Modélisation et reconnaissance active d'objets 3D de forme libre par vision en robotique

Felipe de Jesus Trujillo-Romero

THESE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivrée par l' *Institut National Polytechnique de Toulouse*

Discipline ou spécialité : *Systèmes Informatiques Critiques*

Présentée et soutenue par *Felipe de Jesús Trujillo Romero*
Le *10 décembre 2008*

Titre : *Modélisation et Reconnaissance active d'objets 3D de forme libre par Vision en Robotique*

Directeur de Thèse : *M.Devy*

JURY :

Président :
M. CATTOEN, professeur INPT, Toulouse

Rapporteurs :
F. CHAUSSE, maître de conférences HDR, Univ. Clermont Ferrand
M. MALLEM, professeur Univ.Evry Val d'Essonne

Examinateur:
M. DEVY, DR, LAAS-CNRS, Toulouse

Ecole doctorale : Ecole Doctorale de Systèmes (EDSYS)
Unité de recherche : Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS-CNRS)

Remerciements

Agradezco principalmente al Consejo Nacional de Ciencia y Tecnología CONACyT por los recursos otorgados mediante la beca 167939/200653 para la realización de la presente tesis de grado doctoral.

Je veux remercier, en premier lieu, mon encadrant Michel DEVY pour sa patience et pour sa direction de mes travaux tout au long du déroulement de cette thèse.

Je suis très reconnaissant envers Monsieur Raja CHATILA pour m'avoir accueilli au sein du groupe RIA du LAAS-CNRS afin de pouvoir y préparer mon doctorat.

Merci à mes parents et à ma famille pour toute la confiance, amour et compréhension inconditionnel qu'ils m'ont toujours donnés, particulièrement dans les moments difficiles de mon séjour en France.

Je ne peux pas oublier ma femme et mon fils qui m'ont donné les forces et le courage pour traverser cette étape de ma vie.

Je tiens à remercier mes amis et collègues pour les moments partagés au laboratoire pendant près des 4 ans et demi de mon séjour au LAAS.

Finalement, je veux exprimer toute ma gratitude à tous ceux qui ont contribué d'une façon ou d'une autre à l'obtention de ce doctorat, que cela soit à titre professionnel et à titre personnel.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

La présente thèse traite de la reconstruction d'objets 3D de forme libre ainsi que de la reconnaissance de ces objets 3D, dans le contexte de tâches de manipulation exécutées par un robot personnel, un robot capable d'exécuter des tâches au domicile d'une personne (handicapée, âgée...), donc dans un lieu non contrôlé. Un tel robot devra par lui-même avoir les capacités d'apprentissage nécessaires pour acquérir les représentations des objets présents dans l'environnement : comment les manipuler ? comment les reconnaître, les localiser ?

Ces deux fonctions de reconstruction et reconnaissance restent toujours un grand défi pour les chercheurs en vision par ordinateur, puisqu'il n'existe pas un système qui soit capable de développer ces deux fonctions dans un contexte robotique, d'une manière précise et avec suffisamment de généralité. La plupart des articles écrits à ce sujet, décrivent des méthodes exploitables uniquement dans des conditions bien précises de travail. Généralement, dans les nombreuses bases d'images qu'il est possible de trouver sur Internet (la fameuse base COIL par exemple), des hypothèses sont souvent faites, par exemple pour l'éclairage (grand problème à traiter dans le champ d'étude de la vision par ordinateur) ou pour le fond des scènes analysées en phase d'apprentissage ; les auteurs valident leurs travaux sur des objets faciles à modéliser ou à reconnaître, et très peu traitent de la saisie de ces objets.

La Reconstruction d'objets 3D

La reconstruction d'objets est un sujet qui a été beaucoup abordé par les chercheurs en vision par ordinateur[133] [111] [110]. Cela est attesté par la grande quantité d'articles qui ont été écrits dans ce domaine. Nous pouvons d'abord citer Lorensen *et al.* qui ont développé l'algorithme de *Marching Cubes*[84], et les auteurs qui ont proposé des modifications à cet algorithme, comme Nielson avec le *Dual Marching Cubes* [104] ; citons aussi les algorithmes de *Ball Pivoting* proposé par Bernardini *et al.*[23], *Power Crust* implementé par Amenta *et al.* [12] [11], les surfaces déformables, les *Alpha Shapes* [48] [51]... Ces méthodes sont validées dans des conditions bien définies de laboratoire, et avec des données 3D acquises par des capteurs 3D non exploitables en Robotique (capteur Cyberware par exemple) ; peu de ces travaux exploitent la vision, et quand c'est le cas, c'est avec des conditions constantes d'illumination.

Dans notre projet on cherche à obtenir un algorithme qui prenne en entrée des images acquises depuis des caméras embarquées sur un robot, algorithme qui soit robuste devant des changements d'éclairage, et qui permette de reconstruire en 3D, tout type d'objets de petite taille, qu'un robot personnel pourrait manipuler. C'est la raison pour laquelle nous parlons d'objets 3D de forme libre, donc qui ne sont pas représentables simplement par un modèle structuré, comme dans le cas d'une boîte ou d'une balle. Tout au contraire, un robot personnel devra représenter et reconnaître des objets d'usage courant dans notre maison ou lieu de travail.

Des objets simples peuvent être modélisés sans beaucoup de problèmes par les méthodes fondées sur la CSG, pour *Constructive Shape Geometry* : ces méthodes utilisent des formes géométriques 3D de base comme des polyèdres, des cylindres, des sphères. Les objets de forme libre, comme des objets d'usage commun dans un bureau (verre, téléphone, bouteille, brosse, outil...), ne peuvent être représentés par des formes prédéfinies. Une agrafeuse, bon exemple de ce type d'objets, a une grande quantité de détails et de zones creuses ainsi que des reliefs qui ne facilitent en rien la construction de son modèle 3D. En ce

cas, la seule représentation possible est un maillage de facettes élémentaires, généralement des facettes planes triangulaires.

