aloimonos, 1990) mais accusent malheureusement un manque évident d'expérimentations.

Dans ce domaine, citons toutefois les travaux d'Eric Marchand (Marchand, 1996) qui met en œuvre des stratégies de reconstruction et d'exploration dans une scène statique. Les déplacements de la caméra sont effectués à l'aide d'une boucle d'asservissement visuel et permettent de reconnaître et de reconstruire les différentes

2. Dans leurs travaux, ces approches ne considèrent pas réellement la commande de la caméra mais se restreignent plutôt à l'aspect perceptif.

primitives de la scène.

1.2.4 Notre problématique

L'étude que nous nous proposons d'aborder dans ce manuscrit est le **contournement d'objet inconnu**. Cette problématique s'intègre complètement dans le domaine de la navigation au sens où notre objectif est de pouvoir déplacer une caméra autour d'un objet en conservant à chaque instant l'interaction entre la caméra et l'objet. La finalité de notre approche devrait alors permettre de générer des mouvements autour d'objets inconnus afin d'en effectuer un premier apprentissage et de répondre ainsi à des **problèmes d'initialisation de processus de perception/ reconnaissance/action**.

A partir de là, on peut alors imaginer différentes applications comme un système **d'inspection automatique d'objets** basé sur la recherche de points caractéristiques à valider. Une autre application peut être la **classification automatique de pièces** suivant des caractéristiques données.
Le domaine de la **téléopération assistée** permet aussi d'envisager de nombreux débouchés. Citons pour exemple, le projet BAROCO du CNES qui a pour objectif la réalisation d'une maquette expérimentale permettant d'étudier des fonctions "télérobotiques" en environnement spatial. Dans ce projet, l'aspect "perception" est fortement associé au concept de "robotique avancée", concernant en particulier la planification temporelle d'actions, le contrôle d'exécution de scénarios planifiés, la téléopération niveau "tâche", ...

Toutefois, ces ambitieuses applications font appel à de nombreux points dont

- la définition de tâches robotiques réalisables autour d'un objet inconnu,
- la possibilité d'estimer la situation *observateur/observé* ,
- des stratégies de déplacement permettant d'optimiser la navigation directement dans l'espace image,
- ...

et plus particulièrement dans notre cas :

- la génération et le suivi de trajectoire dans l'espace image,
- l'enchaînement de tâches référencées vision,
- les problèmes de perception dans le cas d'un objet inconnu ou non modélisé.

Nous pouvons constater que les domaines abordés sont multiples et couvrent aussi bien le traitement d'image que l'automatique ou bien encore l'intelligence artificielle sans oublier bien sûr la robotique!

1.3 Différentes approches de la navigation

Les différents concepts que nous présentons ici ont tous été développés initialement dans le contexte de la robotique mobile. Toutefois, ils peuvent sans aucune ambiguité être appliqués aux autres domaines de la robotique et en particulier à celui des manipulateurs.

En reprenant le problème fondamental énoncé au §1.2.3, on trouve dans la littérature trois démarches:

- Les **approches hiérarchiques**, qui comportent plusieurs niveaux décisionnels et exécutifs. Les actions du robot ne peuvent être que le résultat d'un raisonnement.
- Les **approches réactives**, selon lesquelles le robot réagit uniquement aux "stimuli" externes de l'environnement. Les mouvements du robot résultent alors directement de comportements réflexes.
- Les **approches hybrides**, qui tendent à tirer parti au mieux des avantages des deux approches précédentes.

1.3.1 Systèmes hiérarchiques

L'approche hiérarchique est relativement classique en intelligence artificielle et repose sur la décomposition fonctionnelle des différentes tâches à exécuter. Classiquement cette décomposition suit la structure suivante:

Capteurs
⇓
Perception de l'environnement
⇓
Modélisation de l'environnement
⇓
Planification
⇓
Exécution des trajectoires
⇓
Effecteur

Suivant ce développement, le robot doit tout d'abord traiter les données issues du ou des capteurs, à la suite de quoi il élabore une représentation de ce qu'il perçoit.

C'est à partir de cette représentation qu'il va alors planifier un itinéraire lui permettant d'effectuer sa mission. S. Lacroix dans sa thèse (Lacroix, 1995) redécompose le problème général de planification hiérarchique en trois sous-ensembles :

- La planification d'itinéraires qui à partir de la connaissance de l'environnement et de la mission à effectuer, permet de définir un chemin nominal et d'éventuels chemins alternatifs. Le planificateur d'itinéraire détermine ainsi une succession ordonnée de tronçons (sous-buts) prenant en considération non seulement le but à atteindre, mais aussi d'éventuelles contraintes à respecter. Il a un rôle de superviseur dans la tâche de planification.

- La planification de chemins dont le rôle est de déterminer le chemin qui mène au prochain sous-but planifié par le planificateur d'itinéraires. Le planificateur de chemins dispose de la description initiale du tronçon courant ainsi que des informations perçues par le capteur.

- La planification de trajectoires consiste à déterminer la séquence de déplacements élémentaires permettant de suivre le chemin à effectuer. Le rôle de ce module est donc de rallier le sous-but trouvé par le planificateur de chemins.

Cette approche typiquement issue de la robotique mobile prend son origine dans l'idée de décomposer une tâche de navigation conséquente (plusieurs centaines de mètres) en un ensemble de sous-tâches permettant alors des processus de recalage par rapport à l'environnement.

En ce qui concerne les problèmes de planification, notons qu'une abondante littérature a été réalisée dans le domaine. Le lecteur désireux d'approfondir l'étude de la planification pourra se référer au livre de Jean Claude Latombe, "Robot Motion Planning" (Latombe, 1991) qui est à notre connaissance le plus complet sur le sujet.

Le principal avantage des systèmes hiérarchiques est la possibilité d'intégrer des raisonnements de haut niveau (mission, ...) s'appuyant sur des modèles de types cartes.

Les inconvénients majeurs de cette approche sont le manque de souplesse (adaptation à des conditions légèrement différentes) et la nécessité de modéliser l'environnement. Ce dernier est pénalisant à deux niveaux puisqu'il nécessite dans la plupart des cas un temps de calcul élevé et un environnement d'évolution modélisable.

1.3.2 Systèmes réactifs

Dans ce type de systèmes, le robot ne fait que réagir aux stimuli de son environnement. Intrinsèquement parlant, il ne dispose pas de tâche particulière à effectuer et

n'a, par conséquent, pas de plans d'action à exécuter. Par tradition, les architectures réactives s'opposent aux architectures hiérarchiques.

Cette approche consiste à abandonner le concept centré sur la modélisation de l'environnement et le raisonnement, au profit d'une structure totalement distribuée privilégiant des réflexes perception $\Leftrightarrow$ action. Ce type d'approche peut être retrouvé dans la nature chez les insectes par exemple. Une mouche ne dispose pas de grandes capacités de raisonnement, mais peut se déplacer sans encombre dans une pièce et effectuer des trajectoires relativement complexes.

L'un des premiers a avoir introduit la notion de réactivité est R. Brooks (Brooks, 1985), qui définit une structure en couche de comportements dont les premiers niveaux sont les suivants :

$$\text{Capteurs} \Rightarrow \begin{array}{c} \text{4 - Remarquer les changements dans l'environnement} \\ \hline \text{3 - Construire une carte de l'environnement et des routes} \\ \hline \text{2 - Explorer le monde en repérant des points atteignables} \\ \hline \text{1 - Se déplacer sans but précis} \\ \hline \text{0 - Eviter les objets mobiles ou non} \end{array} \Rightarrow \text{Effecteurs}$$

On parle alors d'approche comportementale par opposition aux approches cognitives. Chaque niveau de compétences traduit un comportement désiré du robot directement entre le capteur et l'effecteur. Par exemple le niveau 0, dont le but est d'éviter les obstacles, peut consister à faire virer la trajectoire du robot de façon à éviter l'obstacle.
Dans cette architecture, l'action ne résulte pas d'un raisonnement mais de plusieurs comportements actifs simultanés.

Zapata (Zapata, 1991) propose d'utiliser une structure hiérarchiquement organisée en 4 niveaux : l'arrêt d'urgence, l'évitement d'obstacles, l'exécution de déplacement prédéfinis par une mission donnée et le suivi de cible. La gestion des couches de plus bas niveau (arrêt d'urgence et évitement d'obstacles) fait intervenir la notion de Zones Virtuelles Déformantes (ZVD).

Une ZVD est une enveloppe entourant le robot et dont la forme est déterminée par :

- L'état cinématique du robot : déformations contrôlées,
- Les informations capteur : déformations non contrôlées.

La navigation consiste alors à générer des commandes en fonction des modules haut niveau tout en minimisant la déformation de la ZVD. Cette technique a notamment été mise en œuvre par Cyril Novales dans le cadre de sa thèse (Novales, 1994), où il cherchait à commander de façon réflexe de petits robots mobiles.

Les points forts d'une structure réactive sont la robustesse (les différents couches sont indépendantes), la rapidité de réponse, le faible coût et l'aspect incrémental permettant d'insérer une couche sans pour autant modifier l'état global de l'architecture.
En contrepartie, ce type d'approche accuse un manque de capacités de raisonnement et se trouve par conséquent limité dans des applications complexes.

1.3.3 Systèmes hybrides

Comme nous pouvons le constater, les deux approches précédentes présentent à travers leurs spécificités des caractéristiques intéressantes. L'architecture hiérarchique permet d'envisager une grande autonomie grâce à ses capacités de raisonnement, tandis que les structures réactives proposent des réponses relativement rapides et robustes. Afin de combiner les avantages de ces deux approches, certains chercheurs ont proposé des approches hybrides combinant les avantages de chacune des deux architectures (Payton, 1986) (Arkin, 1987).

Le principe des systèmes hybrides est d'élaborer une architecture hiérarchisée dont les différents modules sont exécutés en parallèle.

Payton décompose le problème en quatre étapes :

Haut niveau

- Planification de mission: Mission=ensemble de buts géographiques
- Planification à partir de cartes : Liaison des buts géographiques par des routes
- Planification locale : Parcours de la route
- Planification réflexe : Contrôle du robot en temps réel

Bas niveau

Le contrôle est effectué de manière descendante tandis qu'un retour sur l'état des différents niveaux remonte du *"bas niveau"* vers le *"haut niveau"*. Un module superviseur de perception multi-niveau permet en cas de conflits d'arbitrer les priorités entre les différentes couches.

1.3.4 Position du problème

A partir de ces trois types d'architecture, essayons à présent de replacer le problème du **contournement d'objet par robot manipulateur** dans le contexte de la navigation. Pour cela, intéressons-nous aux propriétés de ces différentes architectures.

• **Hiérarchique** :

Le point fort de l'architecture hiérarchique réside dans la possibilité d'intégrer des processus de raisonnement de haut niveau dans la tâche de navigation. En contrepartie, ces processus nécessitent une connaissance assez complète de l'environnement dans lequel évolue le robot. Le fait de ne pas avoir de connaissances *a priori* sur la scène nécessite donc obligatoirement une étape de reconstruction. Dans le cadre de notre application, cette étape coûteuse en temps de calcul paraît être inadaptée. De plus, dans le cas d'un environnement totalement inconnu, il paraît relativement difficile de vouloir planifier des mouvements alors que l'on ne sait pas où l'on doit aller!

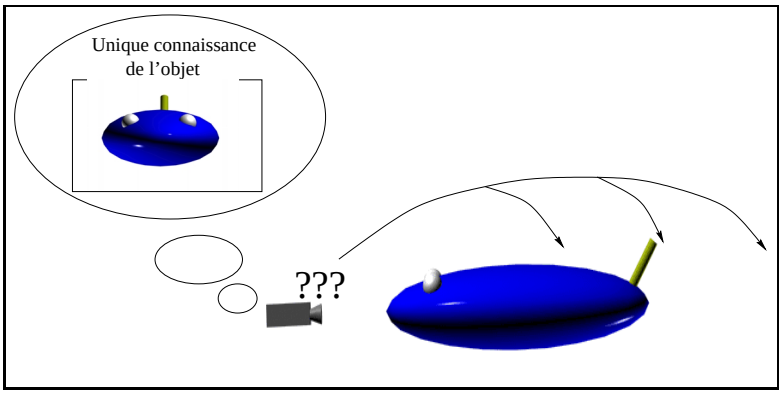


FIG. 1.8 – *Approche hiérarchique autour d'un objet inconnu*

Pour illustrer ce cas de figure, considérons notre manipulateur "en face" d'un objet quelconque (Fig.1.8), dont il n'a aucune connaissance.
Une mission pouvant être donnée par l'utilisateur au système est :

```
"Va derrière l'objet"
```

En considérant que le capteur utilisé permet uniquement d'obtenir l'image de l'objet (ce qui sera notre cas dans la suite), comment peut-on déterminer la position finale `[derrière l'objet]`, alors que l'on ne connaît pas la taille de l'objet. A fortiori,

comment planifier un chemin dans cette direction[3]?

Il est donc nécessaire d'effectuer avant la planification une étape d'exploration permettant de modéliser l'objet à contourner. L'exploration nécessite une combinaison de déplacements et d'actions réflexes permettant une "découverte" de l'objet tout en évitant les collisions.

• **Réactive** :

Comme nous l'avons vu précédemment, l'approche réactive est particulièrement bien adaptée pour la gestion des actions réflexes, mais reste pénalisante dés qu'il s'agit de raisonner à un certain niveau.
Dans notre cas, une telle architecture peut permettre au robot de se déplacer autour de l'objet sans pour autant lui donner une grande autonomie. Cette approche semble donc relativement intéressante pour une phase d'exploration où il est nécessaire de pouvoir "voir" sous différents angles en gardant une "distance de sécurité" objet-caméra.

Une des caractéristiques majeures de l'architecture réactive est l'absence d'interprétation et un lien direct entre action et perception réalisant ainsi une réponse réflexe. Dans notre cas de figure, le manipulateur est équipé d'une caméra CCD fournissant une image monochrome de l'objet visé. Par conséquent, un des choix naturels pour réaliser ce lien perception-action est d'utiliser une boucle d'asservissement visuel. Ce type de commande permet d'intégrer directement dans la commande les informations issues du capteur.

Dans un système réactif, les mouvements de la caméra autour de l'objet sont alors assurés "sans encombre", mais pas pour autant coordonnés par un module de plus haut niveau fixant les déplacements à réaliser.

• **hybride** :

Il semble qu'une architecture hybride de par sa prise en compte tant des aspects haut niveau (mission d'exploration, ...) que des aspects bas niveau (contrôle réactif) soit la mieux adaptée à notre problématique.

Le problème revient alors à définir les différentes couches de notre système de façon à pouvoir **réaliser notre objectif**. Notre système doit pouvoir :

- Percevoir l'objet,
- Se déplacer autour de l'objet,
- Contrôler ses déplacements,
- Modéliser l'objet,

3. Dans le sens où une direction peut être considérée comme une succession de positions.

- Se situer par rapport à l'objet,
- Planifier des actions autour de l'objet.

Ce cahier des charges peut donc se diviser en deux étapes chronologiques : **l'exploration** et **la planification d'action** ayant pour dénominateur commun **le contrôle des déplacements autour de l'objet**.

Ce contrôle constitue donc une couche bas niveau purement réactive que nous avons choisie de réaliser avec une boucle d'asservissement visuel.

1.3.5 Asservissement visuel

La commande d'un robot par un capteur visuel a fait l'objet de nombreuses recherches. Cependant, si l'utilisation de la caméra comme capteur principal dans la boucle de commande semble aujourd'hui naturelle (grâce en particulier aux puissances de calcul disponibles), il y a une vingtaine d'années ceci nécessitait une lourde mise en œuvre pas toujours disponible. On explique par ce fait les différentes structures étudiées dans le cadre de l'asservissement visuel.

Ces dernières peuvent généralement être classées suivant deux catégories[4]:

- **Asservissement dans l'espace cartésien**. Dans ce type de structure, l'image est tout d'abord acquise puis interprétée. Cette interprétation utilise, dans la plupart des cas, une connaissance *a priori* (modèle) de la scène observée ce qui confère à ce type d'approche un haut niveau de hiérarchisation. Dans la littérature, de tels travaux se rencontrent sous la dénomination *Dynamic Look-and-Move*, *Position-based Look-and-Move* ou encore *Look-and-Move*. Dans la suite de ce manuscrit, nous appellerons ce type d'asservissement **Position-based Look-and Move**.
- **Asservissement dans l'image**. Au contraire, dans cette structure, les primitives extraites de l'image sont directement injectées dans la loi de commande. On évite toute l'étape d'interprétation et par conséquent les problèmes de calibrage et de modélisation. Cette approche trouve là encore de nombreux qualificatifs tels *Image-based Look-and-Move*, *Visual Servo*, *Direct Visual Servo*, *Visual Servoing* ou bien encore *Commande Référencée Vision* dont la dénomination est propre à l'approche *fonction de tâche*.
  Par contraste avec la structure précédente, dans la suite de ce manuscrit nous nommerons ce type d'asservissement **Image-based Look-and-Move**.

4. Dans sa classification, Weiss propose 4 schémas en incluant ou non la présence d'une boucle interne au niveau des actionneurs du robot.

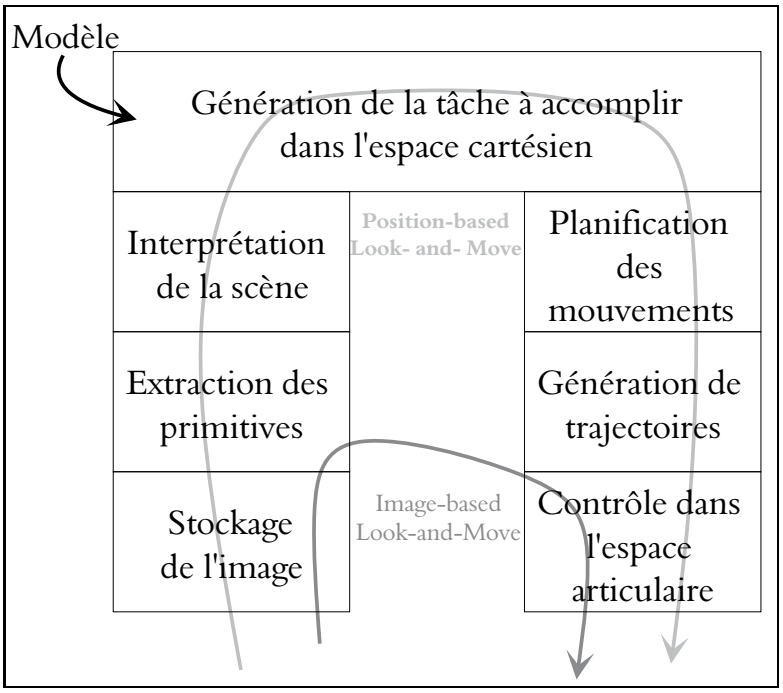


Fig. 1.9 – *Comparaison hiérarchique de l'approche Position-based Look-and-Move et de la Commande Référencée Vision (Corke, 1996)*

Dans l'approche "Position-based Look-and-Move", le déplacement du robot est effectué après avoir évalué la situation courante du robot et la pose[5] à atteindre. Comme on peut le constater sur la figure 1.9, ce type de technique nécessite une modélisation de la scène observée ainsi qu'une large phase d'interprétation. En comparaison, la structure "Image-based Look-and-Move" intervient comme un "*court-circuit bas niveau*". Comme l'explique Corke dans (Corke, 1996), d'un point de vue biologique cette technique peut être considérée comme un comportement réflexe. En effet, le principe de ces méthodes est d'élaborer une relation différentielle entre les variations dans l'espace du capteur et les mouvements du robot. Cette relation permet alors de spécifier la tâche à accomplir directement dans l'espace du capteur et non plus dans l'espace cartésien ou articulaire.

5. On entend par pose les 6 paramètres représentant la position et l'orientation

1.3.5.1 Position-based Look-and-Move

Ces méthodes ont été les premières à être mises en œuvre dans le courant des années 70. Les limitations technologiques de l'époque ont contraint les recherches à s'orienter vers l'approche:

1 - On regarde la scène,

2 - On interpréte l'image,

3 - On estime la situation robot/scène,

4 - On bouge le robot.

Dans ce type de structure, l'espace de commande du robot est l'espace cartésien. L'interprétation des données visuelles fournit la pose du robot par rapport à la scène et permet ainsi un asservissement sur les 3 paramètres de position et les 3 paramètres de rotation (Fig. 1.10).

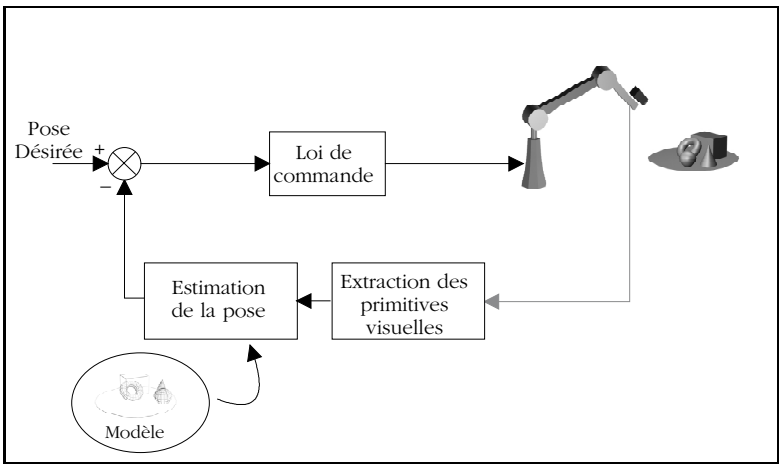


FIG. 1.10 – *Struture "Position-based Look and Move"*

On pourra cependant noter deux approches chronologiques:

– Dans les premières expérimentations réalisées à la fin des années 70, C.G. Agin(Agin, 1977), (Agin, 1979) met en œuvre **une commande en position** à des cadences comprises entre 500ms et 2s. Dans cette approche, les phases d'acquisition/interprétation s'enchaînent séquentiellement avec la phase de déplacement. Nous sommes alors dans le cas d'une pseudo-boucle ouverte puisque l'action n'est pas simultanément contrôlée par la perception.

– Depuis le début des années 90, on note différents travaux basés sur des asservissements de poses (Wilson, 1993) (Fagerer, Dickmanns and Dickmanns, 1994) (Westmore and Wilson, 1991) (Rizzi and Koditschek, 1996) (Wilson, Hulls and Bell, 1996) (Martinet, Gallice and Khadraoui, 1996b). Le principe de ces travaux consistent à utiliser une modélisation de la scène pour estimer la pose de l'outil par rapport à cette dernieère. On utilise alors **une commande en vitesse** pour amener l'outil de la pose initale à la pose finale.

Le principal avantage de cette approche est de pouvoir décrire la tâche à réaliser en terme de positionnement dans l'espace cartésien.

Même si d'un point de vue géométrique, le passage de la position courante à la position désirée se résume à une matrice de transformation homogène, la commande en vitesse du robot nécessite inévitablement la prise en compte de la dynamique de ce dernier, de la cinématique scène-robot et de différentes erreurs dues à:

– l'extraction des primitives dans l'image,

– la modélisation de la caméra,

– la modélisation du robot.

De plus, il faut souligner la nécessité d'hypothèses fortes pour estimer la situation de la caméra telle que la connaissance des dimensions de l'objet ou la distance caméra/scène. Par conséquent, l'approche *Position-based Look-and-Move* ne semble pas appropriée pour réaliser des actions réflexes autour d'objets inconnus ou non modélisés (chapitre 3).

Notons que dans le cas d'objets connus (chapitre 2), les différents types d'erreur énumérés ci-dessus nous ont là encore incités à nous orienter sur l'approche *Image-based Look-and-Move*.

1.3.5.2 Image-based Look-and-Move

L'originalité de cette technique réside dans le fait que la tâche à réaliser est spécifier directement dans l'espace image (Fig.1.11). On évite ainsi les problèmes d'interprétation (Fig.1.9) dûs à la modélisation et aux erreurs de calibrage.

Dans ce type de struture, on utilise une relation entre les mouvements de l'effecteur et la variation des motifs dans l'image. Cette relation différentielle caractérisant l'interaction entre la position du manipulateur et la scène est connue sous le nom de *Jacobien d'image*. Cette notion a été introduite pour la première fois par Weiss (Sanderson, Weiss and Neuman, 1987) sous la dénomination *feature sensitivity matrix* . On la retrouve aussi dans l'approche *fonction de tâche* sous le nom de *matrice*

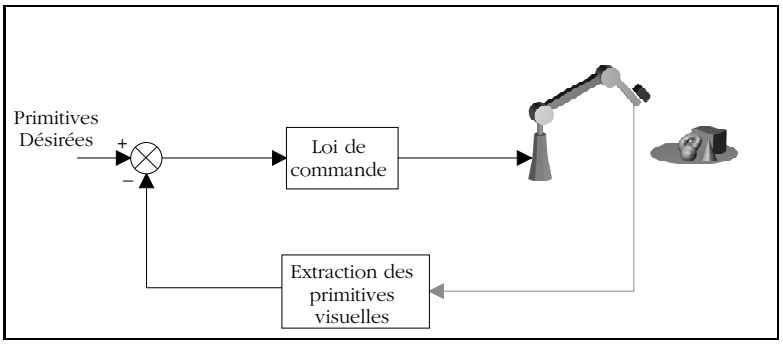


FIG. 1.11 – *Structure "Image-based Look-and-Move"*

d'interaction (Espiau, Chaumette and Rives, 1992) ou encore dans les travaux de Papanikolopoulos qui se place dans l'espace d'état et l'utilise en tant que *matrice* $\boldsymbol{B}$ (Papanikolopoulos, Khosla and Kanade, 1993). Ainsi le jacobien d'image noté J_i vérifie la relation:

$$\underline{\dot{s}} = J_i.\underline{\dot{r}}$$

où $\underline{\dot{s}}$ représente le vecteur des variations des primitives dans l'image et $\underline{\dot{r}}$ est le torseur cinématique appliqué à l'effecteur du robot.

Cette relation exprime les variations des primitives visuelles en fonction des mouvements de la caméra. On retrouve ici la notion d'actions réflexes où les stimuli du capteur ($\underline{\dot{s}}$) sont aussitôt transcrits en action ($\underline{\dot{r}}$).

Toutefois l'évaluation de cette matrice jacobienne n'est pas toujours triviale et dépend du type de primitives considérées. Une première technique repose sur un calcul analytique en fonction des primitives utilisées. Cette approche a été employée par de nombreux auteurs (Chaumette, 1990) (Feddema and Mitchell, 1989) (Papanikolopoulos, 1992), mais se heurte toutefois au problème d'évaluation de la profondeur entre la caméra et la cible. Chaumette (Chaumette, 1990) fixe la profondeur à la valeur finale désirée, tandis que Papanikolopoulos (Papanikolopoulos, 1992) réalise une estimation basée sur une méthode adaptative.
Une autre approche consiste à éviter le calcul de cette matrice en évaluant les différentes composantes par des techniques numériques. Suh (Suh, 1993) effectue une estimation du jacobien d'image par un réseau de neurones tandis que M. Jägersand (Jägersand et al., 1997) et K. Hosoda (Hosoda and Asada, 1994) proposent une estimation numérique en ligne durant la tâche robotique.

Un des principaux avantages de cette approche réside dans la précision et la facilité de mise en œuvre des tâches réalisées. On évite ici le calcul de pose à partir de mesures faites dans l'image, ce qui permet d'être moins sensible à la calibration de la caméra dans les phases d'interprétation.

En contrepartie, le principal désavantage de l'asservissement dans l'image est dû aux problèmes de singularités du jacobien d'image qui interviennent principalement à deux niveaux:

- Singularités de représentation: Ce sont des singularités intervenant lors d'une dégénérescence du motif visuel. Par exemple, une droite se projette en un point, ...
- Singularités isolées: Ces problèmes surviennent dans le calcul du jacobien d'image inverse. Localement, le jacobien d'image ne devient plus inversible et le système peut alors être amené à diverger.

ainsi qu'aux problèmes de minimas locaux récemment étudiés par F. Chaumette (Chaumette, 1998). Ce type de problèmes se rencontre lorsque l'erreur calculée dans l'image $(s - s^*)$ appartient au noyau de la matrice d'interaction inverse, soit

$$(s - s^*) \in Ker\widehat{J_i^+}$$

Dans cette configuration, le torseur cinématique appliqué au robot sera nul alors que le motif mesuré dans l'image n'aura pas convergé vers la situation désirée.

1.3.5.3 Extraction des primitives

Une tâche d'asservissement visuel nécessite l'extraction de primitives visuelles réduisant ainsi l'image à un vecteur de paramètres. Cette étape délicate et souvent coûteuse en temps de calcul oblige à contourner le problème en utilisant des conditions expérimentales "parfaites": Objet blanc sur fond noir, cibles irréalistes, ... (Feddema and Mitchell, 1989) (Chaumette, 1990). D'autres auteurs se placent dans les conditions particulières d'une cible en déplacement et segmente la scène au sens du mouvement (Allen, Timcenko, Yoshimi and Micheman, 1993).

Dans le cas de scènes structurées, la recherche de primitives géométriques classiques telles que les points (Espiau et al., 1992) (Castano and Hutchinson, 1994), la distance entre deux points (Feddema and Mitchell, 1989), la dimension des bords d'un objet (Sanderson et al., 1987), les paramètres d'une droite ou d'une ellipse (Espiau et al., 1992), (Motyl, 1993), (Khadraoui, Motyl, Martinet, Gallice and Chaumette, 1996) ... est devenue classique en asservissement visuel.

L'utilisation du mouvement dans l'image a aussi fait l'objet de plusieurs travaux. Toutefois, conscients des temps d'extraction prohibitifs, la plupart des auteurs évitent de calculer le champ dense du mouvement (flot optique) et préfèrent utiliser des modélisations. V. Sundareswaran (Sundareswaran, Bouthemy and Chaumette, 1996) et A. Crétual (Cretual and Chaumette, 1997) se sont intéressés à

un modèle basé sur les paramètres quadratiques du mouvement. Pour notre part (Martinet, Berry and Gallice, 1996a), nous nous sommes penchés sur le couplage d'informations géométriques et dynamiques dans le cas de primitives "points d'intérêt". Nos expérimentations basées sur la variation de position entre deux instants nous ont permis de mettre en évidence un effet stabilisant des primitives dynamiques. E. Grosso dans (Grosso, Andrea and Sandini, 1996) propose d'observer la scène de l'extérieur (Fig.1.1.a.). Un champ dense du mouvement est calculé autour de l'effecteur et permet d'aligner le manipulateur avec la cible.

Toutefois, le traitement de l'image entière n'est souvent pas indispensable et nécessite une architecture matérielle relativement performante. En effet, il est plus judicieux de ne traiter que les parties autour des primitives considérées. Pour cela, la notion de fenêtres d'intérêt permettant de réaliser un traitement donné dans une zone définie (Martinet, Rives, Fickinger and Borrelly, 1991) peut être introduite. Dans le cadre de l'asservissement visuel, il est important de pouvoir suivre les primitives tout au long de la tâche robotique et par conséquent de pouvoir rafraîchir la position et les traitements des fenêtres d'intérêt. Ces problèmes ont été abordés par plusieurs auteurs (Rizzi and Koditschek, 1996), (Dickmanns and Graefe, 1988). Hager dans (Hager and Toyama, 1998) propose un module logiciel dédié au suivi de primitives (XVision). Le principe de son programme repose sur un calcul de corrélation par une méthode de "Block Matching" autour des primitives à suivre.

Cependant, ce genre d'approche ne permet pas de résoudre le problème de la sélection automatique de primitives qui consiste à choisir les indices visuels qui permettront de réaliser au mieux la tâche désirée. Dans la plupart des travaux, la sélection est basée sur des mesures de confiance qui prennent en compte différents paramètres. Feddema dans (Feddema, Lee and Mitchell, 1991) effectue un suivi de joint de carburateur en mouvement et utilise sept critères pour sélectionner des primitives circulaires. La sélection est faite à deux niveaux : Vision (robustesse, unicité, coût calculatoire, représentation) et Contrôle (observabilité, contrôlabilité et sensibilité).
Toutefois, de nombreux autres travaux proposent l'utilisation de critères variés. Moravec (Moravec, 1980) et Thorpe (Thorpe, 1984) choisissent des zones de l'image ayant de forts contrastes de luminance tandis que Kitchen (Kitchen and Rosenfeld, 1980) et Dreschler (Dreschler and Nagel, 1981) se focalisent sur des primitives de type coin. Papanikolopoulos (Papanikolopoulos, 1995) considère trois types de primitives : coin, bord et surface homogène pour effectuer un asservissement visuel sur des objets plans. La sélection est là encore basée sur différentes mesures de confiance. Ce principe est aussi employé par Janabi (Janabi-Sharifi and Wilson, 1997) qui sélectionne les primitives à partir d'un modèle C.A.O de l'objet afin d'en effectuer le suivi par asservissement visuel.

1.3.5.4 Navigation et Asservissement visuel

Une autre application déjà évoquée de l'asservissement visuel concerne la reconstruction de primitives géométriques. M. Xie (Xie, 1989) et S. Boukir (Boukir, 1993) se sont intéressés au problème dans le cas de formes connues et facilement paramétrables (cylindre, segment, ...). Leur approche consiste à rechercher des mouvements permettant une reconstruction optimale. Pour cela, ils utilisent la commande référencée vision afin de se déplacer autour des formes à reconstruire. La suite de ces travaux effectuée par E. Marchand (Marchand, 1996) a consisté à définir des stratégies de perception capables d'aboutir de façon autonome à une description complète de la scène.

Dans ce type de travaux, l'asservissement visuel est utilisé comme un outil bas niveau permettant le déplacement de la caméra à travers la scène. Une des méthodes généralement employées en Commande Référencée Vision consiste à utiliser une tâche hybride. Cette tâche est construite à partir d'une tâche primaire qui assure un maintien caméra/cible et d'une tâche secondaire qui réalise le déplacement du capteur sans perturber la tâche primaire. Ce type d'approche a été mis en œuvre par plusieurs auteurs dont G. Motyl dans le cadre de déplacement autour d'une scène sphérique (Motyl, 1993) et par P. Rives (Rives and Borrelly, 1997) dans le cadre de la robotique sous-marine. Toutefois, les principaux inconvénients de cette méthode sont le non contrôle de la trajectoire[6] et la nécessité d'avoir un motif visuel compatible avec le mouvement à réaliser.

Pour cette raison, nous avons mis au point une méthode (Berry, Martinet and Gallice, 1997) permettant de réaliser le suivi d'une trajectoire autour d'un objet connu. L'unique condition pour réaliser ce suivi dans de bonnes conditions est la réalisation d'une liaison rigide entre le capteur et la cible. Par cette technique, il est alors relativement facile de suivre des trajectoires planifiées par un module superviseur.

Toutefois, cette approche relativement performante ne s'applique qu'à des primitives facilement paramètrables, et dans le cas d'objets complexes ou inconnus peu de travaux proposent une alternative. En effet, à notre connaissance, seules deux études permettent d'envisager un asservissement et *a fortiori* une tâche de navigation autour d'objet inconnu. C. Collewet dans (Collewet, Chaumette, Wallian and Marchal, 1998) réalise une tâche de positionnement sur un objet plan de forme quelconque en utilisant des critères de minimisation. Malheureusement cette approche ne se limite qu'à des positionnements. D'autres recherches ont été entreprises par A. Crétual (Cretual, 1998), dans le cadre de sa thèse où il réalise une trajectoire d'observation parallèlement à une face. Cette approche basée sur un asservissement

6. Dans ce type d'approche, on contrôle le chemin (lieu géographique), mais non la trajectoire faisant intervenir le temps.

$\frac{d}{dt}2D$ présente toutefois l'inconvénient de ne pouvoir être utilisée que sur des surfaces à géométrie relativement plane.

1.3.5.5 Apparence d'objets (Analyse en Composantes Principales)

Dans ce type de techniques on ne cherche plus à extraire une série de primitives données mais à trouver un sous-espace de paramétrage décrivant au mieux la vue de la scène. Le but de ce sous-espace est de pouvoir établir une relation entre la pose de l'objet (donc la situation caméra/scène) et un ensemble de paramètres (Fig. 1.12). Dans la plupart des travaux effectués, l'espace choisi est l'espace propre. Il est

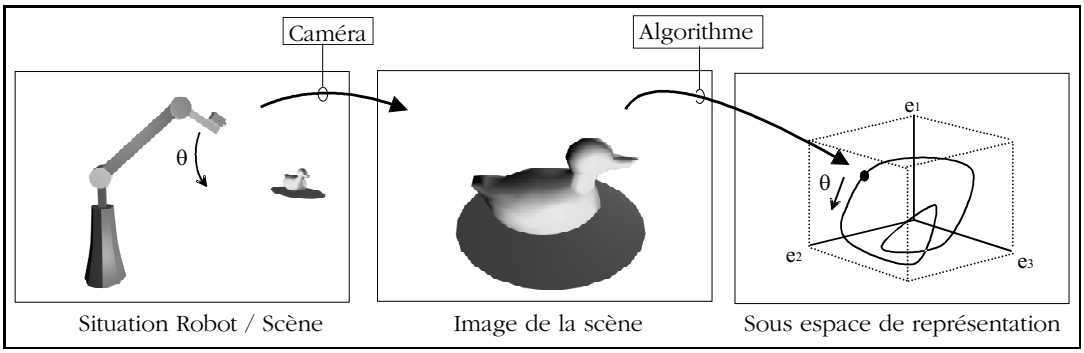


FIG. 1.12 – *Relations entre la pose et un paramètrage*

constitué par une base orthonormale dont les vecteurs unitaires sont les principaux vecteurs propres de l'image. Pour effectuer leurs calculs, on considère l'image de la scène observée comme une matrice à laquelle on applique une décomposition en valeur propre. Le nombre de vecteurs propres choisis conditionne la précision que l'on désire sur la connaissance de l'image. Généralement quelques vecteurs seulement suffisent pour obtenir des caractéristiques significatives. Le nombre de vecteurs choisis constitue alors la dimension de l'espace propre. Dans le cas de la navigation passive[7], une condition fondamentale de validité est la bijectivité de la transformation. En d'autres termes, un paramétrage ne doit correspondre qu'à une seule vue de la scène et réciproquement. Cette condition paraît être vérifiée dans la plupart des travaux avec l'utilisation des trois principaux vecteurs propres (vecteurs correspondants aux plus grandes valeurs propres). Un autre intérêt de l'espace propre est sa "continuité": un déplacement autour d'un objet produit une suite d'images qui seront codées de

7. La notion de passivité est ici entendue dans le sens, où le déplacement du capteur n'est pas contrôlé et sert uniquement à améliorer la perception de l'environnement.

façon continue dans l'espace propre (Sur la Fig. 1.12, le déplacement du robot d'un angle θ revient à suivre continuement la courbe codée).

L'utilisation des sous-espaces de paramétrage, telle que la décomposition en vecteurs propres, trouve son origine dans la reconnaissance d'objet et la compression d'image. Sirovich en 1987 et Turk et Pentland en 1991 proposent une analyse en composantes principales pour reconnaître les visages. Précédemment dans (Murase, Kimura, Yoshimura and Miyake, 1981) Murase *et al.* utilisent ce type d'approche pour la reconnaissance de caractères écrits. Cependant d'autres travaux ayant trait à la classification et à la reconnaissance de formes ont aussi été effectués (Oja, 1983).

En 1994, Nayar (Nayar, Murase and Nene, 1994) propose un système permettant de se positionner et de suivre un objet à partir de son apparence. Cependant leur approche, comme la plupart, nécessite une phase d'apprentissage afin d'élaborer les courbes paramétrées dans l'espace propre. Cet apprentissage se fait en échantillonnant toutes les vues possibles de l'objet (Rotation sur une sphère de vues en "θ/ϕ"). L'estimation de la position caméra/objet se fait en comparant la mesure codée par rapport à la courbe de référence. Dans (Nayar, Nene and Murase, 1995), les résultats présentent l'inspection et le suivi de divers objets tels qu'une carte électronique, un petit jouet en plastique et une pièce manufacturée. Edwards dans (Edwards, 1996) a aussi abordé le problème en se focalisant plus sur le problème d'estimation de la pose d'objets complexes. Dans son approche il utilise un carter d'alternateur en aluminium qu'il échantillonne en 266 vues comme précédemment. Un des intérêts de son approche est l'utilisation d'un objet métallique spéculaire donc très sensible aux conditions d'illumination.

Globalement ce type d'approche peut sembler relativement intéressant, cependant on peut noter deux problèmes majeurs:

- Le paramètrage utilisé ne permet pas de relier analytiquement la position de la caméra à une image de la scène. Par conséquent, il paraît difficile de se passer de l'étape d'apprentissage.

- Le calcul de l'espace propre (apprentissage) nécessite un temps de calcul proportionnel aux nombres de vecteurs calculés et au nombre de vues désirées. Dans (Nayar et al., 1995), les expérimentations faites sur une station de travail Sun IPX mettent en évidence des temps de calcul allant d'une dizaine de minutes à plusieurs heures. En ce qui concerne le calcul de la position à partir de l'image (calcul des vecteurs propres et recherche dans la base propre), différents algorithmes ont été mis en place amenant les temps de reconnaissance autour de quelques centaines de millisecondes (Nayar, Nene and Murase, 1996).

Une récente approche basée sur le codage de l'image dans l'espace propre a été introduite par I.Takahashi dans (Takahashi and Deguchi, 1998). La méthode utilisée semble particulièrement originale puisqu'elle permet de réaliser un positionnement et un suivi d'objets autour d'objet inconnus et complexes en utilisant la notion de matrice d'interaction. Toutefois, là encore, une phase d'apprentissage est nécessaire pour créer une base d' "images propres de référence".

1.4 Conclusion

Dans ce chapitre nous nous sommes intéressés au problème de la navigation à l'aide d'un système de vision monoculaire. Après avoir présenté quelques travaux afin de décrire les différentes tâches robotiques envisageables à l'aide d'un manipulateur (Positionnement, Suivi, Navigation), nous nous sommes focalisés sur la tâche de navigation.

Nous avons rappelé les différents types d'architectures robotiques mis en œuvre dans le cadre d'une tâche de navigation, puis nous avons situé le problème du **contournement d'objet** dans ce contexte. Cette problématique semble pouvoir être résolue avec un système hybride alliant une struture hiérarchique à une structure réactive. Dans ce manuscrit, nous nous intéressons uniquement à l'aspect réactif du contournement.