Cette approche est utilisée par de nombreux chercheurs dans le monde, en utilisant des systèmes actifs, tels que des télémètres laser ou des capteurs à lumière structurée. Il existe de nombreux systèmes commerciaux (MINOLTA, RIEGL, LEICA,...), fondés essentiellement sur la technologie laser, que cela soit pour la télémétrie ou pour illuminer la scène. Ces capteurs 3D sont très précis, mais ils ont plusieurs inconvénients dans le contexte de la robotique personnelle : ce sont des capteurs chers, car ils nécessitent d'intégrer une diode laser (chère et fragile par elle-même) dans un montage mécanique très précis (balayage de la scène), avec une électronique complexe (triangulation, mesure de déphasage ou du temps de vol à la picoseconde) ; ils sont encombrants et fragiles, donc pas faciles à monter sur un robot, et ils sont très sensibles à la nature des matériaux constituant les objets à modéliser. Enfin, mettre des diodes laser dans un environnement humain présente toujours un risque pour la santé des personnes. On peut dire que la reconstruction réalisée au moyen de ce type de capteurs laser est très efficace, mais qu'étant donné les inconvénients qu'on vient de présenter, ce n'est pas un capteur adapté à la robotique personnelle.

Notre méthode de reconstruction doit exploiter des données acquises par stéréo vision, capteur 3D bioinspiré qui utilise une paire de caméras [75]. Ce système récupère deux images de la même scène et à partir de la corrélation entre ces deux images, il obtient l'information sur la profondeur de la scène analysée. Par rapport aux autres capteurs 3D, ce système est passif, donc sans problème de sécurité oculaire ; il est moins coûteux, moins fragile et moins difficile à intégrer sur un robot. Cependant la précision des données acquises est très variable ; de plus, elle dépende à 100% du calibrage de la paire stéréo. Mais, la fusion de plusieurs vues acquises par notre capteur stéréo déplacé autour de l'objet à modéliser, devrait permettre d'améliorer la précision et de supprimer des artefacts dûs aux erreurs de corrélation entre les deux images acquises depuis un seul point de vue.

La Reconnaissance d'objets 3D

Pour ce qui est de la reconnaissance d'objets 3D, il existe aussi une grande quantité de publications sur ce sujet, par exemple citons les travaux de Lowe [88] [86] [87], qui a proposé d'exploiter des primitives ponctuelles invariantes appelées *Sift* pour la reconnaissance d'objets à partir de leur apparence, ceux de Hebert et al qui ont développé l'approche de *Spin Images* pour la reconnaissance d'objets 3D [64] [63] [169], mais aussi les travaux de Fergus [49] et de Ke *et al.* [72] qui ont chacun de son côté, utilisé l'approche de *PCA* pour améliorer la méthode fondée sur les primitives *Sift*. Citons encore les travaux de Sadiq [118] sur la logique floue appliquée à ce domaine, et enfin, nos propres travaux [148] menés au Mexique sur une approche active de la reconnaissance, en exploitant des attributs couleur.

Mais comme dans le cas de la reconstruction d'objets, ces travaux sont validées dans des conditions bien définies en utilisant des objets plutôt facile à reconnaître comme des livres, des cannettes de soda, des objets polyédriques... dont la représentation est déjà disponible. Dans la plupart des investigations et articles, les auteurs utilisent généralement une seule image pour reconnaître un objet. Lorsqu'on utilise juste les caractéristiques 2D d'objets 3D, on perd une dimension, ce qui peut générer des erreurs dans la reconnaissance entre deux objets dont les caractéristiques 2D sont similaires mais qui sont très différents en 3D.

Notre travail cherche à traiter ce problème : comment prendre en compte les caractéristiques tridimensionelles des objets ? Nous avons un avantage car nous traitons en même temps de la reconstruction et de la reconnaissance d'objets 3D. Lors de cette reconstruction nous obtenons les caractéristiques géométriques et photométriques des objets ; ces mêmes caractéristiques seront exploitées pour reconnaître, localiser et manipuler ces objets. Nous cherchons à réaliser un système de reconnaissance plus robuste en exploitant tant les caractéristiques 2D de l'apparence des objets (primitives SIFT par exemple) que des propriétés 3D de ces mêmes objets (graphes des divers points de vue). Par ailleurs notre principale contribution consistera à contrôler les positions successives du capteur pour acquérir des données permettant de valider des hypothèses de reconnaissance. C'est la raison pour laquelle nous parlons de reconnaissance *active* d'objets 3D.

Objectif de la thèse

La perception d'un objet en vue de sa manipulation, nécessite donc deux étapes : l'apprentissage des caractéristiques 2D (apparence) et 3D (forme) des objets, puis la reconnaissance de ces caractéristiques pour identifier et localiser les objets à proximité de notre robot afin de les saisir.

L'objectif de la thèse est donc de proposer un système complet de reconnaissance d'objets 3D de forme libre, perçus par un robot manipulateur mobile équipé de cameras stéréo ; on peut contrôler la position de ce capteur vis-à-vis de la scène à analyser. Les objets d'intérêt sont placés dans un espace de travail réduit ; ils sont typiquement posés sur une surface de travail plane (par exemple, une table). Initialement la thèse comprenait seulement la partie de reconnaissance active d'objets 3D, mais le sujet s'est étendu pour couvrir aussi la partie de reconstruction de ces mêmes objets. Cela s'est révélé nécessaire vu le lien important entre les fonctions d'apprentissage et de reconnaissance des objets.

La motivation principale est l'obtention d'un système de reconstruction et de reconnaissance d'objets 3D de forme libre. Ce système devra être utilisé dans un environnement humain, par exemple une maison ; il aura pour mission, d'aider l'être humain dans ses activités ou travaux quotidiens, en particulier d'assister une personne âgée ou handicapée à avoir de meilleures conditions de vie à son domicile.

A notre arrivée en thèse, nous devions participer au projet européen, *Cogniron : the robot companion*. Notre objectif essentiel était de développer des fonctions visuelles sur un robot autonome, qui puisse réaliser une démonstration complète appelée *Curious robot* : le robot devait être capable d'apprendre en ligne, des objets a priori inconnus afin de les reconnaître, les localiser, les saisir et les donner à l'homme. Mes fonctions de perception devaient être intégrées avec des fonctions de planification, de commande et de supervision, étudiées par d'autres collègues. Pour différentes raisons, nous n'avons pas participé à cette expérimentation robotique ; notre contribution porte donc uniquement sur le développement d'algorithmes de vision, qui à terme, seront intégrés sur un robot personnel pour la reconstruction et la reconnaissance en ligne d'objets 3D.