Pour cela, nous avons choisi d'utiliser une technique d'asservissement visuel. La **Commande Référencée Vision** pour laquelle nous avons opté présente à notre sens le meilleur compromis entre possibilité de contrôle et robustesse. Son principe consiste à introduire directement les informations extraites de l'image dans la boucle de commande.

Dans le cas du contournement d'objet inconnu et non modélisé, un des problèmes majeurs est la détermination des informations visuelles pouvant être utilisées dans la boucle d'asservissement visuel. La dernière partie de ce chapitre aborde le problème de perception visuelle dans le cadre de l'asservissement visuel.

Le chapitre suivant est consacré à l'utilisation de la commande référencée vision pour générer et suivre des trajectoires autour d'une cible connue. Dans un troisième et dernier chapitre, nous abordons le problème du déplacement autour d'un objet volumique complexe et inconnu en présentant les primitives retenues et les lois de commande associées.

Chapitre 2

Génération et suivi de trajectoire par asservissement visuel

2.1 Introduction

DANS le cadre de déplacements en environnement connu, les mouvements d'un robot peuvent se résumer en une série de tâches canoniques du type "**aller à But**". Toutefois, la connaissance de son environnement ne confère pas pour autant au robot la connaissance du chemin à emprunter. Comme nous l'avons vu dans le chapitre précédent, déplacer l'outil du robot d'une configuration A à une configuration B, requiert une hiérarchisation d'actions (Fig. 2.1) permettant à partir d'informations sur l'environnement,

- de planifier un chemin réalisant l'itinéraire désiré. Cette étape consiste, dans la plupart des cas, à prendre des décisions en fonction de la tâche à réaliser. De nombreux travaux ont été proposés dans ce domaine et différentes méthodologies ont été développées (Méthode des potentiels, Décomposition en Cellules, Graphes de visibilité, ...). Le lecteur intéressé trouvera une bonne synthèse des travaux afférant au *"Path planning"* dans le livre de J.C Latombe (Latombe, 1991)

- de générer une trajectoire permettant de déplacer le robot le long de ce chemin. Les données de la planification sont exprimées sous la forme d'une trajectoire définie par une suite de repères (appelés points de passage) correspondant aux situations successives de l'effecteur.

- de suivre cette trajectoire à partir de capteurs et d'actionneurs. Cette dernière étape se résume souvent à une boucle d'asservissement permettant de réguler la trajectoire du robot par rapport à la trajectoire idéale.

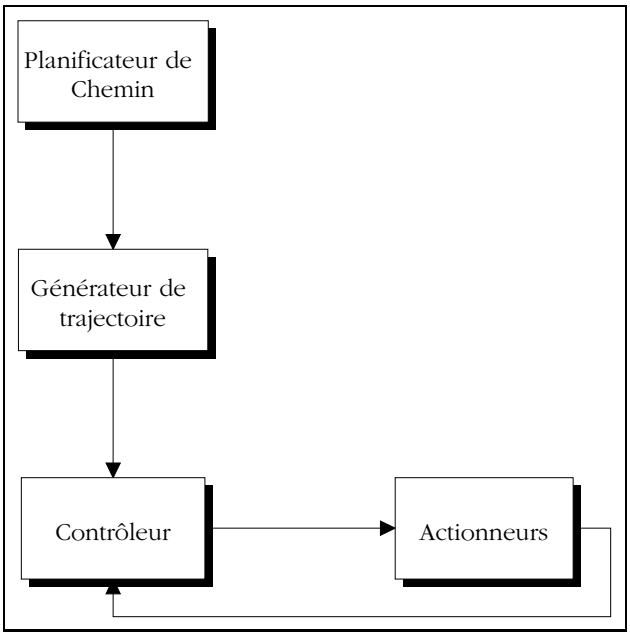


FIG. 2.1 – *Planification, Génération de trajectoire et Contrôle de la trajectoire.*

Si la première étape est plus du domaine de l'intelligence artificielle, les deux dernières étapes rentrent pleinement dans le domaine de la robotique et de l'automatique. Le problème de la génération de mouvement consiste à calculer la consigne $\underline{q}_d(t)$ (ou $\underline{X}_d(t)$) à appliquer à la boucle de commande pour assurer le passage du robot par des points prédéfinis (points de passage). Dans la littérature (Dombre and Khalil, 1988), on considère généralement deux types d'approche correspondant aux espaces de contrôle :

↪ La génération et le suivi de trajectoire peuvent se faire directement dans l'espace articulaire. Cela se traduit alors par une séquence de positions articulaires (voire de vitesses ou d'accélérations) constituant les consignes des asservissements (Fig.2.2).

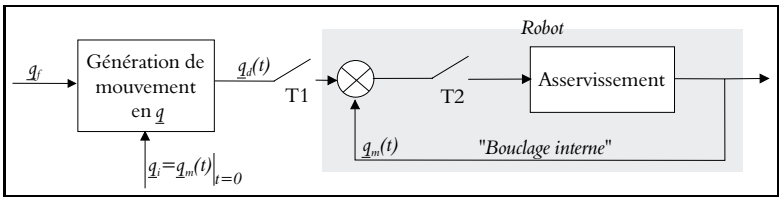


FIG. 2.2 – *Génération de trajectoire dans l'espace articulaire.*

Les mouvements sont donc directement appliqués dans l'espace des actionneurs,

ce qui évite les problèmes de singularités dûs aux calculs de modèles géométriques inverses. En contrepartie, la géométrie de déplacement du robot n'est pas contrôlée dans l'espace cartésien, ce qui peut poser certaines difficultés en présence d'obstacles. Un autre inconvénient de cette méthode est la difficulté de description d'une tâche robotique donnée directement dans l'espace articulaire.

↪ Une seconde approche consiste à exprimer la trajectoire dans l'espace cartésien. Les coordonnées opérationnelles doivent alors être transformées en coordonnées articulaires par le *modèle géométrique inverse* du robot (MGI) (Fig.2.3).

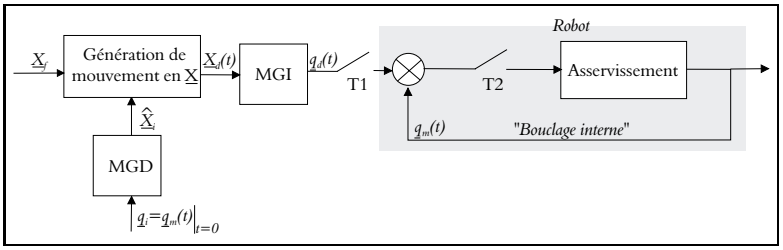


FIG. 2.3 – *Génération de trajectoire dans l'espace cartésien.*

Il est alors facile d'exprimer une tâche robotique en fonction de la géométrie de déplacement du robot. Toutefois dans certains cas de figure, le temps de calcul des modèles géométriques peut être prohibitif. L'utilisation de modéles entraîne alors des biais qui peuvent conduire à des singularités de représentation.

Notre préoccupation dans ce chapitre est d'utiliser un capteur extéroceptif, en l'occurrence une caméra, comme "mesure indirecte" de la situation du robot dans l'espace opérationnel. La régulation du suivi de trajectoire sera réalisée à partir d'informations visuelles. La boucle de commande utilisée se transforme alors en un asservissement visuel où les consignes sont directement spécifiées dans l'image (Fig. 2.4).

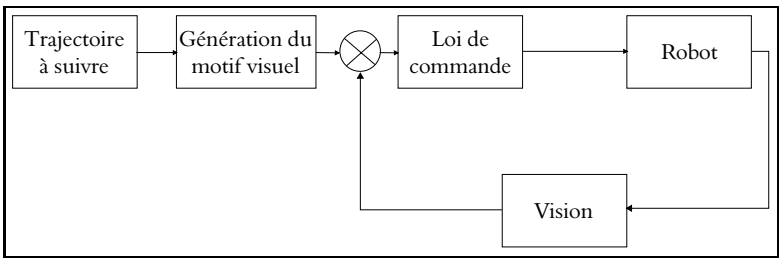


FIG. 2.4 – *Génération et suivi de trajectoire dans l'espace image.*

Dans cette approche basée sur la notion de fonction de tâche (Samson et al., 1991), nous intégrons directement les données issues de la perception dans le suivi de la trajectoire. Ceci nous a amené à appliquer le formalisme de la fonction de tâche au cas de la *trajectoire visuelle*. A partir de là, nous avons développé une loi de commande permettant d'effectuer le suivi des primitives visuelles tout en minimisant l'erreur de traînage.

Un autre aspect de notre travail a été de générer les motifs visuels de référence pour une trajectoire donnée. Nous avons alors développé un formalisme permettant de décrire l'évolution des primitives dans l'image en fonction des mouvements à réaliser.

Dans une dernière partie, nous présentons différents résultats obtenus sur notre plate-forme robotique et sur un simulateur développé en language C sur station de travail.

2.2 Choix d'une fonction de tâche

2.2.1 Présentation

Comme nous l'avons rappelé dans le précédent chapitre, la notion de fonction de tâche a été introduite par Claude Samson, Bernard Espiau et Patrick Le Borgne en 1991 dans leur ouvrage *Robot Control* (Samson et al., 1991). Le but d'une fonction de tâche est de traduire en termes mathématiques, une tâche robotique donnée. Cette mise en forme mathématique réalisée directement dans l'espace du capteur, permet alors d'élaborer une loi de commande et de la rendre exploitable pour la commande du robot. Samson insiste sur la "délicatesse" de cette phase de traduction en arguant que cette opération n'est pas formalisable de façon précise et que par conséquent, des formulations *a priori* assez semblables peuvent conduire à de grandes différences de fonctionnement.

Rappelons à présent le principe de la fonction de tâche en précisant ce que nous entendons par *traduction d'une tâche robotique en termes mathématiques*. Le principe de ce formalisme consiste à réaliser une tâche robotique par la régulation (à zéro) d'une fonction d'erreur $\underline{e}(\underline{q}, t)$ de vecteur configuration $\underline{q}$ et de variable temporelle t (t est considéré comme une variable indépendante).

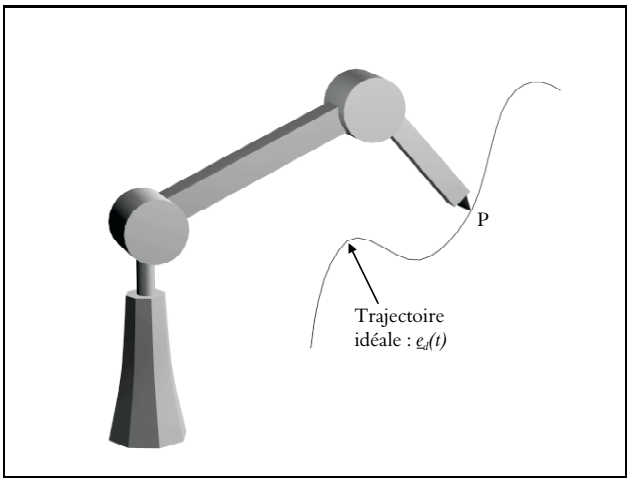


Fig. 2.5 – *Suivi d'une trajectoire*

Afin de préciser notre idée, considérons un manipulateur et supposons que l'on veuille déplacer son effecteur P suivant une trajectoire donnée $\underline{e}_d(t)$ (Fig. 2.5)

Si l'on décrit par $\underline{r}$ la situation de l'effecteur P, alors une fonction de tâche

correspondant à notre intention peut s'écrire:

$$\underline{e}(\underline{q},t) = \underline{q} - \underline{e}_d(t) \tag{2.1}$$

La régulation à zéro de $\underline{e}(\underline{q},t)$ permet bien de déplacer P suivant la trajectoire désirée.

Cependant, d'autres types de tâches robotiques décrites dans différents espaces peuvent être spécifiés suivant ce formalisme:

- dans l'espace articulaire, une fonction de tâche permettant une régulation en position peut s'écrire:
$$\underline{e}(\underline{q},t) = \underline{q} - \underline{q}_d(t)$$
où $\underline{q}$ représente la position articulaire et $\underline{q}_d(t)$ décrit la trajectoire idéale à suivre dans l'espace articulaire.

- dans l'espace cartésien, en suivant le même principe et pour un contrôle en situation, on obtient:
$$\underline{e}(\underline{q},t) = \underline{r}(\underline{q}) - \underline{r}_d(t)$$
où $\underline{r}(\underline{q})$ est un type de paramétrage donné de l'attitude de l'effecteur et $\underline{r}_d(t)$ représente la trajectoire idéale exprimée dans l'espace cartésien.

Toutefois, le problème général de régulation de la fonction de tâche $\underline{e}$ est bien posé si celle-ci possède certaines propriétés. Cet ensemble de propriétés conditionne l'admissibilité de la tâche. Dans le paragraphe suivant, nous nous proposons d'aborder de façon plus formel ce problème en rappelant les conditions énoncées dans (Samson et al., 1991).

2.2.2 Admissibilité d'une tâche

2.2.2.1 Rappels

Une fonction de tâche $\underline{e}(\underline{q},t)$ est une application vectorielle de classe C^2 d'un ouvert Ω de $\mathbb{R}^n \times \mathbb{R}$ dans $\mathbb{R}^n$. Par cette définition, on considère une fonction vectorielle deux fois dérivable ayant la dimension de $\underline{q}$ et dont la régulation s'effectue sur un horizon tel que $t \in [0,T]$.

Supposons qu'à $t = 0$, le manipulateur occupe une position $\underline{q}_0$ telle que $\underline{e}(\underline{q}_0,0) = 0$. Une trajectoire idéale du robot est une trajectoire $\underline{e}(\underline{q}_d(t),t)$ telle que:

$$\underline{e}(\underline{q}_d(t),t) = 0 \text{ avec } \underline{q}_d(0) = \underline{q}_0 \qquad \forall t \in [0,T]$$

Cette trajectoire est idéale dans le sens où elle réalise parfaitement la tâche durant l'intervalle de temps $[0, T]$. Cependant $\underline{e}(\underline{q}_d(t), t)$ doit satisfaire deux critères:

$\hookrightarrow$ Pour des raisons physiques imposées par les lois de la dynamique, une tâche n'est possible que si la trajectoire idéale $\underline{q}_d(t)$ est **deux fois dérivable** sur $[0, T]$. Il existe alors une commande idéale $\Gamma_d(t)$ [1] qui permet de suivre exactement la trajectoire idéale $\underline{q}_d(t)$.

$\hookrightarrow$ Cependant une dernière condition (et non des moindres), est l'unicité de cette trajectoire. Cette **propriété d'unicité** est nécessaire en pratique pour assurer la répétabilité du mouvement.

Ces deux propriétés conditionnent ce que l'on appelle l'admissibilité de la fonction de tâche.

Une définition plus formelle peut être donnée par: $\underline{e}(\underline{q}, t)$ est une fonction ρ-admissible [2], sur un ensemble $C_{\rho,t} \subset \Omega$ pendant l'intervalle $[0, T]$, si et seulement si la fonction $\underline{F}(\underline{q}, t) = (\underline{e}(\underline{q}, t), t)$ est un difféomorphisme [3] de classe C^2 de $C_{\rho,t}$ sur la boule fermée $\overline{B}_\rho \times [0, T]$ centrée à l'origine et de rayon $\rho > 0$.

A travers cette définition, on retrouve les différentes propriétés citées ci-dessus, à savoir:

- **l'unicité de la trajectoire**: la bijectivité de $\underline{F}(\underline{q}, t)$ assure qu'une valeur de $\underline{e}(\underline{q}, t)$ ne correspond qu'à une et une seule valeur de $\underline{q}$ pour t donné et réciproquement.
- **la classe C^2 de cette bijection**: il existe une commande Γ permettant de réguler notre tâche.

On montre que dans la pratique, ces propriétés imposent:

- la régularité du jacobien de tâche, à savoir que $\det(\frac{\partial \underline{e}}{\partial \underline{q}}(\underline{q}, t)) \neq 0$
- que le jacobien inverse soit borné: $||\frac{\partial \underline{e}}{\partial \underline{q}}(\underline{q}, t)^{-1}|| < m_{\rho,T} < +\infty$, uniformément sur [0,T]

1. Le comportement dynamique d'un robot rigide est décrit par l'équation de la dynamique: $\underline{\Gamma} = \underline{M}(\underline{q})\ddot{\underline{q}} + \underline{N}(\underline{q}, \dot{\underline{q}}, t)$ où $\underline{\Gamma}$ est le vecteur des forces extérieures appliquées, $\underline{M}$ est la matrice d'énergie cinétique et $\underline{N}$ rassemble les contributions des forces de gravité, de Coriolis, centrifuge et de frottement.

2. S'il on considère un élément $O \in C_{\rho,t}$, alors la définition précédente signifie que la trajectoire à réaliser n'est pas isolée en O et que par conséquent il existe un ouvert de trajectoires ρ-admissibles autour.

3. Par difféomorphisme de classe C^2, on entend une bijection de classe C^2, dont la réciproque est aussi de classe C^2.

- que les variations de $\underline{e}(\underline{q},t)$ dans le temps soient bornées : $||\frac{\partial \underline{e}}{\partial t}(\underline{q},t)|| < m'_{\rho,T} < +\infty$, uniformément sur [0,T]

2.2.2.2 Notions de ρ-admissibilité

Dans la réalité, il est impossible de produire la commande idéale $\underline{\Gamma_d}$ correspondant à la trajectoire idéale $\underline{q}_d(t)$ pour différentes raisons assez simples à comprendre. En effet, la modélisation du robot se réduit toujours à une approximation en raison de la méconnaissance des grandeurs intervenant dans l'équation de la dynamique du robot. De plus, les différentes mesures sont sujettes à divers types de perturbations impliquant des incertitudes plus ou moins importantes. De nombreuses autres raisons peuvent être invoquées, mais il paraît évident que dans la pratique nous ne pouvons pas annuler strictement la fonction $\underline{\epsilon}(\underline{q},t)$, mais seulement se contenter de la rendre la plus faible possible. Cependant, tolérer une erreur $\underline{\epsilon}(t)$ ne peut être valable que si la trajectoire suivie par le robot est proche de la trajectoire idéale.

Cette prise en compte d'une erreur $\underline{\epsilon}(t)$ le long de la trajectoire a amené à définir la notion de ρ-admissibilité. Cette dernière permet de borner l'intervalle dans lequel la fonction $\underline{\epsilon}(\underline{q},t)$ assure la convergence de la trajectoire suivie vers la trajectoire désirée. On dit que ρ définit le degré d'admissibilité de la tâche sur [0,T].

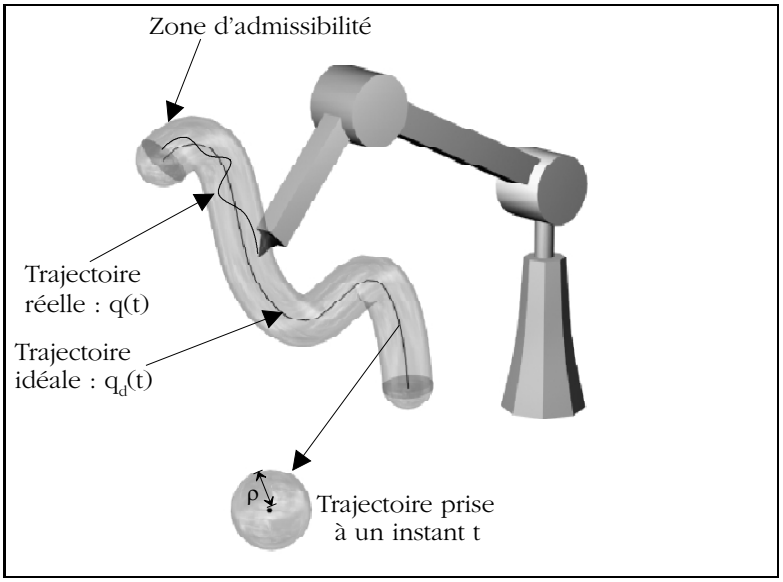


FIG. 2.6 – *Tâche ρ-admissible*

Tant que $\underline{e}(\underline{q},t) = \underline{q}(t) - \underline{q}_d(t)$ reste à l'intérieur de la boule centrée sur l'origine et de rayon ρ, la convergence à tout moment de $\underline{e}(\underline{q},t)$ vers 0 est assurée. Par contre,

si durant l'exécution de la tâche $||\underline{e}(\underline{q},t)|| > \rho$ alors, il n'est pas évident de ramener $\underline{e}(\underline{q},t)$ à 0. Le degré d'admissibilité détermine en quelque sorte le degré de robustesse de la tâche vis à vis du problème posé.

Remarque: Comme nous le verrons plus loin dans le paragraphe 2.3.1, dans la plupart des cas, les commandes mises en œuvre nécessitent la connaissance du jacobien de tâche inverse soit $(\frac{\partial \underline{e}}{\partial \underline{q}})^{-1}$. Cette quantité suppose une connaissance parfaite des termes constituant la commande, ce qui n'est dans la réalité jamais le cas. Aussi, utilise-t-on un modèle noté $(\widehat{\frac{\partial \underline{e}}{\partial \underline{q}}})^{-1}$ et principalement basé sur des approximations ou des estimations. Toutefois, comme le souligne Chaumette dans (Chaumette, 1990), le comportement de la fonction de tâche dans le cas d'une connaissance parfaite du modèle dynamique du robot[4] s'écrit alors :

$$\ddot{\underline{e}} = -\lambda \frac{\partial \underline{e}}{\partial \underline{q}} \left(\widehat{\frac{\partial \underline{e}}{\partial \underline{q}}} \right)^{-1} G \left(\mu D \underline{e} + \left(\widehat{\frac{\partial \underline{e}}{\partial \underline{q}}} \right) \dot{\underline{q}} + \left(\widehat{\frac{\partial \underline{e}}{\partial \underline{t}}} \right) \right)$$

Dans cette équation apparaît clairement la condition de positivité nécessaire à la convergence de $(\underline{e}(t))$ soit:

$$\frac{\partial \underline{e}}{\partial \underline{q}} \left(\widehat{\frac{\partial \underline{e}}{\partial \underline{q}}} \right)^{-1} > 0$$

Cette condition sera retrouvée plus loin dans le cas du capteur visuel (Eq. 2.9).

2.2.3 Tâches hybrides

Dans le cas où le nombre de degré de liberté à commander n est supérieur à la dimension de la fonction vectorielle $\underline{e}(\underline{q},t)$, la trajectoire idéale $\underline{q}_d(t)$ n'est plus définie de façon unique et la tâche considérée n'est plus admissible. On dit alors que la tâche est **redondante**.

Une tâche redondante est une tâche mal conditionnée, qu'il faut compléter de manière à la rendre admissible. Une façon de modifier la tâche consiste à construire une fonction de tâche $\underline{e}(\underline{q},t)$ avec les m premières composantes indépendantes ($m < n$) de la fonction de tâche. Les $(n-m)$ dernières composantes sont choisies de sorte que l'ensemble des n composantes de la fonction $\underline{e}(\underline{q},t)$ soient indépendantes le long de la trajectoire idéale $\underline{q}_d(t)$). La tâche se décompose alors en deux sous

4. On considére une commande linéarisante dans l'espace de la tâche de la forme $\Gamma = M \left(\frac{\partial \underline{e}}{\partial \underline{q}} \right)^{-1} . \underline{u'} + \underline{N} - M \left(\frac{\partial \underline{e}}{\partial \underline{q}} \right)^{-1} . \underline{f}$, où $\underline{u'}$ représente un retour proportionnel dérivé de forme général: $\underline{u'} = -\lambda G \left(\mu D \underline{e} + \dot{\underline{e}} \right)$.

tâches réalisant chacune un objectif indépendant. La première tâche, appelée tâche principale $\underline{e}_1(\underline{q},t)$, assure à travers sa régulation le suivi de la trajectoire idéale $\underline{q}_d(t)$. La seconde tâche, appelée tâche secondaire $\underline{e}_2(\underline{q},t)$ est souvent utilisée pour réaliser un objectif indépendant en ne perturbant pas la réalisation de la tâche principale. La plupart du temps, cette tâche secondaire est spécifiée comme la minimisation d'un coût h_s sous la contrainte $\underline{e}_1(\underline{q},t)$=0.

Cette minimisation de h_s nécessite alors la détermination du sous-espace des mouvements laissés libres par cette contrainte. En d'autres termes, cela revient à connaître le noyau de $\frac{\partial \underline{e}_1}{\partial \underline{q}}$, noté usuellement $Ker(\frac{\partial \underline{e}_1}{\partial \underline{q}})$, le long de la trajectoire idéale $\underline{q}_d(t)$ donc à connaître n'importe quelle matrice W de taille $m \times n$ et de rang plein, telle que:

$$Ker\ (W) = Ker(\frac{\partial \underline{e}_1}{\partial \underline{q}}) \tag{2.2}$$

lorsque $\underline{q}(t)$ parcourt $\underline{q}_d(t)$.

Une fois que l'on a déterminé cette matrice W, on peut alors construire une fonction de tâche réalisant les deux objectifs indépendamment. Cette fonction de tâche réalise ce que l'on appelle une tâche hybride et se formule généralement sous la forme:

$$\underline{e} = W^+\underline{e}_1 + \alpha(\mathbb{I}_n - W^+W)\underline{g}_s^T \tag{2.3}$$

où:

- α est un scalaire permettant de pondérer l'importance du coût secondaire par rapport à la tâche principale,
- W^+ est la pseudo-inverse de W,
- $(\mathbb{I}_n - W^+W)$ est un opérateur de projection orthogonal sur le noyau de W, donc sur celui de $(\frac{\partial \underline{e}_1}{\partial \underline{q}})$,
- $\underline{g}_s^T$ représente le gradient du coût h_s à minimiser, tel que $\underline{g}_s^T = \frac{\partial h_s}{\partial \underline{q}}$.

L'expression précédente fait donc apparaître les deux termes correspondants respectivement à la tâche principale et à la tâche secondaire. Cependant si la tâche principale consiste généralement en une fonction à réguler dans l'espace de mesure,

la tâche secondaire quant à elle peut prendre plusieurs formes. Samson dans (Samson et al., 1991) distingue trois types de tâches secondaires :

1 *Les tâches basées sur des critères de trajectoire "interne"*, c'est-à-dire ne dépendant pas de l'environnement. Dans ce type de tâche, $h_s(\underline{q}, t)$ dépend uniquement des paramètres articulaires $\underline{q}$ et se trouve indépendant du temps. Parmi les objectifs secondaires les plus courants citons l'évitement de butées articulaires ou encore l'évitement de singularités.

2 *Les tâches basées sur des critères de trajectoire "externe"*. Dans ce cas, la fonction de coût $h_s(\underline{q}, t)$ dépend d'informations extérieures au système ou de l'environnement. Un des principaux objectifs de ce genre de tâche est l'évitement d'obstacles.

3 *Les tâches basées sur des critères d'énergie*. Dans cette dernière classe, les grandeurs prises en compte sont généralement les couples et les forces appliqués aux actionneurs. Les mesures effectuées se révèlent souvent délicates puisque la fonction de coût ne dépend plus simplement de $\underline{q}$ et t, mais aussi de $\dot{\underline{q}}$ et $\ddot{\underline{q}}$.

2.2.4 Le cas du capteur visuel

Contrairement aux capteurs de proximité qui fournissent une grandeur physique mesurable (force, position, ...), la caméra vidéo délivre un signal vidéo relativement riche permettant d'avoir des informations sur l'environnement du robot ainsi que sur sa situation dans cet environnement.

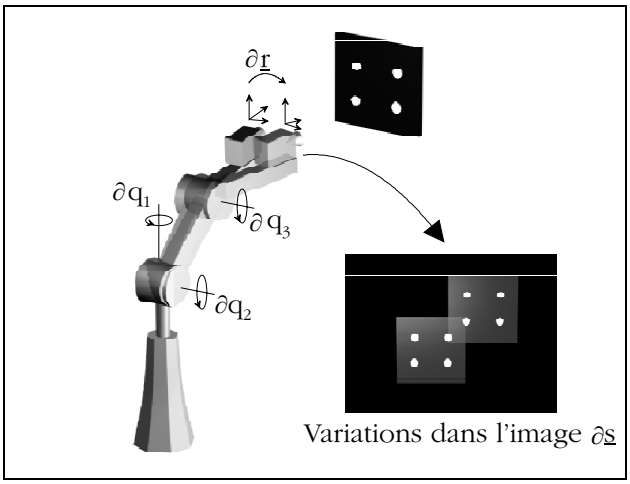


Fig. 2.7 – *Caméra fixée sur l'effecteur d'un manipulateur.*

Cette extraction d'indices visuels fournit alors un ensemble de primitives géométriques paramétrables nous permettant d'intégrer la caméra directement dans la boucle de commande. Comme nous l'avons évoqué dans le chapitre précédent, la configuration que nous avons étudiée est constituée d'un bras manipulateur au bout duquel est montée une caméra CCD. Ainsi tout mouvement du robot entraîne le déplacement du capteur et par voie de conséquence une variation de l'angle de vue sous lequel est observé l'environnement(Fig. 2.7).

Dans notre cas de figure, cette caractéristique est particulièrement importante puisque notre objectif est de déplacer l'effecteur à partir des données visuelles. Ainsi la génération de trajectoire ne sera pas effectuée dans l'espace articulaire ou dans l'espace cartésien, mais directement dans l'espace image. Une fonction de tâche permettant de réguler un suivi de trajectoire dans l'espace image pourra donc s'écrire:

$$\underline{e}(\underline{q}, t) = C(\underline{s}(\underline{q}, t) - \underline{s}^*(t)) \tag{2.4}$$

où

- $\underline{s}(\underline{q}, t)$, est le vecteur représentant l'ensemble des informations visuelles pour réaliser la tâche,
- $\underline{s}^*(t)$, correspond à la trajectoire idéale devant être suivie dans l'espace image,
- C est une matrice de découplage permettant de prendre en compte un nombre d'informations visuelles supérieur au nombre de degrés de liberté à commander.

Toutefois, la fonction de tâche proposée permettra d'atteindre l'objectif fixé si elle décrit de façon unique et entière la tâche à réaliser. Les signaux capteurs devront donc être choisis de manière à assurer le déplacement de l'effecteur suivant une trajectoire donnée. L'attitude de l'outil (effecteur) est classiquement décrite dans l'espace des configurations par trois paramètres de position et trois paramètres d'orientation. Ainsi, assurer le déplacement sans ambiguïté de l'effecteur requiert au moins six informations visuelles indépendantes. On construit alors entre le capteur et son environnement une **liaison dite rigide** dans le sens où n'importe quel mouvement du robot et donc de la caméra implique une variation du signal capteur.

Il est en effet facile de constater que dans le cas où la dimension du signal visuel est inférieure au nombre de degrés de liberté du robot, la tâche est redondante. Ceci est par exemple le cas d'un axe se projetant sous la forme d'une droite horizontale centrée dans l'image. Un mouvement latéral de translation ou une rotation autour de cet axe laisse invariant l'information visuelle et par conséquent ne peut permettre de commander ce dernier.

Toutefois, le choix judicieux des informations visuelles, s'il est nécessaire, ne constitue cependant pas une condition suffisante au bon déroulement de la tâche.

En effet, comme nous l'avons rappelé précédemment, une des conditions les plus fortes est la régularité du jacobien de tâche $\frac{\partial \underline{e}}{\partial \underline{q}}(\underline{q}, t)$. Pour notre suivi de trajectoire, ce jacobien s'écrit comme suit:

$$\frac{\partial \underline{e}}{\partial \underline{q}}(\underline{q}, t)) = C.\frac{\partial \underline{s}}{\partial \underline{r}}.\frac{\partial \underline{r}}{\partial \underline{q}} \tag{2.5}$$

Le terme $\frac{\partial \underline{r}}{\partial \underline{q}}$ correspond au jacobien du robot que l'on considérera comme parfaitement connu et inversible, et l'expression $\frac{\partial \underline{s}}{\partial \underline{r}} = L_{\underline{s}}^T(\underline{r})$ correspond naturellement à la matrice d'interaction (Jacobien d'image).

A partir de la relation 2.5, on en déduit que la régularité du jacobien de tâche est uniquement conditionnée par la régularité de $CL_{\underline{s}}^T$, donc par le choix d'une matrice C tel que $CL_{\underline{s}}^T$ soit de rang plein dans un voisinage autour de la trajectoire idéale.

2.3 Trajectoires dans l'espace image

Dans cette partie, nous présentons la loi de commande utilisée afin de suivre une trajectoire directement dans l'espace image. Ensuite, nous proposons une méthode pour générer le motif de référence à partir des données de la trajectoire à réaliser.

2.3.1 Loi de commande développée

La loi de commande que nous avons mise en œuvre, repose sur les bases de la Commande Référencée Vision. Notre motivation principale dans cette approche a été de spécifier la trajectoire que nous voulions réaliser directement dans l'espace image (Berry et al., 1997). Traditionnellement en Commande Référencée Vision, un suivi de trajectoire est spécifié dans une tâche secondaire sous la forme d'un coût à minimiser. Cette technique si elle est facile et naturelle à mettre en œuvre, présente cependant le défaut d'être en boucle ouverte (au niveau du contrôle de la trajectoire, la régulation peut être réalisée uniquement qu'à partir des informations odométriques) et de ne pas pouvoir s'appliquer dans tous les cas. En effet dans ce type d'approche, seule la tâche principale est régulée par les informations visuelles tandis que la tâche secondaire se comporte comme une simple contrainte sur les axes laissés libres. Ainsi la réalisation d'une tâche plus "élaborée", en utilisant une simple tâche secondaire devient vite conditionnée par un certain nombre d'impératifs : la tâche primaire doit laisser libre les degrés de liberté nécessaires, l'action à accomplir par la tâche secondaire doit être compatible avec la liaison "virtuelle".

Dans le cadre d'un suivi de trajectoire, la méthode que nous proposons consiste à maintenir une liaison de type rigide entre la caméra et la cible. Comme nous l'avons expliqué dans la partie précédente, cette liaison est obtenue en contraignant les 6 degrés de liberté de l'effecteur, donc en utilisant un signal capteur tel que $dim(\underline{s}) \geq 6$. Dans ce type de liaison et sous la condition que le jacobien d'image soit inversible autour de la trajectoire idéale, une configuration d'un motif dans l'image correspond à une position et une seule dans l'espace cartésien. Ainsi, une variation de la consigne visuelle sous la condition d'une bonne régulation permet de générer un mouvement de la caméra. Cette variation de consigne est directement prise en compte dans le choix de la fonction de tâche à réguler:

$$\underline{e}(\underline{r}, t) = C(\underline{s}(\underline{r}, t) - \underline{s}^*(t)) \tag{2.6}$$

où $\underline{s}^*(t)$ représente la trajectoire de référence à suivre par les signaux capteur.

Cette relation met en évidence le fait que les variations de $\underline{e}(\underline{r}, t)$ sont fonction

de $\underline{r}$ et du temps et peuvent par conséquent être développées sous la forme:

$$\frac{d\underline{e}}{dt}(\underline{r},t) = \frac{\partial \underline{e}}{\partial \underline{r}}.\frac{d\underline{r}}{dt} + \frac{\partial \underline{e}}{\partial t} \tag{2.7}$$

Lors de la régulation de la fonction de tâche, nous allons chercher à obtenir un comportement exponentiel découplé qui peut s'écrire:

$$\frac{d\underline{e}}{dt}(\underline{r},t) = -\lambda.\underline{e}(\underline{r},t) \tag{2.8}$$

où λ est une matrice de gain diagonale permettant de fixer la vitesse de décroissance de la fonction de tâche.

Sous la condition de régularité du jacobien de tâche, et en remarquant que

$$\frac{\partial \underline{e}}{\partial \underline{r}} = C.\frac{\partial \underline{s}}{\partial \underline{r}} = C.L_{\underline{s}}^T$$

les relations 2.7 et 2.8 permettent d'obtenir la commande en vitesse suivante:

$$T = (C.L_{\underline{s}}^T)^+ \left(-\lambda \underline{e}(\underline{r},t) - \frac{\partial \underline{e}}{\partial t}\right) \tag{2.9}$$

Nous pouvons remarquer qu'une condition de convergence de cette loi de commande est la positivité du jacobien de tâche soit $(C.L_{\underline{s}}^T) > 0$.

Intéressons-nous à présent au terme $\frac{\partial \underline{e}}{\partial t}$. Ce terme décrit les variations de la fonction de tâche ne dépendant pas du mouvement du robot. Son développement à l'aide de l'équation (2.6) donne:

$$\frac{\partial \underline{e}}{\partial t} = C\left(\frac{\partial \underline{s}(\underline{r},t)}{\partial t} - \frac{d\underline{s}^*(t)}{dt}\right) \tag{2.10}$$

$\hookrightarrow$ Le premier terme $\frac{\partial \underline{s}(\underline{r},t)}{\partial t}$ exprime la contribution d'un éventuel mouvement autonome de la cible. Dans la suite de notre exposé, nous considérerons uniquement le cas d'une cible fixe, par conséquent $\frac{\partial \underline{s}(\underline{r},t)}{\partial t}$=0.

$\hookrightarrow$ Le second terme $\frac{d\underline{s}^*(t)}{dt}$ représente la variation du signal de référence dans le temps. Dans notre cas, ce terme est non nul et intervient explicitement dans la loi de commande.

En reprenant la définition de la fonction de tâche (Eq.2.6) et les relations 2.9 et 2.10, la loi de commande peut alors se réécrire sous la forme:

$$T = -(C.L_{\underline{s}}^T)^+ \lambda C(\underline{s}(\underline{r},t) - \underline{s}^*(t)) + (C.L_{\underline{s}}^T)^+ C\frac{d\underline{s}^*(t)}{dt} \tag{2.11}$$

Le calcul littéral de cette loi de commande fait intervenir de manière explicite la matrice d'interaction $L_{\underline{s}}^T$ et requiert donc la connaissance à chaque instant de l'information tridimensionnelle des primitives observées. Dans la plupart des cas, cette information n'est pas disponible et il est alors nécessaire d'utiliser un modèle $\hat{L}$ de L. A travers de nombreux travaux, il a été montré expérimentalement que le choix de la matrice d'interaction calculée à l'équilibre permettait la convergence de la loi de commande, même dans des configurations initiales relativement éloignées de la position finale.
Pour notre part, nous avons choisi de modéliser l'expression $(C.L_{\underline{s}}^T)^+C$ par $L_{\underline{s}^*}^{T+}$. Notre loi de commande s'exprime alors:

$$T = \underbrace{-L_{\underline{s}^*}^{T+}\Lambda(\underline{s}(\underline{r},t) - \underline{s}^*(t))}_{Régulation} + \underbrace{L_{\underline{s}^*}^{T+}\frac{d\underline{s}^*(t)}{dt}}_{Traînage} \tag{2.12}$$

où $\Lambda = C^+\lambda C$ représente une matrice de gain permettant le réglage de la commande.

Dans cette expression deux termes peuvent se distinguer. Le premier permet la régulation de $\underline{s}(\underline{r},t)$ vers $\underline{s}^*(t)$. On reconnaît ici le terme classique d'une simple loi de positionnement. Le second membre contribue pour sa part à la prise en compte du mouvement de la référence. Comme nous pouvons le voir sur le schéma 2.8, ce terme apparaît comme une branche prédictive de l'évolution du système et permet en quelque sorte de "compenser" l'erreur de traînage dûe à la variation de consigne par une anticipation de cette variation de consigne.

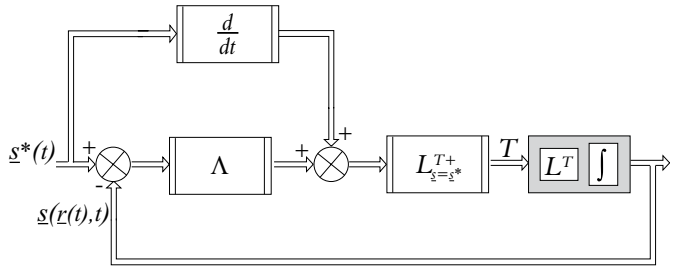


FIG. 2.8 – *Schéma bloc équivalent de la loi de commande (2.12)*

Remarques: Une analogie relativement triviale peut être faite entre la loi de commande présentée ci-dessus et les lois de commande développées dans le cadre du suivi de cible mobile. On notera simplement que la différence se situe au niveau du second terme (traînage), où dans notre cas, celui-ci se comporte comme un terme anticipatif tandis que dans le cas du suivi d'objets mobiles nous retrouvons un terme prédictif prenant en compte l'évolution de la cible dans le temps.

2.3.2 Analyse de la loi de commande

Nous allons à présent étudier cette loi de commande en nous plaçant dans différentes configurations idéalisant plus ou moins les conditions réelles. L'ensemble robot et caméra embarqué sur l'effecteur est modélisé par le bloc grisé contenant la matrice d'interaction suivie d'un intégrateur.

Analysons tout d'abord le système d'un point de vue continu à partir du schéma bloc 2.8. Nous obtenons l'équation d'évolution temporelle de notre système soit:

$$\frac{d\underline{s}}{dt} = -L_{\underline{s}}^{T} L_{\underline{s}^*}^{T+} \Lambda(\underline{s}(t) - \underline{s}^*(t)) + L_{\underline{s}}^{T} L_{\underline{s}^*}^{T+} \frac{d\underline{s}^*}{dt} \tag{2.13}$$

posons $\mathbb{K}(\underline{s}, \underline{s}^*) = L_{\underline{s}}^{T} L_{\underline{s}^*}^{T+}$ et il vient:

$$\frac{d\underline{s}}{dt} = -\mathbb{K}(\underline{s}, \underline{s}^*)\Lambda(\underline{s}(t) - \underline{s}^*(t)) + \mathbb{K}(\underline{s}, \underline{s}^*)\frac{d\underline{s}^*}{dt} \tag{2.14}$$

Cette équation est non linéaire et particulièrement difficile à intégrer compte tenu de la complexité de la matrice d'interaction. Nous pouvons cependant émettre quelques hypothèses réalistes permettant d'analyser le comportement du système.

$\hookrightarrow$ **Hypothèse 1**: La modélisation choisie est supposée parfaite, nous avons alors $\mathbb{K} = \mathbb{I}_m$, où $m = dim(\underline{s}(t))$. La solution générale de 2.14 est donnée par la relation:

$$\underline{s}(t) = \underline{s}^*(t) + e^{-\Lambda(t-t_0)}(\underline{s}(t_0) - \underline{s}^*(t_0)) \tag{2.15}$$

Il suffit alors de choisir la matrice de gain Λ, diagonale et à valeur positive pour assurer la stabilité et la convergence. La matrice Λ peut alors s'écrire:

$$\Lambda = \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_q \end{pmatrix} = \begin{pmatrix} \frac{1}{\tau_1} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{\tau_m} \end{pmatrix}$$

Dans la relation 2.15, nous voyons que l'équilibre ($\underline{s}(t) = \underline{s}^*(t)$) sera atteint d'autant plus rapidement que les gains λ_i (diagonale de Λ) sont importants (dans la limite de stabilité du système). Nous avons ainsi:

$$\underline{s}(t)|_{t \gg max(\lambda_i^{-1})} = \underline{s}^*(t) \tag{2.16}$$

Dans ce cas idéal, la loi de commande permet donc un suivi parfait de la trajectoire de référence décrite par $\underline{s}^*(t)$ après un temps $t \gg max(\lambda_i^{-1})$.