Organisation du document

La suite du manuscrit est structuré en cinq chapitres, de la manière suivante.

Dans le second chapitre, nous présentons notre sujet de manière générale : quels sont les défis de l'apprentissage en ligne de représentations nécessaires pour reconnaître, localiser puis, manipuler un objet 3D ? Nous présenterons également le contexte général sur la Robotique Personnelle, le robot Compagnon de l'Homme exécutant des tâches à domicile, ainsi que l'état de l'art sur les fonctionnalités visuelles requises pour contrôler une tâche de manipulation complexe : comment représenter un objet que le robot devra saisir, comment apprendre cette représentation, comment l'exploiter pour reconnaître et localiser un objet déjà appris ? Enfin, nous définirons les thématiques que nous avons abordées lors de notre travail, et préciserons nos contributions dans chacun de ces thèmes.

Le troisième chapitre traite de la modélisation 3D d'un objet que le robot doit saisir. Ce modèle est généré à partir d'images de points 3D acquis par stéréovision. Afin de réaliser la reconstruction tridimensionnelle d'un objet il est nécessaire d'exécuter plusieurs traitements :

- L'acquisition d'images
- La stéréo-corrélation pour obtenir des images de points 3D
- La décimation de ces images pour garder les points essentiels
- Le recalage et la fusion incrémentale de ces images décimées, dans un nuage de points 3D
- La construction d'un maillage à partir de ce nuage.

Chacune des étapes mentionnées dans cette liste, est par elle-même, un sujet de grand intérêt pour la vision par ordinateur. Nous avons d'abord exploité et adapté des outils développés précédemment dans le pôle Robotique du LAAS-CNRS, notamment par A.Restrepo Specht [133]. Du fait des difficultés rencontrées, nous avons ensuite développé nos propres solutions plus simples pour traiter essentiellement du recalage incrémental des données acquises par stéréovision dans un nuage de points 3D, et de la construction d'un maillage 3D à partir de ce nuage fusionné.

Le quatrième chapitre présente nos travaux sur l'apprentissage d'un modèle d'apparence d'un objet,

et sur l'exploitation d'un tel modèle pour la reconnaissance d'objets 3D. La reconnaissance est essentielle pour un système robotique qui doit savoir quels sont les objets présents dans l'environnement où il évolue et à quels endroits ils se trouvent. Par exemple, en robotique mobile un tel système de reconnaissance pourrait permettre à un robot de se localiser, en connaissant la position respective d'objets particuliers présents dans le monde où il se déplace [147][16] [15]. Dans notre cas, nous ne cherchons pas à développer un système de navigation autonome, mais plutôt un système robotique capable de modéliser, de reconnaître et de manipuler des objets 3D.

La saisie requiert la reconnaissance et la localisation de l'objet, le choix d'une position de prise en fonction de la position courante du robot, et l'exécution d'un mouvement pour atteindre cette position. Nous ne traitons que des fonctions perceptuelles :

- **la segmentation** : un point essentiel, en particulier pour l'apprentissage, consiste à isoler l'objet d'intérêt dans l'image : on parle en général de séparer le *foreground* du *background*. Nous avons exploité l'hypothèse que l'objet est posé sur une table de surface non texturée. Une méthode classique de contour actif est exploitée pour extraire dans les images, les régions correspondant aux objets posés sur la table.
- **Pour l'apprentissage**, nous exploitons plusieurs représentations locales ou globales : la couleur, la signature polaire pour la silhouette de l'objet, le *shape context* pour ses contours, les points d'intérêt, tels que ceux proposées par D.Lowe... Nous visons à terme l'exécution en ligne de ces fonctions d'apprentissage.
- **Pour la reconnaissance** lorsque l'objet est revu à un autre instant, il peut être isolé ou occulté par d'autres objets. Nous exploitons des attributs globaux (couleur, signatures diverses) et locaux (appariement de points d'intérêt) pour le reconnaître.

Surtout, notre méthode de reconnaissance est une méthode active, permettant de vérifier une hypothèse quant à la classe de l'objet perçu, par acquisition d'une nouvelle image depuis un point de vue sélectionné pour maximiser la quantité mutuelle d'informations acquises. Nous avons étendu une méthode simple que nous avions développée au Mexique avant de venir en France. Initialement la reconnaissance n'exploitait qu'un attribut global, la couleur ; dans la version actuelle, nous prenons en compte de nombreux attributs, et en particulier des attributs locaux tels que des points d'intérêt.

Notre système pourrait reconnaître des objets à l'aide de caractéristiques 2D ou 3D, c'est à dire soit à partir d'informations photométriques extraites des images, soit à partir d'informations géométriques extraites de données 3D acquises par stéréovision sur la scène en cours d'analyse. Notre méthode active pourrait guider le processus de reconnaissance pour exploiter ces deux types de caractéristiques ; néammoins, nous ne l'avons validée qu'en vision monoculaire.

Nous disposons donc d'algorithmes de reconstruction et de reconnaissance ; notre travail ne serait pas complet sans application pratique en robotique. Aussi le cinquième chapitre présente l'intégration de nos modules avec d'autres fonctions déjà existantes sur le démonstrateur robotique JIDO du LAAS ; nous décrivons spécialement les interactions à considérer pour traiter de la saisie d'objets 3D posés sur une table par un robot personnel.

Deux situations peuvent survenir : dans la première, le robot connaît l'objet qu'il doit saisir, tandis que dans la deuxième, l'objet est inconnu et nous devons, d'abord, le modéliser pour pouvoir trouver comment le prendre (planification des positions de prise), et pour pouvoir le reconnaître et le localiser à une autre occasion.