↪ **Hypothèse 2**: Intéressons-nous à présent, au cas plus réaliste selon lequel même proche de l'équilibre, nous avons une erreur de modélisation que nous considérons comme constante en première approximation soit $\mathbb{K} \neq \mathbb{I}_m$. Ce cas plus réaliste permet de tenir compte des erreurs inévitables de modélisation, calibrage, ...
$\mathbb{K}$ représente donc une matrice de gain traduisant l'erreur résiduelle entre le système réel décrit par $L^T_{\underline{s}(t)}$ et sa modélisation $L^{T+}_{\underline{s}^*(t)}$.

En faisant l'hypothèse que le système est linéaire, nous pouvons utiliser le formalisme de Laplace pour en étudier le comportement (Fig. 2.9).

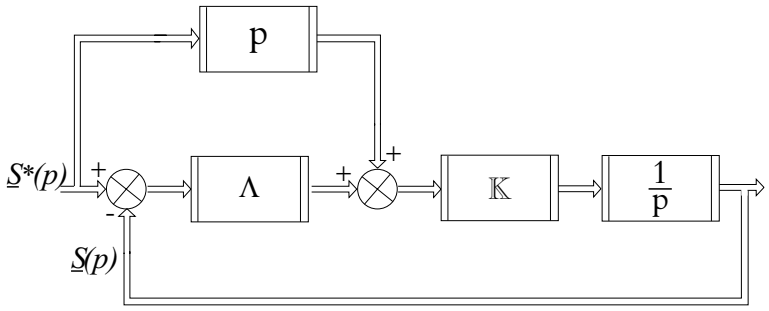


FIG. 2.9 – *Schéma équivalent dans le formalisme de Laplace.*

Le signal capteur $\underline{S}(p)$ s'exprime par :

$$\underline{S}(p) = \frac{1}{p}\mathbb{K}[p.\underline{S}^*(p) + \Lambda.(\underline{S}^*(p) - \underline{S}(p))]$$

soit

$$\underline{S}(p) = (p.\mathbb{I}_m + \mathbb{K}\Lambda)^{-1}\,\mathbb{K}(p.\mathbb{I}_m + \Lambda)\,\underline{S}^*(p)$$

Ce système est stable si les pôles sont à partie réelle négative. Il suffit donc de choisir Λ tel que la matrice $\mathbb{K}\Lambda$ soit diagonale et s'écrive:

$$\mathbb{K}\Lambda = \begin{pmatrix} \frac{1}{\tau_1} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{\tau_m} \end{pmatrix}$$

où les termes τ_i sont des constantes positives.

↪ **Erreurs**: La sortie de notre système $\underline{S}(p)$ peut se réécrire sous la forme:

$$\underline{S}(p) = (p\mathbb{I}_m + \mathbb{K}\Lambda)^{-1}(p\mathbb{I}_m + \mathbb{K}\Lambda + \mathbb{K}p - p\mathbb{I}_m)\underline{S}^*(p)$$

soit

$$\underline{S}(p) = \underline{S}^*(p) - (p\mathbb{I}_m + \mathbb{K}\Lambda)^{-1}(\mathbb{K} - \mathbb{I}_m)p\underline{S}^*(p)$$

et l'erreur $\underline{\epsilon}(p) = \underline{S}(p) - \underline{S}^*(p)$ est donnée par:

$$\underline{\epsilon}(p) = \begin{pmatrix} \frac{\tau_1}{1+\tau_1 p} & & 0 \\ & \ddots & \\ 0 & & \frac{\tau_m}{1+\tau_m p} \end{pmatrix} (\mathbb{K} - \mathbb{I}_m) p \underline{S}^*(p) \tag{2.17}$$

soit en régime établi:

$$\underline{\epsilon}(\infty) = \lim_{p \to 0} \begin{pmatrix} \frac{\tau_1}{1+\tau_1 p} & & 0 \\ & \ddots & \\ 0 & & \frac{\tau_m}{1+\tau_m p} \end{pmatrix} (\mathbb{K} - \mathbb{I}_m) p^2 \underline{S}^*(p) \tag{2.18}$$

- Erreur en position: On applique un vecteur échelon dont les amplitudes constituent le vecteur $\underline{a}$ soit $\underline{S}^*(p) = \frac{1}{p}\underline{a}$ et il vient

$$\underline{\epsilon}_p(\infty) = 0$$

- Erreur en vitesse: De la même manière, l'erreur est calculée en envoyant un vecteur consigne de type rampe dont les pentes sont les composantes de $\underline{b}$ soit $\underline{S}^*(p) = \frac{1}{p^2}\underline{b}$ et l'on obtient:

$$\underline{\epsilon}_v(\infty)_i = \tau_i \sum_j (\mathbb{K} - \mathbb{I}_m)_{ij} b_j$$

  Cette erreur de traînage est d'autant plus faible que la modélisation de notre système est correcte c'est à dire $(\mathbb{K} - \mathbb{I}_m)$ proche de la matrice nulle. Dans le cas inverse, on obtient un effet de couplage qui augmente l'erreur de traînage.

Dans le cas où l'erreur de modélisation existe mais n'implique pas de forts couplages entre les différentes composantes, la matrice $\mathbb{K}$ est diagonale et peut se mettre sous la forme $\mathbb{K} = \mathbb{I}_m + \Delta\mathbb{K}$. L'erreur de traînage peut alors s'écrire:

$$\underline{\epsilon}_v(\infty)_i = \tau_i \Delta\mathbb{K}_i b_i = [\tau \Delta\mathbb{K} b]_i$$

Dans ces conditions, chaque composante vérifie la relation:

$$s(p)_i = \frac{(1 + \tau_i(1 + \Delta\mathbb{K}_i).p)}{(1 + \tau_i.p)} s_i^*(p) \tag{2.19}$$

Dans cette expression, on reconnaît la forme d'un système à décalage de phase, où τ_i représente la constante de temps du système et $(\tau\mathbb{K})_i$ le facteur de déphasage.

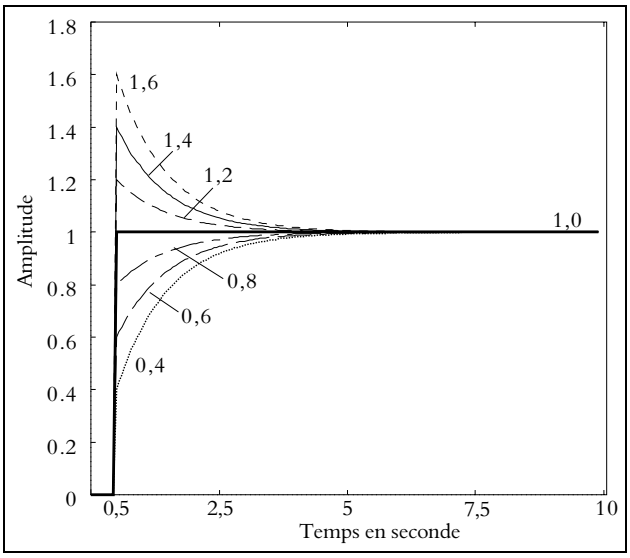


FIG. 2.10 – *Réponse d'un système à décalage de phase pour une entrée indicielle.*

On notera que pour une modélisation parfaite du système ($\Delta\mathbb{K} = 0$), on retrouve le comportement décrit par l'équation 2.15.

La figure 2.10 représente la réponse indicielle du système pour différentes valeurs de $(1 + \Delta\mathbb{K}_i)$ avec τ=1s.

On remarque que l'erreur de modélisation n'intervient pas dans le temps de réponse du système mais agit seulement sur l'amplitude du front de réponse. Sur ces courbes on voit nettement que notre système réagit comme un transfert direct entachée d'une constante de temps parasite $\frac{1}{1+\tau_i p}$. On obtiendra donc un temps de réponse court pour $\tau_i \ll 1$, ce qui pour un $\Delta\mathbb{K}$ donné impose d'avoir un gain λ relativement important.

Ce résultat n'est pas surprenant mais il nécessite de prendre en compte la stabilité du système. Dans notre cas, le système est apparemment toujours stable, toutefois la plate-forme robotique est un système physique comportant des retards parasites dûs au temps de réponse des actionneurs et surtout au traitement des données issues de la caméra. Ces différents retards rajoutent une constante de temps parasite au système, de sorte que notre système devient instable pour de grandes valeurs de gain.

↪ **Discrétisation du système**: Nous donnons ici les principaux résultats d'une étude dans le cas discret avec le formalisme en z. Dans le cas discret il convient de prendre en compte l'effet de blocage du système et le retard éventuel dû au traitement (délai égal à n itérations) (Fig. 2.11).

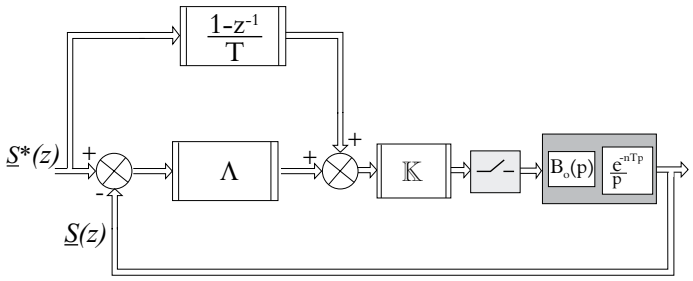


Fig. 2.11 – *Schéma équivalent en discret.*

Le calcul de la trajectoire dans l'image $\underline{S}(z)$ peut donc être exprimé en fonction de la consigne $\underline{S^*}(z)$ par:

$$\underline{S}(z) = \left(\mathbb{I}_m(1-z^{-1}) + \mathbb{K}.\Lambda.T.z^{-1-n}\right)^{-1} \left(\mathbb{K}.\left[(1-z^{-1})\mathbb{I}_m + \Lambda.T\right]\right).z^{-1-n}.\underline{S^*}(z) \tag{2.20}$$

Le calcul de l'erreur statique est donné par:

$$\begin{aligned}\underline{\epsilon}(\infty) = \lim_{z\to 1}(1-z^{-1})^2 \Big[&\left(\mathbb{I}_m(1-z^{-1}) + \mathbb{K}.\Lambda.T.z^{-1-n}\right)^{-1} \\ &\left(\mathbb{I}_m - \mathbb{K}.z^{-1-n}\right)\Big].\underline{S^*}(z)\end{aligned} \tag{2.21}$$

Selon les consignes désirées, on obtient alors les erreurs statiques suivantes:
$\rightarrow$ Echelon (Positionnement) : $\underline{S^*}(z) = \frac{z}{z-1}.\underline{U} \Longrightarrow \underline{\epsilon}_p(\infty) = 0$
$\rightarrow$ Rampe (Translation, ...) : $\underline{S^*}(z) = \frac{Tz}{(z-1)^2}.\underline{V} \Longrightarrow \underline{\epsilon}_v(\infty) = (\mathbb{K}\Lambda)^{-1}(\mathbb{I}_m - \mathbb{K}).\underline{V}$

où $\underline{U}$ et $\underline{V}$ représentent des vecteurs constants dont les composantes correspondent aux amplitudes des consignes respectives.

On constate que notre loi de commande permet bien sûr d'annuler l'erreur statique dans le cas d'un positionnement et que pour un suivi de trajectoire à vitesse constante (consigne de type rampe), on obtient une erreur proportionnelle à l'erreur de modélisation. On vérifie ainsi que pour un système parfait ($\mathbb{K} = \mathbb{I}_6$) l'erreur de traînage dans le cas d'un suivi de trajectoire à vitesse constante est nulle. Notons aussi que cette erreur peut être diminuée pour des gains λ_i élévés comme dans le cas continu.

2.3.2.1 Influence du terme dérivée

Nous proposons à présent de mettre en évidence l'influence du terme dérivée en comparant l'erreur statique de notre système avec et sans la branche dérivée.

Sans la branche dérivée, la transmittance de notre système peut s'écrire:

$$\underline{S}(z)|_{sans} = \left(\mathbb{I}_m(1-z^{-1}) + \mathbb{K}.\Lambda.T.z^{-1-n}\right)^{-1}.\mathbb{K}\Lambda.T.z^{-1-n}.\underline{S}^*(z) \tag{2.22}$$

Par conséquent, l'erreur $\underline{\epsilon}(z)|_{sans} = \underline{S}^*(z) - \underline{S}(z)|_{sans}$ de notre système s'écrit en reprenant le théorème de la valeur finale :

$$\underline{\epsilon}(\infty)|_{sans} = \lim_{z\to 1}\left(\mathbb{I}_m(1-z^{-1}) + \mathbb{K}.\Lambda.T.z^{-1-n}\right)^{-1}.(1-z^{-1})^2.\underline{S}^*(z) \tag{2.23}$$

Nous pouvons vérifier que pour une tâche de positionnement (échelon), nous obtenons une erreur statique nulle.

Dans le cas d'un suivi de trajectoire à vitesse constante, la consigne sera de type rampe et il vient:

$$\underline{\epsilon}_v(\infty)|_{sans} = \left(\mathbb{K}.\Lambda.T\right)^{-1}.T.\underline{V} = \left(\mathbb{K}.\Lambda\right)^{-1}.\underline{V} \tag{2.24}$$

où $\underline{V}$ représente un vecteur constant de dimension (6×1) et dont les composantes correspondent aux pentes des rampes de consigne.

Les erreurs de traînage avec et sans terme dérivée dans le cas d'un suivi de trajectoire peuvent être résumées comme suit:

- Avec une action dérivée: $\underline{\epsilon}_v(\infty) = (\mathbb{K}\Lambda)^{-1}(\mathbb{I}_m - \mathbb{K}).\underline{V}$
- Sans le terme dérivée: $\underline{\epsilon}_v(\infty)|_{sans} = \left(\mathbb{K}.\Lambda\right)^{-1}.\underline{V}$

On constate donc que le terme dérivée permet de diminuer notablement l'erreur de traînage du système en rajoutant à l'information position $(\underline{s}^*(t))$ une information de direction $(\frac{d}{dt}\underline{s}^*(t))$ renseignant le système sur le comportement à adopter pour suivre au mieux la consigne. Comme nous l'avons déjà fait remarquer précédemment, lorsque la modélisation du système est parfaite, le terme dérivée annule totalement l'erreur de traînage tandis qu'en l'absence de ce terme, le système converge vers une erreur inversement proportionnelle à l'amplitude du gain λ.

Dans ce paragraphe nous avons développé la loi de commande que nous allons utiliser afin d'effectuer le suivi de trajectoire. Intéressons-nous à présent à la génération de la consigne $\underline{s}^*(t)$ en fonction de la trajectoire à réaliser.

2.3.3 Génération du motif de référence

Comme nous l'avons évoqué dans la section précédente, la trajectoire suivie par la caméra est générée directement à partir des variations de la consigne. De ce fait, notre loi de commande nécessite la réalisation d'une liaison de type rigide entre la caméra et la scène, ce qui se concrétise par un choix de $\underline{s}(t)$ tel que $\dim(\underline{s}(t))\geq 6$. Rappelons

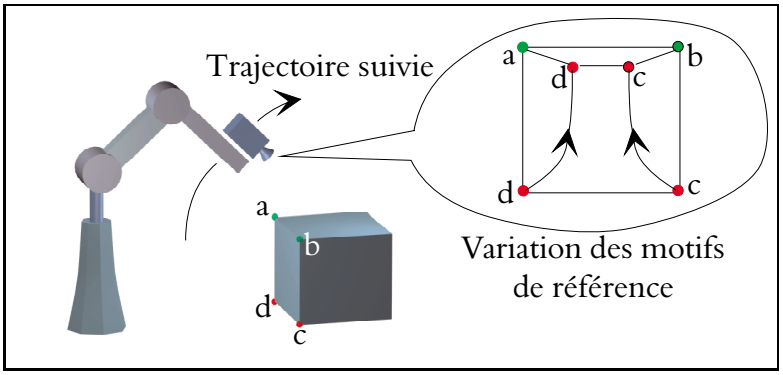


Fig. 2.12 – *Suivi de trajectoire à partir des informations visuelles.*

que l'on entend par liaison rigide, une relation d'interaction complètement contrainte par la tâche visuelle. Ainsi un mouvement de la caméra entraîne obligatoirement une variation des motifs visuels et réciproquement. Le contrôle de la variation de ces motifs nous permet alors de réguler la trajectoire suivie par la caméra (Fig.2.12).

Le calcul de la trajectoire de référence dans l'espace capteur est effectué en considérant la cible (objet) comme la référence absolue à partir de laquelle sont faits les déplacements. Ainsi la connaissance d'un modèle 3D de la cible est nécessaire pour suivre la trajectoire proposée.

Dans un premier temps, on exprime les coordonnées tridimensionnelles de la cible dans le repère de la caméra en fonction des vitesses désirées le long de la trajectoire. On a ainsi les coordonnées idéales de la cible dans le repère caméra en fonction des mouvements à effectuer. Le calcul des motifs de références dans l'image est simplement déduit à partir d'une projection perspective. La figure (2.13) présente une configuration simplifiée de notre approche. Sur cette figure, nous présentons les deux étapes permettant de calculer les motifs visuels de référence $\underline{s}^*(t)$[5]: la transformation P_1 permet d'exprimer les coordonnées 3D $\underline{p}^*(t)$ de la cible dans le

5. On peut remarquer que le calcul du motif de référence aurait pu être de manière itérative par la relation $\underline{S}_{t+\Delta t} = \underline{S}_t + L^T.\Delta T$. Toutefois, nous n'avons pas choisi cette approche, en raison du caractère intégrateur de cette formule. Ainsi, nous évitons d'accumuler une erreur présente à chaque itératrion sur $\underline{S}$ et L^T

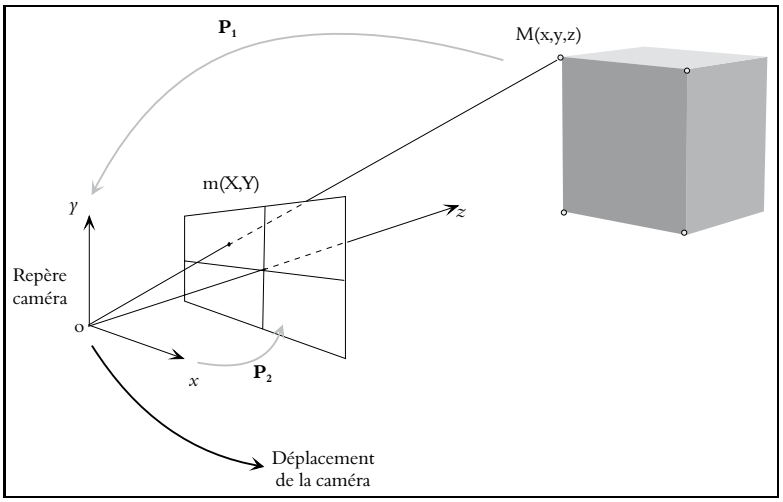


FIG. 2.13 – *Génération du motif de référence.*

repère caméra R_{cam}, puis la transformation P_2 effectue la projection perspective de ces coordonnées sur le plan image R_{im}:

$$P_1 : \underline{p}^*(t)/R_{cib} \longrightarrow \underline{p}^*(t)/R_{cam}$$
$$P_2 : \underline{p}^*(t)/R_{cam} \longrightarrow \underline{p}^*(t)/R_{im} = \underline{s}^*(t)$$

Le développement de P_1 est basé sur le fait que les coordonnées d'un point M (la cible) peuvent être exprimées dans un repère animé d'une vitesse $(\underline{V}, \underline{\Omega})$ par:

$$\frac{d}{dt}\overrightarrow{OM} = -\overrightarrow{V(O)} - \vec{\Omega} \wedge \overrightarrow{OM} \tag{2.25}$$

Nous pouvons redévelopper cette expression en utilisant une notation matricielle où l'on note le point considéré $\underline{p} = (x, y, z)^T$, la translation $\underline{V} = (V_x, V_y, V_z)^T$ et la rotation $\omega\tilde{r}$. Dans l'expression $\omega\tilde{r}$, ω correspond à la norme du vecteur rotation (i.e $\omega = ||\vec{\Omega}|| = \sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2}$) et $\tilde{r}$ représente la matrice antisymétrique[6] associée au vecteur unitaire $\vec{r}$ représentant l'axe de la rotation $\vec{\Omega} = \omega\vec{r}$. On a donc

$$\tilde{r} = \begin{pmatrix} 0 & -\bar{\omega}_z & \bar{\omega}_y \\ \bar{\omega}_z & 0 & -\bar{\omega}_x \\ -\bar{\omega}_y & \bar{\omega}_x & 0 \end{pmatrix}$$

6. L'utilisation de la matrice antisymétrique permet de "transformer" le produit vectoriel en un produit matriciel, plus facile à manipuler dans la suite des calculs. Ainsi on considère que $\tilde{r}. \equiv \vec{r}\wedge$

avec $\bar{\omega}_i = \frac{\omega_i}{\omega}$.

et la relation (2.25) peut alors se réécrire sous la forme:

$$\frac{d}{dt}\underline{p} = -\underline{V} - \omega\tilde{r}\underline{p} \tag{2.26}$$

Dans le cas où les vitesses $\underline{\Omega}$ et $\underline{V}$ sont constantes, la résolution de cette équation différentielle permet d'exprimer le vecteur $\underline{p}$ en fonction de sa position initiale et des vitesses appliquées au repère caméra. La solution de (2.26) est alors donnée par:

$$\underline{p}(t) = e^{-\omega\tilde{r}(t-t_0)}.\underline{p}(t_0) - e^{-\omega\tilde{r}(t-t_0)} \int_{\alpha=t_o}^{\alpha=t} e^{\omega\tilde{r}(\alpha-t_0)} d\alpha.\underline{V} \tag{2.27}$$

Le terme exponentiel peut être développé en une série de Taylor sous la forme:

$$e^{\omega\tilde{r}} = 1 + \frac{(\omega\tilde{r})}{1!} + \frac{(\omega\tilde{r})^2}{2!} + \cdots + \frac{(\omega\tilde{r})^n}{n!}$$

or on montre que:

$$(\tilde{r})^{2n+1} = (-1)^n(\tilde{r}) \quad \text{pour} \quad n \geq 0$$

$$(\tilde{r})^{2n} = (-1)^{n+1}(\tilde{r})^2 \quad \text{pour} \quad n \geq 1$$

et en regroupant dans le développement de l'exponentielle les termes paires et impaires, on retrouve une des formules de Rodrigues :

$$e^{\omega\tilde{r}} = \mathbb{I}_3 + (sin\omega).\tilde{r} + (1 - cos\omega)\tilde{r}^2 \tag{2.28}$$

La solution finale (2.27) peut alors s'écrire :

$$\underline{p}(t) = \left(\mathbb{I}_3 - sin(\omega(t-t_0)).\tilde{r} + (1 - cos(\omega(t-t_0)))\tilde{r}^2\right)\underline{p}(t_0) -$$

$$(t-t_0)\left(\mathbb{I}_3 - \frac{1 - cos(\omega(t-t_0))}{\omega(t-t_0)}\tilde{r} + \frac{\omega(t-t_0) - sin(\omega(t-t_0))}{\omega(t-t_0)}\tilde{r}^2\right).\underline{V} \tag{2.29}$$

A partir de cette expression, nous pouvons exprimer les coordonnées des primitives choisies (points, droites, ...) dans le repère de la caméra et en fonction des vitesses appliquées à ce dernier. Cette relation constitue la transformation P_1.

La seconde transformation P_2 est simplement réalisée par une projection perspective basée sur un modèle sténopé de caméra, à savoir:

$$X(t) = \frac{x(t)}{z(t)}.F_u \qquad\qquad Y(t) = \frac{y(t)}{z(t)}.F_v \tag{2.30}$$

La méthode proposée dans ce paragraphe permet de calculer les motifs visuels de référence $\underline{s}^*(t)$ à partir de la connaissance de la cible et du mouvement à effectuer. La relation (2.29) peut paraître relativement complexe, aussi allons-nous en proposer quelques applications dans les cas les plus classiques.

2.3.4 Trajectoires particulières

Toute trajectoire aussi complexe soit-elle, peut se décomposer en une suite de translations et de rotations suivant différents axes. Pour cette raison et afin d'illustrer notre propos, nous présentons dans les deux paragraphes suivants le calcul des motifs de référence pour une translation le long d'un axe quelconque et pour une rotation autour d'un axe quelconque et à une distance donnée.

2.3.4.1 Translation

2.3.4.1.1 Mouvement à vitesse constante Dans le cas d'une translation à vitesse constante, nous avons naturellement $\omega = 0$ et la relation (2.29) se simplifie singulièrement pour se réduire à:

$$\underline{p}(t) = \underline{p}(t_0) - \underline{V}(t - t_0)$$

en faisant apparaître la "période d'échantillonnage" $\Delta t = (t - t_0)$, la relation précédente nous donne:

$$\underline{p}(t + \Delta t) = \underline{p}(t) - \underline{V}.\Delta t \tag{2.31}$$

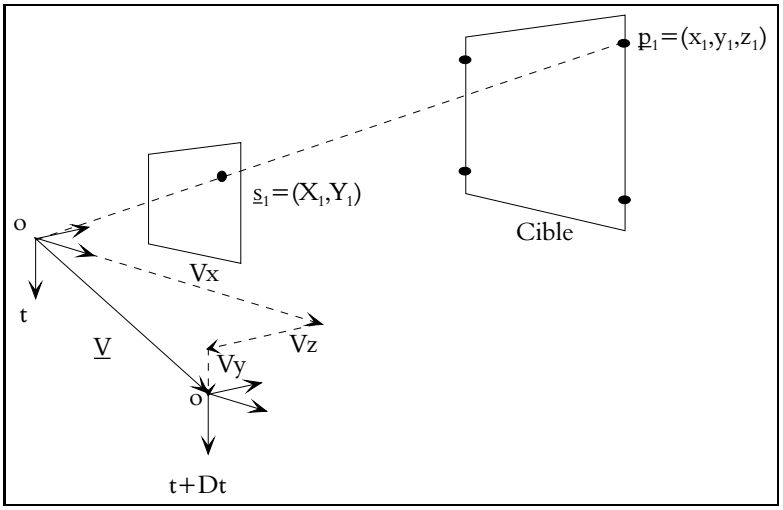


FIG. 2.14 – *Mouvement de translation*

L'expression finale des motifs visuels de référence $\underline{s}^*(t) = (X_1, Y_1, \ldots, X_n, Y_n)^T$

sera donnée par la projection perspective, soit:

$$X_i(t+\Delta t) = \frac{x_i(t) - V_x.\Delta t}{z_i(t) - V_z.\Delta t}.F_u$$
$$Y_i(t+\Delta t) = \frac{y_i(t) - V_y.\Delta t}{z_i(t) - V_z.\Delta t}.F_v \tag{2.32}$$

avec $i \in [1, n]$.

2.3.4.1.2 Mouvement à vitesse variable Lorsque le mouvement désiré est simplement une translation dont l'amplitude varie avec le temps, l'équation 2.26 se réécrit sous la forme :

$$\frac{d}{dt}\underline{p} = -\underline{V}(t) \tag{2.33}$$

dont une solution générale sera donnée par :

$$\underline{p}(t) = \underline{p}(t_0) - \int_{\alpha=t_o}^{\alpha=t} \underline{V}(\alpha)d\alpha \tag{2.34}$$

Ce cas de figure est mis en œuvre dans la section suivante consacrée aux expérimentations en réalisant une trajectoire hélicoïdale parallèlement à la face d'un cube.

2.3.4.2 Rotation

Une rotation autour d'un axe quelconque, à une distance donnée $||\underline{\rho}||$, est réalisée par la combinaison de deux mouvements élémentaires : une rotation orientant les axes du repère et une translation permettant de faire évoluer l'effecteur le long de la trajectoire circulaire (Fig. 2.15). Nous allons tout d'abord chercher à exprimer le mouvement de translation en fonction de la vitesse angulaire et du rayon de courbure de la trajectoire.

Pour cela, utilisons le trièdre de Frenet qui forme une base orthonormale directe à partir du vecteur normal, du vecteur tangent et du vecteur binormal. Dans notre cas de figure, la vitesse de translation $\vec{V}$ est portée par le vecteur tangent et la distance $\vec{\rho}$ entre la caméra et l'axe est décrite suivant le vecteur normal. L'axe de rotation porté par le vecteur unitaire $\vec{r}$ est quant à lui aligné avec le vecteur binormal ce qui vérifie :

$$\vec{V} = -\omega\vec{r} \wedge \vec{\rho}$$

en reprenant l'écriture à l'aide de la matrice antisymétrique nous avons:

$$\underline{V} = -\omega\tilde{r}\underline{\rho}$$

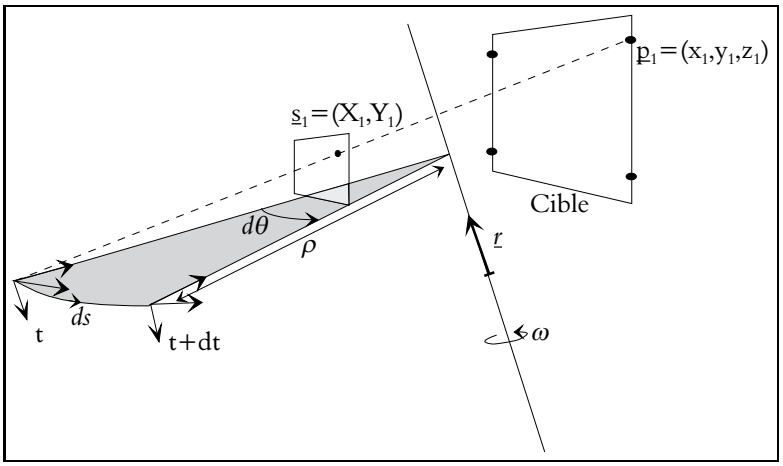


FIG. 2.15 – *Mouvement de rotation*

et d'après 2.29, il vient en réorganisant les termes :

$$\underline{p}(t) = \underline{p}(t_0) - \left(sin(\omega(t-t_0)).\tilde{r} - (1 - cos(\omega(t-t_0)))\tilde{r}^2\right).(\underline{p}(t_0) - \underline{\rho}) \qquad (2.35)$$

2.4 Résultats expérimentaux

Dans ce dernier paragraphe nous présentons les différents résultats obtenus sur notre plate-forme expérimentale et sur le simulateur développé au laboratoire. La description de notre cellule robotique et du simulateur font respectivement l'objet des annexes A et B.

Les résultats proposés dans ce paragraphe ont pour but de montrer la validité de l'approche que nous avons développée et d'en présenter quelques applications potentielles.

L'objet utilisé pour nos différentes expérimentations est un cube de 25 cm de côté marqué par des amers lumineux (Fig. 2.16). Ces amers sont disposés par deux sur les arêtes latérales à 2.5cm des sommets.

Durant les expérimentations, nous utilisons un repère absolu noté $\mathcal{R}_a$ et centré sur le cube.

Les différents amers admettent comme coordonnées dans $\mathcal{R}_a$:

$$\begin{cases} x = \pm 12.5cm \\ y = \pm 7.5cm \\ z = \pm 12.5cm \end{cases}$$

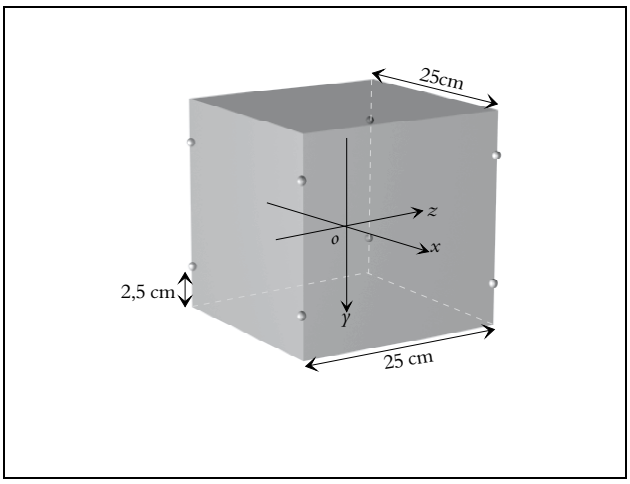


Fig. 2.16 – *Disposition des amers sur le cube*

D'un point de vue temporel, les différentes simulations et expérimentations ont été effectuées en temps réel vidéo, soit $40ms$ par itération. Durant les simulations, un retard de deux itérations a été introduit afin de prendre en compte les délais

d'acquisition et de traitement vidéo. De plus, sauf spécifications contraires, toutes les expérimentations ont été faites avec un gain de boucle $\lambda = 0.5s^{-1}$ et les unités utilisées sont le mètre et le radian.

Nous rappelons que la loi de commande que nous avons utilisée pour toutes ces expérimentations est:

$$T = -L_{\underline{s}^*}^{T+} \Lambda(\underline{s}(\underline{r}, t) - \underline{s}^*(t)) + L_{\underline{s}^*}^{T+} \frac{d\underline{s}^*(t)}{dt} \tag{2.36}$$

dans laquelle la matrice d'interaction $L_{\underline{s}^*}^{T+}$ correspond à la pseudo-inverse de $L_{\underline{s}^*}^{T}$ telle que:

$$L_{s=s^*}^T = \begin{pmatrix} \frac{-F_u}{z_1^*} & 0 & \frac{X_1}{z_1^*} & \frac{X_1^* Y_1^*}{F_v} & -\left(F_u + \frac{X_1^{*2}}{F_u}\right) & Y_1^* \frac{F_u}{F_v} \\ 0 & \frac{-F_v}{z_1^*} & \frac{Y_1}{z_1^*} & \left(F_v + \frac{Y_1^{*2}}{F_v}\right) & -\frac{X_1^* Y_1^*}{F_u} & -X_1^* \frac{F_v}{F_u} \\ \frac{-F_u}{z_2^*} & 0 & \frac{X_2}{z_2^*} & \frac{X_2^* Y_2^*}{F_v} & -\left(F_u + \frac{X_2^{*2}}{F_u}\right) & Y_2^* \frac{F_u}{F_v} \\ 0 & \frac{-F_v}{z_2^*} & \frac{Y_2}{z_2^*} & \left(F_v + \frac{Y_2^{*2}}{F_v}\right) & -\frac{X_2^* Y_2^*}{F_u} & -X_2^* \frac{F_v}{F_u} \\ \frac{-F_u}{z_3^*} & 0 & \frac{X_3}{z_3^*} & \frac{X_3^* Y_3^*}{F_v} & -\left(F_u + \frac{X_3^{*2}}{F_u}\right) & Y_3^* \frac{F_u}{F_v} \\ 0 & \frac{-F_v}{z_3^*} & \frac{Y_3}{z_3^*} & \left(F_v + \frac{Y_3^{*2}}{F_v}\right) & -\frac{X_3^* Y_3^*}{F_u} & -X_3^* \frac{F_v}{F_u} \\ \frac{-F_u}{z_4^*} & 0 & \frac{X_4}{z_4^*} & \frac{X_4^* Y_4^*}{F_v} & -\left(F_u + \frac{X_4^{*2}}{F_u}\right) & Y_4^* \frac{F_u}{F_v} \\ 0 & \frac{-F_v}{z_4^*} & \frac{Y_4}{z_4^*} & \left(F_v + \frac{Y_4^{*2}}{F_v}\right) & -\frac{X_4^* Y_4^*}{F_u} & -X_4^* \frac{F_v}{F_u} \end{pmatrix} \tag{2.37}$$

où X_i^* et Y_i^* correspondent respectivement à l'abscisse et à l'ordonnée désirées dans l'image du point i et la matrice Λ a été choisie telle que:

$$\Lambda = \begin{pmatrix} \lambda & & 0 \\ & \ddots & \\ 0 & & \lambda \end{pmatrix} \tag{2.38}$$

2.4.1 Résultats de simulation

Les résultats présentés dans ce paragraphe ont été réalisés sur un simulateur que nous avons développé au laboratoire. Les différentes conditions de simulations sont les suivantes :

- Caméra : Modèle sténopé sans distorsions.
- Robot : Intégrateur parfait sans dynamique et sans couplage.
- Système de traitement : Retard de 2 itérations.
- Calibrage parfait de la matrice de passage bras-œil.

2.4.1.1 Translation

Dans un premier temps, une tâche classique de positionnement place la caméra sur la trajectoire à effectuer. La situation désirée de la caméra après ce positionnement est telle que la caméra soit à 67.5cm de la face du cube et que les deux amers de droite apparaissent centrés dans l'image.

A partir de ce positionnement, on effectue une trajectoire de translation parallèlement à la face du cube (Fig. 2.17). Pour réaliser ce déplacement, les motifs de référence dans l'image sont calculés à partir des équations appliquées aux quatre points.

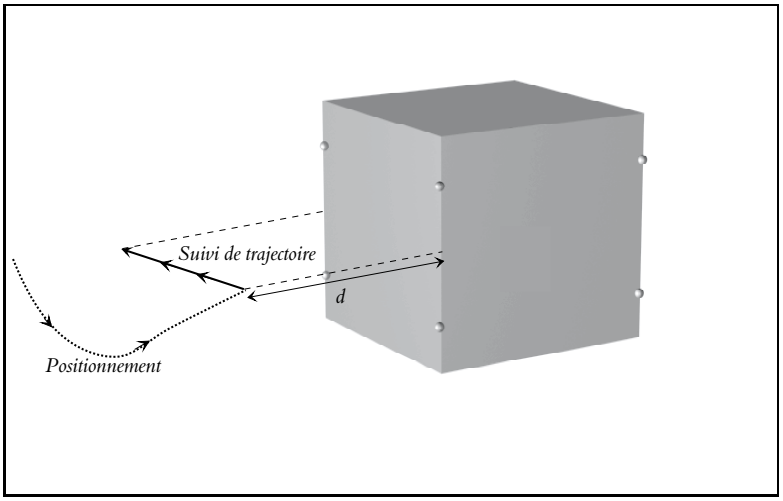


Fig. 2.17 – *Translation parallèlement à une face.*

Les différents paramètres de cette expérimentation sont les suivants :

- Position d'origine de l'effecteur dans $\mathcal{R}_a$: $(-0.15m, -0.22m, -1.1m)$
- Orientation d'origine de l'effecteur dans $\mathcal{R}_a$: $(-17^o, 22^o, 28^o)$
- Positionnement du début de trajectoire dans $\mathcal{R}_a$: $(0.125m, 0m, -0.80m)$
- Vitesse de déplacement de la caméra : $V_x = -0.05m.s^{-1}$

Remarque: Les différentes coordonnées sont données suivant l'ordre x, y, z.

Les motifs de référence réalisant cette trajectoire sont :

$$X_i^*(t+\Delta t) = \frac{x_i(t) - V_x \Delta t}{z_i(t)}.F_u$$

$$Y_i^*(t+\Delta t) = \frac{y_i(t)}{z_i(t)}.F_v$$

avec $i = 1, 2, 3, 4$.

Dans un premier paragraphe nous présentons les résultats simulés dans un cas idéal. Dans un second paragraphe, des bruits de mesure ont été rajoutés à la mesure dans l'image et sur le torseur cinématique.

2.4.1.1.1 Simulation sans bruit de mesure Les Figures 2.18.a et 2.18.b représentent les courbes en fonction du temps et des composantes du torseur cinématique appliquées à la caméra. Ces courbes sont respectivement les trois vitesses de translation et les trois composantes en rotation.

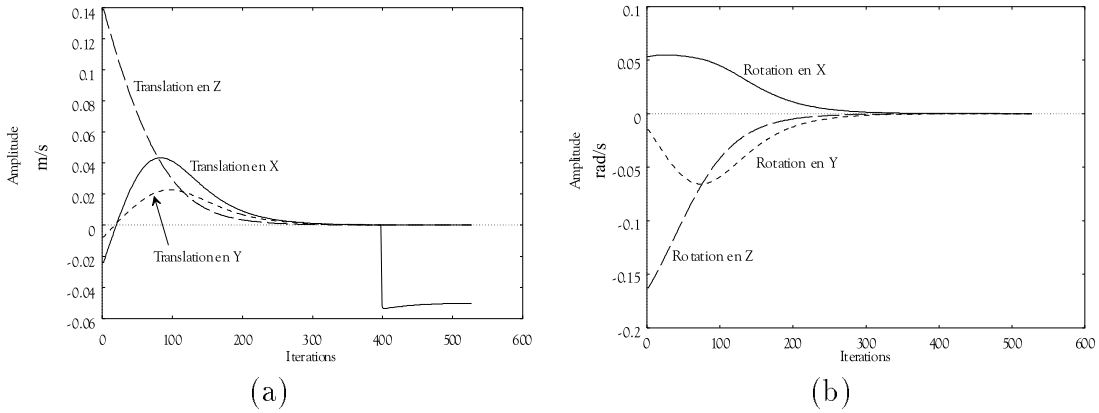


FIG. 2.18 – *Vitesses de translation et de rotation de l'effecteur*

On retrouve sur ces courbes la tâche de positionnement qui suit une décroissance exponentielle puis la tâche de suivi de trajectoire parfaitement réalisée. En effet la translation suivant l'axe est bien de $-5cm.s^{-1}$ tandis que les autres composantes sont nulles.