La démonstration de saisie visée initialement, devait comporter plusieurs étapes : l'accostage d'une plate-forme mobile le long de la table, la reconnaissance du ou des objets posés sur la table, la modélisation 3D en ligne d'un objet inconnu s'il y en a un, et enfin la planification et l'exécution d'une tâche de saisie sur un des objets présents dans la scène.

Malheureusement l'intégration de nos fonctions de reconstruction et reconnaissance n'a pu être finalisée. Nous ne présenterons donc que des résultats partiels obtenus en simulation.

Comme tout travail qui s'achève et apporte ses fruits, cette thèse a donné quelques contributions. Pour conclure, dans le sixième et dernier chapitre, nous tirerons un bilan du travail accompli et commenterons nos résultats. Finalement, nous indiquerons les possibles améliorations ainsi que les perspectives pour la continuation de recherches sur les thématiques abordées dans ce document.

Chapitre 2

Contexte : Robotique Cognitive et Interactive

Ce chapitre va décrire la problématique traitée dans cette thèse. Nous nous intéressons à la modélisation d'objets 3D de forme libre ainsi qu'à la reconnaissance de ces mêmes objets : l'objectif final de ces travaux concerne l'interaction entre une machine autonome et son environnement, et particulièrement la manipulation d'objets 3D.

Ces deux sujets sont un grand défi pour les chercheurs en vision par ordinateur, puisqu'il n'existe pas vraiment un système qui soit capable d'exécuter ces deux fonctionnalités d'une manière générique, comme un Homme peut le faire par exemple. La plupart des articles écrits sur ces sujets, décrivent des méthodes ou systèmes qui ne fonctionnent que sous des conditions bien précises de travail. Citons les conditions d'illumination, souvent considèrées comme constantes entre les points de vue depuis lesquels sont acquises les images pour réaliser la reconstruction ou la reconnaissance. Citons aussi le problème de la segmentation : un fond uniforme est la plupart du temps utilisé, d'une couleur différente de celles présentes sur les objets. Par ailleurs, souvent, en robotique, les méthodes sont validées uniquement avec des objets simples.

Dans ce chapitre nous introduisons dans une perspective historique, les contributions sur la machine autonome et sur le robot social. Puis nous évoquons le projet européen COGNIRON auquel nous avons contribué et nous présentons précisément notre problématique dans la modélisation et la reconnaissance d'objets 3D et situons rapidement nos travaux dans l'état de l'art de ces domaines. Enfin nous définirons nos contributions dans ces deux domaines.

2.1 De l'automate au robot sociable.

La robotique est la science ou la branche de la science qui s'occupe de l'étude, le développement et les applications des robots. Les robots sont des dispositifs composés de capteurs qui reçoivent des données d'entrée, d'actionneurs qui leur permettent d'agir dans l'environnement et d'ordinateurs pour les contrôler. Initialement cet ordinateur était déporté (on parle de *Remote-Brain System*), mais aujourd'hui, les unités de calcul sont assez puissantes et miniaturisées pour être embarquées sur les robots.

Le mot robot évoque souvent dans notre esprit l'image d'une machine avec forme humaine, donc avec tête et membres. Cette association est le fruit de l'influence de la télévision ou du cinéma, dont les annonces ou les films montrent des machines avec forme humaine, appelées androïdes, qui relèvent évidemment de la fiction pure. Ce sont des hommes déguisés en machine ou, si ce sont réellement des machines, elles n'effectuent pas de travaux utiles pour l'Homme. Actuellement, les avancées technologiques et scientifiques n'ont pas permis encore de construire un robot réellement intelligent, bien qu'il existe des espoirs que ceci soit possible dans un futur proche.

De nos jours, les robots sont construits essentiellement pour intervenir dans les processus de fabrication. Ces robots, qui n'ont absolument pas une forme humaine, remplacent les personnes chargées d'effectuer des travaux répétitifs dans les chaînes de fabrication, comme par exemple : peindre au spray, souder des carrosseries d'automobile, transférer des matériels, etc.. Dans une usine sans robot, ces travaux sont effectués par des techniciens spécialistes : avec les robots, le technicien peut être libéré de la routine

et du risque que ces tâches comportent, d'où un gain en rapidité, sécurité, qualité et précision.

2.1.1 Les automates, ou la génèse de la robotique

Depuis très longtemps, on a créé des artefacts capables d'effectuer des tâches à la place des hommes, ou bien, pour leur rendre des services au quotidien. Les hommes se sont rendus compte qu'il y avait des tâches répétitives qui pouvaient être réalisées avec un système complexe, et c'est ainsi que l'Homme a commencé à créer des machines capables d'accomplir certaines tâches qu'il effectuait. Comme premiers exemples de telles machines nous pouvons citer :

- La roue comme moyen de transport ou comme outil,
- Le pignon,
- La catapulte comme arme de combat,
- Le moulin, pour moudre du grain à partir de la force de l'eau ou du vent.

Toute une variété de machines ont été créées pour rendre des services aux hommes. Mais ces artefacts n'avaient pas tous une utilité ; quelques machines servaient seulement à divertir leurs propriétaires, et ne faisaient rien de plus que d'effectuer des mouvements répétitifs ou d'émettre des sons. Ce sont les arabes qui ont été des précurseurs dans la construction d'automates, du fait de la précision de leurs calculs ; rappelons qu'ils ont inventé l'horloge mécanique, qu'ils ont grandement contribué à l'astrologie... Les ingénieurs grecs ont produit aussi de grandes connaissances sur les automates, bien que leur intérêt se portait plutôt vers le savoir humain plus que vers les applications pratiques.