La figure 2.19 décrit l'erreur des différents signaux capteurs. Cette erreur est calculée en faisant la somme des différences quadratiques entre la mesure et la consigne soit:

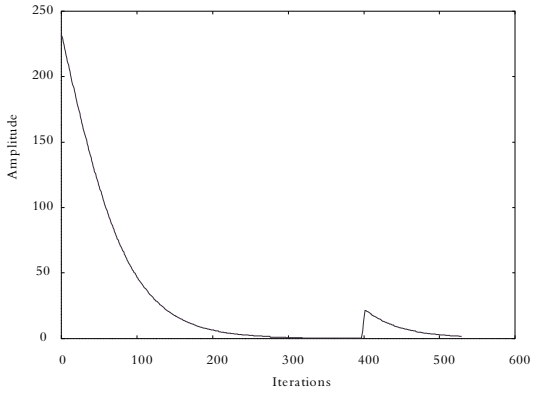


Fig. 2.19 – *Erreur durant la tâche*

$$e(t) = \sqrt{\sum_{i=1}^{i=4} [(X_i(t) - X_i^*(t))^2 + (Y_i(t) - Y_i^*(t))^2]} \tag{2.39}$$

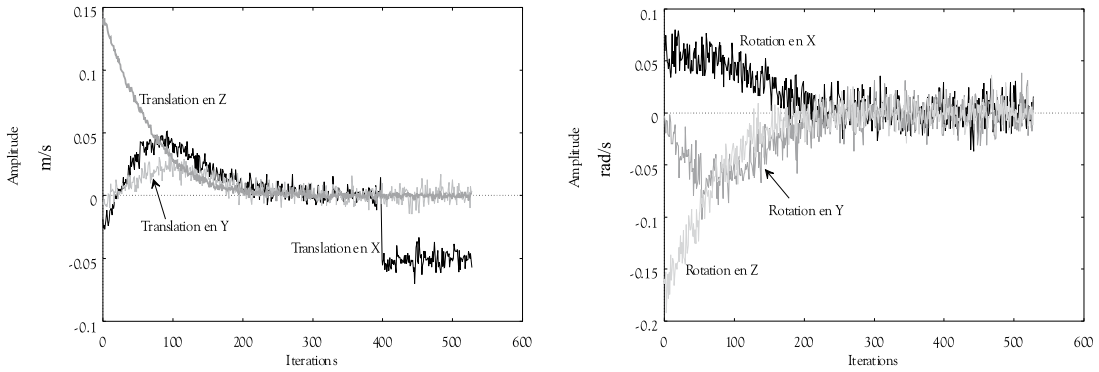
On constate la décroissance exponentielle de l'erreur ainsi qu'un petit pic lors de la transition entre la tâche de positionnement et la tâche de suivi. Ce pic s'explique par la combinaison de 2 phénomènes:

- l'initialisation du calcul de la dérivée numérique et
- le retard pur introduit par le système de traitement d'image,

qui imposent au système une certaine inertie. On constatera toutefois que cette erreur décroît de façon exponentielle.

2.4.1.1.2 Simulation bruitée Afin de valider notre approche dans des conditions plus proches de la réalité, nous avons refait cette simulation en rajoutant dans la boucle de commande des bruits sur la mesure image et sur le torseur cinématique. Toutes les autres conditions d'expérimentation ont été conservées.

Les mesures image sont bruitées avec un bruit gaussien d'écart type $\sigma = 0.5$ pixel qui est directement rajouté aux coordonnées (X_i, Y_i). Au niveau du torseur cinématique, un bruit gaussien d'écart type $\sigma = 0.001\ m.s^{-1}$ est appliqué aux vitesses de translation et un bruit gaussien d'écart type $\sigma = 0.01\ rad.s^{-1}$ a été rajouté aux composantes de rotation.

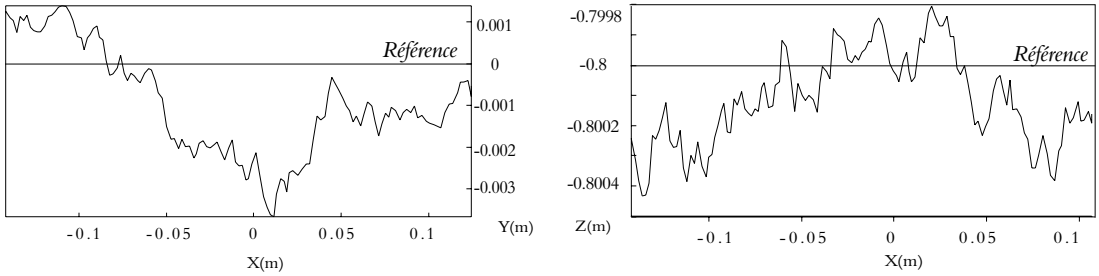


(a) (b)

FIG. 2.20 – *Vitesses de translation et de rotation de l'effecteur*

Les valeurs du torseur cinématique en translation et rotation sont présentées sur la figure 2.20.

Comme on le remarque. malgré des bruits relativement importants par rapport aux amplitudes, la convergence et le suivi sont assurés. De plus les figures 2.21a et b, nous permettent de constater que la trajectoire n'est que très faiblement déformée : moins de 1 mm suivant l'axe optique (Z), et environ 5 mm suivant l'axe Y.



(a) (b)

FIG. 2.21 – *Trajectoire vue de derrière (a) et de dessus (b).*

De la même manière, nous présentons dans le prochain paragraphe une simulation réalisant une rotation autour de l'arête de notre cube.

2.4.1.2 Rotation

Nous effectuons là encore une tâche de positionnement qui nous permet de se placer sur la trajectoire à suivre. Ce positionnement nous place à 67.5cm de l'arête

du cube, parallèlement à la face et de manière à centrer dans l'image les deux amers de droite (Fig. 2.22).

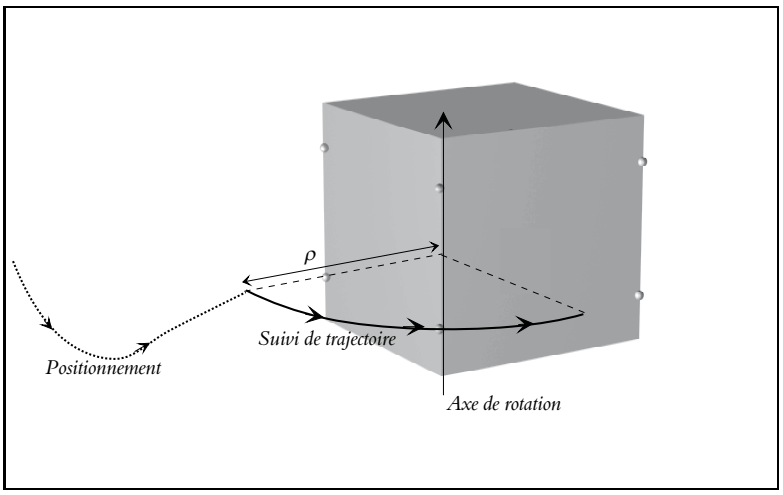


Fig. 2.22 – *Rotation autour d'une arête du cube*

Le motif de référence est calculé à partir de la relation 2.35 et de la projection perspective. L'axe de rotation que nous avons choisi est l'arête du cube. Cette dernière est parallèle à l'axe Y du repère caméra, par conséquent il vient $\underline{r} = [0, r_y, 0]^T = [0, 1, 0]^T$ et nous obtenons:

$$\tilde{r} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} \text{ donc } \tilde{r}^2 = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

De même, le rayon de la trajectoire est orienté suivant l'axe optique donc il vient :

$$\underline{\rho} = \begin{pmatrix} 0 \\ 0 \\ \rho \end{pmatrix}$$

Nous pouvons alors redévelopper la relation 2.35 suivant chaque axe et nous obtenons:

$$\begin{cases} x(t+\Delta t) = x(t)\cos(\omega\Delta t) - (z(t)-\rho)\sin(\omega\Delta t) \\ y(t+\Delta t) = y(t) \\ z(t+\Delta t) = x(t)\sin(\omega\Delta t) + (z(t)-\rho)\cos(\omega\Delta t) + \rho \end{cases}$$

La projection perspective nous donne alors :

$$X^*(t+\Delta t) = \frac{x(t)\cos(\omega\Delta t) - (z(t)-\rho)\sin(\omega\Delta t)}{x(t)\sin(\omega\Delta t) + (z(t)-\rho)\cos(\omega\Delta t) + \rho}.F_u$$

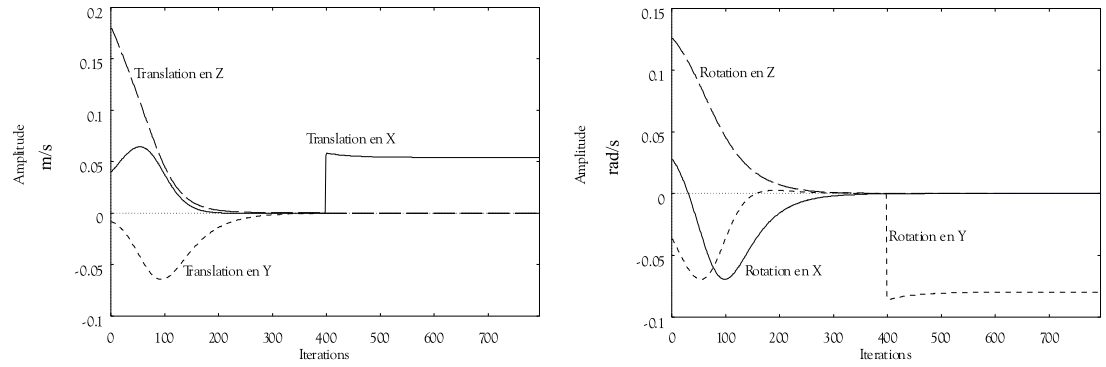
$$Y^*(t+\Delta t) = \frac{y(t)}{x(t)\sin(\omega\Delta t) + (z(t)-\rho)\cos(\omega\Delta t) + \rho}.F_v$$

Les différentes caractéristiques de cette expérimentation sont les suivantes :

- Position d'origine de l'effecteur dans $\mathcal{R}_a$: $(-0.15m, 0.52m, -1.2m)$
- Orientation d'origine de l'effecteur dans $\mathcal{R}_a$: $(17^o, 17^o, -24^o)$
- Positionnement sur la trajectoire dans $\mathcal{R}_a$: $(0.125m, 0m, -0.80m)$
- Vitesse de déplacement de la caméra : $\omega = -0.08 rad.s^{-1}$

Comme précédemment, nous présentons tout d'abord une simulation sans bruit de mesure, puis dans un second temps nous appliquons le même type de bruit que pour la translation.

2.4.1.2.1 Simulation sans bruit de mesure La combinaison d'un mouvement de translation et de rotation apparaît très bien sur les courbes de vitesses (Fig. 2.23a et b). En effet, on peut noter que le déplacement de la caméra s'effectue



(a) (b)

FIG. 2.23 – *Vitesses de translation et de rotation de l'effecteur*

bien avec une vitesse angulaire de $\omega = 0.08 rad.s^{-1}$ et une vitesse tangentielle de $V = -0.08 \times 0.675 = 0.054 m.s^{-1}$. Comme le montre la figure 2.24.a, la trajectoire obtenue est parfaitement circulaire et le maintien du rayon ρ est assuré à moins de 2 millimètres durant tout le suivi de trajectoire(Fig. 2.24.b).

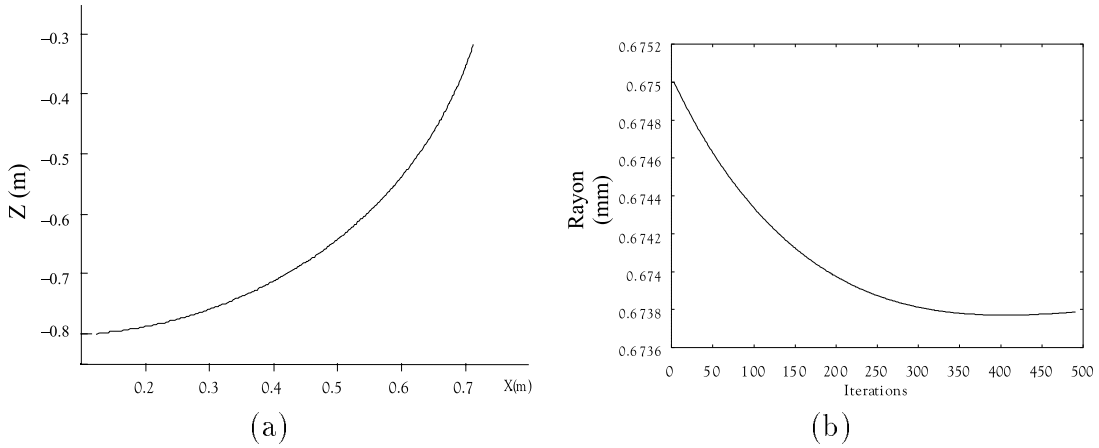


FIG. 2.24 – *(a) Trajectoire de l'effecteur. (b) Evolution du rayon*

2.4.1.2.2 Simulation bruitée Ce paragraphe présente la simulation précédente à laquelle nous avons rajouté les mêmes bruits de mesure que pour la translation soit :

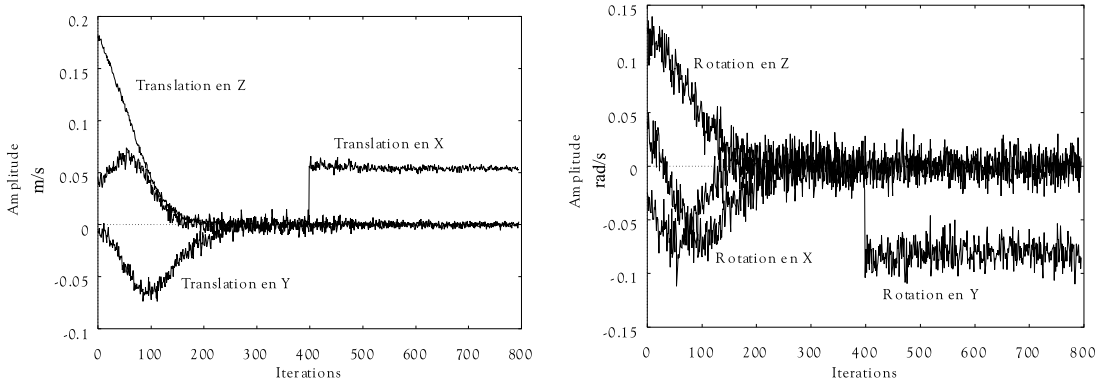
- Un bruit gaussien d'écart type 0.5 pixel sur les mesures image $\underline{s}$,
- Un bruit gaussien d'écart type $1mm.s^{-1}$ sur les composantes en translation du torseur cinématique,
- et un bruit gaussien d'écart type 0.01 $rad.s^{-1}$ sur les rotations.

Les figures 2.25a et b représentant les différentes composantes du torseur cinématique nous permettent de constater un très bon suivi de la trajectoire malgré les bruits appliqués. Ces résultats sont confirmés par la figure 2.26 qui représente l'évolution du rayon de la trajectoire en fonction du temps.

En effet, comme dans le cas non bruité, le rayon de la trajectoire ne varie que de quelques millimètres durant le suivi, ce qui permet d'obtenir là encore une trajectoire parfaitement circulaire. La trajectoire de l'effecteur n'est pas présentée puisqu'elle est similaire au cas non bruité.

2.4.2 Influence du terme dérivée

La loi de commande que nous avons présentée dans le paragraphe 2.3.1, nous permet d'effectuer différents types de suivi de trajectoire. Cette loi de commande



(a) (b)

FIG. 2.25 – *Vitesses de translation et de rotation de l'effecteur*

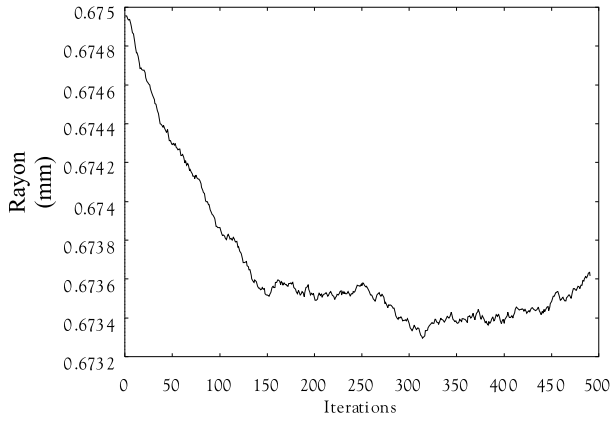


FIG. 2.26 – *Evolution du rayon de la trajectoire durant le suivi*

diffère d'une simple loi de positionnement par l'ajout d'un terme dérivée permettant de prendre en compte l'évolution de la consigne au cours du temps.

Dans ce paragraphe, nous présentons une simulation permettant de mettre en évidence l'influence de ce terme. Pour cela, nous avons repris la première simulation (translation le long d'une face), et nous comparons l'évolution de l'erreur de traînage en fonction de la présence du terme $\frac{d\underline{s}^*}{dt}$. Dans un premier temps, nous allons calculer l'erreur de traînage théorique à partir du modèle proposé dans le paragraphe 2.3.2.3, puis nous la comparerons à l'erreur effective trouvée lors de la simulation.

Les paramètres utilisés pour cette simulation sont les mêmes que précédemment

soit :

- Vitesse de déplacement de la caméra : $V_x = -5cm.s^{-1}$.
- Gain de la boucle de commande : $\lambda = 0.5s^{-1}$
- Distance entre la caméra et la face : $z = 0.675cm$

Sachant que la trajectoire suivie est une translation suivant l'axe x, nous pouvons considérer que l'erreur totale $\epsilon(t)$ (calculée par la relation 2.39) est la résultante de l'erreur de chacun des points. Dans le cas présent (simulation), le déplacement est effectué suivant l'axe x, par conséquent nous pouvons considérer que l'erreur totale $\epsilon(t)$ résulte d'une erreur identique sur chacun des 4 points. En d'autres termes, l'erreur globale $\epsilon(t)$ peut être déduite de l'erreur sur un point $\epsilon(t)$ par :

$$\epsilon(t) = \sqrt{\sum_{i=1}^{i=4}(X_i(t) - X_i^*(t))^2} = \sqrt{4.(X_1(t) - X_1^*(t))^2} = 2.\epsilon_X(t)$$

où $\epsilon_X(t)$ correspond ici à l'erreur de traînage de l'abscisse du point 1.

Nous allons donc maintenant calculer l'erreur $\epsilon(t)|_{sans}$ pour une translation à partir de $\epsilon_X(t)|_{sans}$ sans le terme dérivée.

Pour le point i, le motif de référence $X_i^*(t)$ est de la forme:

$$X_i^*(t) = \frac{x_i(t_o) - V_x(t - t_o)}{z}.F_u$$

où t_o désigne l'instant initial. Pour des raisons évidentes de simplification et sans perte de généralité, nous considérerons le cas d'un point tel que $x(t_o) = 0$ à $t_o = 0$ et il vient:

$$X_1^*(t) = \frac{-V_x.t}{z}.F_u$$

Comme nous pouvons le constater, cette consigne est un signal de type rampe de pente $\frac{-V_x}{z}.F_u$, donc la relation 2.24 dans le cas monodimensionnel de $\mathcal{X}_1^*(\breve{z})$ nous donne:

$$\epsilon_X(\infty)|_{sans} = \lim_{\breve{z}\to 1} \frac{(1-\breve{z}^{-1})^2}{\lambda_1.k_1.T.\breve{z}^{-3} - \breve{z}^{-1} + 1}.\frac{-V_x.F_u}{z}.\frac{T.\breve{z}^{-1}}{(1-\breve{z}^{-1})^2} = \frac{-V_x.F_u}{\lambda_1.k_1.z} \qquad (2.40)$$

où $\breve{z}$ correspond à la variable complexe de la transformation en z et T à la période d'échantillonnage et où k_1 et λ_1 correspondent respectivement au composante (1,1) des matrices $\mathbb{K}$ et Λ.

L'erreur de traînage globale aura donc comme amplitude :

$$\epsilon_X(\infty)|_{sans} = 2.\epsilon_X(\infty)|_{sans} = 2.\frac{V_x.F_u}{\lambda_1.k_1.z}$$

Dans notre cas de figure, nous considérons que l'erreur de modélisation k_1 est négligeable, donc $k_1 \approx 1$ et la distance focale utilisée est de 650 pixels. L'erreur de traînage correspondante est $2 \times (\frac{0.05 \times 650}{0.5 \times 0.675}) = 192$ pixels.

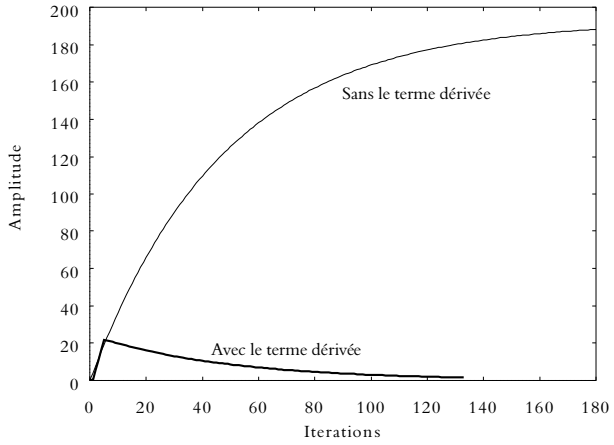


FIG. 2.27 – *Evolution de l'erreur avec et sans terme dérivée.*

Nous retrouvons bien cette valeur sur la figure 2.27 représentant l'évolution de l'erreur de traînage avec et sans terme dérivé. En effet, lorsque la loi de commande utilise le terme dérivée alors l'erreur de traînage décroît exponentiellement vers zéro. Notons là encore le pic initial dû à l'inertie du système (retard pur) et à l'initialisation du calcul du terme dérivée.

2.4.3 Résultats sur site expérimental

Dans ce paragraphe nous présentons quelques résultats obtenus sur notre plate-forme expérimentale. Durant ces manipulations "en réel", nous avons utilisé un cube respectant les cotes de la cible de simulation. Les amers sont constitués par de petites lampes à incandescence permettant une bonne détection quelque soit la direction selon laquelle est vu le cube. La caméra CCD fixée sur l'effecteur du robot perçoit ainsi une image telle que le montre la figure 2.28.

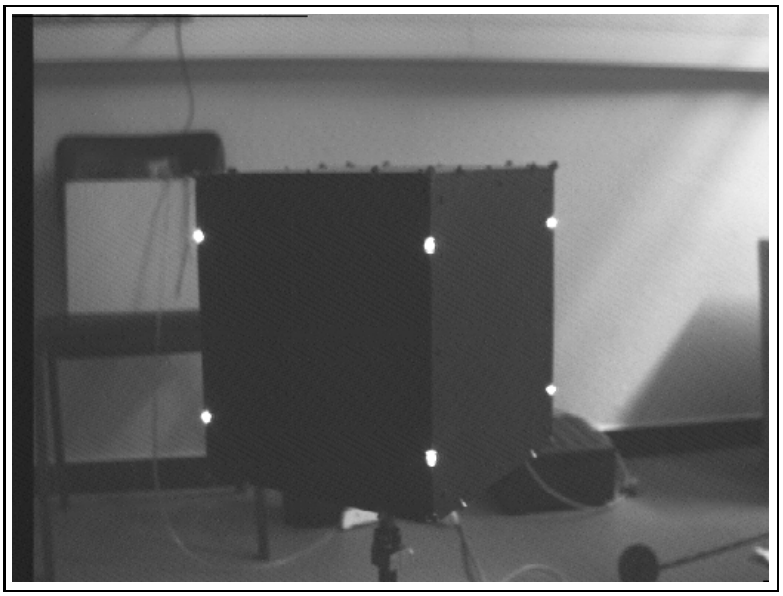


Fig. 2.28 – *Vue du cube par la caméra*

2.4.3.1 Hélicoïde parallèle à une face

Le principe de cette expérimentation est de mettre simultanément à contribution tous les axes de translation de la caméra afin d'étudier le comportement de notre loi de commande dans un cas plus complexe que le simple mouvement rectiligne. Pour cette raison, nous avons choisi d'effectuer une trajectoire helicoïdale dont l'axe est perpendiculaire à une face du cube. La figure 2.29 présente l'expérimentation envisagée.

La trajectoire réalisée ici, se décompose en deux mouvements principaux : une translation suivant l'axe optique (z), et une translation circulaire appliquée aux deux autres axes (x et y). Ainsi, en se plaçant dans le plan XoY face au cube, la caméra décrit un cercle de rayon R avec une vitesse de pulsation α. Les vitesses appliquées à l'effecteur sont alors du type:

$$\begin{cases} V_x = -\alpha \mathrm{R} \sin(\alpha t) \\ V_y = \alpha \mathrm{R} \cos(\alpha t) \\ V_z = \mathrm{V} \end{cases}$$

Dans ce cas de figure (mouvement parallèle à un plan), aucune vitesse de rotation n'est appliquée à l'effecteur, par conséquent la relation 2.29 se simplifie en posant $\omega = 0$ et l'on retrouve la relation 2.34 soit :

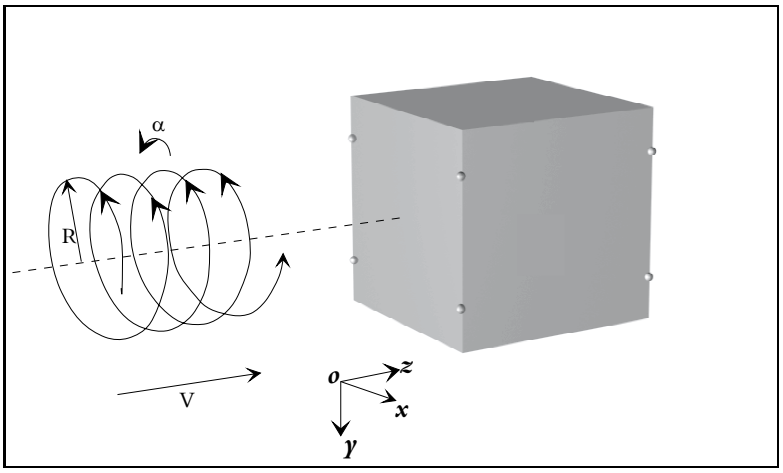


FIG. 2.29 – *Trajectoire helicoïdale*

$$\underline{p}(t) = \underline{p}_0(t) - \int_{\beta=t_o}^{\beta=t} \underline{V}(\beta) d\beta$$

En reprenant la démarche utilisée précédemment, cette expression se développe suivant chaque axe:

$$\begin{cases} x_i(t+\Delta t) = x(t) - \mathrm{R}\cos(\alpha t)\Delta t \\ y_i(t+\Delta t) = y(t) - \mathrm{R}\sin(\alpha t)\Delta t \\ z_i(t+\Delta t) = z(t) - \mathrm{v}\Delta t \end{cases}$$

et la projection perspective appliquée sur chacun des points nous donne les motifs de référence:

$$X_i^*(t) = \frac{x_i(t) - \mathrm{R}\sin(\alpha t)\Delta t}{z_i(t) - \mathrm{v}\Delta t}.F_u$$
$$Y_i^*(t) = \frac{y_i(t) - \mathrm{R}\cos(\alpha t)\Delta t}{z_i(t) - \mathrm{v}\Delta t}.F_v$$

Les conditions que nous avons choisies pour cette expérimentation sont :

- Rayon de l'hélicoïde $R = 5cm$
- Translation suivant l'axe optique $V = 20mm.s^{-1}$
- Pulsation $\alpha = \frac{4\pi}{5} rad.s^{-1}$

– Gain de la boucle de commande $\lambda = 0.2s^{-1}$

La figure 2.30 présente la trajectoire effectuée sous trois angles de vue.

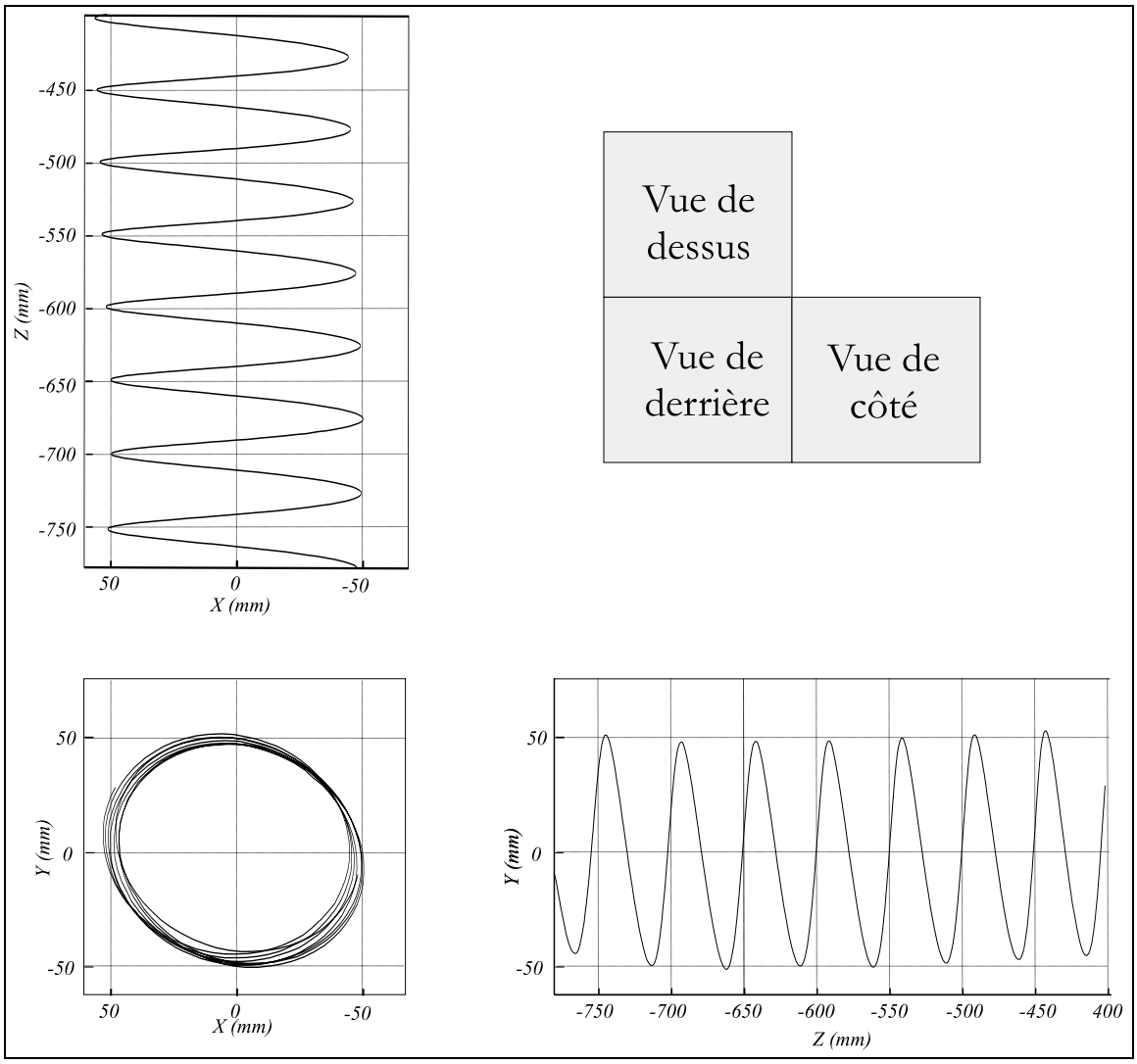
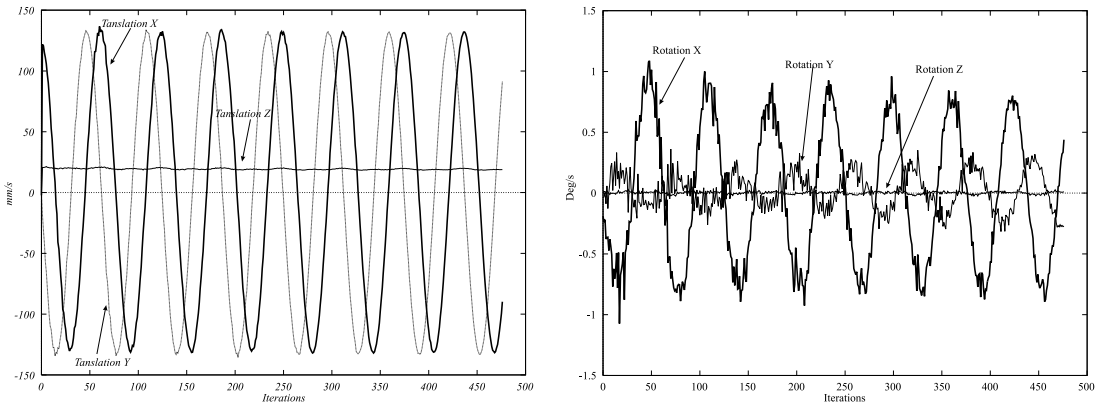


Fig. 2.30 – *Vues de la trajectoire*

A la vue des ces tracés, nous pouvons constater que la trajectoire subit une légère déformation probablement due au fait que toutes les expérimentations ont été réalisées sans calibrage particulier. En effet, la matrice de passage "bras-œil" n'est que grossièrement estimée et seules les focales de la caméra sont prises en compte.

Ce manque de calibrage semble être confirmé par les courbes de vitesse qui démontre un léger couplage entre les vitesses de translation et les vitesses de rotation.



(a) (b)

FIG. 2.31 – *Vitesses de translation et de rotation de l'effecteur*

On note toutefois que sur la courbe 2.31b, l'amplitude de la rotation suivant X est nettement plus importante que celle en Y. Ce comportement dissymétrique est dû en réalité aux distances focales qui sont différentes suivant chaque axe (pour notre caméra, $\frac{F_v}{F_u}$=1.5).

2.4.3.2 Contournement du cube

Dans cette seconde expérimentation, nous proposons d'enchaîner des tâches de suivi afin de réaliser le contournement du cube. Ces différentes tâches élémentaires sont des translations parallèles à chacune des faces et une rotation autour de l'arête commune à ces deux faces. La trajectoire réalisée est représentée sur la figure 2.32.

Ce type de trajectoire appelle toutefois à deux réflexions fondamentales :

- Comment lier les portions de trajectoires?
- Quels critères utilisés pour valider un changement de trajectoires?

La première de ces réflexions nous amène au concept d'enchaînement de tâches référencées vision. Ce problème a initialement été abordé par R. Pissard Gibolet dans le cadre de sa thèse (Pissard-Gibollet, 1993). Son but était de commander un robot mobile dans une pièce en utilisant les murs et les coins comme primitives visuelles. Le passage d'une portion de mur à une autre oblige alors à commuter les tâches et à considérer la tâche résultante comme un mélange de deux tâches (l'une se finissant et l'autre débutant). Le formalisme utilisé tient compte de plusieurs cas de figure en fonction de la dimension des signaux capteur à mélanger et des sous-espaces

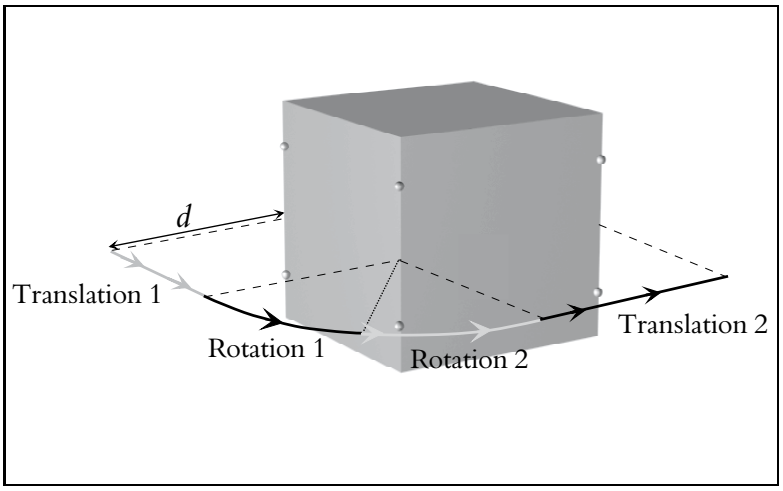


Fig. 2.32 – *Trajectoire choisie pour le contournement.*

engendrés. Ainsi le problème consiste à effectuer un changement de paramètrage ou de combinaison de signaux pour deux tâches de même liaison virtuelle.

Dans le cas général où les dimensions entre la tâche (n) et $(n+1)$ sont différentes, les matrices d'interaction et de combinaison sont calculées de façon à prendre en compte simultanément les 2 tâches. La loi de commande doit alors réguler une fonction de tâche du type:

$$\underline{e}(\underline{r},t) = \gamma(t)\underline{e}_1(\underline{r},t) + (1-\gamma(t))\underline{e}_2(\underline{r},t)$$

où $\gamma(t)$ représente une fonction de pondération évoluant au cours de la tâche et où $\underline{e}_1$ et $\underline{e}_2$ décrivent les deux tâches référencées à lier. Cette approche est relativement élégante mais présente cependant un inconvénient lorsque les tâches $\underline{e}_1$ et $\underline{e}_2$ ne sont pas de la même dimension. En effet, si dans certaines configurations la tâche de liaison peut se définir sans trop de difficultés, en revanche il existe des cas où le calcul de la matrice d'interaction est loin d'être trivial. De plus, dans le cas où les tâches à commuter sont de même dimension, il convient aussi que ces dernières soient de même type. En effet, le mélange d'une tâche basée sur trois droites non parallèles (dimension $3 \times 2 = 6$) avec une tâche basée sur une ellipse et l'abscisse d'un point (dimension $5 + 1 = 6$) demande un bon ajusement des matrices d'interaction. Pour cette raison, nous avons choisi d'éviter ce problème lié aux dimensions des vecteurs de commande en effectuant simplement le mélange des torseurs cinématiques résultants. L'étape de transition n'est alors plus une tâche de transition, mais seulement un ajustement des vitesses entre les deux tâches. Cette approche si elle est nettement moins élégante que la précédente, présente toutefois l'avantage de fonctionner quelque soit la dimension des signaux capteurs. La transition se fait alors par la

relation:

$$T = \gamma(t)T_1 + (1 - \gamma(t))T_2$$

où T_1 et T_2 représentent respectivement les torseurs cinématiques résultant des tâches 1 et 2. La fonction de pondération $\gamma(t)$ que nous avons choisie est une sigmoïde garantissant un passage "en douceur" d'une phase à l'autre.

La seconde remarque aborde le problème du critère de déclenchement de la transition. A partir de quelle mesure peut-on élaborer un critère permettant d'effectuer une transition? Les différentes grandeurs dont nous disposons peuvent être des mesures faites dans l'image et des mesures de positions calculées par intégration du torseur cinématique. Ce dernier cas a été volontairement exclu pour deux raisons évidentes: tout d'abord l'intégration du torseur cinématique ne peut pas être représentative de la position puisque à chaque itération nous intégrons une erreur et en second lieu, notre choix était de réaliser une commande exclusivement dans l'espace image. Par conséquent, les mesures permettant le déclenchement des phases de transition sont effectuées dans l'espace image.

Chaque portion de trajectoire correspond alors à une tâche paramétrée par :

- Le type de trajectoire (translation, rotation).
- Les primitives à utiliser pour la trajectoire.
- La trajectoire suivante.
- Les critères de transition.

Le suivi de trajectoire peut alors être représenté comme une mise en parallèle d'une boucle de commande référencée vision assurant le suivi de la trajectoire et d'un automate à états finis gérant la succession des portions de trajectoires et "alimentant" la boucle de commande en primitives visuelles (Fig. 2.33).

Dans notre cas, la trajectoire a été décomposée en quatre parties correspondant aux deux translations suivant chaque face et aux deux demi-rotations permettant de tourner autour de l'arête. La décomposition de la partie circulaire en deux demi-rotations permet d'effectuer complètement (90^o) cette portion de trajectoire sans avoir de problèmes de discontinuité aux extrémités (disparition d'un amer).

Pour notre expérimentation, les différents paramètres de la trajectoire ont été exprimés en fonction des 6 amers fixés sur le cube. Ces différentes caractéristiques sont résumées dans le tableau 2.1

7. Distance amer supérieur gauche/coin supérieur gauche -Distance amer supérieur droit/coin supérieur droit inférieure à 5 pixels

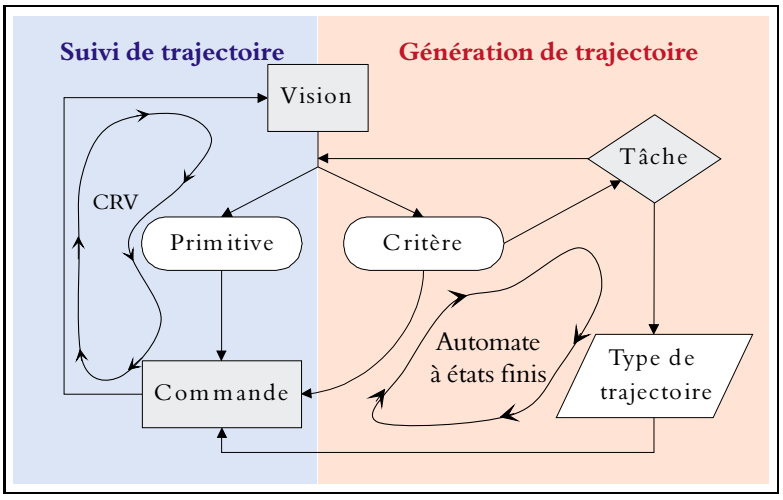


Fig. 2.33 – *Enchaînement de trajectoires*

La génération des motifs de référence se fait de la même manière que dans les cas simulés. Les caractéristiques des trajectoires sont:
Translation:

- Vitesse : $V_x = 0.02m.s^{-1}$
- Distance caméra/face : $d = 0.7m$
- Gain : $\lambda = 0.2s^{-1}$

Rotation:

- Vitesse : $\omega = 0.028rad.s^{-1}$
- Rayon : $\rho = 0.7m$
- Gain : $\lambda = 0.2s^{-1}$

La largeur de la sigmoïde a été choisie à 40 itérations. Comme dans les précédentes expérimentations, une tâche préliminaire de positionnement permet de placer la caméra sur la trajectoire.

La vue de dessus, présentée sur la figure 2.34, nous permet de constater que la trajectoire est bien réalisée. Notons toutefois une légère déformation durant les phases de translation que l'on retrouve sur les courbes de vitesses (Fig. 2.35).

En effet, on note durant les translations un couplage de l'axe de rotation suivant y avec la translation en x. Nous pensons que ce phénomène est dû (comme dans le cas

Tâche	Nombre d'amers visibles	Type de trajectoire	Primitives utilisées pour la commande	Critère de transition
1	4 à 6	Translation	Les 4 amers les plus à gauche	L'abscisse la plus à gauche<20
2	6	Rotation	Les 4 amers les plus à gauche	Symétrie du motif[7]
3	6	Rotation	Les 4 amers les plus à droite	Les ordonnées de droite égale à celles de gauche
4	4 à 6	Translation	Les 4 amers les plus à droite	L'abscisse la plus à droite centrée

TAB. 2.1 – *Paramètres des différentes portions de trajectoires*

précédent) au manque de calibrage de notre système d'expérimentation. Remarquons toutefois que la trajectoire circulaire est très bien assurée à la vue des courbes de vitesses. En effet, le couplage des axes de rotation et de translation est parfait et la complémentarité de ces deux mouvements nous permet de réaliser un bon suivi de trajectoire. Nous pouvons facilement vérifier la constance du rayon de courbure, en regardant les variations cinématiques sur chaque axe durant cette portion de trajectoire. On retrouve bien un rayon $r = \frac{V}{\omega} = \frac{0.02}{0.028} = 0.7$[8].

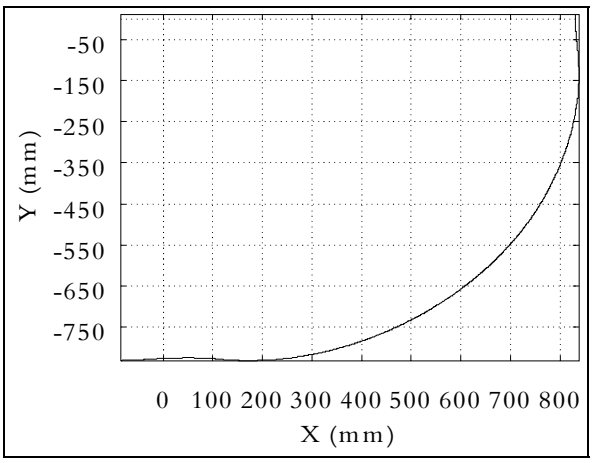


FIG. 2.34 – *Vue supérieure de la trajectoire*

8. Il est à noter que sur la figure 2.34, les mesures ont été effectuées dans le repère centré sur le cube.

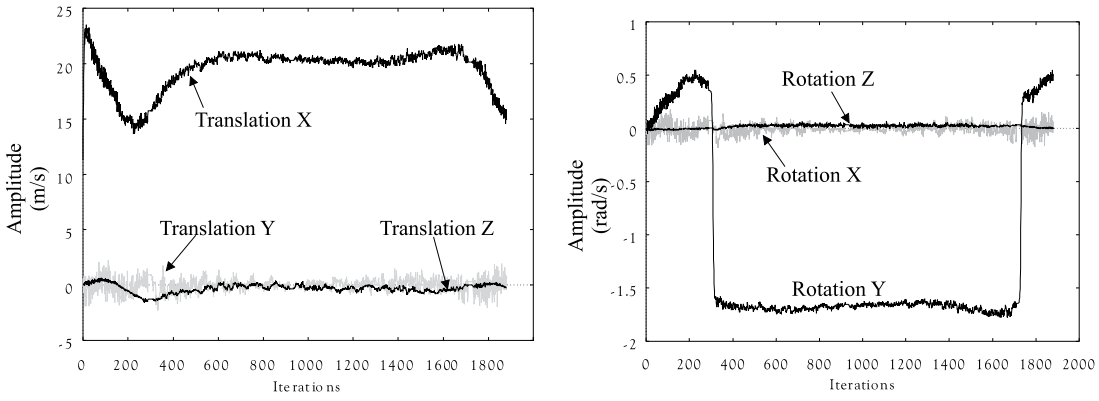


FIG. 2.35 – *Vitesses de translation et de rotation de l'effecteur*

2.5 Conclusion

Dans ce chapitre, nous nous sommes intéressés à la génération et au suivi de trajectoires à partir d'une boucle d'asservissement visuel. L'approche que nous avons choisie est originale dans le sens où elle permet de générer un mouvement de caméra à partir du contrôle d'une trajectoire dans l'espace image.

La commande que nous avons utilisée est une loi de type référencée vision, dans laquelle nous avons rajouté un terme permettant de prendre en compte le caractère dynamique de la consigne visuelle. Ce terme construit à partir de la dérivée de la consigne visuelle effectue une correction sur l'erreur de traînage en permettant ainsi un bon suivi des motifs dans l'image.

Nous avons proposé une méthode permettant de calculer les motifs visuels en fonction des mouvements que l'on désire appliquer à la caméra. Cette méthode a été mise en œuvre durant des simulations sur stations de travail, mais aussi sur site réel. Les résultats obtenus sont satisfaisants même s'ils demandent des approfondissements en particulier au niveau du calibrage du système d'expérimentation.

Dans le chapitre suivant, nous nous intéressons au problème du contournement d'objet inconnu. Nous présenterons entre autres les différentes primitives visuelles utilisables et les tâches robotiques que l'on peut définir à partir de celles-ci.

Chapitre 3

Navigation autour d'objets complexes

3.1 Introduction

AMENER *l'effecteur d'un robot en face d'un carré blanc sur un fond noir.*
Et si pour une pointe de fantaisie (ou par simple curiosité scientifique!), un "non-initié" demande si ça marche aussi avec un cercle, un paquet de cigarettes ou la photo d'Elvis Presley, alors il est souvent difficile d'expliquer que l'algorithme d'asservissement visuel a été calculé pour une forme carrée, que la mesure dans l'image est grandement facilitée par de forts contrastes de luminosité, etc...
Et à la fin de toutes ces explications, l'expérience qui paraissait tout auréolée de mystère et de science en est alors réduite à des réflexions du type *"Ça marche même plus, si on remplace le carré par un cercle!"*
Cette considération bien naturelle du néophyte s'explique facilement. En effet, pour le commun des mortels, contourner un obstacle tel qu'une chaise au milieu de son chemin paraît normal. Cette réaction ne demande pas à l'intéressé d'avoir en tête une modélisation exacte de l'objet pour l'éviter ou de savoir quelle distance il doit maintenir entre l'objet et lui-même pour passer son chemin sans encombre.

Dans le cadre de tâches de navigation sur des objets inconnus et non modélisés, le problème change totalement de point de vue. Comme nous l'avons remarqué dans les chapitres précédents, une commande de type *asservissement visuel* nécessite une formalisation, en termes géométriques ou dynamiques, de l'information visuelle perçue. Cette étape de traduction permet alors à la loi de commande d'utiliser les données issues de la caméra afin de réaliser la tâche envisagée. Cependant lorsque l'objet observé ne répond plus à un signalement précis, tel deux droites, un cercle ou bien encore quatre points judicieusement positionnés, le problème d'asservisse-

ment prend une toute autre dimension. En effet, la nécessité de traduire l'image en termes géométriques (coordonnées de points, longueurs de segments,...) peut se faire de plusieurs manières. Comme nous l'avons présenté dans le premier chapitre, les méthodes de reconstruction à partir du mouvement peuvent fournir une carte des coordonnées de la scène observée. Toutefois ces méthodes sont basées sur des mesures de mouvements ou des procédés itératifs, les rendant totalement incompatibles avec les fréquences "temps réel" utilisées en robotique. Une autre approche consiste à modéliser la scène à partir de la connaissance des objets observés. Cependant, la nécessité de connaître exactement les objets sur lesquels on réalise la tâche robotique est fortement contraignante et réduit singulièrement le nombre des applications potentielles.

Dans notre approche, notre ambition est de pouvoir réaliser un asservissement visuel en temps réel vidéo sur des objets dont la forme est inconnue et dont aucune modélisation n'est donnée (Berry, Martinet and Gallice, 1998). Pour cette raison, nous allons tout d'abord nous intéresser aux différentes propriétés géométriques d'image d'objets. Nous proposerons ensuite pour chacune d'elles une loi de commande permettant de mettre en évidence les mouvements autorisés par celles-ci. Enfin dans un dernier paragraphe, nous évoquerons les problèmes dûs à la mise en œuvre réelle des manipulations ainsi que les conditions dans lesquelles nous avons réalisé nos expérimentations. Dans une dernière partie, différents résultats expérimentaux seront présentés.

3.2 Asservissement visuel sur un objet quelconque

En mécanique, il est classique de paramétrer les objets afin de pouvoir appliquer les différentes lois régissant la cinématique et la dynamique des corps rigides. Dans le cadre des corps statiques, le paramètrage que l'on utilise traditionnellement permet d'obtenir une description de l'objet sous la forme d'une série de coefficients tels que sa masse, sa longueur, sa largeur, sa forme, sa position, ... Ces différentes données nous permettent alors de répondre à des questions tant qualitatives que quantitatives comme :
" Est-ce que l'objet est gros ?"
"Est-ce qu'il est long ou large ? "
"Comment est-il posé ?", etc...
Ainsi, qu'un pot de fleurs soit rouge ou bleu ne change en aucun cas ses propriétés mécaniques et une personne désireuse de s'en saisir prendra avant tout en compte sa taille et sa position plutôt que sa couleur. Seuls les reflets et les ombrages à la surface de l'objet peuvent permettre de se faire une idée sur les formes du vase (Sont-elles plutôt cylindriques ou cubiques, ...).

Dans notre approche, nous avons volontairement évité de prendre en compte les paramètres tels que la texture ou les ombres portées. En effet ces mesures sont relativement sensibles aux conditions d'éclairement et imposent la plupart du temps des hypothèses simplificatrices (surface lambertienne, luminosité contrainte, ...) qui ne sont pas réalistes dans le cadre de la robotique. Pour cette raison, nous avons choisi de n'utiliser que les propriétés géométriques des objets. Nous allons à présent en établir un rapide panorama, en explicitant pour chacune d'entre elles, leurs différentes caractéristiques ainsi que la matrice jacobienne d'image associée.

3.2.1 Modélisation

Lorsque l'on se trouve en face d'un objet, une simple observation de celui-ci peut nous donner trois types d'informations nous permettant alors d'agir. Il s'agit de sa **position**, de son **orientation** et de sa **taille**. A chacune de ces caractéristiques physiques, on peut associer des primitives géométriques qui nous permettent d'agir directement sur l'objet. Rappelons cependant que le seul organe de perception dont nous disposons est une caméra CCD monochrome fixée sur l'effecteur du manipulateur. Ainsi toutes les mesures sur l'objet seront faites à partir d'une image de celui ci. De plus, comme nous l'avons évoqué précédemment, nous ne nous intéressons pas aux variations photogramétriques dans l'image, ni aux différentes textures de l'objet. Par conséquent, l'image de l'objet sera réduite à une image binaire définie telle que:

$$\begin{cases} P_{i,j} = 1 & si\ P_{i,j} \in objet \\ P_{i,j} = 0 & si\ P_{i,j} \notin objet \end{cases}$$

où $P_{i,j}$ correspond au pixel de la colonne i et de la ligne j.

Les détails sur la segmentation et les conditions de prises de vue sont explicités au paragraphe (3.2.3).

3.2.1.1 Position de l'objet

Dans ce paragraphe, nous nous attachons à quantifier la position de l'objet dans le plan image. Par position, nous entendons l'endroit où est disposé l'objet dans l'image mais en aucun cas, son orientation qui fait l'objet du paragraphe 3.2.1.3.

3.2.1.1.1 Centre de gravité et centroïde A la vue de l'image binarisée d'un objet quelconque, comment peut-on quantifier sa position? Une réponse naturelle et souvent utilisée consiste à ramener simplement la position de l'objet dans l'image en la position d'un point caractéristique. Ce point peut être choisi de différentes manières. Dans de nombreux cas de figure, le centre de gravité paraît un choix judicieux. Cependant ce dernier fait intervenir la notion de répartition de masse à travers l'objet. Aussi dans certain cas, il peut être intéressant de définir l'équivalent du centre de gravité en ne considérant non plus la masse, mais le volume occupé par le corps. On parle alors de centroïde. Ainsi dans le cas d'un objet homogène le centroïde est confondu avec le centre de gravité. Dans notre cas, sachant que l'on travaille sur une image de l'objet il paraît évident d'utiliser le centroïde de l'objet observé.

Ce dernier se calcule de la manière suivante :

$$x_c = \frac{1}{A}\sum_i \sum_j i.P(i,j) \qquad y_c = \frac{1}{A}\sum_i \sum_j j.P(i,j) \tag{3.1}$$

où A représente l'aire occupée par la forme dans l'image, soit

$$A = \sum_i \sum_j P(i,j) \tag{3.2}$$

Remarque: Notons que dans la relation précédente, les termes $\sum_i \sum_j i.P(i,j)$ et $\sum_i \sum_j j.P(i,j)$ sont nommés moment d'ordre un de l'objet respectivement par rapport à l'axe y et x et sont notés Q_y et Q_x.

3.2.1.1.2 Autres points caractéristiques Nous allons à présent chercher à savoir si le centroïde est vraiment la meilleure primitive permettant de caractériser la position de l'objet dans l'image. Ce que l'on entend par positionnement de l'objet revient en réalité à positionner la surface occupée par l'objet dans l'image. Toutefois, cette surface (Eq. 3.2) n'est pas toujours représentative de l'objet observé. Pour cela, prenons plusieurs cas de figure représentés sur le schéma (3.1).

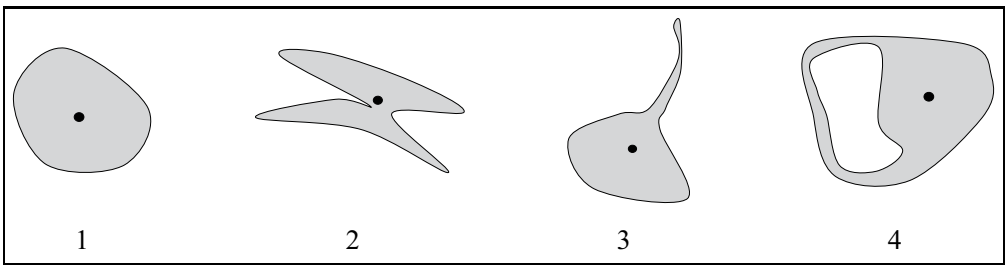


Fig. 3.1 – *Centroïde de plusieurs objets binarisés.*

L'objet 1 est représenté par une forme convexe de répartition uniforme, assurant une bonne symétrie par rapport au centroïde (représenté par le point noir). Le positionnement du centroïde entraînera ici le positionnement de l'objet. Dans le second cas de figure, l'objet n'est plus convexe mais conserve une certaine symétrie. Là encore le centroïde permet une assez bonne représentation de la position de l'objet dans l'image. A contrario, l'objet 3 présente une forte dissymétrie et la relation (3.1) ne prend que faiblement en compte la partie supérieure. Une tâche de centrage pourrait alors exclure une partie de l'objet (Fig. 3.2) et les différents mouvements exécutés autour de la cible se trouveraient totalement biaisés.

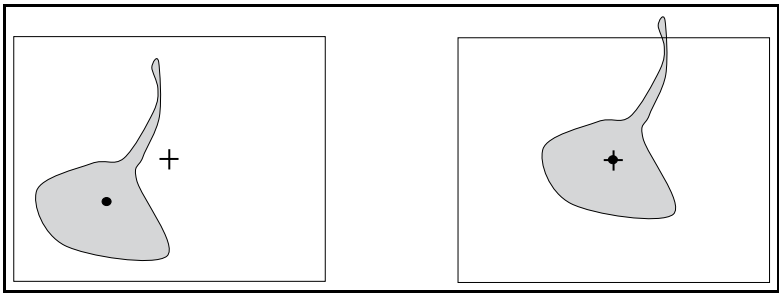


Fig. 3.2 – *Centrage d'un objet dissymétrique*

La quatrième forme pose le même problème, bien que cela se manifeste de façon différente. En effet dans ce dernier cas l'objet présente une forme presque symétrique,

mais n'est pas uniforme. On se retrouve alors dans le cas d'un décentrage pouvant amener le même type de problèmes que le cas précédent.

Afin de palier ce type de problèmes, une méthode consiste à "convexifier" la forme ou à l'uniformiser (Fig. 3.3)

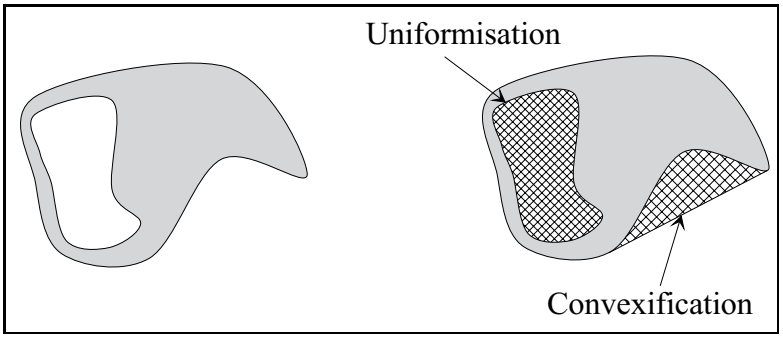


FIG. 3.3 – *"Convexification" et uniformisation d'une forme.*

Cependant si l'uniformisation est relativement simple à exécuter, la "convexification" nécessite une connaissance topologique de la forme qui n'est pas toujours triviale à obtenir.

Pour notre part, nous avons choisi de travailler non plus directement sur l'objet dans l'image mais sur un rectangle englobant l'objet dans l'image. Cette approche présente à notre avis plusieurs avantages. Tout d'abord, le calcul du rectangle englobant est relativement simple puisqu'il nécessite seulement les coordonnées extrêmes de l'objet (MinMax en x et en y). De plus, le positionnement se fait dans l'image qui est de forme rectangulaire, ce qui revient à positionner un rectangle dans un autre rectangle; on évite ainsi les problèmes de sortie de champs évoqués sur la figure 3.2. Enfin il n'y a pas de prépondérance donnée à certaine partie de l'objet dont la surface est plus importante (Fig. 3.4).

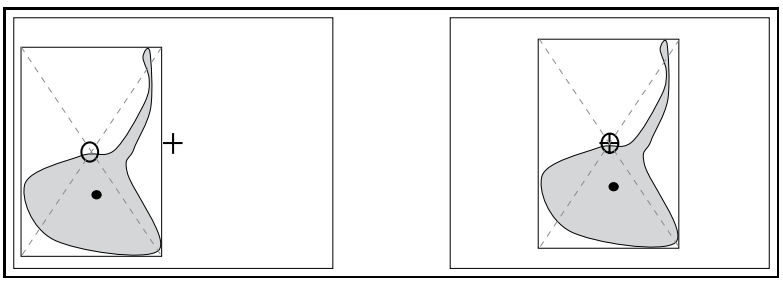


FIG. 3.4 – *Centrage d'une forme dissymétrique à partir d'un rectangle englobant*

Enfin comme nous le verrons dans la suite, l'intérêt d'utiliser un rectangle englobant l'objet nous permet de savoir exactement dans quelle partie de l'image est située la cible et quelle dimension elle occupe.

3.2.1.1.3 Etude de l'interaction A l'aide du rectangle englobant l'objet, nous allons chercher à déplacer le robot de façon à centrer l'image de l'objet dans le plan image. Pour cela, il est nécessaire d'établir une relation d'interaction entre le motif image (dans notre cas le centre du rectangle englobant l'objet dans l'image) et les mouvements du robot. Dans le cas présent, la relation d'interaction mise en œuvre pour cette tâche de positionnement se ramène à celle d'un point m ayant pour coordonnées $\underline{x} = (x\ y\ z)^T$.

Ce point m se projette dans le plan image par projection perspective et les coordonnées image $\underline{X} = (X\ Y)^T$ du point projeté M s'exprime par

$$X = \frac{x}{z} \qquad\qquad Y = \frac{y}{z} \tag{3.3}$$

En différentiant l'équation 3.3 par rapport au temps, et en utilisant la relation cinématique

$$\frac{d\underline{x}}{dt} = -V - \Omega \wedge \underline{x} \tag{3.4}$$

qui exprime la variation des coordonnées de m exprimée dans le repère caméra et en fonction de la vitesse (V, Ω) de la caméra, on peut alors expliciter la matrice d'interaction suivante (avec une focale unité):

$$\begin{pmatrix} \dot{X} \\ \dot{Y} \end{pmatrix} = \begin{pmatrix} \frac{-1}{z} & 0 & \frac{X}{z} & XY & -1-X^2 & Y \\ 0 & \frac{-1}{z} & \frac{Y}{z} & 1+Y^2 & -XY & -X \end{pmatrix} \cdot \begin{pmatrix} V_x \\ V_y \\ V_z \\ \Omega_x \\ \Omega_y \\ \Omega_z \end{pmatrix} \tag{3.5}$$

A l'aide de cette relation, nous décrivons l'interaction entre les mouvements du robot et un point m de l'espace opérationnel ayant comme coordonnées dans le repère caméra $(x\ y\ z)^T$. Toutefois, contrairement à de nombreux cas le point m ne décrit pas une primitive réelle de la scène, telle que le coin d'un objet ou le centre de gravité d'un motif plan.

Afin de qualifier la validité de la relation 3.5, cherchons à quel motif de la scène correspond le point m. Le rectangle encadrant l'objet dans l'image est la projection d'une boîte de forme pyramidale à base rectangulaire dont le sommet principal est

le centre optique de la caméra et dont les dimensions sont telles que l'objet observé soit inscrit à l'intérieur (Fig. 3.5). Cette pyramide que nous notons P admet une face tronquée parallèle au plan image et est traversée par un axe γ passant par le milieu de cette face et le centre optique. Ainsi le point M est l'image d'un point m de l'axe γ par projection perspective sur le plan image.

La question qui vient alors à l'esprit est: "Où se trouve le point m sur l'axe γ?"

En d'autres termes, où se trouve le point 3D sur lequel nous sommes censés nous asservir? Un premier élément de réponse peut être fourni en considérant m au milieu de la boîte P. Ce choix paraît naturel puisque l'on va ainsi chercher à se rapprocher du centre de gravité de l'objet. Toutefois, comme nous l'avons déjà rappelé, aucune information *a priori* sur l'objet n'est disponible. Par conséquent, il est impossible de réguler un tel point. Une autre approche consiste à considérer le fait que nous utilisons comme information non pas une donnée tridimensionnelle ponctuelle, mais une vue de l'objet à un instant donné. Le point recherché m sera alors le point d'intersection entre l'axe γ et la surface de l'objet vue par la caméra. On remarquera ainsi que la régularité de la surface entraînera la régularité de l'erreur en z. Ainsi pour un objet parfaitement lisse et convexe (la sphère étant le cas parfait), on aura une erreur constante, tandis que pour une forme fortement bosselée et concave l'erreur suivant z pourra admettre de forts gradients. Malgré cette interprétation, comme

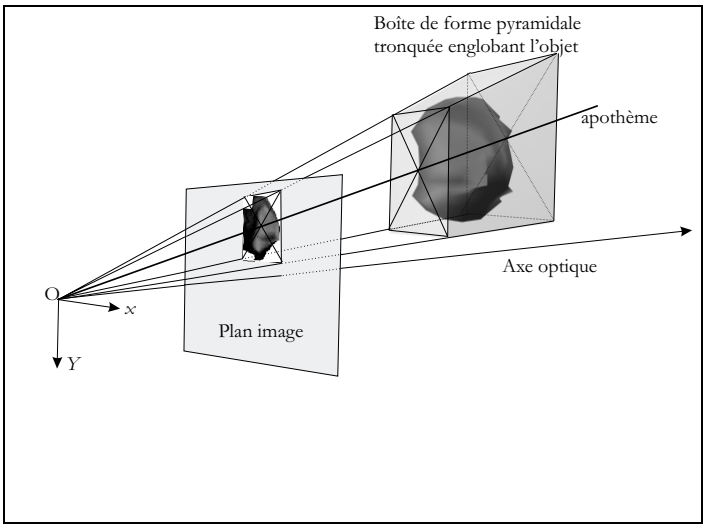


FIG. 3.5 – *Relation entre le centre de gravité image* $(X\ Y)$ *et l'objet réel.*

dans beaucoup de cas en asservissement visuel 2D, la quantité z n'est pas explicitement accessible et il est ainsi nécessaire de la fixer arbitrairement en fonction des conditions expérimentales (taille de l'objet, régulation de la distance, ...). Toutefois, pour une tâche de centrage pur, il est possible de s'affranchir de la connaissance de z

en ne commandant simplement que les axes de rotations. En effet comme le montre la matrice d'interaction (Rel. 3.5), seules les informations dans l'image soit $(X\ Y)$ sont nécessaires.

3.2.1.2 Dimension de l'objet

3.2.1.2.1 Définition des primitives La dénomination "dimension" est en réalité bien prétentieuse puisque, comme précédemment, les seules mesures que nous pouvons effectuer sont faites dans l'image. Généralement, on définit les dimensions d'un objet dans l'espace cartésien par trois grandeurs: la longueur, la largeur et la hauteur. Dans le cas présent, l'espace de mesure est l'espace image, par conséquent les différentes dimensions que l'on peut acquérir seront uniquement apparentes et totalement dépendantes du point de vue (Fig. 3.6).

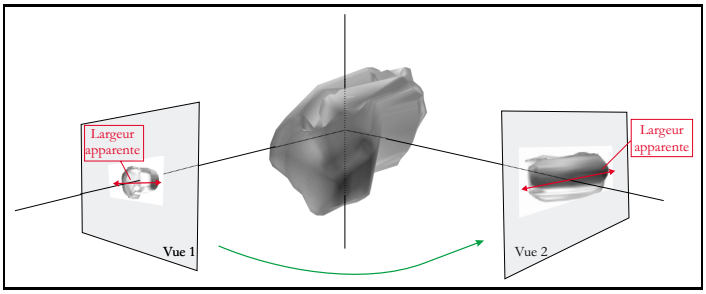


Fig. 3.6 – *Variations de la largeur en fonction du point de vue.*

Dans le paragraphe précédent, nous avons défini une primitive (en l'occurence, le centre du rectangle englobant l'objet dans l'image) pour réguler la position de l'objet dans l'image. Dans ce paragraphe, nous utilisons les dimensions de l'objet dans l'image afin de réguler la distance entre la caméra et la cible. Cette approche provient simplement du fait que la dimension de l'objet dans l'image est inversement proportionnelle à la distance caméra/cible (Projection perspective). Aussi différentes grandeurs peuvent être utilisées pour quantifier la dimension de l'objet:

- L'aire de la forme binaire,
- La hauteur ou la largeur,
- La projection sur un axe particulier,
- Le rayon du cercle inscrit ou du cercle circonscrit,...

Pour notre part, nous avons choisi d'utiliser la hauteur ou la largeur dans les tâches de positionnement et la projection sur un axe particulier dans les tâches de déplacement [1] autour de l'objet.

Intéressons-nous à présent à la projection d'une forme binarisée suivant un axe particulier (Fig. 3.7). La forme générale de la projection d'une forme suivant un axe

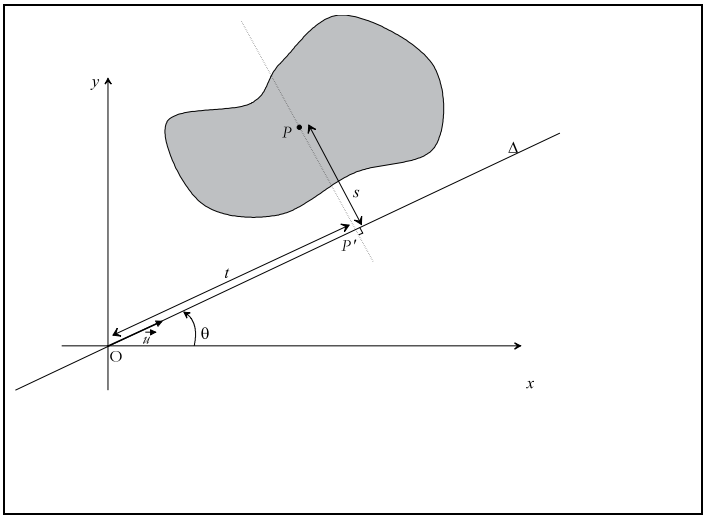


FIG. 3.7 – *Projection d'une forme suivant un axe donné*

Δ faisant un angle θ avec l'axe des abscisses est donnée par:

$$Proj_\theta(t) = \int_\Delta P(t\cos\theta - s\sin\theta, t\sin\theta + s\cos\theta)ds \tag{3.6}$$

où t est la distance suivant l'axe Δ de la projection P' d'un point P sur Δ à l'origine et s est la distance entre le point considéré P et sa projection P'.

Dans notre cas, une approche analytique en utilisant le produit scalaire entre le vecteur directeur unitaire de Δ noté $\overrightarrow{u}$ et le vecteur $\overrightarrow{OP}$ semble plus appropriée. En effet, il vient alors simplement:

$$t = \overrightarrow{u}.\overrightarrow{OP} = x.\cos\theta + y.\sin\theta \tag{3.7}$$

et l'on en déduit facilement les coordonnées du point P' (projeté de P sur Δ) par:

$$\begin{aligned} x_{P'} &= t.\cos\theta = (x.\cos\theta + y.\sin\theta).\cos\theta \\ y_{P'} &= t.\sin\theta = (x.\cos\theta + y.\sin\theta).\sin\theta \end{aligned} \tag{3.8}$$

1. Cf. §3.2.2.1.2

La projection totale de la forme s'effectue alors en réalisant cette transformation pour tous les pixels de la forme binarisée et en sélectionnant les extremums.

Dans les cas particuliers qui nous intéressent pour le positionnement (largeur et hauteur de l'objet) il suffit d'appliquer les relations précédentes avec respectivement θ=0 et θ=$\frac{\pi}{2}$. On obtient alors la projection sur l'axe horizontal par:

$$l(y) = \int_{\Delta} P(x,y)dx \Big|_{\theta=0} = x \tag{3.9}$$

La largeur est alors calculée dans le cas d'une image (n×m) par

$$Larg = Max(l(y)) - Min(l(y)) \qquad y \in [0,n]$$

De la même façon, la hauteur est déduite de la projection suivant l'axe des ordonnées par:

$$h(x) = \int_{\Delta} P(x,y)dy \Big|_{\theta=\frac{\pi}{2}} = y \tag{3.10}$$

et la hauteur est donnée par:

$$Haut = Max(h(x)) - Min(h(x)) \qquad x \in [0,m]$$

3.2.1.2.2 Etude de l'interaction Dans ce paragraphe, nous allons chercher à caractériser les variations des primitives décrites ci-dessus, à savoir la largeur et la hauteur et plus généralement la longeur du segment $\mathcal{L}$ issue de la projection de la forme binaire sur un axe donné Δ (Fig. 3.8).

De façon générale, la projection du motif binaire sur un axe Δ se ramène à la projection d'un segment $\mathcal{S}$ dépendant de la configuration de l'objet. La longueur $L_{\mathcal{S}}$ de ce segment que nous mesurons dans l'image peut être aisément calculée à partir de la projection perspective de ses deux points extrêmes notés a et b soit:

$$\begin{aligned} L_{\mathcal{S}} &= \sqrt{(\tfrac{x_a.f_u}{z_a} - \tfrac{x_b.f_u}{z_b})^2 + (\tfrac{y_a.f_v}{z_a} - \tfrac{y_b.f_v}{z_b})^2} \\ L_{\mathcal{S}} &= \sqrt{(X_a - X_b)^2 + (Y_a - Y_b)^2} \end{aligned} \tag{3.11}$$

en notant α, l'angle que fait $\mathcal{S}$ avec l'axe des abscisses et β, l'angle que fait l'axe Δ avec l'axe des abscisses, nous pouvons alors exprimer $L_{\mathcal{L}}$ en fonction de $L_{\mathcal{S}}$ et l'on obtient:

$$L_{\mathcal{L}} = L_{\mathcal{S}}.cos(\alpha - \beta) \tag{3.12}$$

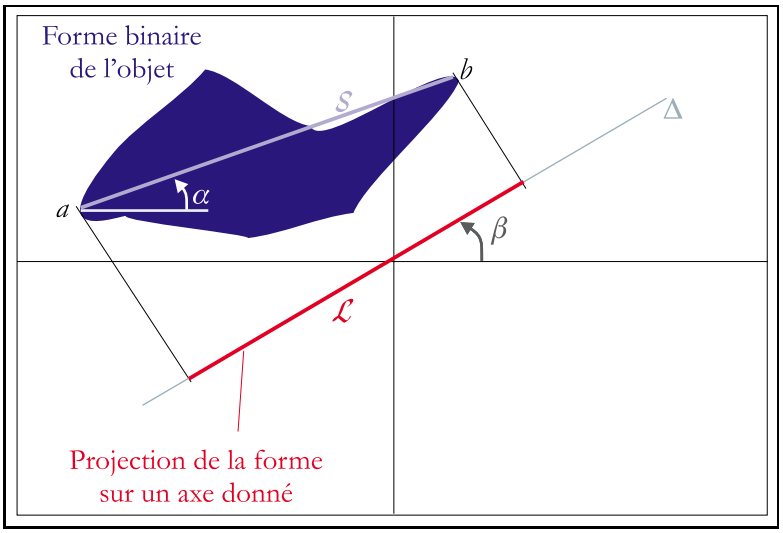


FIG. 3.8 – *Segment $\mathcal{L}$ issu de la projection du motif binaire sur un axe*

avec $\alpha = \arctan\left(\frac{Y_a - Y_b}{X_a - X_b}\right)$.

La matrice d'interaction d'un segment de longueur L_S faisant un angle α avec l'axe des abscisses et positionné en $(X_S \; Y_S)$ dans l'image a déjà été étudiée par F. Chaumette. Nous en rappelons simplement le résultat, en renvoyant le lecteur intéressé pour les détails calculatoires à (Chaumette, 1990). Cette matrice est donnée par:

$$\begin{pmatrix} \dot{X}_S \\ \dot{Y}_S \\ \dot{L}_S \\ \dot{\alpha} \end{pmatrix} = \begin{pmatrix} \begin{bmatrix} -\nu_2 & 0 \\ \nu_2 X_c - \nu_1 L\cos\alpha/4 & X_c Y_c + L^2\cos\alpha\sin\alpha/4 \\ -(1+X_c^2+L^2\cos^2\alpha/4) & Y_c \end{bmatrix} \\ \begin{bmatrix} 0 & -\nu_2 \\ \nu_2 Y_c - \nu_1 L\sin\alpha/4 & 1+Y_c^2+L^2\sin^2\alpha/4 \\ -X_c Y_c - L^2\cos\alpha\sin\alpha/4 & -X_c \end{bmatrix} \\ \begin{bmatrix} \nu_1\cos\alpha & \nu_1\sin\alpha \\ \nu_2 L - \nu_1(X_c\cos\alpha + Y_c\sin\alpha) & L(X_c\cos\alpha\sin\alpha + Y_c(1+sin^2\alpha)) \\ -L(X_c(1+\cos^2\alpha) + Y_c\cos\alpha\sin\alpha) & 0 \end{bmatrix} \\ \begin{bmatrix} -\nu_1\sin\alpha/L & \nu_1\cos\alpha/L \\ \nu_1(X_c\sin\alpha - Y_c\cos\alpha)/L & -X_c\sin^2\alpha + Y_c\cos\alpha\sin\alpha \\ X_c\cos\alpha\sin\alpha - Y_c\cos^2\alpha & -1 \end{bmatrix} \end{pmatrix} \cdot \begin{pmatrix} V_x \\ V_y \\ V_z \\ \Omega_x \\ \Omega_y \\ \Omega_z \end{pmatrix} \tag{3.13}$$

avec $\nu_1 = \frac{z_a - z_b}{z_a z_b}$ et $\nu_2 = \frac{z_a + z_b}{2 z_a z_b}$.

Cherchons à présent à calculer la matrice d'interaction pour la projection du segment $\mathcal{S}$ sur l'axe Δ. Cette transformation surjective est dans notre cas de figure

caractérisée par les relations suivantes:

$$\begin{cases} X_{\mathcal{L}} = (X_{\mathcal{S}}.\cos\beta + Y_{\mathcal{S}}.\sin\beta).\cos\beta \\ Y_{\mathcal{L}} = (X_{\mathcal{S}}.\cos\beta + Y_{\mathcal{S}}.\sin\beta).\sin\beta \\ L_{\mathcal{L}} = L_{\mathcal{S}}.cos(\alpha - \beta) \end{cases} \tag{3.14}$$

En notant le vecteur décrivant $\mathcal{S}$ par $\underline{P_{\mathcal{S}}}$, et celui de $\mathcal{L}$ par $\underline{P_{\mathcal{L}}}$, soit:

$$\underline{P_{\mathcal{S}}} = \begin{pmatrix} X_{\mathcal{S}} \\ Y_{\mathcal{S}} \\ L_{\mathcal{S}} \\ \alpha \end{pmatrix} \quad \text{et} \quad \underline{P_{\mathcal{L}}} = \begin{pmatrix} X_{\mathcal{L}} \\ Y_{\mathcal{L}} \\ L_{\mathcal{L}} \end{pmatrix}$$

on peut alors calculer simplement la matrice d'interaction correspondant a $\underline{P_{\mathcal{L}}}$ à partir de celle calculée pour $\underline{P_{\mathcal{S}}}$ en utilisant le jacobien associé aux relations 3.14. En effet, il vient:

$$\underline{\dot{P}_{\mathcal{L}}} = \frac{\partial \underline{P_{\mathcal{L}}}}{\partial \underline{P_{\mathcal{S}}}}.\underline{\dot{P}_{\mathcal{S}}}$$

avec

$$\frac{\partial \underline{P_{\mathcal{L}}}}{\partial \underline{P_{\mathcal{S}}}} = \begin{pmatrix} \cos^2\beta & \sin\beta\cos\beta & 0 & 0 \\ \sin\beta\cos\beta & \sin^2\beta & 0 & 0 \\ 0 & 0 & \cos(\beta-\alpha) & L_{\mathcal{S}}\sin(\beta-\alpha) \end{pmatrix}$$

et la matrice d'interaction correspondant à la primitive $\mathcal{L}$ est donnée par:

$$M_{\mathcal{L}}^T = \frac{\partial \underline{P_{\mathcal{L}}}}{\partial \underline{P_{\mathcal{S}}}}.M_{\mathcal{S}}^T$$

où $M_{\mathcal{L}}^T$ et $M_{\mathcal{S}}^T$ représentent respectivement les matrices d'interaction associées à $\mathcal{L}$ et $\mathcal{S}$.

Dans la matrice $M_{\mathcal{S}}^T$, seule la sous-matrice correspondant à la longueur de la projection sur l'axe Δ nous intéresse. Nous pouvons la réécrire en fonction de la longueur projetée $L_{\mathcal{L}}$ et il vient:

$$\begin{aligned}
M_{L_{\mathcal{L}}}^T[1,1] &= \nu_1.\cos\beta \\
M_{L_{\mathcal{L}}}^T[1,2] &= \nu_1.\sin\beta \\
M_{L_{\mathcal{L}}}^T[1,3] &= \nu_2.L_{\mathcal{L}} - \nu_1.X_{\mathcal{S}}.\cos\beta - \nu_1.Y_{\mathcal{S}}.\sin\beta \\
M_{L_{\mathcal{L}}}^T[1,4] &= L_{\mathcal{L}}(X_{\mathcal{S}}.\cos\alpha.\sin\alpha + Y_{\mathcal{S}}(1+\sin^2\alpha) + \tan(\beta-\alpha).(-X_{\mathcal{S}}.\sin^2\alpha + Y_{\mathcal{S}}.\cos\alpha.\sin\alpha)) \\
M_{L_{\mathcal{L}}}^T[1,5] &= -L_{\mathcal{L}}(Y_{\mathcal{S}}.\cos\alpha.\sin\alpha + \tan(\beta-\alpha).(-X_{\mathcal{S}}.\cos\alpha.\sin\alpha + Y_{\mathcal{S}}.\cos^2\alpha) + X_{\mathcal{S}}(\cos^2\alpha+1)) \\
M_{L_{\mathcal{L}}}^T[1,6] &= -L_{\mathcal{L}}.\tan(\beta-\alpha)
\end{aligned} \tag{3.15}$$

A partir de celle-ci il est alors facile de déduire les matrices correspondant à la largeur et à la hauteur de l'objet comme elles ont été définies précédemment. On obtient en effet pour

la largeur ($\beta = 0$),

$$M_{Larg}^T = \begin{bmatrix} \nu_1 & 0 & \nu_2.L_{\mathcal{L}} - \nu_1.X_{\mathcal{S}} \\ L_{\mathcal{L}}(Y_{\mathcal{S}} + X_{\mathcal{S}}.\tan\alpha) & -2L_{\mathcal{L}}X_{\mathcal{S}} & L_{\mathcal{L}}.\tan\alpha \end{bmatrix} \tag{3.16}$$

et pour la hauteur ($\beta = \frac{\pi}{2}$),

$$M_{Haut}^T = \begin{bmatrix} 0 & \nu_1 & \nu_2.L_{\mathcal{L}} - \nu_1.Y_{\mathcal{S}} \\ 2L_{\mathcal{L}}.Y_{\mathcal{S}} & -L_{\mathcal{L}}.(Y_{\mathcal{S}}.\cot\alpha + X_{\mathcal{S}}) & -L_{\mathcal{L}}.\cot\alpha) \end{bmatrix} \tag{3.17}$$

Nous étudierons dans la prochaine partie les différentes approximations pouvant être faites sur ces matrices, et en particulier les quantités ν_1 et ν_2 qui prennent en compte la profondeur des points extrèmes de l'objet.

3.2.1.3 Orientation de l'objet

3.2.1.3.1 Les moments d'inertie Dans les premiers paragraphes de ce chapitre nous avons considéré la position et la dimension de l'objet. Nous nous proposons d'étudier maintenant un troisième paramètre : l'orientation.

En mécanique classique l'étude de la répartition de masse dans un corps peut être faite à partir des moments d'inertie. Dans le cadre de la vision artificielle de nombreux travaux se sont basés sur le calcul des moments, en particulier dans le domaine de la reconnaissance des formes. Le lecteur intéressé trouvera une bibliographie relativement complète dans l'article de (Prokop and Reeves, 1992) et plus récemment dans celui de (Mukundan and Ramakrishnan, 1995).

Pour notre part, nous avons choisi de travailler avec les moments cartésiens d'ordre $(n + m)$ que l'on définit généralement de la manière suivante:

$$I_{nm} = \int\int_{aire} x^n.y^m.dxdy \tag{3.18}$$

En effet, l'orientation d'un motif binaire dans une image peut être obtenue à partir des moments cartésiens d'ordre deux encore appelés moments d'inertie. Leur calcul suivant un axe est effectué simplement en multipliant un élément de surface considéré dA par le carré de la distance à l'axe(Fig. (3.9).