Depuis des centaines d'années avant Jésus Christ, on a commencé à créer des automates, prédécesseurs des robots actuels. Nous citons ci-après, une liste d'automates créés par des civilisations très variées et dans des époques proches [36] :

- en 1500 A.C., Amenhotep, frère de Hapu, construit une statue de Memon, le roi de l'Ethiopie, qui émet des sons quand elle est illuminée par les rayons du soleil au matin.
- en 500 A.C., King-su Tse, en Chine, invente une pie volante en bois et bambou et un cheval de bois qui sautait.
- Entre 400 et 397 A.C., Archytar de Tarente construit un pigeon en bois suspendu à un pivot, qui est mû par une machine à vapeur.
- en 206 A.C., a été trouvé le trésor de Chin Shih Hueng Toi : un orchestre mécanique de marionnettes.
- en l'année 62 de notre ère, Hero de l'Alexandrie produit un traité sur les automates. C'est un célèbre registre d'applications scientifiques qui peuvent être validées par le biais d'un automate. Il exhibe aussi son théâtre automatique où des figurines montées dans une caisse, changent de position devant les yeux des spectateurs : oiseaux chanteurs, trompettes qui sonnent...
- en 700, Huang Kun a construit des bateaux avec des figures d'animaux, des chanteurs, musiciens et danseurs qui se déplaçaient.
- en 770 D., Yang Wu-Lien construit un singe qui étend ses mains et dit "Aumône ! Aumône !", en mettant sa collecte dans une bourse quand elle atteint un poids déterminé.
- En 890, Chih Ho fait un chat de bois qui chasse des rats.
- Alberto Magno (1204 - 1272) crée un serviteur mécanique.
- Roger Bacon (1214 - 1294) parvient à construire, après 7 années, une tête qui parle.
- en 1235, Villard d'Honnecourt fait un livre d'ébauches qui incluent des plans de dispositifs mécaniques, comme un ange automate, ainsi que des indications pour la construction de figures humaines et animales.
- Citons aussi l'horloge avec une forme de coq qui chante dans la cathédrale de Strasbourg ; cet automate a fonctionné depuis 1352 jusqu'à 1789.
- Leonard de Vinci construit durant l'année 1500 un lion automatique en honneur de Louis XII.
- en 1640, René Rejets a inventé un automate auquel il se réfère comme "mon fils Francis".
- en 1662, on ouvre à Osaka le théâtre Takedo d'automates.
- Jacques de Vaucanson, construit son canard, probablement l'automate le plus connu ; un canard fait de cuivre, qui boit, mange, crie, barbote dans l'eau et digère son repas comme un canard réel.
- Les Maillardet (Henri, Jean-David, Julien-Auguste, Jacques-Rodolphe) construisent un dessinateur, avec la forme d'un garçon agenouillé avec un crayon dans sa main : il écrit en Anglais et en Français et dessine des paysages. Ils construisent aussi un mécanisme "magique" qui répond à des questions et un oiseau qui chante dans une caisse.

– Robert Houdin construit une poupée qui écrit, et aussi, un danseur sur corde, un trapèziste... un homme qui signale un événement avec un canon.
– et pour finir, Thomas Alva Edison a construit durant l'année 1891 une poupée qui parle.

Nous pouvons constater que les automates construits jusqu'à ce moment, servaient seulement au divertissement ; ils n'avaient pas d'application pratique dans un secteur spécifique. Ces machines fonctionnaient généralement par le biais de mouvements ascendants d'air ou d'eau chaude ; ce mouvement provoquait des ruptures d'équilibre ou bien la chute d'un poids dans divers récipients pourvus de valves. D'autres mécanismes se basaient sur des leviers ou des contrepoids. Dans le treizième siècle, Al-Djazari est apparu comme l'héritier de toutes ces tendances, avec la publication de son *Livre de la connaissance des procédures mécaniques*.

Peu de temps après, les automates ont été les protagonistes principaux d'une infinité d'histoires de science-fiction. La majorité des romanciers de ces temps, considéraient les automates comme une menace pour l'existence de la race humaine. Avec ce type d'histoires, la crainte envers les automates a cru considérablement. Durant l'année 1920, l'auteur d'origine tchèque Karel Capek, a publié son roman RUR (Russum's Universel Robots), qui a été ensuite adapté, puis présenté comme oeuvre de théâtre dans le Théâtre National de Prague le 25 janvier 1921. *Cette oeuvre traite de deux petits êtres artificiels de forme humaine qui répondent parfaitement aux ordres de leur créateur, bien qu'à la fin ils finissent par se rebeller contre lui.* Pour se référer à ces êtres, l'auteur a proposé le mot **robot**, dérivation du mot tchèque *robota*, qui signifie "travail obligatoire". Et c'est ainsi qu'apparaît le mot *robot* pour désigner les automates mécaniques de ces époques. A partir de ce roman, on appelle les automates, des robots.

Il s'est créé rapidement un sentiment de peur des robots, étant donné l'évolution tellement accélérée avec laquelle ce concept a été projeté dans des romans de science-fiction. Dans son oeuvre "moi robot" publiée en 1940, Isaac Asimov postule trois lois que les robots devront suivre :
– Un robot ne doit pas endommager un être humain, ou laisser un être humain souffrir de dommages du fait de son inaction.
– Un robot doit obéir aux ordres que lui donne un être humain, excepté quand ces ordres sont en contradiction avec la première loi.
– Un robot doit protéger son existence propre, tant que cette protection n'entre pas en conflit avec la première ou la seconde loi.

Encore après cette publication d'Isaac Asimov, les romanciers continuaient à s'interroger sur la nature d'un robot. Plusieurs concrétisent dans leurs romans l'idée qu'un certain jour, l'homme sera esclave des machines. Citons par exemple le roman de Jack Williamson "Avec les mains croisées" ; il y est montré comment la liberté humaine s'est transformée en esclavage pour des robots ; les hommes doivent obéir à tous les ordres que les robots leur donnent.

L'Homme a imaginé une infinité d'histoires en rapport avec les robots ; beaucoup de ces histoires ont annoncé l'arrivée de nouvelles technologies. De bons exemples sont les romans de Jules Verne, spécialement celui appelée "Voyage autour de la Lune" où il rapporte avec luxe de détail, comment trois hommes pourraient arriver sur la lune.

Un des premiers films qui traitent le sujet de la robotique est intitulé "Métropolis". Il introduit un robot féminin qui possède une intelligence propre, mais obéit à tous les ordres de son créateur. Bien qu'il s'agisse d'un ancien film, il est significatif de comment l'homme percevait les robots dans les années 1930.