Ainsi on définit (et l'on notera) les moments d'inertie par rapport à x et à y par:

$$I_x = \int\int_{aire} y^2 dA \qquad\qquad I_y = \int\int_{aire} x^2 dA \tag{3.19}$$

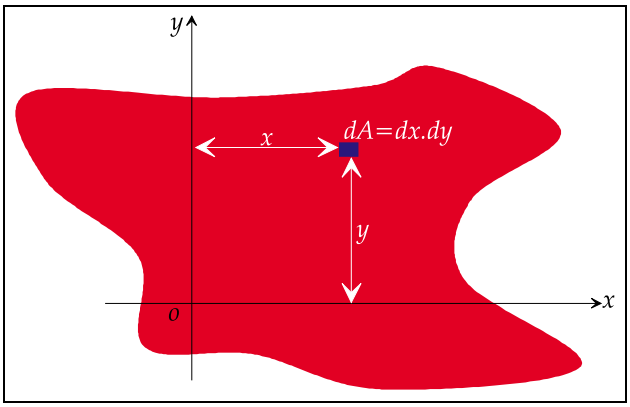


Fig. 3.9 – *Moments d'inertie suivant les axes x et y*

De la même manière, on définit le produit d'inertie par rapport aux axes x et y par l'expression:

$$I_{xy} = \int\int_{aire} xydA \tag{3.20}$$

Contrairement à I_x et I_y, le produit d'inertie peut aussi bien être négatif que positif. On remarquera que pour une forme symétrique (et centrée!), la répartition de masse est isotrope et l'on a I_{xy}=0.

Nous cherchons dans ce paragraphe à caractériser l'orientation de l'objet observé à partir de son image binarisée. La méthode que nous allons utiliser consiste à chercher les axes d'inertie principaux encore appelés axes principaux correspondant aux axes selon lesquels le moment d'inertie est respectivement maximal et minimal.

Pour fixer les idées, cherchons à faire passer au mieux (par exemple, par une résolution au sens des moindres carrés) une ellipse sur les contours de l'objet. L'ellipse obtenue admet deux axes orthogonaux qui sont les axes principaux de la forme observée (Fig. 3.10).

L'orientation de l'objet observé peut alors être déduite simplement en mesurant l'inclinaison de l'axe principal de plus grand moment avec par exemple l'axe des abscisses.

Pour calculer les axes principaux de la forme binaire, repartons de la définition initiale (3.19) appliquée au cas de l'image binaire:

$$I_x = \sum_{i=1}^{m}\sum_{j=1}^{n} j^2 P(i,j) \qquad I_y = \sum_{i=1}^{m}\sum_{j=1}^{n} i^2 P(i,j) \tag{3.21}$$

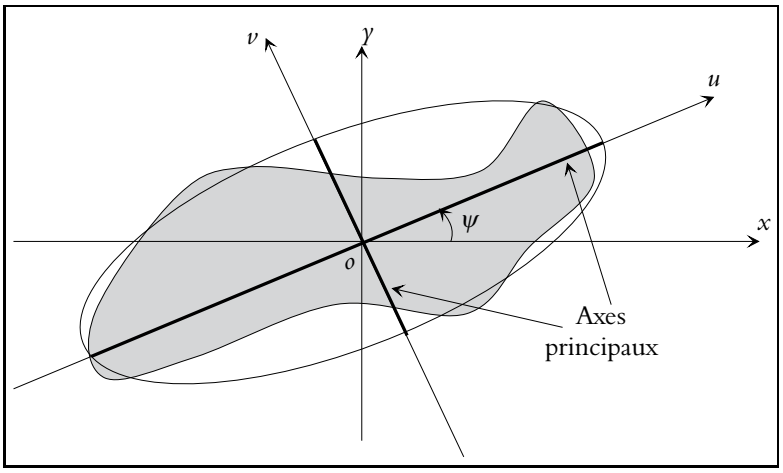


FIG. 3.10 – *Axes principaux d'une forme*

Nous allons maintenant chercher à calculer les moments d'inertie suivant un repère centré en l'origine mais ayant subi une rotation d'un angle δ. Nous pourrons ainsi exprimer les nouveaux moments I_u et I_v en fonction de I_x, I_y et de l'angle δ. Les axes du repère d'origine après avoir subi une rotation d'un angle δ s'expriment par:

$$\begin{aligned} u &= j\sin\delta + i\cos\delta \\ v &= j\cos\delta - i\sin\delta \end{aligned} \tag{3.22}$$

En reprenant la définition, il vient

$$I_u = \sum_{i=1}^{m}\sum_{j=1}^{n} v^2 P(i,j) = \sum_{i=1}^{m}\sum_{j=1}^{n} (j\cos\delta - i\sin\delta)^2 P(i,j) \tag{3.23}$$

et le développement de l'identité remarquable, nous donne:

$$\begin{aligned} I_u = \cos^2\delta \sum_{i=1}^{m}\sum_{j=1}^{n} j^2 P(i,j) - 2\sin\delta\cos\delta \sum_{i=1}^{m}\sum_{j=1}^{n} ijP(i,j) + \\ \sin^2\delta \sum_{i=1}^{m}\sum_{j=1}^{n} i^2 P(i,j) \end{aligned} \tag{3.24}$$

Il est alors aisé de faire apparaître dans cette équation les moments suivant x et y ainsi que le produit d'inertie. En effet, il vient:

$$I_u = I_x\cos^2\delta - 2I_{xy}\sin\delta\cos\delta + I_y\sin^2\delta \tag{3.25}$$

L'orientation de la forme binarisée correspond à l'orientation de l'axe principal d'inertie qui correspond lui-même à un maximum du moment d'inertie. Ainsi l'orientation δ cherchée correspondra à un extremum de l'équation (3.25) soit $\frac{dI_u}{d\delta}$=0 d'où on en déduit:

$$\delta = -\frac{1}{2}\arctan\left(\frac{2I_{xy}}{I_x - I_y}\right) \tag{3.26}$$

3.2.1.3.2 Etude de l'interaction Comme nous venons de l'expliquer, le calcul des moments d'inertie d'une forme binaire revient à chercher l'orientation de ces axes principaux d'inertie. Ces axes correspondent aux axes d'une ellipse passant au mieux par les points de contours du motif considéré (Fig. 3.10). Toutefois pour établir la relation d'interaction, il est nécessaire d'établir un lien entre le motif visuel et la primitive tridimensionnelle l'ayant produit. Dans ce cas de figure, quelle primitive 3D peut correspondre à une ellipse englobant au mieux le motif?

Pour répondre à cette question nous nous retrouvons dans la situation évoquée dans le cas du point où nous cherchions une correspondance physique entre le centre du rectangle englobant et l'objet lui-même. En effet là encore, la primitive visuelle que nous avons choisie n'a *a priori* pas de réalité physique, toutefois, nous pouvons la considérer comme la projection d'un cône elliptique tronqué dont l'orientation de la base correspond à celle de l'objet (Fig. 3.11). Les dimensions de ce cône sont

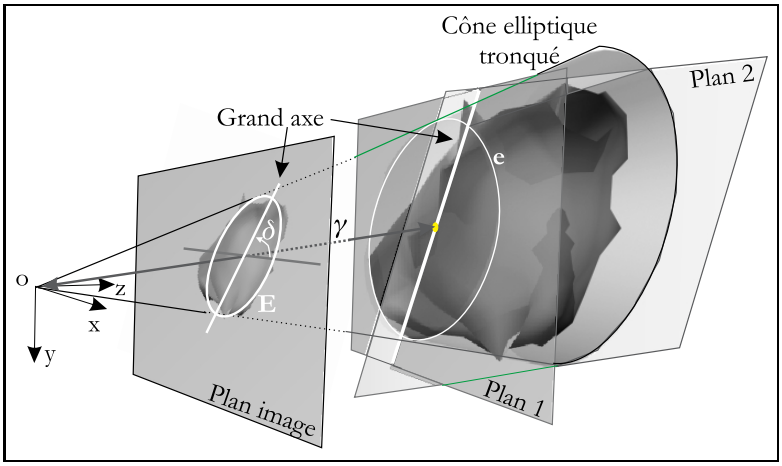


Fig. 3.11 – *Orientation d'un objet binaire à partir de sa projection perspective*

fixées par l'objet de manière à ce que les limbes du cône soient le plus près possible de la surface de l'objet. On peut ainsi considérer que l'ellipse E dans l'image est la projection de la face elliptique tronquée e du cône. Cette face est située à une distance γ correspondant à la cote du centre de la face. On se retrouve avec les

mêmes considérations que précédemment (§1.2.1.1.3) en ce qui concerne l'évolution de cette primitive au cours du temps. Pour notre part, nous nous intéresserons exclusivement à l'orientation du grand axe dans l'image, en d'autres termes à la régulation de l'angle δ.

Le grand axe de l'ellipse dans l'image peut ainsi être assimilé à la projection du grand axe de la face elliptique tronquée. Ce dernier peut être représenté comme l'intersection de deux plans (plan 1 et plan 2) dont l'un (plan 1) sera parallèle au plan image. Les équations de ces deux plans peuvent s'écrire :

$$\begin{cases} \text{Plan 1: } c_1.z + d_1 = 0 \\ \text{Plan 2: } a_2.x + b_2.y + c_2.z + d_2 = 0 \end{cases} \tag{3.27}$$

et la projection dans l'image du grand axe peut naturellement s'écrire:

$$A.X + B.Y + C = 0$$

La relation d'interaction caractérisant l'inclinaison δ du grand axe avec l'axe des abscisses peut être facilement obtenue à partir de la matrice d'interaction de la droite paramétrée en (ρ, δ). Ces résultats ont là encore été largement étudiés dans (Chaumette, 1990) et nous n'en rappelerons que les grandes lignes.

Le principe du paramétrage (ρ, δ) consiste à considérer la distance notée ρ entre la droite et l'origine, ainsi que l'inclinaison de cette droite que l'on note δ (Fig. 3.12) Dans le cas présent la relation d'interaction concernant l'inclinaison du grand axe

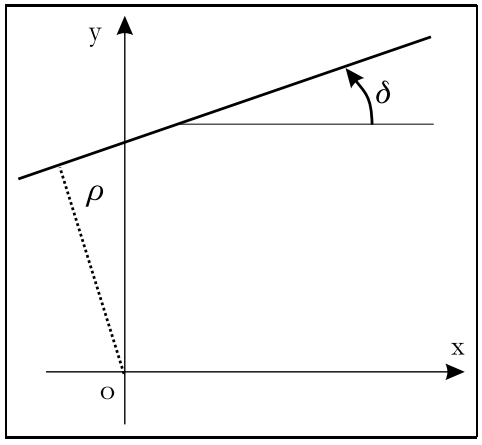


FIG. 3.12 – *Représentation (ρ, δ) des droites 2D.*

est donnée par la matrice suivante:

$$M_\delta^T = \begin{pmatrix} \nu.\sin\delta & -\nu.\cos\delta & \nu.\rho & -\rho.\sin\delta & \rho.\cos\delta & -1 \end{pmatrix} \tag{3.28}$$

avec $\nu = (a_2 b_1 - a_1 b_2)/\sqrt{A^2 + B^2}$

Comme le montre les équations 3.27, le plan 1 est parallèle au plan image et l'on a donc $a_1 = b_1 = 0$, ce qui implique que $\nu = 0$. La matrice d'interaction correspondante s'écrit alors:

$$M_\delta^T = \begin{pmatrix} 0 & 0 & 0 & -\rho.\sin\delta & \rho.\cos\delta & -1 \end{pmatrix} \tag{3.29}$$

On peut remarquer que seuls les axes de rotation permettent de réguler l'orientation de la forme dans l'image et lorsque le motif est centré (ρ=0), alors une rotation suivant l'axe optique suffit à réaliser la tâche désirée.

3.2.1.4 Les limites de ces primitives

Comme nous l'avons expliqué en préambule de ce chapitre, nous avons choisi de ne considérer que l'aspect géométrique du motif binaire dans l'image. A partir de cette approche nous avons alors mis en évidence trois types de primitives géométriques nous permettant de réaliser des tâches robotiques autour de l'objet. Ces primitives sont:

- le centre dans l'image d'un rectangle englobant l'objet,
- les dimensions de ce rectangle englobant dans l'image et plus généralement, la longueur d'un segment projeté sur un axe,
- l'orientation du motif binaire dans l'image.

Toutefois dans chacun des cas précédents, l'écriture de la relation d'interaction a été faite sans réellement prendre en compte l'aspect de l'objet et en cantonnant ces primitives à de simples segments disposés dans l'espace des configurations. Il est bien évident que ce cas est totalement irréaliste puisque dans la plupart des circonstances l'objet est volumique et l'extraction de primitives que l'on exécutera sous un point de vue peut être totalement remise en cause sous un autre point de vue.

Prenons pour exemple le cas d'une tâche de positonnement à une distance donnée de l'objet où l'on considère la largeur dans l'image comme primitive permettant de réguler la distance entre la caméra et l'objet. Durant la tâche de positionnement, la caméra peut "découvrir" d'autres aspects de l'objet et la largeur mesurée dans l'image n'aura pas de réalité physique (Fig. 3.13). En effet, sur cet exemple la largeur dans l'image notée par le segment [AB], correspond suivant la position de la caméra soit au segment [a1b1], soit à [a2b2]. On notera toutefois que malgré les discontinuités entre les points de vue 1 et 2, l'évolution de la largeur dans l'image sera continue.

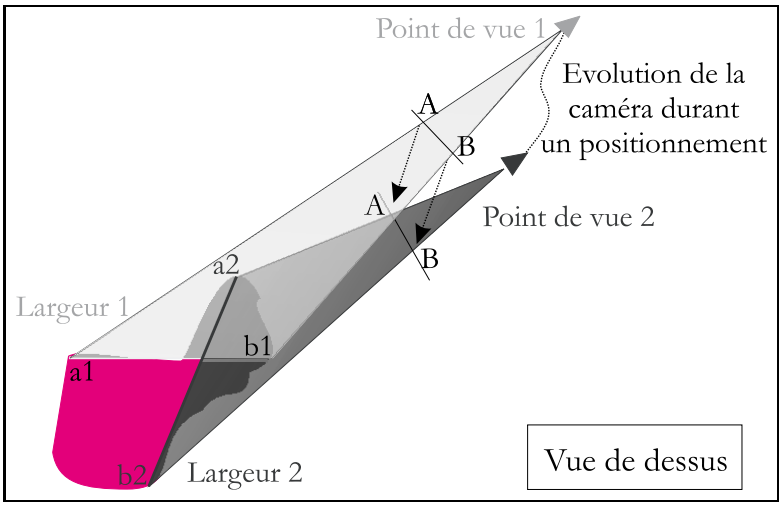


FIG. 3.13 – *Evolution de la caméra durant un positionnement et mesure de la largeur.*

Ceci est tout simplement dû au fait qu'il existe une position limite pour chaque extrémité (A et B) telle que :

$$x_A = \frac{x_{a1}}{z_{a1}} = \frac{x_{a2}}{z_{a2}}$$

et de même pour l'extrémité B.

Dans notre cas, nous n'avons pas accès aux coordonnées tridimensionnelles des extrémités, ce qui implique des approximations (discutées dans le paragraphe suivant) ainsi que la spécification d'une consigne directement dans l'image.

Une seconde limite concerne la régulation de l'orientation du motif dans l'image. En effet de la même façon que précédemment, durant une tâche de positionnement l'aspect de l'objet peut évoluer et l'orientation de ce dernier peut être totalement modifiée. Un exemple significatif peut être donné en considérant une boite fine et très allongée (Fig. 3.14). Suivant le point de vue admis, l'orientation de l'objet peut se trouver entièrement modifiée. Dans le cas présent, on passe d'un angle de 0^o à 90^o! Nous noterons toutefois que cette variation d'orientation (tout comme les dimensions) est relativement faible dans les tâches de positionnement et pour notre part nous n'avons pas observé de divergence lors des expérimentations sur site réel.

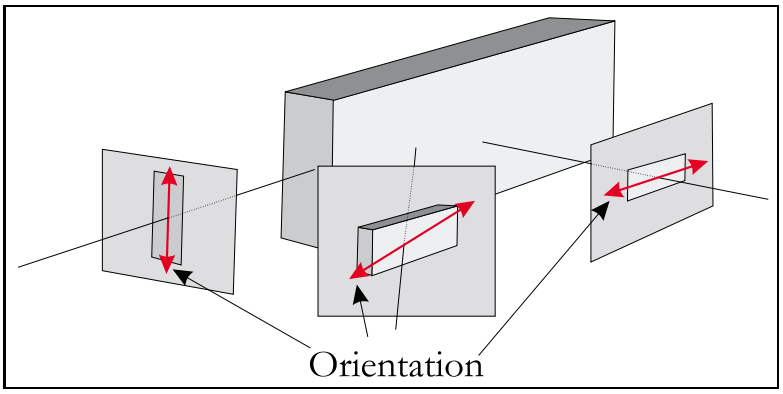


Fig. 3.14 – *Axes principaux d'une boîte allongée*

3.2.2 Commande

Dans le paragraphe précédent nous avons présenté les différentes primitives géométriques que l'on pouvait extraire d'une image binarisée. Ces primitives sont la position de l'objet dans l'image, la taille dans l'image et l'orientation. Nous avons proposé dans chacun des cas une relation d'interaction permettant de lier les variations de la primitive considérée avec le mouvement de la caméra. Nous allons à présent passer en revue les différentes tâches robotiques réalisables à partir de ces primitives ainsi que les mouvements laissés libres pour d'autres tâches.

Toutefois dans un premier temps, avant de nous intéresser à l'aspect "contrôle", nous allons fixer les principales conditions permettant de se déplacer sans problème autour d'un objet:

1 Tout d'abord, l'objet autour duquel l'asservissement est fait doit toujours être vu dans l'image. Pour cette raison, il est nécessaire de réaliser une tâche de pointage permettant de garder l'objet considéré au centre de l'image.

2 En second lieu, les déplacements effectués autour de l'objet doivent être réalisés à une certaine distance évitant ainsi toute collision entre le robot et l'objet. Il convient donc de définir une seconde tâche robotique permettant de maintenir un écart constant entre la caméra et l'objet.

3 Enfin, les mouvements réalisés autour de l'objet doivent être entièrement compatibles avec les conditions 1 et 2.

Les conditions précédentes contraignent donc les mouvements de la caméra sur une sphère centrée sur l'objet et ayant un rayon permettant d'éviter les collisions ro-

bot/objet. Dans les deux paragraphes suivants nous allons successivement étudier la tâche de positionnement par rapport à l'objet, puis la navigation.[2]

3.2.2.1 Positionnement par rapport à un objet complexe

Pour ce type de tâche robotique, comme nous l'avons expliqué dans le chapitre introductif, nous cherchons à atteindre un équilibre entre la caméra et l'objet observé. Dans notre approche cet équilibre sera commandé à partir des informations visuelles précédemment décrites, à savoir la position dans l'image, la projection sur un axe et l'orientation des axes principaux.

3.2.2.1.1 Signaux capteur et spécification de la consigne Ces quatres informations visuelles constituent ainsi un vecteur de mesure noté $\mathcal{F}$ tel que:

$$\mathcal{F} = \begin{pmatrix} X \\ Y \\ L_{\mathcal{L}} \\ \delta \end{pmatrix}$$

Ce dernier va donc nous permettre de contrôler quatre degrés de liberté en l'occurence les trois axes de rotations (R_x, R_y, R_z) et la translation suivant l'axe optique (T_z). Ce choix des axes est purement arbitraire en ce qui concerne les rotations suivant x et y, et l'on aurait très bien pu utiliser les translations suivant ces mêmes axes. Notons toutefois qu'il nous aît paru plus intuitif de garder les translations pour générer des déplacements autour de l'objet.
La consigne visuelle $\mathcal{F}^*$ est construite à partir des spécifications données ci-dessus. En d'autres termes, la régulation devra centrer l'objet dans l'image soit $\begin{pmatrix} X^* \\ Y^* \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$. Dans les tâches de positionnement envisagées aucune connaissance *a priori* sur les dimensions de l'objet n'est fournie. Pour cette raison, nous avons été contraints de choisir les critères permettant de réguler la distance entre la caméra et l'objet directement dans l'espace image. Ainsi ces derniers sont spécifés sous la forme: `"hauteur de l'objet=x% de la taille image"` ou plus généralement "$L^*_{\mathcal{L}}$`= n-pixels"`.

En ce qui concerne l'orientation de l'objet, nous avons choisi d'aligner les axes principaux avec les axes du repère image, par conséquent $\delta^* = 0$ et la consigne

2. simplement au sens "déplacement" autour de l'objet.

visuelle est donnée par:

$$\mathcal{F}^* = \begin{pmatrix} 0 \\ 0 \\ L^*_{\mathcal{L}} \\ 0 \end{pmatrix}$$

3.2.2.1.2 Spécification de la matrice d'interaction La matrice d'interaction que nous utilisons pour réaliser cette tâche de positionnement est déduite à partir des différentes relations d'interactions étudiées ci-dessus.

En ne considérant que les axes de commande concernés (soit T_z, R_x, R_y, R_z), la matrice d'interaction prend la forme suivante:

$$M^T_{\mathcal{F}} = \begin{pmatrix} \frac{X}{z} & XY & -1-X^2 & Y \\ \frac{Y}{z} & 1+Y^2 & -XY & -X \\ \begin{bmatrix} \nu_2.L_{\mathcal{L}} - \nu_1.X_{\mathcal{S}}. \\ \cos\beta - \nu_1.Y_{\mathcal{S}}.\sin\beta \end{bmatrix} & \begin{bmatrix} -L_{\mathcal{L}}(-X_{\mathcal{S}}.\cos\alpha.\sin\alpha + \\ Y_{\mathcal{S}}(1+\sin^2\alpha) + \tan(\beta-\alpha). \\ X_{\mathcal{S}}.\sin^2\alpha - \tan(\beta-\alpha). \\ Y_{\mathcal{S}}.\cos\alpha.\sin\alpha) \end{bmatrix} & \begin{bmatrix} -L_{\mathcal{L}}(Y_{\mathcal{S}}.\cos\alpha.\sin\alpha - \\ \tan(\beta-\alpha).X_{\mathcal{S}}.\cos\alpha. \\ \sin\alpha + X_{\mathcal{S}}(\cos^2\alpha+1) + \\ \tan(\beta-\alpha).Y_{\mathcal{S}}.\cos^2\alpha) \end{bmatrix} & -L_{\mathcal{L}}.\tan(\beta-\alpha) \\ 0 & -\rho.\sin\delta & \rho.\cos\delta & -1 \end{pmatrix} \quad (3.30)$$

où
- $\rightarrow (X, Y)$: centre M du rectangle englobant l'objet dans l'image,
- $\rightarrow z$: cote du point M sur la surface de l'objet,
- $\rightarrow (X_{\mathcal{S}}, Y_{\mathcal{S}})$: milieu de $[ab]$,
- $\rightarrow \alpha$: inclinaison de $[ab]$,
- $\rightarrow \beta$: inclinaison de l'axe de projection Δ,
- $\rightarrow L_{\mathcal{L}}$: longueur de la projection de [ab] sur Δ,
- $\rightarrow \left.\begin{array}{l} \nu_1 = \frac{z_a - z_b}{z_a.z_b} \\ \nu_2 = \frac{z_a + z_b}{2.z_a.z_b} \end{array}\right\}$ où z_a et z_b correspondent aux cotes des points extrêmes a et b,
- $\rightarrow \delta$: inclinaison du grand axe d'inertie,
- $\rightarrow \rho$: distance du grand axe d'inertie à l'origine

Comme nous pouvons le constater, cette matrice fait intervenir de nombreuses quantités qui ne sont pas toutes mesurables. Pour cette raison, nous nous restreindrons à utiliser cette même matrice calculée à l'équilibre pour $\mathcal{F} = \mathcal{F}^*$. Ainsi il vient $X = 0$, $Y = 0$, $\delta = 0$ et la quantité z n'intervient plus. De plus, à l'équilibre, l'objet est centré, donc les axes d'inertie concourent à l'origine dont $\rho = 0$.
Pour des raisons de simplification évidente, nous choisirons comme longueur de mesure dans l'image $L_{\mathcal{L}}$, la hauteur ou la largeur de l'objet. Pour la suite de ce paragraphe, nous utiliserons la largeur sachant qu'un développement identique peut être effectué pour la hauteur. Dans cette configuration, nous avons alors $\beta = 0$,

$X_{\mathcal{S}} = 0$ et la matrice d'interaction est donnée par:

$$M_{\mathcal{F}^*}^T = \begin{pmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ \nu_2^*.L_{\mathcal{L}}^* & L_{\mathcal{L}}^*.Y_{\mathcal{S}}^* & 0 & L_{\mathcal{L}}^*.\tan(\alpha^*) \\ 0 & 0 & 0 & -1 \end{pmatrix} \tag{3.31}$$

Dans cette expression nous exprimons la relation d'interaction à l'équilibre entre la caméra et l'objet; toutefois cette relation est fonction de α, $Y_{\mathcal{S}}$ et ν_2 c'est-à-dire fonction de l'objet observé. Dans un premier temps attachons-nous à discuter les deux premiers termes soit α, $Y_{\mathcal{S}}$.

La tâche d'orientation va chercher à ramener à l'horizontale le grand axe d'inertie de l'objet, ce qui signifie que le maximum de répartiton de masse[3] dans l'image sera ramené parallèlement à l'axe de projection Δ. Partant de ce fait et exceptés des cas très particuliers comme celui de la figure 3.15, nous pouvons considérer que les points extrêmes (a et b) se projetant sur Δ vont former un segment proche de l'horizontale. Nous aurons alors $\alpha \approx 0$. Comme le montre Fig. 3.15, cette approximation n'est

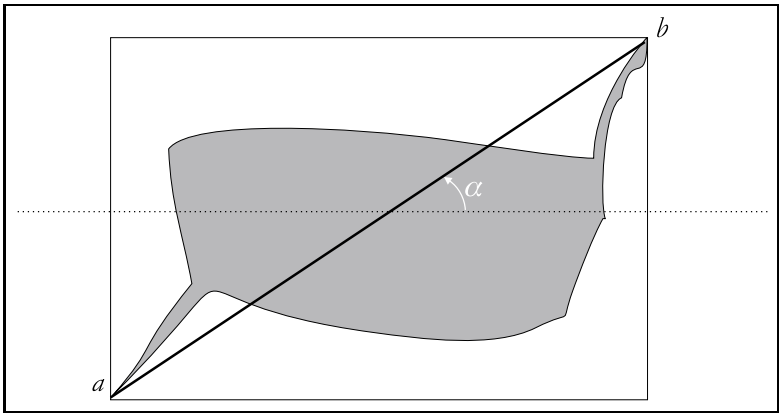


FIG. 3.15 – *Exemple d'objets ne validant pas* $\alpha \approx 0$.

valable que si les points a et b sont proches du grand axe d'inertie.

La seconde approximation concerne le terme $Y_{\mathcal{S}}$. Lors de la tâche de centrage le centre du rectangle englobant l'objet est ramené à l'origine. En se plaçant là encore dans une configuration selon laquelle l'objet ne présente pas de particularités telles que la figure 3.16, nous pouvons considérer que $Y_{\mathcal{S}} \approx 0$.

On peut ainsi construire une loi proportionnelle de la forme $T = -\lambda.\underline{e}$, où

3. Le terme masse est ici un abus de langage, puisque nous travaillons uniquement dans l'image.

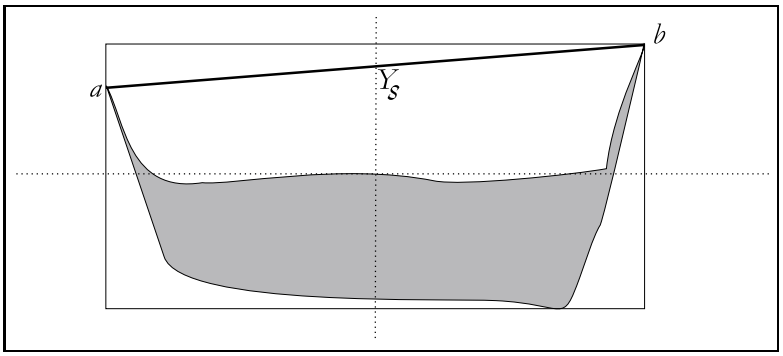


Fig. 3.16 – *Exemple d'objets ne validant pas* $Y_S \approx 0$.

$\underline{e}(\underline{s}(t), t)$ définit une fonction de tâche telle que

$$\underline{e}(\underline{s}(t), t) = C. \begin{pmatrix} X - 0 \\ Y - 0 \\ L_{\mathcal{L}} - L^*_{\mathcal{L}} \\ \delta - 0 \end{pmatrix}$$

Comme il a été précédemment expliqué, on peut choisir la matrice de combinaison C telle que $C = L^{T-1}$ soit:

$$C = \begin{pmatrix} 0 & 0 & \frac{1}{\nu_2 . L^*_{\mathcal{L}}} & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \tag{3.32}$$

et il vient alors:

$$\begin{pmatrix} T_z \\ R_x \\ R_y \\ R_z \end{pmatrix} = \lambda . \begin{pmatrix} -\frac{1}{\nu_2 . L^*_{\mathcal{L}}} . (L_{\mathcal{L}} - L^*_{\mathcal{L}}) \\ -Y \\ X \\ \delta \end{pmatrix} \tag{3.33}$$

On peut constater que cette commande au prix de quelques approximations souvent réalistes permet un découplage de chaque axe vis à vis des primitives mesurées. Cette condition est particulièrement intéressante dans notre cas puisque l'on ne connait pas l'objet.

Il est toutefois nécessaire de connaître le terme $\nu_2 = \frac{z_a + z_b}{2 . z_a . z_b}$. Naturellement cette quantité ne peut pas être connue ni même estimée, cependant elle peut être approximée par $\nu'_2 = \frac{2}{z_a + z_b} = \frac{1}{z_{moy}}$, où z_{moy} correspond à la profondeur moyenne des

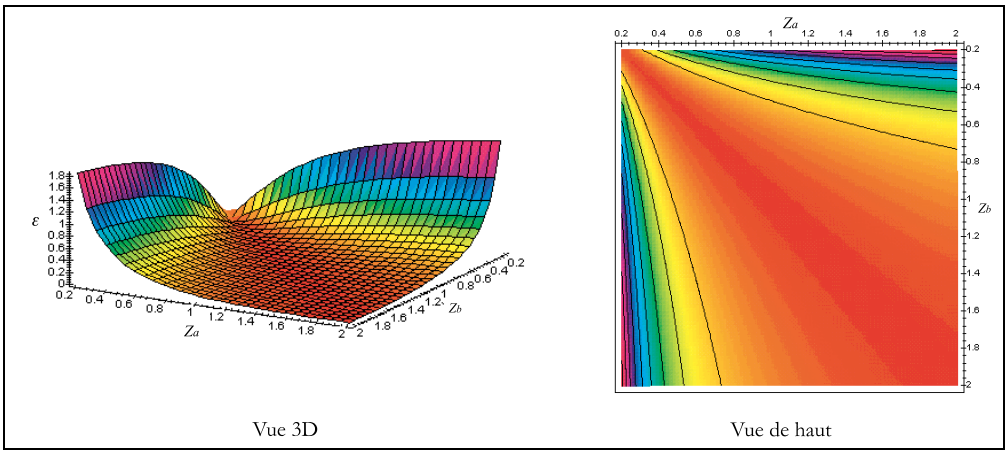


FIG. 3.17 – *Erreur entre ν_2 et ν_2'. Les courbes de niveaux correspondent à des pas de 0.2 et l'erreur ϵ croît graduellement du rouge vers le bleu.*

extrémités a et b. Sur la figure 3.17, on représente l'erreur $\epsilon = (\nu_2 - \nu_2')$ entre la valeur vraie de ν_2 et son approximation ν_2'. On constate naturellement que plus on se trouve éloigné de l'objet et plus cette approximation devient juste. En effet les dimensions de l'objet deviennent alors négligeables devant la distance caméra/objet. De plus on remarquera que pour des positionnements à une distance supérieure à 50cm l'approximation proposée permet de garantir une erreur inférieure à 0.2.

En ce qui concerne la valeur de ν_2', elle est calculée pour z^*_{moy} à l'équilibre. La quantité z^*_{moy} peut être fixée arbitrairement dans un domaine de valeurs raisonnables, puisque z^*_{moy} se comportera dans notre loi de commande comme un simple gain suivant T_z.

3.2.2.2 Navigation autour d'un objet complexe

Attachons-nous maintenant dans ce paragraphe à construire une commande permettant de se déplacer autour d'un objet complexe. Cette loi de commande est élaborée à partir d'une tâche hybride constituée :

- d'une tâche primaire dont le rôle est de centrer l'objet et de maintenir la distance caméra/objet,

- d'une tâche secondaire contrôlée par les translations suivant les axes x et y et dont le rôle est de produire le mouvement autour de l'objet.

Remarque: Dans cette loi de commande nous ne pouvons pas assurer la régulation de l'orientation à partir des primitives utilisées. En effet, comme nous l'avons évoqué au §3.2.1.4, durant une tâche de contournement la mesure de l'orientation basée sur les axes d'inertie de l'objet dans l'image est instantanée et ne correspond pas à l'orientation globale de l'objet dans l'espace.

La tâche primaire sera donc régulée par:

$$\mathcal{F} = \begin{pmatrix} X \\ Y \\ L_{\mathcal{L}} \end{pmatrix}$$

Contrairement au cas précédent, nous ne considérerons pas que les trois axes à commander, mais cinq degrés de liberté, à savoir les 3 axes de translation et les rotations suivant x et y. En effet, la mise en place de la tâche secondaire nécessite l'utilisation d'un projecteur orthogonal prenant en compte tous les axes commandés.

3.2.2.2.1 Relation d'interaction Pour cette loi de commande, nous ferons l'approximation selon laquelle le point $(X_{\mathcal{S}}, Y_{\mathcal{S}})$ est proche de l'origine; il vient ainsi $X_{\mathcal{S}} \approx Y_{\mathcal{S}} \approx 0$. La matrice d'interaction correspondante calculée à l'équilibre sera donc:

$$M^T_{\mathcal{F}^*} = \begin{pmatrix} -\frac{1}{z} & 0 & 0 & 0 & -1 \\ 0 & -\frac{1}{z} & 0 & 1 & 0 \\ \nu 1.\cos\beta & \nu 1.\sin\beta & \nu_2.L^*_{\mathcal{L}} & 0 & 0 \end{pmatrix} \tag{3.34}$$

Afin d'observer les mouvements autorisés par cette interaction, calculons le noyau de cette matrice d'interaction et il vient:

$$Ker(M^T_{\mathcal{F}^*}) = \left\{ \begin{pmatrix} 1 & 0 & \frac{-\nu_1.\cos\beta}{\nu_2.L^*_{\mathcal{L}}} & 0 & -\frac{1}{z} \end{pmatrix}, \begin{pmatrix} 0 & 1 & \frac{-\nu_1.\sin\beta}{\nu_2.L^*_{\mathcal{L}}} & \frac{1}{z} & 0 \end{pmatrix} \right\}$$

Si l'on considère un mouvement de translation de la forme $\underline{T} = \begin{pmatrix} T_x \\ T_y \end{pmatrix}$ que l'on peut décomposer dans le repère caméra sous la forme $T = \begin{pmatrix} A.\cos\gamma \\ A.\sin\gamma \end{pmatrix}$, ainsi les mouvements autorisés par cette interaction sont de la forme:

$$\begin{pmatrix} A.\cos\gamma & A.\sin\gamma & A.\frac{-\nu_1(\cos\beta.\cos\gamma+\sin\beta.\sin\gamma)}{\nu_2.L^*_{\mathcal{L}}} & \frac{A.\sin\gamma}{z} & -A.\frac{T.\cos\gamma}{z} \end{pmatrix} \tag{3.35}$$

Nous pouvons remarquer que ces mouvements sont constitués par une combinaison de translation et de rotation suivant les axes x et y, mais interviennent aussi suivant l'axe optique. La seule possibilité permettant de découpler ce dernier axe par rapport

aux "axes de pointage" (T_x, T_y, R_x, R_y), est d'imposer $(\cos\beta.\cos\gamma + \sin\beta.\sin\gamma) = 0$. Cette condition est aisément obtenue par le choix de $\gamma = \beta + \frac{\pi}{2}$. En effet, il vient alors $(\cos\beta.\cos(\beta+\frac{\pi}{2}) + \sin\beta.\cos(\beta+\frac{\pi}{2}) = \cos\beta.\sin\beta - \cos\beta.\sin\beta = 0$.

Cette condition peut être facilement remplie en fonction du mouvement à réaliser. En effet, il suffit simplement de choisir un axe Δ tel que ce dernier soit orthogonal au mouvement dans l'image. Ainsi pour réaliser un déplacement horizontal on choisira pour axe de projection l'axe des ordonnées soit $\beta = \frac{\pi}{2}$.

3.2.2.2.2 Loi de commande La fonction de tâche correspondant à notre objectif peut ainsi se mettre sous la forme:

$$\underline{e} = M_{\mathcal{F}^*}^{T+}(\mathcal{F} - \mathcal{F}^*) + \alpha(\mathbb{I}_5 - M_{\mathcal{F}^*}^{T+} M_{\mathcal{F}^*}^{T})\underline{g}_s^T$$

où $(\mathbb{I}_5 - M_{\mathcal{F}^*}^{T+} M_{\mathcal{F}^*}^{T})$ représente le terme de projection orthogonal sur le noyau de la matrice d'interaction $M_{\mathcal{F}^*}^{T}$. La loi de commande utilisée est donnée par

$$Tc = -\lambda\underline{e} - \alpha(\mathbb{I}_5 - M_{\mathcal{F}^*}^{T+}.M_{\mathcal{F}^*}^{T})\frac{\partial \underline{g}_{\mathcal{F}}^T}{\partial t} \tag{3.36}$$

où $\mathcal{F}$ et $\mathcal{F}^*$ représentent respectivement les vecteurs de mesure et de consigne.

Dans le cas de notre tâche de navigation autour de l'objet, il convient de définir un coût secondaire $h_{\mathcal{F}}$ de gradient $\underline{g}_{\mathcal{F}}^T$. Pour un mouvement autour de l'objet à vitesse constante V_x selon l'axe $\vec{x}$ et V_y selon l'axe $\vec{y}$, le coût secondaire s'écrit alors:

$$h_{\mathcal{F}} = \frac{1}{2}(x - x_o - V_x t)^2 + \frac{1}{2}(y - y_o - V_y t)^2$$

soit le gradient $\underline{g}_{\mathcal{F}}^T = \frac{\partial h_{\mathcal{F}}}{\partial r}$ égal à:

$$\underline{g}_{\mathcal{F}}^T = \begin{pmatrix} (x - x_o - V_x t) \\ (y - y_o - V_y t) \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

3.3 Résultats expérimentaux

Dans cet ultime paragraphe nous présentons les différents résultats que nous avons obtenus sur notre plate-forme expérimentale. Ces résultats ont pour but de montrer l'efficacité de l'approche retenue à travers deux types de tâches robotiques : le positionnement et le contournement d'objet.

3.3.1 Contexte expérimental

Afin de valider les tâches de positionnement et de navigation autour d'objets quelconques, nous avons implémenté une loi de commande référencée vision utilisant les différentes primitives visuelles décrites ci-dessus. Comme nous pouvons le constater, ces primitives essayent de transcrire en termes quantitatifs les conditions du bon déroulement d'une tâche de navigation "intelligente".

Cependant, la loi de commande proposée doit pouvoir fonctionner sur différents types d'objets. Aussi est-il normal d'évoquer les problèmes de traitements d'image ainsi que les choix algorithmiques que nous avons faits en fonction de l'architecture matérielle dont nous disposions [4].

Les quatre primitives requises et que nous avons à extraire de l'image sont:

- le centre du rectangle englobant,
- la taille de ce rectangle,
- la longueur du segment sur lequel sont projetés les points de contours,
- l'orientation de l'axe de plus grand moment d'inertie.

Les trois premières mesures nécessitent la détection des points de contour de l'objet tandis que la dernière primitive requiert tous les points de l'objet. Afin de faciliter la détection de l'objet dans l'image, le fond de la scène a été volontairement assombri.

Cependant, malgré ces précautions les objets observés peuvent comporter différentes teintes donnant un large éventail de niveau de gris. Pour cette raison, nous utilisons une détection basée sur un seuillage adaptatif des niveaux de gris, ce qui nous permet de binariser l'image en deux régions : objet et fond.

La méthode utilisée (Tsai, 1985) est basée sur la conservation des modes de l'histogramme des niveaux de gris entre l'image originale et l'image seuillée. Dans

4. Une description de cette architecture est proposée en Annexe A.

cette approche, l'algorithme cherche les deux lobes principaux dans l'histogramme de l'image d'origine, puis il détermine un seuil tel que l'histogramme de l'image seuillée admette les deux mêmes lobes. Cette méthode a donné de bons résultats dans l'ensemble même si quelque fois des contrastes trop forts ont entraîné quelques parasites.

L'extraction des points de contour est effectuée à l'aide de la carte bas niveau *Windis* permettant de calculer en temps réel et directement sur le flot vidéo différents masques de convolution. Nous utilisons dans notre cas deux masques 3×3 effectuant un gradient en x et en y et un masque unité pour garder tous les points de l'image.

Les résultats de ces convolutions sont alors seuillés par le seuil adaptatif, et envoyés sur la carte moyen niveau *Winproc* . Sur cette carte munie de 4 DSP travaillant en parallèle, nous effectuons le calcul du seuil adaptatif, le calcul du rectangle englobant, la projection du contour sur l'axe de rotation et le calcul des moments d'inertie dans l'image. Les résultats sont alors directement envoyés sur une carte haut niveau *Winman* s'occupant de la commande du robot et de la gestion des deux cartes précédentes.

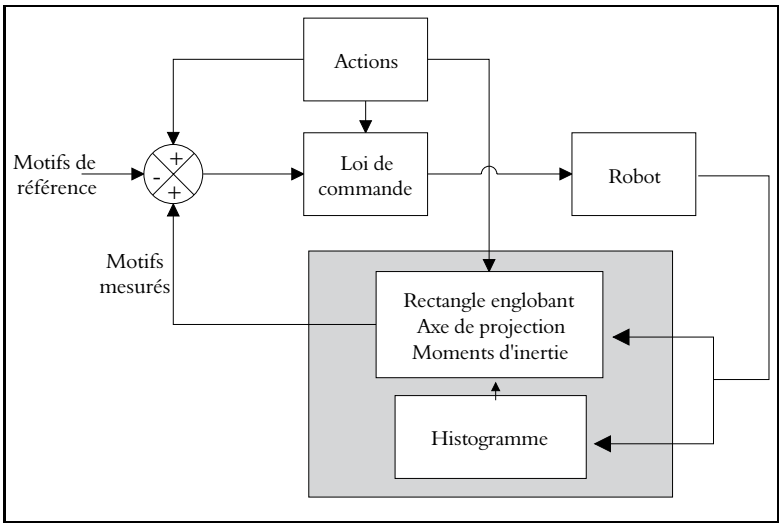


FIG. 3.18 – *Synoptique de l'application*

Au niveau architecture, la plate-forme utilisée nous permet de rafraîchir en temps réel tous les paramètres de traitement: taille des fenêtres d'intérêt, traitement dans les fenêtres, orientation de l'axe de projection,... Les différents traitements ne peuvent cependant être appliqués à la commande qu'avec un retard pur de 2 itérations (80ms) dû à la numérisation, au stockage et au traitement de l'image.
Au niveau algorithmique, après avoir remarqué des détections de points parasites, nous avons été contraints de sélectionner non pas un point unique pour qualifier

chaque extremum mais un ensemble de points. Le choix du point extrême est alors effectué par un tri statistique.

Les différents objets utilisés sont de petits jouets aux formes relativement variées et ne présentant pas de particularités visuelles (points caractéristiques, ...). Pour chacune des manipulations présentées, une photo montre l'objet utilisé. Les expérimentations ont été réalisées à partir d'applications tournant à la cadence vidéo soit 40ms.

3.3.2 Tâche de positionnement

Dans ce type de tâche, on cherche à positionner le robot vers l'objet en respectant des conditions uniquement spécifiées dans l'espace image. Nous présentons deux tâches de positionnement effectuées sur une petite voiture en plastique et un petit canard en bois peint.

3.3.2.1 Voiture

L'objet retenu dans cette expérimentation est présenté sur la figure 3.19.

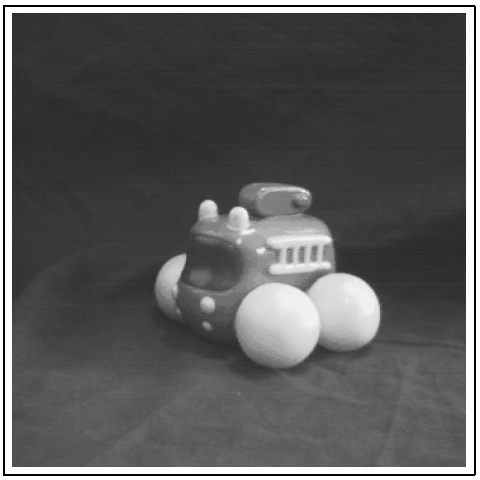


Fig. 3.19 – *Cible utilisée pour le positionnement*

Les différents paramètres utilisés pour cette expérimentation sont les suivants:

- Gain de boucle: $\lambda = 0.5$
- Distance désirée entre l'objet et la caméra: $z^* = 0.7m$

Ce dernier paramètre a été fixé arbitrairement en fonction de la scène observée et ne fait l'objet d'aucune estimation ou calcul particulier.

La consigne visuelle a été spécifiée comme suit:

- Objet centré dans l'image,
- Hauteur de l'objet: 20% de l'image soit environ 102 pixels,
- Orientation: Grand axe horizontal.

Les figures 3.20, présentent les courbes des vitesses appliquées à l'effecteur.

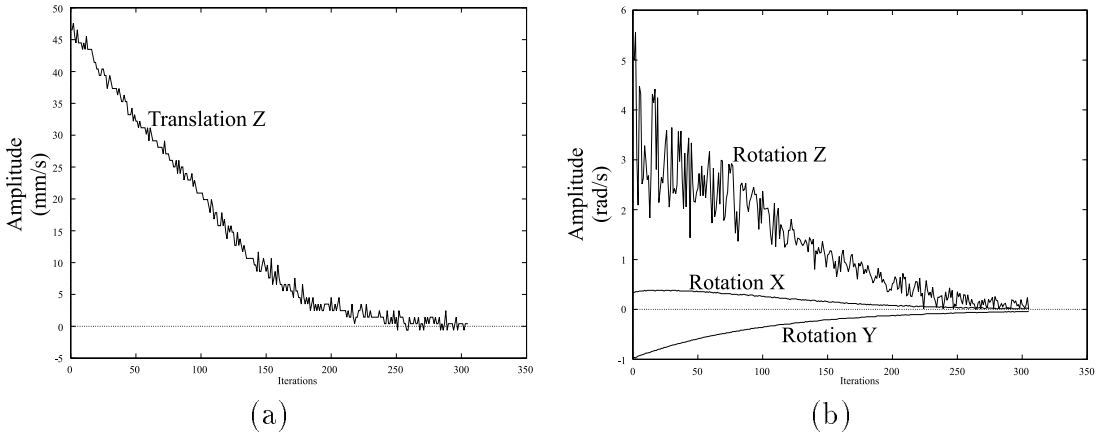


FIG. 3.20 – *Vitesses de translation et de rotation de l'effecteur*

Pour cette expérimentation, nous avons disposé l'objet comme le montre la première image de la séquence Fig.3.21. Comme nous pouvons le constater sur les courbes 3.20, les différentes composantes commandées à savoir (T_z, R_x, R_y, R_z) effectuent une décroissance exponentielle. Nous pouvons toutefois remarquer les perturbations sur l'axe de rotation en z, liées à la mesure de l'orientation qui s'avère relativement sensible au bruit de mesure.

Fig. 3.21 – *Evolution de l'objet dans l'image durant le positionnement.*

3.3.2.2 "Patatoïde"

Pour cette tâche de positionnement, l'objet utilisé est un "patatoïde" muni de deux "cornes" comme décrit sur la figure 3.22. L'intérêt de cet objet réside dans le fait qu'il ne valide pas les conditions d'approximations exposées précédemment, à savoir $\alpha \approx 0$ et $Y_S \approx 0$. Pour cette manipulation, nous avons disposé l'objet de façon à avoir $\alpha \not\approx 0$ (Fig. 3.15).

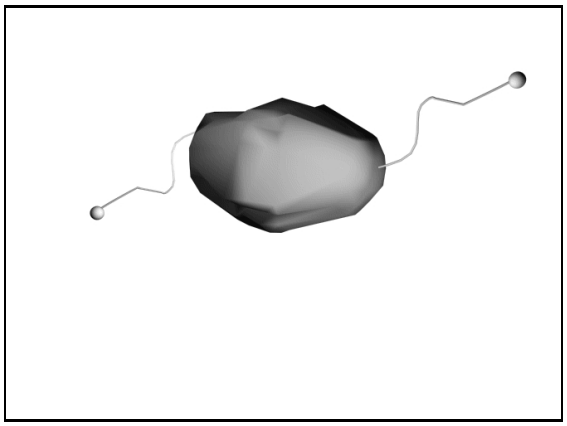


FIG. 3.22 – *Description de l'objet "Patatoïde"*

Les différents paramètres utilisés pour cette expérimentation sont les suivants:

- Gain de boucle: $\lambda = 0.3$
- Distance désirée entre l'objet et la caméra: $z^* = 0.7m$

Là encore, le paramètre z a été fixé arbitrairement.

La consigne visuelle a été spécifiée comme suit:

- Objet centré dans l'image,
- Largeur de l'objet: 30% de l'image soit 170 pixels,
- Orientation: Grand axe horizontal.

Sur les courbes de vitesse (Fig. 3.23) nous pouvons noter les perturbations auxquelles sont sujets les axes de rotation et de translation suivant z. Le bruit, particulièrement conséquent suivant l'axe de rotation en z s'explique par la méthode de mesure de l'angle d'orientation. En effet malgré des objets relativement constrastés,

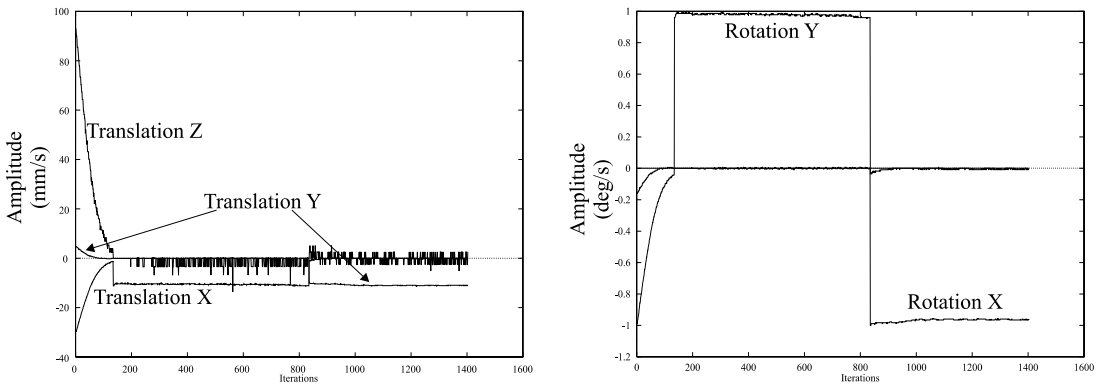


FIG. 3.23 – *Vitesses de translation et de rotation*

nous avons pu noter que la mesure de l'orientation est entachée d'un bruit résiduel d'environ 1.5^o.

Toutefois, comme le montrent les différentes courbes et la séquence 3.25, les approximations réalisées dans la loi de commande ne semblent pas perturber la convergence de la loi de commande.

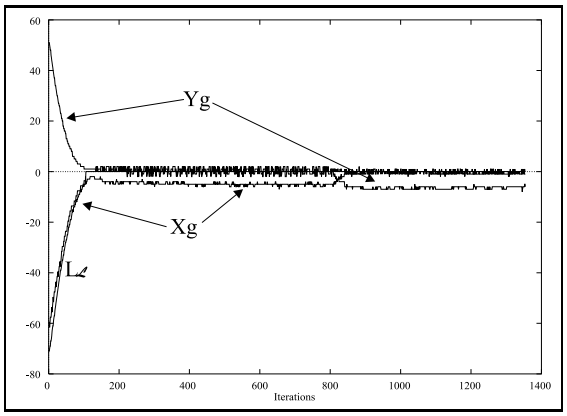


FIG. 3.24 – *Evolution de l'erreur suivant les 4 primitives.*

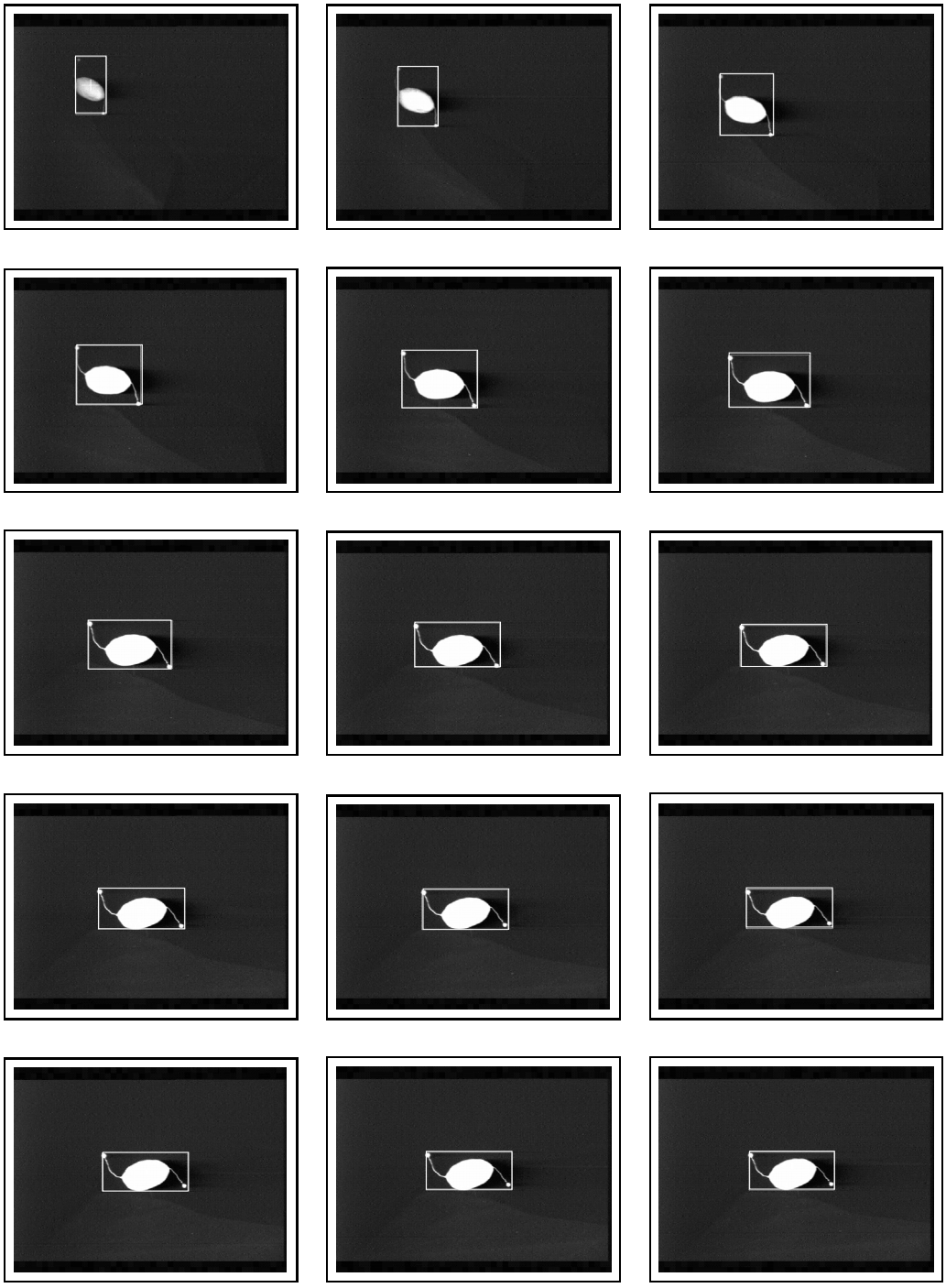


FIG. 3.25 – *Evolution de l'objet dans l'image durant le positionnement.*

3.3.3 Tâche de contournement

Cette tâche succède au positionnement et permet d'effectuer un déplacement autour de l'objet. Dans ce type de tâche, nous utilisons une tâche secondaire nous permettant d'appliquer à la caméra les mouvements désirés tout en restant fixée à une distance constante sur l'objet. Comme précédemment, chaque tâche sera désignée par l'objet sur lequel nous avons réalisé l'asservissement. Dans le cas présent, nous avons réutilisé le "patatoïde" et une petite girafe en caoutchouc.

3.3.3.1 Patatoïde

Durant cet asservissement, le robot effectue une première tâche de positionnement, puis réalise un mouvement de translation vers la gauche suivant l'axe x suivi d'une translation vers le haut suivant l'axe y. Dans cette expérimentation, l'objet est disposé de façon à avoir $Y_{\mathcal{S}} \not\approx 0$ (Fig. 3.16).

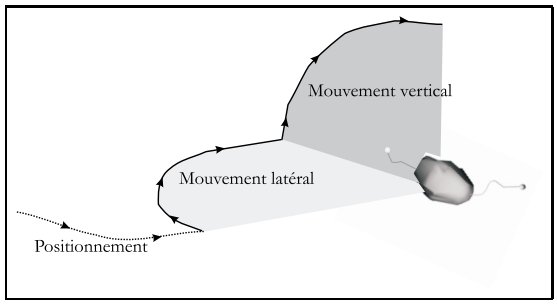


Fig. 3.26 – *Trajectoire de contournement*

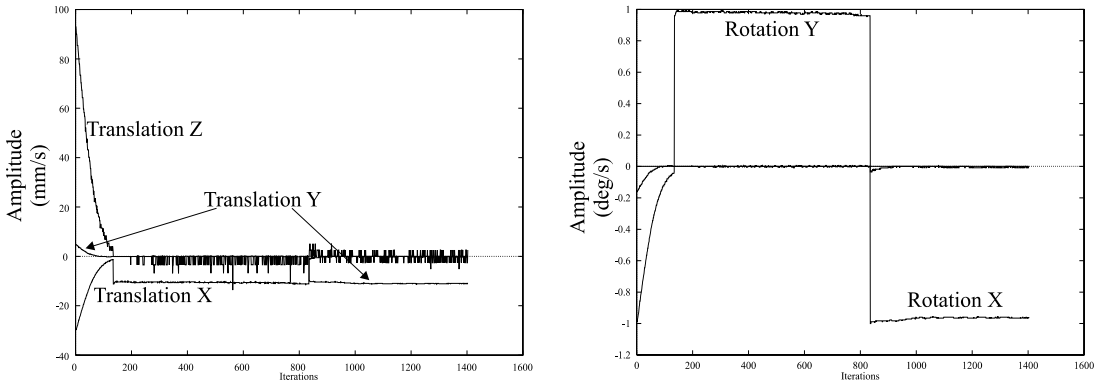
Comme précédemment, nous utilisons les paramètres suivants pour la loi de commande:

- Gain de boucle: $\lambda = 1$
- Pondération de la tâche secondaire: $\alpha = 0.8$
- Distance désirée entre l'objet et la caméra: $z^* = 0.58m$

La consigne visuelle a été spécifiée comme suit pour le positionnement

- Objet centré dans l'image,
- Largeur de l'objet: 30% de l'image soit 170 pixels,

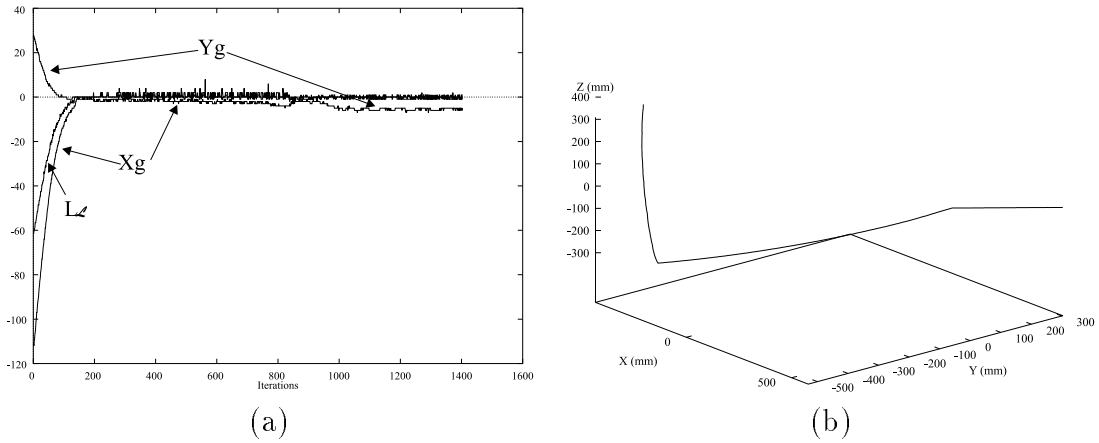
La vitesse appliquée à l'effecteur pour son déplacement est $T_x = -0.08m.s^{-1}$ et $T_y = -0.08m.s^{-1}$. Naturellement, l'axe de rotation est tout d'abord colinéaire à la hauteur puis à la largeur de l'objet.



(a) (b)

FIG. 3.27 – *Vitesses de translation et de rotation*

Nous constatons la bonne complémentarité de la translation suivant x et de la rotation suivant y. Il est toutefois à noter que le choix de z^* est dans le cas d'une tâche secondaire primordial. En effet, lors du calcul du projecteur orthogonal sur le noyau de la matrice d'interaction une bonne évaluation de la distance z^* permet d'assurer correctement la tâche de fixation. Dans notre cas, la valeur de z^* a été fixée de façon "artificielle" en ayant une connaissance *a priori* sur la scène observée.



(a) (b)

FIG. 3.28 – *(a) Erreur des différentes primitives. (b) Trajectoire 3D de l'effecteur.*

Aussi, cette valeur ne correspond pas exactement à la valeur réelle ce qui nous

permet d'observer dans l'image une légère erreur de la tâche de pointage (le centre du rectangle englobant est légèrement décalé) que l'on retrouve sur la figure représentant l'erreur (Fig. 3.28(a)). Une solution à ce type de problème serait d'effectuer une évaluation de la distance par une méthode de reconstruction comme celle proposée par (Boukir, 1993).

La seconde figure (Fig. 3.28(b)) montre les trois phases de la tâche robotique soit le positionnement, le mouvement latéral puis le mouvement vertical. Cet enchainement se retrouve sur la séquence d'image présentée sur la figure 3.29.

De plus, on constatera là encore que malgré la forme de l'objet (qui ne valide pas l'approximation $Y_{\mathcal{S}} \approx 0$), la convergence et les tâches secondaires sont bien réalisées.

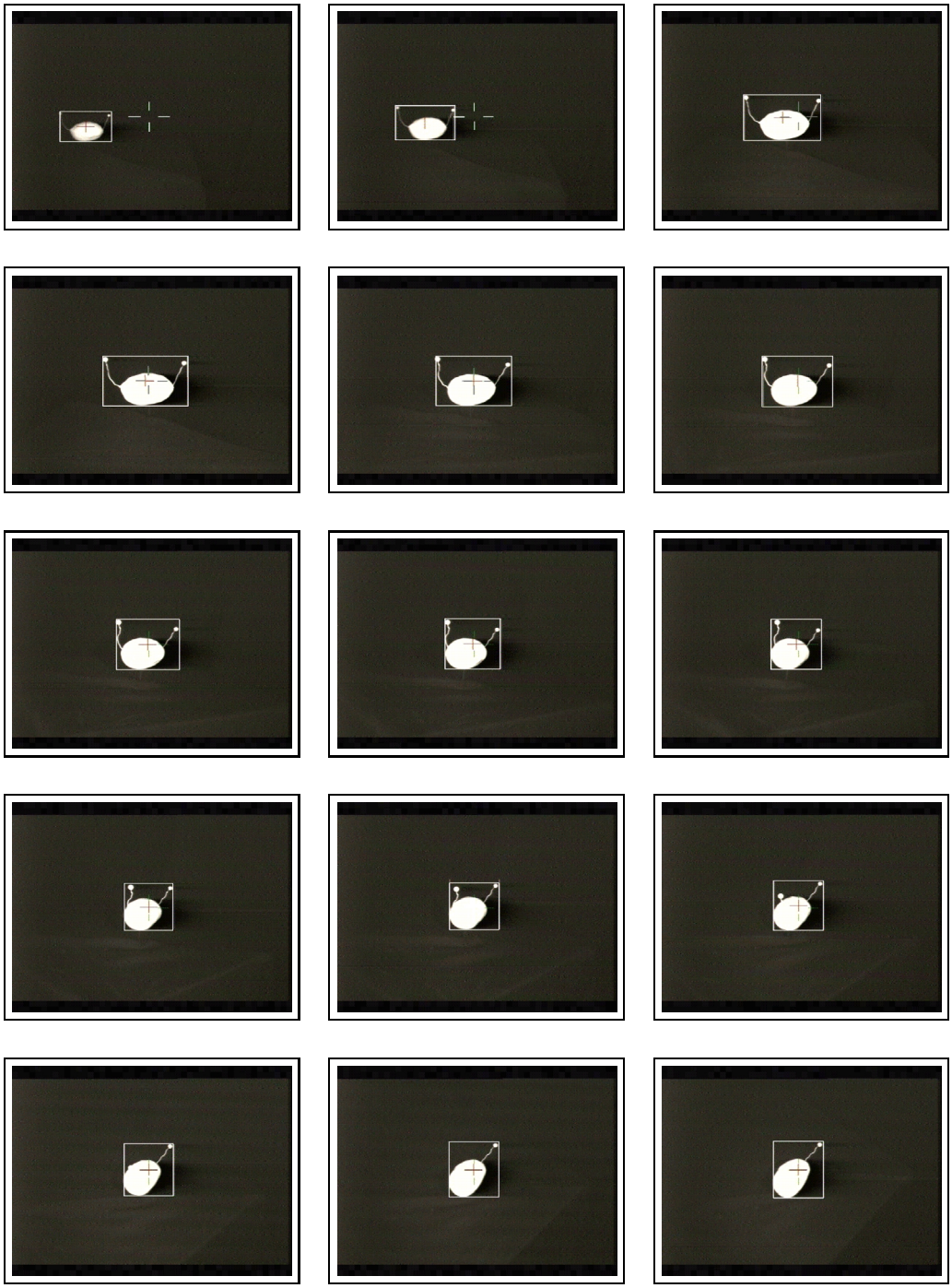


FIG. 3.29 – *Evolution de l'objet dans l'image durant le contournement.*

3.3.3.2 Girafe

Le même type de tâche que précédemment a été réalisé sur une petite girafe en caoutchouc ne présentant pas de points caractéristiques particuliers. Pour cela, nous effectuons deux mouvements enchaînés : un mouvement latéral puis un passage par dessus l'objet (Fig. 3.30)

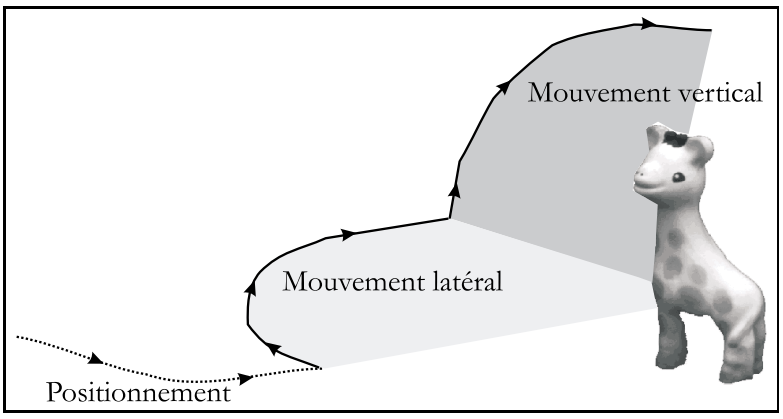


Fig. 3.30 – *Trajectoire effectuée autour de la girafe.*

Comme auparavant, nous utilisons les paramètres suivants pour la loi de commande:

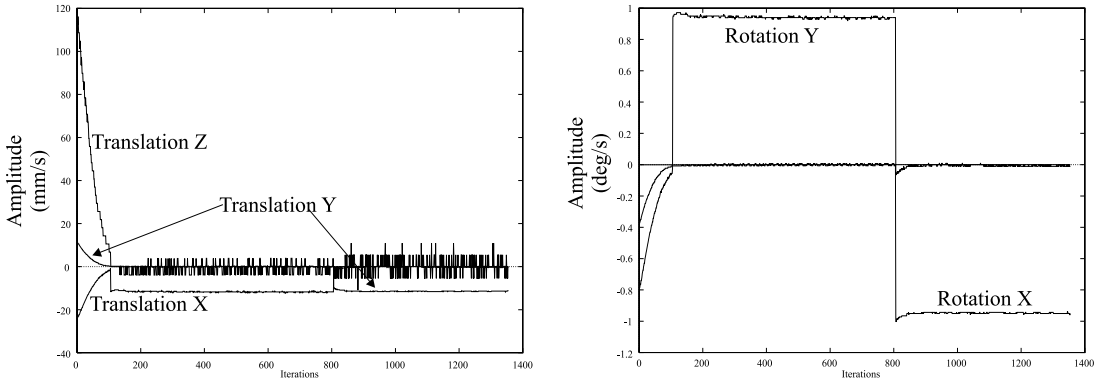
- Gain de boucle: $\lambda = 0.8$
- Pondération de la tâche secondaire: $\alpha = 1.0$
- Distance désirée entre l'objet et la caméra: $z^* = 0.7m$

La consigne visuelle a été spécifiée comme suit pour le positionnement

- Objet centré dans l'image,
- Hauteur de l'objet: 30% de l'image soit 170 pixels,

et la vitesse appliquée à l'effecteur pour son déplacement est $T_x = -0.08m.s^{-1}$ puis $T_y = -0.08m.s^{-1}$.

Les axes de rotation (dans l'image) seront donc respectivement la hauteur du rectangle englobant puis la largeur de ce rectangle. Le choix de la consigne visuelle pour le second mouvement est réalisé en mesurant la dernière valeur de la largeur

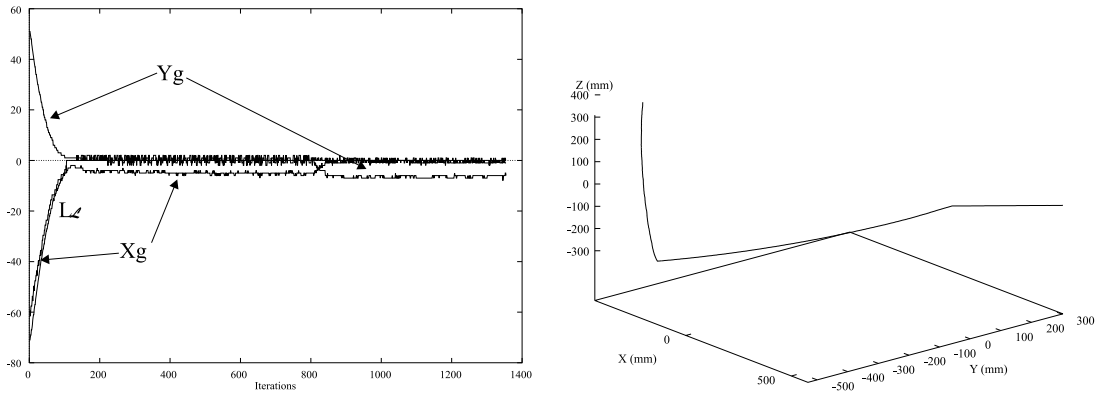


(a) (b)

FIG. 3.31 – *Vitesses de translation et de rotation*

durant le mouvement précédent. Ce choix nous permet ainsi de rester à la même distance de l'objet durant les deux mouvements.

La figure 3.31 présente les composantes du torseur cinématique suivant les axes de translation et de rotation. Toutefois, nous pouvons aussi noter le bruit nettement plus important dans le second mouvement de contournement. Ce dernier s'applique sur la translation en z ce qui nous amène à penser que la mesure de la largeur est nettement plus bruitée que celle de la hauteur. Cette remarque peut s'expliquer en considérant que nous travaillons sur une demi-trame vidéo ce qui a pour effet de "compresser" l'objet suivant la verticale. La hauteur utilisée dans la loi de commande se trouve donc "filtrée" puisque l'on n'utilise qu'une ligne sur deux.



(a) (b)

FIG. 3.32 – *(a)Erreur suivant les primitives. (b) Trajectoire 3D de l'effecteur.*

Après la tâche de positionnement les deux tâches de contournement sont réalisées

(Fig. 3.32.b). Ici encore, la remarque quant au choix de la valeur de z^* est primordiale et fixe les conditions d'une bonne réalisation de la tâche secondaire. Comme précédemment, nous pouvons remarquer une erreur résiduelle (Fig. 3.32.a) dûe à la méconnaissance de la profondeur z^*.

On constatera (Fig. 3.33), là encore, que la convergence durant le positionnement et les deux tâches secondaires sont bien réalisées.

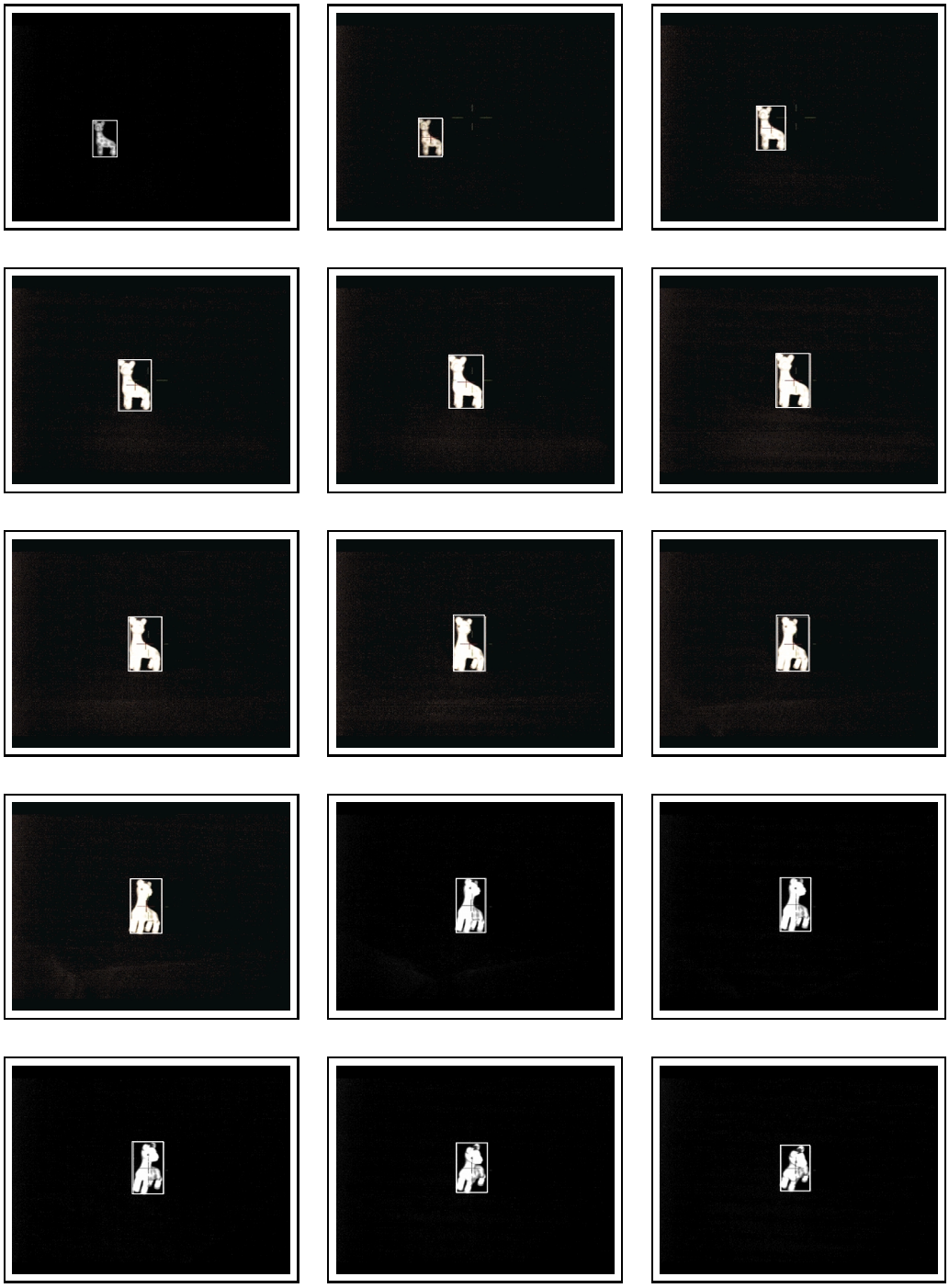


FIG. 3.33 – *Evolution de l'objet dans l'image durant le contournement.*

3.4 Conclusion

Dans ce chapitre, nous nous sommes attachés à proposer un ensemble de primitives visuelles permettant de réaliser différentes tâches robotiques sans connaissance *a priori* de la scène observée. L'approche que nous avons mise en œuvre repose sur la mesure des caractéristiques géométriques de l'objet que l'on désire observer, et permet ainsi d'éviter les problèmes liés aux conditions d'éclairement ou à la surface de l'objet.

Une étude des différentes relations d'interactions sur chacune des primitives considérées nous a alors permis de mettre en évidence les mouvements "autorisés" par chacune des tâches afférantes.

Toutefois, la méthodologie proposée demande encore quelques améliorations à deux niveaux :

- Tout d'abord, comme nous l'avons évoqué dans les résultats expérimentaux, il serait intéressant de disposer d'un estimateur de z nous permettant ainsi de pouvoir initialiser et réaliser les tâches de contournement sans erreur de suivi.
- En second lieu, une amélioration de la technique de seuillage relativement rudimentaire (mais donnant de bons résultats!), permettrait de rendre les mesures images tout à fait robustes aux différents bruits occasionnés par les conditions expérimentales.

Quelques pistes destinées à la poursuite de ces travaux ont alors été tracées et permettent de situer notre contribution au délicat problème qu'est le contournement d'objets par asservissement visuel.

En dernier lieu, une série d'expérimentations mettent en évidence la validité et la faisabilité de l'approche étudiée.

Conclusion

LE travail que nous venons de présenter porte sur la mise en œuvre de techniques d'asservissement visuel permettant d'effectuer des tâches de navigation autour d'un objet inconnu. Ce travail se doit donc de répondre à deux questions:

- Comment réaliser des déplacements contrôlés à l'aide d'une boucle d'asservissement visuel?
- Existe-t-il des primitives visuelles "universelles" communes à tous les types d'objets et quelles tâches robotiques peut-on réaliser avec?

Pour répondre au premier point, nous avons établi une loi de commande basée sur la notion de consigne variant dans le temps. L'approche que nous avons choisie est originale, dans le sens où elle permet de générer un mouvement de caméra à partir du contrôle d'une trajectoire dans l'espace image.

La commande que nous avons utilisée est une loi de type référencée vision, dans laquelle nous avons rajouté un terme permettant de prendre en compte le caractère dynamique de la consigne visuelle. Ce terme construit à partir de la dérivée de la consigne visuelle effectue une correction sur l'erreur de traînage permettant ainsi un bon suivi des motifs dans l'image. Toutefois, ces mouvements ne peuvent être correctement réalisés que si tous les degrés de liberté du robot sont contraints par la loi de commande. Cette condition est facilement réalisable en choisissant une primitive visuelle dont le paramètrage est de dimension suffisamment élevée.

Dans un second temps, nous avons proposé une méthode permettant de calculer les motifs visuels en fonction des mouvements que l'on désire appliquer à la caméra. Cette méthode a été mise en œuvre durant des simulations sur station de travail, mais aussi sur site réel. En considérant une cible parfaitement connue, il est alors possible de déplacer la caméra uniquement à partir des données visuelles. La cible peut alors être vue comme une balise visuelle permettant à la caméra de se situer. Ce type d'approche pourrait ainsi être étendu au cas de la robotique mobile où les déplacements de la plate-forme mobile seront contrôlés uniquement à partir de

l'interaction caméra/cible. Notons toutefois le problème d'observabilité de la cible qui n'a pas été abordé et qui constitue, dans ce cas de figure, une question cruciale.

Dans le cadre de notre étude, nous avons choisi d'utiliser un cube (objet simple) dont les faces sont facilement repérables (4 points caractéristiques). Les expérimentations ont permis de réaliser différents types de mouvements contrôlés à la cadence vidéo. Nous avons en particulier effectué le contournement du cube en mettant en œuvre un enchaînement de tâches référencées vision.

Cependant un large champ d'investigation reste encore à explorer.

- Tout d'abord, en ce qui concerne le déplacement autour d'un objet connu, il serait particulièrement intéressant de quantifier de façon théorique la précision des déplacements en fonction des primitives image et de la trajectoire à suivre.

- Un second point important concerne l'étude de trajectoires plus élaborées et par conséquent les problèmes de liaison inter-tâches. La solution que nous avons proposée, si elle a le mérite de fonctionner, n'est certainement pas optimale et demande de nombreuses améliorations.

- Enfin, il serait intéressant d'étudier l'influence du calibrage sur nos expérimentations, tant au niveau intrinsèque (distorsion, ...) qu'au niveau extrinsèque (calibrage bras/oeil).

En ce qui concerne le second point, nous avons considéré un objet totalement inconnu et ne présentant aucune "particularité visuelle"(pas de points caractéristiques). Il a alors été nécessaire de définir quelles primitives pouvaient être extraites de l'image, sans tenir compte de l'objet. A partir de ces primitives, nous avons alors établi les tâches robotiques réalisables, en considérant pour chacune d'elles le cas du positionnement et de la navigation.
Un algorithme de traitement d'images bas et moyen niveau a alors été implanté sur notre plate-forme expérimentale afin d'extraire en temps réel vidéo les différentes primitives. Nous avons alors pu réaliser plusieurs tâches de positionnement et de contournement autour de différents types d'objets. Cette approche a l'avantage de s'appliquer dans tous les cas de figure, dès lors que l'objet à contourner est facilement détectable. La méthodologie présentée peut ainsi être utilisée pour l'initialisation de processus de reconnaissance où il est alors nécessaire d'avoir une série de vues prises autour de l'objet à reconnaître.

Toutefois cette approche a laissé de nombreuses questions en suspens:

- Les contraintes dues à l'implantation *temps réel* de notre application nous ont amené à choisir des algorithmes relativement rudimentaires et présentant

quelquefois un manque de robustesse vis-à-vis des conditions expérimentales. Le développement d'algorithmes de traitement d'image plus élaborés sur de puissantes machines dédiées devrait permettre de se placer dans les conditions optimales pour la réalisation des différentes tâches robotiques.

- En second lieu, il serait intéressant d'étudier différents cas d'occultation tel que le contournement d'un objet fin et long dont on ne voit initialement que la plus petite face. Ce type de problème pourrait alors être géré par un automate à états finis déclenchant une stratégie de recul contrôlée par la longueur du segment invariant.

- Une suite logique de cette approche serait l'évaluation de la situation entre la caméra et l'objet. Pour ce faire, nous avons suggéré une approche basée sur l'évaluation de l'ellipsoïde d'inertie de l'objet. Cette méthode permettrait alors de paramétrer l'objet suivant son orientation (orientation des axes d'inertie) et le volume qu'il occupe (longueur des axes d'inertie).

- Au regard des difficultés rencontrées au niveau de la perception, il serait aussi intéressant de faire coopérer d'autres types de capteurs avec la caméra, afin de faciliter et d'accélérer la "connaissance" de la scène.

Vers la navigation "intelligente"...

En guise de perspective, nous proposons dans ce paragraphe quelques réflexions permettant d'orienter le lecteur sur quelques applications potentielles de ces travaux. En effet, sous ce titre prometteur nous entendons aborder les suites qui peuvent être données à l'approche étudiée précédemment et dont l'idée maîtresse demeure l' **asservissement visuel sur un objet quelconque**.

Effectuer des tâches de contournement en passant à gauche de l'objet, puis au dessus, puis en redescendant, etc... n'a en soit pas grand intérêt sinon la satisfaction de l'expérimentateur(!).

A contrario, arriver à pouvoir exprimer une situation entre la caméra et l'objet sans avoir de connaissances *a priori* sur l'objet et sans chercher à effectuer de reconstruction trop coûteuse en temps de calcul semble être un des prochains challenges pour les techniques d'asservissement visuel. Cependant ce genre d'approche nécessite indubitablement une coopération entre:

- des méthodes bas-niveau, comme celle que nous avons présentée, qui permettent de gérer les déplacements du capteur autour de forme inconnue,

- des techniques moyen-niveau qui utilisent ces déplacements pour obtenir une modélisation de l'objet,
- une plate-forme haut-niveau, dont le but est de pouvoir effectuer une cartographie de l'objet afin de planifier des actions utilisant la carte préalablement établie.

Situer l'objet

Lorsque l'on se trouve en face d'un objet quelqu'il soit, n'importe quel *Homo Sapiens* digne de ce nom, est tout de suite capable de dire s'il se trouve en face d'un objet plutôt long ou plutôt large. Et s'il désire donner de plus amples renseignements sur ce qu'il observe, il en effectuera alors sans doute le tour afin de s'en faire une idée plus complète. Les renseignements donnés par ce type d'observation seront alors du type : l'objet est posé à tel endroit, il est de forme carrée devant et plutôt allongée vers l'arrière.