Un siècle plus tard, ces romans de science-fiction ne sont pas tellement hors de la réalité que nous vivons aujourd'hui ; néammoins, nous ne devons pas avoir peur des robots, mais au contraire, favoriser le développement de robots capables d'exécuter des tâches pour les hommes, tels que le robot humanoïde Hrp2 présenté en figure 2.1.

2.1.2 Différents types de robots

Avant d'en arriver à l'humanoïde, de nombreux critères ont été proposés pour la classification des robots : la fonctionnalité, la géométrie, l'intelligence... Les caractéristiques les plus employées sont :
– Le but ou la fonction que le robot devra exécuter.
– le système de coordonnées employé, significatif de sa situation dans l'espace.
– le nombre de degrés de liberté, pour qualifier les capacités de mouvement.

FIGURE 2.1 – Humanoid Robotics Project (Hrp2), développé par Kawada Industries.

– le degré d"'intelligence" du système de contrôle.

Pour ce qui est de la classification basée sur le but ou la fonction, citons quelques classes de robot :

1. Industriel... Les éléments qui constituent un robot industriel sont : Bras manipulateurs, Effecteurs finaux, Contrôleurs, Capteurs, Sources d'énergie.
2. Personnel/Éducatif/Service... Il s'agit ici de plates-formes mobiles, agissant en milieu intérieur, sur sol plat.
3. Militaire/Agricole/Exploration planétaire... On parle aussi de *Field Robotics*, avec des robots terrestres (ou Unmanned Ground Vehicles) dotés de systèmes de locomotion permettant de se déplacer sur des sols accidentés, ou pour d'autres milieux, des robots aériens (Unmanned Aerial Vehicles), marins (Unmanned Surface Vehicles) ou sous-marins (Autonomous Underwater Vehicles).

La notion de **générations de robots**, peut être définie à partir des capacités des systèmes de contrôle, ou du degré d"'intelligence" de la machine.

◇ *La première génération* de robots se caractérise par des boucles de commande en boucle ouverte, des méthodes simples style contrôle "bang bang". Citons par exemple ces mécanismes d'horlogerie qui permettent de déplacer des caisses musicales. Ces robots sont utiles pour les applications industrielles de *Pick and Place*, limitées à un petit nombre de mouvements.

◇ *La seconde génération* utilise aussi une structure de contrôle en boucle ouverte, mais au lieu d'utiliser des interrupteurs et des boutons mécaniques, il utilise un programme numérique de contrôle de mouvements stockés dans un disque, un ruban magnétique, un automate de contrôle... Ce programme permet de sélectionner les séquences de mouvement par un interface opérateur (boîte à boutons...). Le plus grand nombre d'applications dans lesquels on utilise les robots de cette génération, concerne l'industrie automobile, la plus ancienne à s'être convertie à la robotique : soudure, peinture avec "spray"... Des auteurs suggèrent que près de 90% des robots industriels aux Etats-Unis appartiennent toujours à cette 2ième génération.

◇ *La troisième génération* de robots utilise une stratégie de contrôle numérique, souvent via des microcontroleurs. Ils ont une certaine connaissance de l'environnement local à travers l'utilisation de capteurs exploités pour acquérir des informations sur l'environnement. Ils sont capables d'adapter leur stratégie de contrôle, en fonction des données capteurs : commandes référencées capteur, contrôle en boucle fermée... On entre dans l'ére des robots intelligents, des langages de programmation pour écrire les programmes de contrôle...

◇ *La quatrième génération* de robots, est dotée d'extensions sensorielles intelligentes avec de plus nombreux et de meilleurs capteurs : le robot peut ainsi comprendre ses actions, modéliser le monde qui l'entoure, adapter sa conduite propre en fonction des évolutions de l'environnement dans laquelle il opère. Les programmes qui contrôlent ces robots, sont à base de connaissances : ils enregistrent l'effet de l'action

du robot sur l'environnement, et adaptent les stratégies et les tâches en cours d'exécution, afin d'améliorer les résultats.

◇ Enfin, *la cinquième génération*, s'inspire des théories comportementalistes : on parle souvent de bioinspiration. Dans ces robots, la stratégie de contrôle émerge d'une autoorganisation de modules réactifs ; des techniques d'apprentissage permettent de sélectionner les plus adéquates. Cette nouvelle architecture est appelée architecture de subsomption.

2.1.3 Le robot social

Les progrès de la robotique durant les dernières années, a permis d'introduire les robots dans l'environnement de l'homme, d'abord sur le lieu de travail, les ateliers, les chantiers... avec des opérateurs humains, généralement dotés d'une formation scientifique et technique, et de plus, formés pour interagir avec le robot. Grâce aux progrès en fiabilité et en sûreté de fonctionnement, de plus en plus de robots sont introduits dans des environnements humains non professionnels, donc en présence d'un public non formé. Ce rapprochement vers l'homme implique que les robots possédent une série de caractéristiques déterminées que l'homme considère nécessaire pour les accepter.

Robots sociaux et sociables

Un point de départ important pour se convaincre de la nécessité d'avoir des robots sociaux ou sociables, est de considérer et de reconnaître que l'homme est une espèce profondément sociale. Avec des performances plus ou moins bonnes, l'homme applique des modèles sociaux pour expliquer, comprendre et prédire le comportement de ce qui l'entoure.

Traditionnellement le terme *robots sociaux* s'appliquait à des systèmes multirobots bioinspirés : la motivation pour l'étude de tels systèmes venait du comportement collectif d'insectes, oiseaux, poissons, etc.. Cependant, sont apparus des travaux récents [25] dans lesquels on utilise le terme *sociable* pour faire la distinction entre le style anthropomorphe de l'interaction homme-robot et les comportements collectifs mis en oeuvre dans une colonie de robots (on parle aussi d'essaim, de flotte...). Cette notion de "sociabilité" des robots a évolué tout au long des années récentes, pour arriver à être associée plus étroitement au comportement social anthropomorphe. Pour cette raison, on utilisera dorénavant le terme social pour designer ce type de comportement, mais on continuera à désigner par le terme sociable, une sous-classe dans les robots sociaux, ceux qui vivent en société entre eux.