Avec ces trois remarques, l'observateur définit ainsi la position, l'orientation et la forme globale de ce qu'il voit. Sans en donner une description précise s'attachant à considérer tous les détails, ces informations une fois quantifiées peuvent suffire pour se déplacer de façon intelligente autour de l'objet.

Partant de cet état de faît, nous considérons qu'il est nullement nécessaire d'avoir une connaissance exacte de l'objet pour se déplacer autour de celui-ci, mais plutôt d'avoir des informations sur sa disposition dans l'espace et sur le volume qu'il occupe (Exploration). A partir de ce type de connaissances, facilement quantifiables, nous disposons alors d'un vecteur de paramètres caractérisant l'objet et nous permettant alors de connaître la situation *Observateur-Observé* (Localisation).

Un certain nombre de réflexions nous ont amené à réfléchir sur l'idée de forme englobante, dont les caractéristiques (position, volume, ...) représenteraient de façon fidèle celles de l'objet observé. Cependant quelle forme utilisée? Une première solution est tout simplement d'utiliser une boîte s'ajustant au mieux avec les cotes de l'objet. Ainsi se déplacer autour de l'objet reviendrait à se déplacer autour de la boîte (Fig. 3.34).

Cependant le point dur de cette approche est l'élaboration de la boîte elle-même. Construire une boîte autour de l'objet revient à trouver une face de la boîte sur l'objet observé. Mais comment caractériser la face d'un objet?

Cette question apparemment simple est en réalité loin d'être triviale. Une approche naturelle consiste à utiliser des techniques de minimisation sur différents critères telles que la hauteur, la largeur ou l'orientation. Toutefois, suivant l'angle sous lequel est vue la scène, et en fonction de la position de l'objet, ce type de mé-

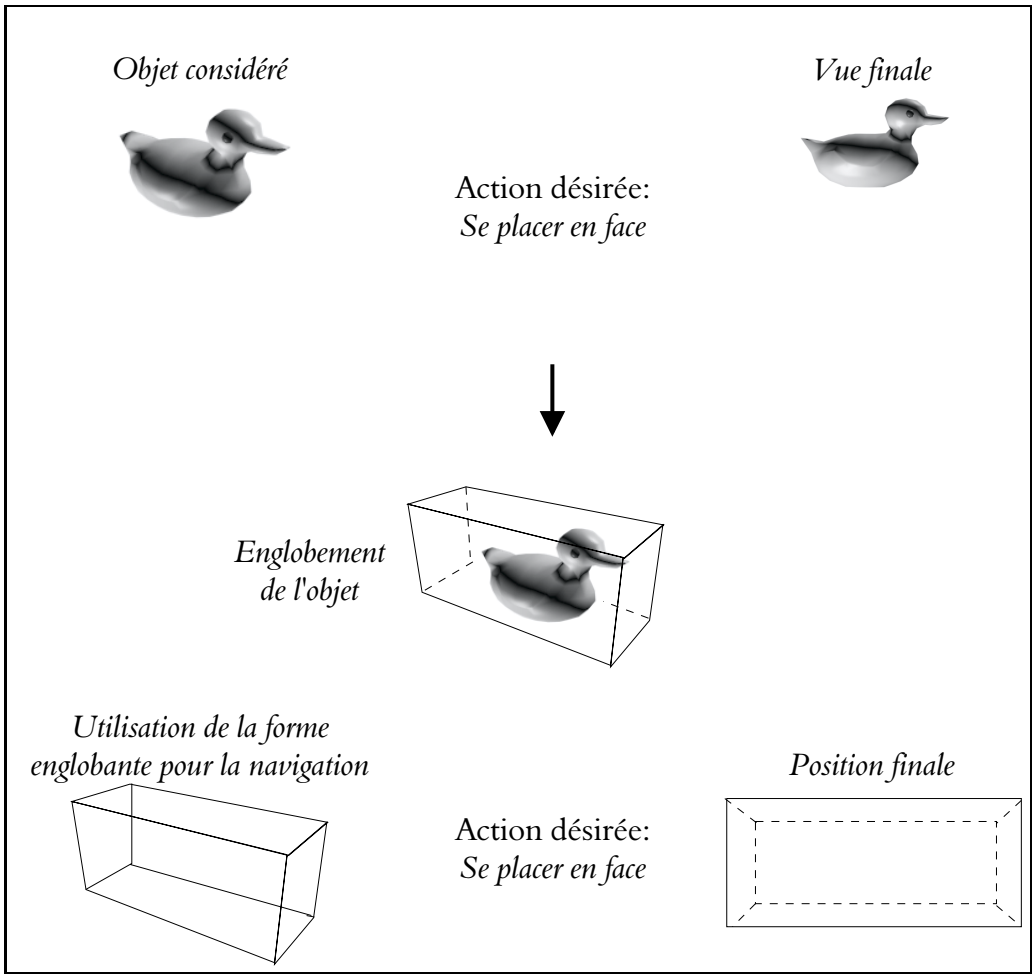


FIG. 3.34 – *Principe de la boîte englobante*

thode ne donne pas toujours les résultats escomptés. De plus, les critères que nous cherchons à minimiser sont des mesures dans l'image qui ont le mauvais goût d'être fortement bruitées.

Le fait d'utiliser une boîte rectangulaire présente aussi l'inconvénient d'être discontinu lors d'un changement de faces. Cette dernière remarque nous a alors amené à réfléchir à un autre type de forme englobante.

A notre sens, les surfaces quadriques et en particulier l'ellipsoïde présentent le meilleur compromis pour réaliser l'englobement de l'objet observé. Ces formes géométriques sont construites à partir de surfaces d'ordre deux et admettent pour équa-

tion générale dans le cas d'une forme centrée en l'origine:

$$\alpha x^2 + \beta y^2 + \gamma z^2 = \Gamma \tag{3.37}$$

Suivant la valeur des coefficients α, β, γ, ainsi que Γ, on obtient des surfaces sphériques, ellipsoïdales, cylindriques, paraboloïdales,...

La simplicité de l'équation générique 3.37 et la continuité de la surface nous amènent à penser que l'ellipsoïde (Fig. 3.35) serait une des formes les plus intéressantes.

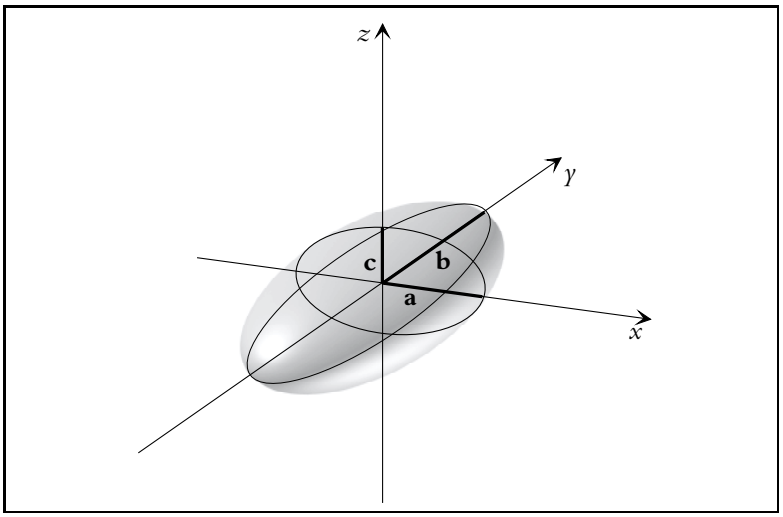


FIG. 3.35 – *Ellipsoïde général*

L'équation correspondant à ce type de forme est donnée par :

$$\frac{x^2}{\mathbf{a}^2} + \frac{y^2}{\mathbf{b}^2} + \frac{z^2}{\mathbf{c}^2} = 1 \tag{3.38}$$

où **a**, **b**, **c** désignent les axes principaux suivant respectivement x, y, z.

La question d'englober l'objet se présente sous un jour nouveau, puisque le but à atteindre n'est plus de rechercher une face de l'objet, mais d'aligner au mieux l'ellipsoïde avec l'objet. Quelques rudiments de mécanique nous montrent alors que cet alignement d'axe revient à chercher l'ellipsoïde d'inertie de l'objet. Cet ellipsoïde est la quadrique admettant pour axes principaux les axes d'inertie de l'objet.

Ainsi trouver les axes principaux de l'objet et leur orientation permet de connaître exactement la situation de l'objet par rapport à l'observateur.

L'idée que nous proposons, revient à utiliser l'ellipsoïde d'inertie comme une modélisation de l'objet. Le problème consiste alors à ajuster les 6 paramètres du système, à savoir les paramètres de l'ellipsoïde (**a**, **b**, **c**) et l'orientation des axes (θ_x, θ_y, θ_z). Ce "fittage" peut être fait itérativement durant le déplacement de la caméra, où pour une orientation donnée, nous pouvons mesurer dans l'image la disposition des axes d'inertie de l'objet et les comparer à ceux de l'ellipsoïde projeté.

Cette approche revient alors à un problème de minimisation sous la contrainte d'une bonne réalisation de la tâche visuelle (pointage et maintien de la distance).

Planifier la navigation autour de l'objet

La dernière étape du processus consiste à établir une "cartographie" de l'objet permettant de faciliter, voire d'optimiser, les mouvements que nous voulons effectuer autour. Une idée peut être de construire une carte de l'objet en échantillonnant l'objet suivant un ensemble de vues, puis d'associer à ces vues discrétisées des relations de connectivité.

Ainsi dans le cas très simple d'une forme discrétisée sous 6 angles de vues différents (cube) nous obtenons le cas de figure présenté figure 3.36.

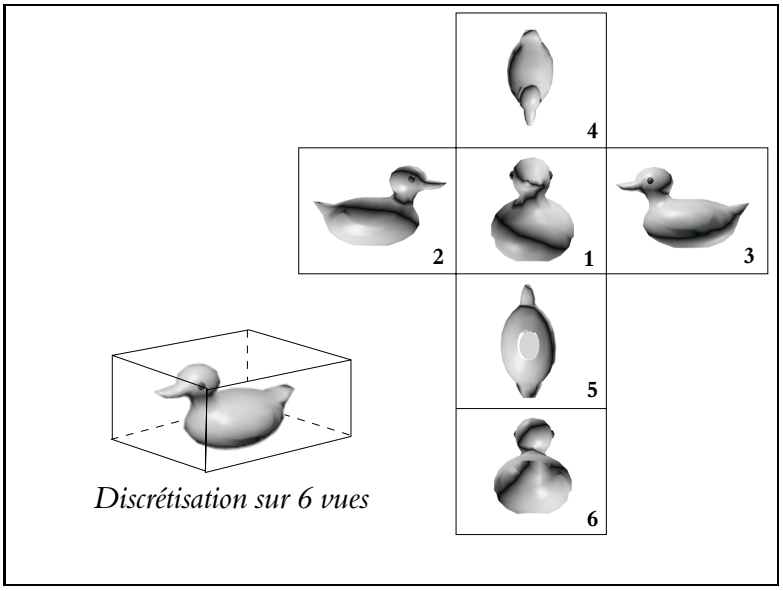


FIG. 3.36 – *Discrétisation d'un objet*

Nous pouvons alors décrire sous différentes formes (tableau, graphe, ...), les chemins nous permettant de transiter d'une vue à l'autre. Ainsi le passage de la vue 1

à 6 pourra se faire par les chemins 1-2-6, 1-3-6, etc...

Toutefois en évoquant ce problème de planification, nous sortons du cadre des techniques d'asservissement pour rentrer de plain-pied dans celui de L'Intelligence Artificielle. Aussi cantonnerons-nous ce petit paragraphe à ces quelques lignes permettant de donner au lecteur intéressé une idée sur les applications potentielles de notre approche.

Annexe A

Plateforme robotique du laboratoire

La plateforme robotique installée au LASMEA est constituée de deux parties:

- Un robot cartésien fabriqué par la société AFMA Robots,
- La machine de vision *Windis* développée au laboratoire en collaboration avec l'INRIA de Sophia-Antipolis.

A.1 Robot

Le robot que nous avons utilisé dans le cadre des travaux présentés dans ce manuscrit est un robot cartésien de type portique (Fig. A.1).

Les caractéristiques cinématiques de ce robot sont :

- Vitesse de translation maximum : 1m.s^{-1}
- Vitesse de rotation maximum : 2rad.s^{-1}

Le débattement suivant chacun des axes est de:

Translation suivant X : 1,5m	Rotation suivant A : $\pm 187{,}5^o$
Translation suivant Y : 1,5m	Rotation suivant B : $\pm 130^o$
Translation suivant Z : 1m	Rotation suivant C : $\pm 130^o$

Chacun de ces axes est commandé par un variateur de vitesse *Télémécanique*. La commande numérique développée par la société *Cap SESA* a été implémentée

sur une carte du commerce MVME147 architecturée autour d'un microprocesseur 68030. Le noyau d'asservissement du robot est cadencé à 5ms dans l'environnement temps réel VxWorks de Wind River Systems.

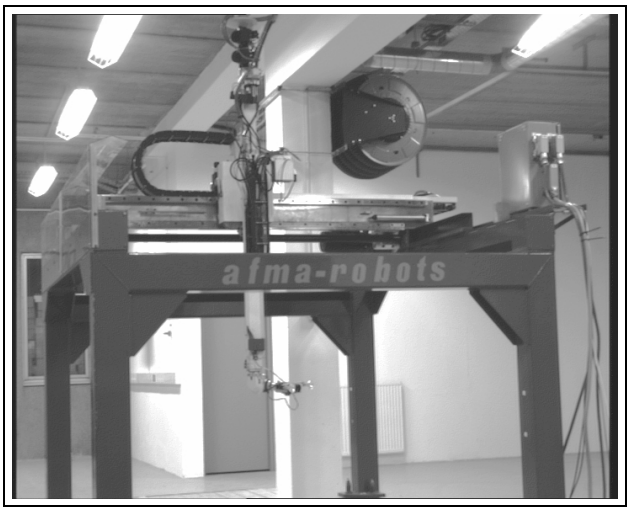


FIG. A.1 – *Vue générale du robot cartésien*

A.2 Système de traitement d'image *Windis.*

La caméra CCD embarquée sur l'effecteur du robot transmet directement l'image au système de vision *Windis*. Dans le cadre de nos travaux, cette caméra a été équipée d'un objectif 8mm, et deux spots halogènes permettant d'éclairer la scène à étudier.

En ce qui concerne le traitement d'image, l'objectif temporel visé dans ce type de tâche robotique est le temps réel vidéo. L'architecture *Windis* permet de satisfaire ce type de contraintes, en ne travaillant que sur des zones d'intérêt. On introduit ici la notion de **fenêtres actives**.

Cette dénomination est utilisée dans le sens où tous les paramètres définissant la fenêtre peuvent être modifiés dynamiquement pendant la séquence vidéo. De plus, afin d'augmenter la puissance de calcul, WINDIS repose sur une architecture parallèle et hybride gérée avec l'exécutif temps réel VxWorks.

La machine WINDIS est construite autour de trois modules principaux qui sont :

- WINDIS (WINdow DIStributor subsystem) : ce module élaboré à partir d'une structure pipeline cablée réalise l'extraction des fenêtres d'interêt et différents

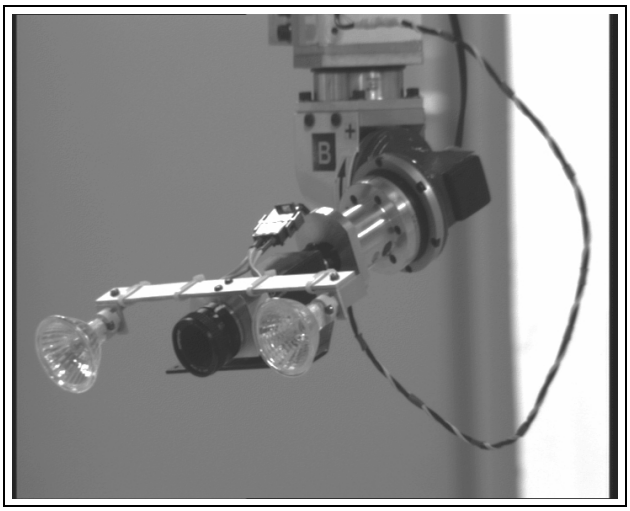


Fig. A.2 – *Caméra fixée sur l'effecteur*

traitements bas niveau. Le cœur de cette carte est constitué de 2 convolueurs IMS-A110 (Inmos) permettant d'appliquer différents masques de convolution (3×3, 2×3×3, 5×5). Cette carte permet aussi d'extraire des points candidats à la sortie des convolueurs suivant différents critères (Min/Max, Hystérésis, ...)

– WINPROC (WINdow PROCessing subsystem) : ce module effectue les traitements moyen niveau. Sa structure MIMD [1] programmable est constituée de quatre DSP [2] 96002 de Motorola sur lesquels sont implantés des algorithmes permettant de calculer les paramètres géométriques des primitives à partir des données issues du module WINDIS.

– WINMAN (WINdow MANager subsystem) : ce dernier module constitue la partie haut niveau du système de traitement d'image. La carte utilisée ici est un module commercialisé par Motorola (MVME165) et élaboré autour d'un 68040. Le rôle de ce sous-système est d'une part de contrôler les modules bas et moyen niveau et d'autre part de réaliser le calcul de la commande du robot à partir des données visuelles traitées.

1. Multiple Instruction stream, Multiple Data stream
2. Digital Signal Processing

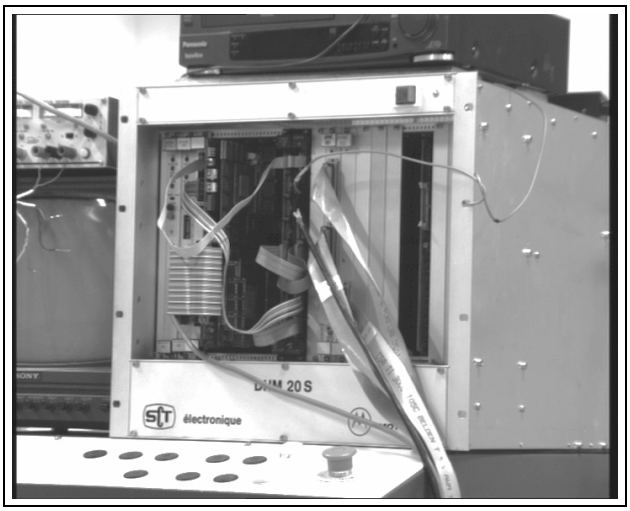


FIG. A.3 – *Machine de vision Windis*

A.3 L'environnement de travail

Les algorithmes sont développés en langage C sur station SUN. Ceux-ci sont téléchargés par réseau ethernet sur le RACK de développement. Celui-ci est relié à un moniteur avec lequel nous pouvons contrôler le robot et lancer les différents programmes de vision et de commande.

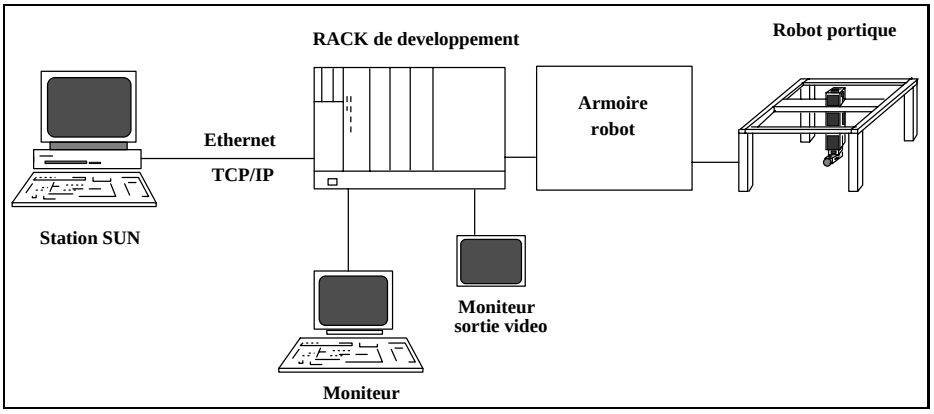


FIG. A.4 – *Environnement de travail.*

A.4 Architecture d'une application

Une application met en parallèle un processus de commande et un processus de vision. Le processus de commande est matérialisé par un noyau qui est activé par une horloge temps réel toutes les 5ms. Il réactualise les consignes articulaires du robot et prend l'état du robot (positions articulaires) à la même cadence.

Le processus de vision, quant à lui, définit des zones d'interêts dans l'image, gére ces zones et élabore la commande à une cadence de 40ms. L'échange entre les deux processus est réalisé à l'aide d'une mémoire partagée constituée d'une double banque de données.

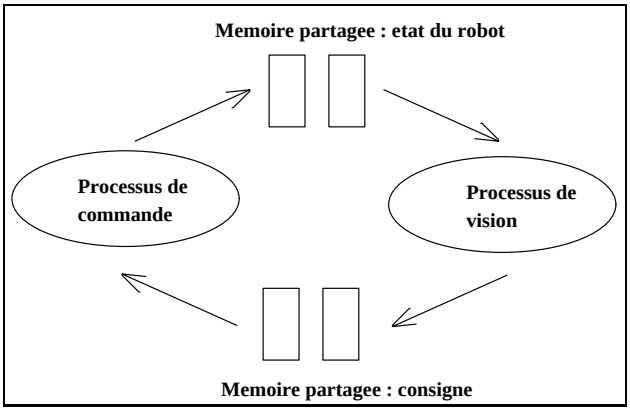


FIG. A.5 – *Echange entre les deux processus.*

Annexe B

Simulateur pour la Commande Référencée Vision

Afin de valider nos différentes lois de commande, nous avons développé un simulateur robotique. Ce simulateur a été programmé en langage C sous l'environnement XMotif. La plateforme de développement est une station de travail HP. Notons toutefois que ce simulateur est tout à fait portable sur d'autres plateformes (Sun, SGI).

B.1 Principe

Dans sa version actuelle, ce programme permet de simuler des boucles d'asservissement visuel telle que celle représentée sur la figure ci-dessous.

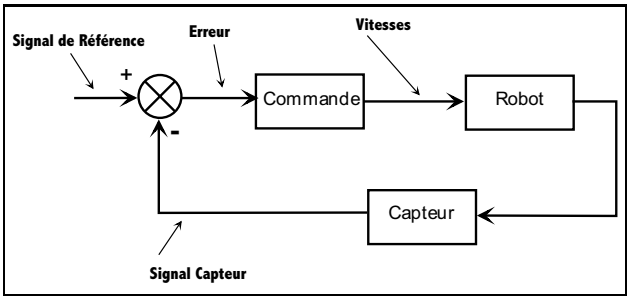


FIG. B.1 – *Commande Référencée Capteur.*

Sur la Figure B.1, on distingue 3 blocs:

– Le bloc Commande est constitué d'une cinquantaine de lignes de code C **di-**

rectement transférable sur la plateforme expérimentale. L'implantation de ce bloc, se résume à une boucle de type do...while. Il comporte une partie concernant l'initialisation des différentes matrices et une description de la loi de commande à réaliser.
On a un bloc du type:

Initialisation des différentes matrices et de la position d'origine.

DO

Calcul de la position du capteur.

Calcul des signaux délivrés par le capteur.(Visualisation)

Calcul de la matrice d'interaction (si nécessaire).

Loi de commande:

$e = C(s - s^*)$

$T = -\lambda L^{T+}(s - s^*)$

$\vdots$

Insertion d'un retard (si nécessaire).

WHILE(condition de fin)

- Le but du bloc Robot est d'évaluer la position de l'effecteur supportant le capteur par rapport à la cible d'asservissement. Pour cela on recalcule à chaque itération la position de l'effecteur à partir du torseur cinématique et de la période d'échantillonnage. Dans sa version actuelle, ce module permet de modifier la dynamique des axes du robot en faisant réagir ces derniers comme des systèmes du 1^{er} ordre. Des commandes simulant différents types de bruit (uniforme, gaussien) permettent aussi de bruiter les différentes consignes appliquées aux 6 axes. Notons enfin la possibilité d'introduire une matrice de couplage entre les axes.

- Le bloc Capteur permet de simuler soit une caméra soit un capteur 3D. Les paramètres de caméra modifiables sont les focales suivant x et y, ainsi que l'origine de la matrice CCD. Dans ce module, il est possible de bruiter les données visuelles et de simuler l'effet d'échantillonage de la matrice CCD.

B.2 Interface utilisateur

Une interface graphique a été réalisée afin de faciliter l'accès au programme.

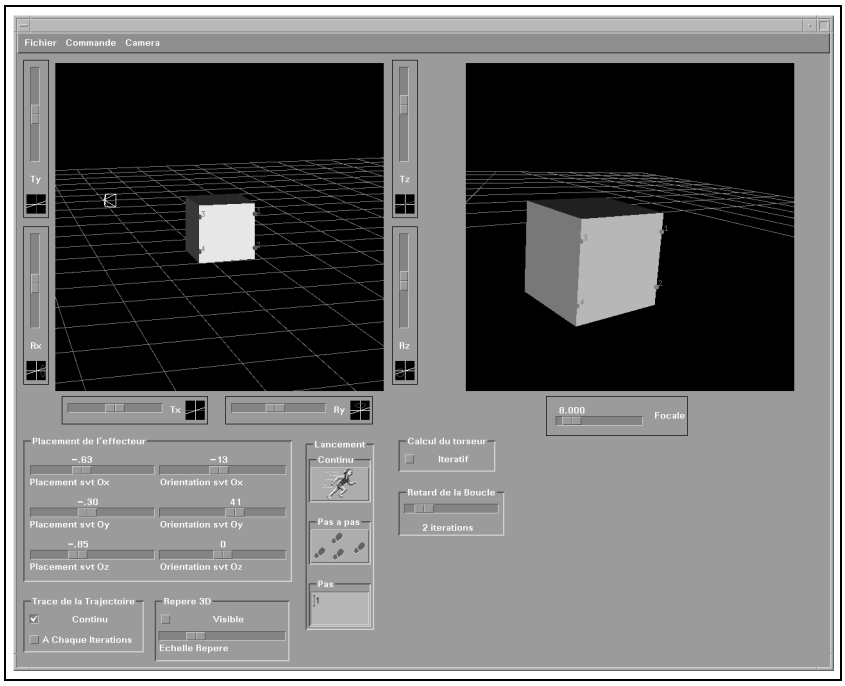


Fig. B.2 – *Interfaçage graphique.*

Cette interface comporte deux parties correspondant à une vue extérieure de la scène et à l'image de la scène dans laquelle sont faites les mesures. Différents curseurs permettent de placer l'effecteur et de modifier différents paramètres (Focale, Gain, ...)

Bibliographie

Abrams, S., Allen, P. and Tarabanis, K.: 1993. Dynamic sensor planning. *DARPA93*. pp. 599–607.

Agin, G. J.: 1977. Servoing with visual feedback. *Human Age and Robot: 7th International Symposium on Industrial Robots*. Tokyo, Japan. pp. 551–560.

Agin, G. J.: 1979. Real-time control of a robot with a mobile camera. *International Symposium on Industrial Robots*. Washington DC, USA. pp. 233–246.

Allen, P. K., Timcenko, A., Yoshimi, B. and Micheman, P.: 1993. *Hand-eyes coordination for robotic tracknig and grasping, Visual Servoing, Hashimoto éd.*. Vol. 7. World Scientific Press, Singapore. pp. 33–69.

Allen, P., Timcenko, A., Yoshimi, B. and Michelman, P.: 1992. Real time visual servoing. *ICRA'92*. IEEE International Conference on Robotics and Automation. pp. 1850–1856.

Allen, P., Yoshimi, B. and Timcenko, A.: 1991. Real time visual servoing. *ICRA'91*. IEEE International Conference on Robotics and Automation. pp. 851–856.

Aloimonos, Y.: 1990. Purposive and qualitative active vision. *IAPR'90*. Vol. 1. IAPR International Conference on Pattern Recognition. Atlantic City, USA. pp. 346–360.

Aloimonos, Y., Weiss, I. and Bandopadhay, A. 1987. Active vision. *International Journal on Computer Vision* **1**(4), 333–356.

Arkin, J.: 1987. Motor schema based navigation for a mobile robot. *ICRA'87*. IEEE International Conference on Robotics and Automation. Raleigh, USA. pp. 264–271.

Bajcsy, R.: 1988. Active perception. *Proceeding IEEE special Computer Vision, Invited paper*. Vol. 76. pp. 996–1005.

Berry, F., Martinet, P. and Gallice, J.: 1997. Trajectory generation by visual servoing. *IROS'97*. Vol. 2. IEEE International Symposium on Robot and System. Grenoble, France. pp. 1065–1071.

Berry, F., Martinet, P. and Gallice, J.: 1998. Visual servoing around a complex object. *MVA'98*. IAPR Workshop on Machine Vision Applications. Chiba, Japon. pp. 254–257.

Boukir, S.: 1993. *Reconstruction 3D d'un environnement statique par vision active*. PhD thesis. Université de Rennes I, IRISA.

Brooks, R.: 1985. A robust layered control system for a mobile robot. *Technical Report memo 864*. MIT, Artificial Intelligence Laboratory.

Castano, A. and Hutchinson, S. 1994. Visual compliance: Task-directed visual servo control. *IEEE Transactions on Robotics and Automation* **10**(3), 334–342.

Chaumette, F.: 1990. *La relation vision-commande: théorie et applications à des tâches robotiques*. PhD thesis. Université de Rennes I, IRISA.

Chaumette, F.: 1998. *Potential problems of stability and convergence in image-based and position-based visual servoing. The Confluence of Vision and Control, D. Kriegman, G. Hager, A.S. Morse (eds.)*. number 237. LNCIS Series, Springer-Verlag. pp. 66–78.

Chaumette, F. and Santos, A.: 1993. Tacking a moving object by visual servoing. *12th World Congress IFAC*. Vol. 3. Sydney, Australie. pp. 643–648.

Chaumette, F., Rives, P. and Espiau, B.: 1991. Positionning of a robot with respect to an object, tacking it and estimating its velocity by visual servoing. *ICRA'91*. Vol. 3. IEEE International Conference on Robotics and Automation. Sacramento, USA. pp. 2248–2253.

Collewet, C., Chaumette, F., Wallian, L. and Marchal, P.: 1998. Positionnement par rapport à un objet plan de forme inconnue par asservissement visuel 2d. *Technical Report RR-3419*. Unité de recherche INRIA, IRISA, Rennes.

Corke, P. I.: 1996. *Visual control of robots. High-performance visual servoing*. number ISBN : 0-86380-207-9. Research Studies Press LTD, Somerset, England.

Cretual, A.: 1998. *Utilisation d'informations visuelles dynamiques en asservissement visuel*. PhD thesis. Université de Rennes I, IRISA.

Cretual, A. and Chaumette, F.: 1997. Positioning a camera parallel to a plane using dynamic visual servoing. *IROS'97*. Vol. 1. IEEE International Symposium on Robot and System. Grenoble, France. pp. 43–48.

Cretual, A. and Chaumette, F.: 1998. Image-based visual servoing by integration of dynamic measurements. *ICRA'98*. IEEE International Conference on Robotics and Automation. Louvain, Belgique.

Dickmanns, E. and Graefe, V.: 1988. Dynamic monocular machine vision. *MVA'88*. Vol. 1. IAPR Workshop on Machine Vision Applications. pp. 223–240.

Dombre, E. and Khalil, W.: 1988. *Modélisation et commande de robots*. number ISBN : 2-86601-142-2. Traité des nouvelles technologies, Série robotique, Hermès.

Dreschler, L. and Nagel, H. H.: 1981. Volumetric model and 3d trajectory of a moving car derived from monocular tv frame sequences of a street scene. *Proceedings of the International Joint Conference on Artificial Intelligence*. Vancouver, Canada. pp. 692–697.

Edwards, J.: 1996. An active, appearance-based approach to the pose estimation of complex objects. *IROS'96*. IEEE International Symposium on Robot and System. Osaka, Japon.

Espiau, B., Chaumette, F. and Rives, P. 1992. A new approach to visual servoing in robotics. *IEEE Transactions on Robotics and Automation* **8**, 313–326.

Fagerer, C., Dickmanns, D. and Dickmanns, E. 1994. Visual grasping with long delay time of a free floating object in orbit. *Autonomous Robots*.

Feddema, J. and Mitchell, O. 1989. Vision-guided servoing with feature-based trajectory generation.. *IEEE Transactions on Robotics and Automation* **5**, 99–108.

Feddema, J., Lee, C. and Mitchell, O. 1991. Weighted selection of image features for resolved rate visual feedback control. *IEEE Transactions on Robotics and Automation* **7**, 31–47.

Gilbert, A., Giles, M., Flachs, G., Rogers, R. and Yee, H. 1980. A real time video tracking system. *IEEE Transactions on Pattern Analysis and Machine intelligence* **2**(1), 47–56.

Grosso, E., Andrea, M. and Sandini, G. 1996. Robust visual servoing in 3d reachong tasks. *IEEE Transactions on Robotics and Automation* **12**, 732–742.

Grosso, E., Metta, G., Oderra, A. and Sandini, G.: 1995. Uncalibrated visual servoing in reaching tasks. *Technical Report 3/95*. LIRA Lab., DIST Université de Géne.

Hager, G. and Toyama, K. 1998. X vision: A portable substrate for real-time vision applications. *Computer Vision and Image Understanding* **69**(1), 23–37.

Horn, B. K. P. and Schunk, B. G. 1981. Determining optical flow. *Artificial Intelligence* **17**, 185–203.

Hosoda, K. and Asada, M.: 1994. Versatile visual servoing without knwledge of true jacobian. *IROS'94*. IEEE International Symposium on Robot and System. Munich, Allemagne. pp. 186–193.

Jagersand, M. and Nelson, R.: 1994. Adaptative differential visual feedback for uncalibrated hand-eye coordination and motor control. *Technical Report TR-579*. Depart. of Computer Science, Univ. of Rochester.

Jägersand, M., Fuentes, O. and Nelson, R.: 1997. Experimental evaluation od uncalibrated visual servoing for precision manipulation. *ICRA'97*. Vol. 3. IEEE International Conference on Robotics and Automation. Albuquerque, USA. pp. 2874–2880.

Janabi-Sharifi, F. and Wilson, W. J. 1997. Automatic selection of image features for visual servoing. *IEEE Transactions on Robotics and Automation* **13**(6), 890–903.

Khadraoui, D., Motyl, G., Martinet, P., Gallice, J. and Chaumette, F. 1996. Visual servoing in robotics scheme using a camera/laser-stripe sensor. *IEEE Transactions on Robotics and Automation* **12**(5), 743–750.

Kitchen, L. and Rosenfeld, A.: 1980. Gray-level corner detection. *Technical Report CS-TR-887*. Computer science, University of Maryland, College Park.

Lacroix, S.: 1995. *Stratégies de perception et de déplacement pour la navigation d'un robot mobile autonome en environnement naturel*. PhD thesis. Université Paul Sabatier, LAAS.

Latombe, J. C.: 1991. *Robot motion planning*. number ISBN : 0-7923-9129-2 in *Robotics : Vision, manipulation and sensors*. Kluwer Academic Press. Norwell, Massachusetts.

Marchand, E.: 1996. *Stratégies de perception par vision active pour la reconstruction et l'exploration de scènes statiques*. PhD thesis. Université de Rennes I, IRISA.

Martinet, P., Berry, F. and Gallice, J.: 1996a. Use of first derivative of geometrical features in visual servoing. *ICRA'96*. Vol. 4. IEEE International Conference on Robotics and Automation. Minneapolis, USA. pp. 3413–3419.

Martinet, P., Gallice, J. and Khadraoui, D.: 1996b. Vision based control law using 3d visual features. *ISRAM'96*. Vol. 3. Second World Autonomous Congress. Montpellier, FRANCE. pp. 497–502.

Martinet, P., Rives, P., Fickinger, P. and Borrelly, J. J.: 1991. Parallel architecture for visual servoing applications. *CAMP'91.* Proceedings of the Workshop on Computer Architecture for Machine Perception. Paris, France. pp. 407–418.

Moravec, H.: 1980. *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover.* PhD thesis. Standford University.

Motyl, G.: 1993. *Couplage d'une caméra et d'un faisceu Laser en commande référencée vision.* PhD thesis. Thèse de l'université Blaise Pascal de Clermont Ferrand, LASMEA.

Mukundan, R. and Ramakrishnan, R. 1995. Fast computation of legendre and zernike moments. *Pattern Recognition* **28**(9), 1433–1442.

Murase, H., Kimura, F., Yoshimura, M. and Miyake, Y. 1981. An improvement of the auto-correlation matrix in patternmatching method and its application to handprinted. *Trans. IECE* **J64-D**(3), 276–283.

Nayar, S., Murase, H. and Nene, S.: 1994. Learning, positionning and tracking visual appearance. *ICRA'94.* IEEE International Conference on Robotics and Automation. San Diego, USA. pp. 3237–3244.

Nayar, S., Nene, S. and Murase, H.: 1995. Subspace methods for robot vision. *Technical Report CUCS-06-95.* Depart. of Computer Science, Columbia Univ.

Nayar, S., Nene, S. and Murase, H.: 1996. Real-time 100 object recognition system. *ICRA'96.* IEEE International Conference on Robotics and Automation. pp. 2321–2325.

Novales, C.: 1994. *Pilotage par actions réflexes et navigation locale de robots mobiles rapides.* PhD thesis. Université de MontpellierII.

Oja, E.: 1983. *Subspace methods of pattern recognition.* Research Studies Press. Hertfordshire.

Papanikolopoulos, N.: 1992. Shared and traded telerobotic visual control.. *ICRA'92.* Vol. 1. IEEE International Conference on Robotics and Automation. Nice, France. pp. 878–885.

Papanikolopoulos, N. 1995. Selection of features and evaluation of visual measurements during robotic visual servoing tasks. *Journal of Intelligent and Robotic Systems* **13**, 279–304.

Papanikolopoulos, N., Nelson, D. J. and Khosla, P. K. 1995. Six degree-of-freedom hand/eye visual tracking with uncertain parameters. *IEEE Transactions on Robotics and Automation* **11**(5), 725–732.

Papanikolopoulos, N. P., Khosla, P. K. and Kanade, T. 1993. Visual tracking of a moving target by a camera mounted on a robot : A combination of control and vision. *IEEE Transactions on Robotics and Automation* **9**(1), 14–35.

Payton, D.: 1986. An architecture for reflexive autonomous vehicle control. *ICRA '86.* IEEE International Conference on Robotics and Automation. San Francisco, USA. pp. 1838–1845.

Pissard-Gibollet, R.: 1993. *Conception et commande par asservissement visuel d'un robot mobile.*. PhD thesis. Thèse de l'école des mines de Paris, Sophia-Antipolis.

Prajoux, R. E.: 1979. *Visual tracking.* SRI International. pp. 17–37.

Prokop, R. J. and Reeves, A. P. 1992. A survey of moments-based techniques for unoccluded object representation and recognition. *Computer Vision, Graphics, and Image Processing* **54**(5), 438–460.

P.Y.Coulon and M.Nougaret: 1983. *Use of a TV camera system in closed-loop position control of mechanisms.* IFS Publications. pp. 117–127.

Rives, P. and Borrelly, J.: 1997. Underwater pipe inspection task using visual servoing techniques. *IROS'97.* Vol. 1. IEEE International Symposium on Robot and System. Grenoble, France. pp. 63–68.

Rizzi, A. and Koditschek, D. 1996. An active visual estimator for dexterous manipulation. *IEEE Transactions on Robotics and Automation* **12**(5), 697–713.

Salganicoff, M., Metta, G., Oddera, A. and Sandini, G.: 1993. A direct approach to vision guided manipulation. *ICAR'93.* icar.

Samson, C., Espiau, B. and Borgne, M. L.: 1991. *Robot control: The task function approach.* number ISBN : 0-19-853805-7 in *The Oxford Engineering Sciences.* Oxford university press.

Sanderson, A., Weiss, L. and Neuman, C. 1987. Dynamic sensor-based control of robots with visual feedback. *IEEE Transactions on Robotics and Automation* **3**, 404–417.

Shirai, Y. and Inoue, H. 1973. Guiding a robot by visual feedback in assembly tasks.. *Pattern Recognition* **5**, 99–108.

Suh, H. I.: 1993. *Visual servoing of robot manipulators by fuzzy menbership function based neural networks.* Vol. 7. World Scientific Press, Singapore. pp. 285–315.

Sundareswaran, V., Bouthemy, P. and Chaumette, F. 1996. Exploiting image motion for active vision in a visual servoing framework. *Int. Journal of Robotics Research* **15**(6), 629–645.

Takahashi, I. and Deguchi, K.: 1998. Image-based control of robot and target object motions by eigen space methods. *MVA'98*. IAPR Workshop on Machine Vision Applications. Chiba, Japon. pp. 472–475.

Thorpe, C. E.: 1984. *Vision and Navigation for a robot rover*. PhD thesis. Carnegie Mellon University.

Tsai, W. 1985. Moment preserving thresholding: A new approach. *Computer Vision, Graphics, and Image Understanding* **29**, 377–393.

Weiss, L.: 1984. *Dynamic Visual Servo Control of Robots: An Adaptative Image-Based Approach*. PhD thesis. Carnegie-Mellon University.

Westmore, D. B. and Wilson, W. J.: 1991. Direct dynamic control of a robot using an end-point mounted camera and kalman filter position estimation. *ICRA'91*. IEEE International Conference on Robotics and Automation. pp. 2376–2384.

Wilson, W.: 1993. *Visual servo control of robots using kalman filter estimates of robot pose relative to work-pieces*. Vol. 7. World Scientific Press, Singapore. pp. 71–104.

Wilson, W., Hulls, C. and Bell, G. 1996. Relative end-effector control using cartesian position based visual servoing. *RA* **12**(5), 684–696.

Xie, M.: 1989. *Contribution à la vision dynamique : reconstruction d'objets 3D polyhédriques par une caméra mobile*. PhD thesis. Université de Rennes I, IRISA.

Yoshimi, B. and Allen, P.: 1996. Closed-loop visual grasping and manipulation. *ARPA96*. pp. 1353–1360.

Zapata, R.: 1991. *Quelques aspects topologiques de la planification de mouvements et des actions réflexes en robotique mobile*. PhD thesis. Université de MontpellierII.

Zheng, J. Y., Chen, Q. and Tsuji, S.: 1991. Active camera guided manipulation. *ICRA'91*. IEEE International Conference on Robotics and Automation. pp. 632–638.