Généralement les personnes appliquent un modèle social quand elles observent les comportements d'un robot social autonome à des fins d'interaction. Un robot autonome perçoit son environnement, prend des décisions par lui-même et exécute des actions destinées à mener à bien la tâche assignée. Comme les êtres vivants, son comportement résulte de son état interne ainsi que de lois physiques. Si nous ajoutons à ce comportement orienté vers la satisfaction de ses propres besoins, des comportements identiques à ceux d'une créature vivante, quand il s'agit de communiquer et coopérer avec des personnes, ou d'apprendre par imitation ou démonstration, alors il s'avère presque impossible de ne pas lui appliquer des modèles anthropomorphes, c'est-à-dire, lui attribuer des caractéristiques humaines ou animales.

Ce type de robots autonomes sera appelé robot social, c'est-à-dire, un robot auquel les personnes appliquent un modèle social en vue d'interagir avec lui et de le comprendre. Le domaine des robots sociaux et de leur conception a une dizaine d'années : l'aventure a commencé avec les robots guides de musée (projet Minerva en Allemagne en 1999, exposition robotique à Neuchatel en 2002, le robot Rackham du LAAS à la cité de l'espace de Toulouse en 2005...), mais les applications de la robotique au contact de l'Homme se multiplient :

- robot personnel ou robot compagnon, capable de rendre des services aux personnes handicapées ou agées à leur domicile,
- trolleys robotisés en grande surface ou dans des centres de transport -aéroports et autres ...-,
- robot domestique, au delà des aspirateurs ou des tondeuses à gazon autonomes (*Roomba...*), qui tiennent plus de la machine que du robot.
- robots de service (nettoyage, logistique...) qui interagissent avec les personnels d'un hopital par exemple ...

Caractéristiques d'un robot social

Bien que cela soit encore assez nouveau, on peut distinguer plusieurs sous-classes de robots à partir de quelques applications et d'exemples existants. Dans tous les cas les personnes tendent à leur associer des caractéristiques anthropomorphes pour parvenir à interagir avec eux. Au fur et à mesure qu'on avance dans la classification, les robots possèdent de plus grandes aptitudes pourse plier à un modèle social dans des environnements réalistes qui s'approchent de l'environnement social humain, et dans des modes d'interaction plus complexes du type des relations humaines en face à face :

◇ *Socialement évocateur* : robots conçus pour que les utilisateurs leur associent des caractéristiques humaines dans le but d'interagir avec eux, mais sans aller plus loin. Ce serait le cas de beaucoup de jouets ou de jeux pour ordinateur ou consoles. Citons les robots Robosapiens et Robosapiens V2 développés par Mark Tilden, un ancien collaborateur de la NASA, du DARPA et de JPL (voir figure 2.2).

◇ *Socialement communicatif via un écran* : cette sous-classe de robots utilise des signes sociaux humains et des modalités de communication afin de faciliter l'interaction avec les gens, en la rendant plus naturelle et familiale. Ce serait le cas des robots avatar, qui ont besoin d'intelligence sociale suffisante pour transmettre de manière appropriée les messages à des personnes, en exploitant le regard, les gestes, l'expression faciale... Récemment des robots hotesses ont été créés au Japon, avec l'apparence d'une femme, avec peau artificielle et simulation des muscles du visage. D'autres exemples sont les robots guide de musées, dans lesquels un avatar ou un clone sur écran, est utilisé pour communiquer l'information aux utilisateurs : cet avatar parle et adopte parfois des expressions faciales. Cette classe de robot adopte un comportement social, mais uniquement via une artefact virtuel sur écran ; de ce fait le modèle social de ces robots, est peu profond.

◇ *Socialement communicatif via des actions* : robots qui apprennent l'interaction avec les gens à travers la démonstration par un homme, appelé tuteur. L'interaction avec les personnes affecte la structure interne du robot (en réorganisant le système moteur pour développer de nouveaux gestes, en associant des étiquettes symboliques à de nouvelles perceptions, etc.). Un tel système tend à donner davantage d'importance à la perception sur l'Homme pour détecter des signes sociaux, et à l'action pour que l'Homme comprenne le comportement du Robot. Toutefois, ils sont socialement passifs : ils répondent aux sollicitations des gens pour interagir avec eux mais ne prennent pas l'initiative de l'interaction.

◇ *Socialement actif (ou sociable)* : les robots sociables sont des créatures socialement participantes avec leurs fins propres et motivations internes. Ils interagissent avec les personnes activement d'une manière sociale, non seulement pour rendre des services à la personne (en l'aidant à effectuer une certaine tâche, en facilitant l'interaction avec le robot, etc.), mais aussi en cherchant son bénéfice propre (chercher sa survie, améliorer ses actions propres, apprendre des hommes, etc.). L'interaction sociale avec les gens n'est pas uniquement évaluée comme un interface, mais il est considéré dans un niveau fonctionnel.

Une des réalisations emblématiques d'un robot sociable est Kismet, présenté en figure 2.3. Ce robot a été développé au MIT dans l'équipe de Rodney Brooks par C.Breazeal [26], dans le projet de Machines Sociables. Kismet n'a pas été développé pour traiter d'une tâche spécifique, mais il a été conçu comme une créature robotique qui pourrait physiquement interagir, affectivement et socialement avec des hommes dans le but d'apprendre d'eux. Il possède l'infrastructure nécessaire pour être capable d'évoluer depuis un niveau d'interactions infantiles plus destinées à des jeux, jusqu'à atteindre un développement social complet .

Interaction homme-robot

L'incorporation des robots dans des environnements humains a rendu indispensable de les doter de mécanismes qui permettent à ces robots, d'interagir avec les hommes qu'ils ont autour, ainsi que de répondre aux sollicitations des hommes. Ces robots destinés à être directement mis en rapport avec les hommes, doivent communiquer de manière optimale afin d'être acceptés par l'Homme (reconnaissance fiable de la parole et des gestes de l'homme ... voix de synthèse audible et agréable... écran tactile placé de manière ergonomique).

FIGURE 2.2 – Robots Robosapiens développés par Mark Tilden.

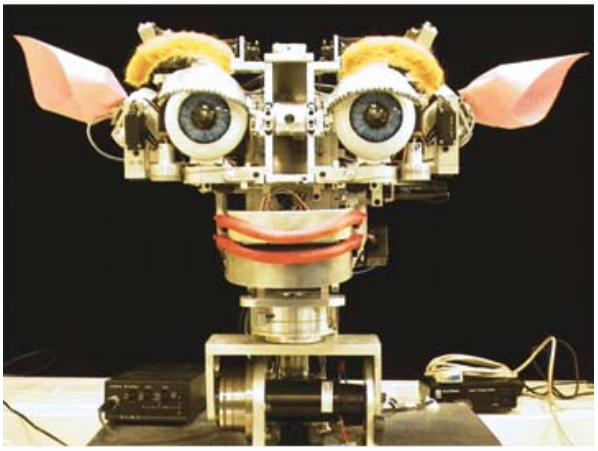


FIGURE 2.3 – Robot Kismet développé par le MIT

Plus concrètement pour promouvoir les robots personnels ou sociaux, de nombreux travaux sur la robotique de service, portent sur des mécanismes de communication basés sur le langage parlé, qui est une caractéristique typique de la communication entre les hommes ; le dialogue est un moyen de communication explicite entre humains [38]. Mais dans la communication entre les hommes il existe aussi un autre canal, implicite, au moyen duquel on transmet une information sur les individus, généralement sous forme d'émotions. Ce canal implicite est une caractéristique essentielle de la communication humaine. Par conséquent, si on souhaite obtenir une interaction réellement effective des robots avec les hommes, il faudra aussi aborder ce type de communication.

L'interaction qu'on souhaite obtenir est bidirectionnelle : (1) l'Homme espère que le Robot soit capable de reconnaître les signes implicites qu'il souhaite transmettre et (2) l'Homme doit reconnaître et distinguer dans le comportement du Robot, des signes qu'un homme montrerait comme réaction à l'information échangée ; la nature de ces signes dépend du milieu culturel, ils sont différents au Japon, aux USA, en Europe...

Pour ce faire, l'interface intégrée sur un robot pour la communication homme-machine sera formée, principalement, de deux parties : une entrée multimodale et une sortie multimédia ou mécanique. Les recherches en rapport avec l'entrée multimodale comprennent l'analyse d'images et de vidéo (vision), l'analyse de la parole (voix), l'analyse de l'écriture manuelle, etc.. La sortie multimédia, pour sa part, comprend la synthèse de voix, d'images, de graphiques par ordinateur et d'animation (clones). Pour une communication plus effective, il est développé des structures mécaniques sous forme de face anthropomorphe ou de simples gadgets mécaniques comme les oreilles ou les sourcils de Kismet.

2.1.4 Le projet COGNIRON

Un des projets importants ces dernières années, pour l'étude d'un robot social, a été le projet COGNIRON [46], pour *the Cognitive Robot Companion*, projet intégré européen (Janvier 2004-Avril 2008), dont la thématique portait sur le robot compagnon de l'homme. L'ambition de ce projet était grande : *Développer des robots qui interagissent avec les hommes, capables de percevoir, décider, communiquer et apprendre de manière permanente.*

Les travaux étaient organisés en lots ou *Research Activities (RA)*, dont les plus significatifs étaient :
- RA1 : dialogue multi-modal,
- RA2 : détection et interprétation des activités humaines,
- RA3 : comportement social et Interaction Homme-Machine,
- RA4 : apprentissage de tâches,
- RA5 : cognition spatiale et reconnaissance de situations,
- RA6 : intentionnalité et initiative
- RA7 : intégration et évaluation.

Le but était de concevoir, développer et évaluer des instances de robots physiques, qui intégrent plusieurs des contributions produites dans les domaines évoqués ci-dessus. Dès le début ont été définies des expérimentations objectifs du projet, sous la forme de scénarios très précis impliquant le robot et un ou plusieurs hommes. Le but du projet a été d'étudier et d'intégrer sur un démonstrateur, les fonctions nécessaires pour jouer ces scénarios de manière fiable et répétable. Trois *Key Experiments* ont été considérés :

- KE1 : *The Robot Home Tour* (figures 2.4 et 2.5 gauche), dédié à l'apprentissage supervisé par l'Homme, de représentations de l'environnement.
- KE2 : *The Curious Robot* (figures 2.4 et 2.5 milieu), dédié à des tâches plus locales impliquant l'apprentissage de représentations, la reconnaissance et la manipulation d'objets usuels dans les environnements humains.
- KE3 : *Learning Skills and Tasks* (figures 2.4 et 2.5 droite), dédié à l'apprentissage de tâches sur le Robot par imitation de l'Homme.

Si on replace ces expérimentations dans le contexte applicatif du robot personnel, KE1 intervient quand le Robot est réceptionné dans une maison : l'Homme le promène, le Robot apprend un modèle spatial de chaque lieu composant l'environnement, l'Homme nomme les lieux appris : le Robot pourra ensuite se déplacer si l'Homme commande *Va à la cuisine*. KE2 intervient quand l'Homme présente au Robot des objets qu'il devra manipuler ; l'Homme montre les objets selon tous les points de vue, et les

FIGURE 2.4 – Symboliques associées aux trois *Key Experiments* du projet COGNIRON.

FIGURE 2.5 – (a) KE1 : robot BIRON apprenant un lieu avec son tuteur (Univ.Bielefeld) ; (b) KE2 : robot JIDO donnant un objet à l'homme (LAAS-CNRS) ; (c) KE3 : robot HOAP3 apprenant une tâche avec son tuteur (EPFL).

nomme : le Robot pourra ensuite exécuter un ordre du type *Va à la cuisine et ramène un verre.* Enfin KE3 concerne l'apprentissage de tâche plus complexe que *Aller à* ou *Prendre*, *Déposer* ou *Donner à* : l'Homme présente des objets outils au Robot, et lui montre comment les utiliser ; le robot, s'il est doté de tous les accessoires nécessaires, en particulier s'il dispose d'un ou de deux bras robotiques, pourra ensuite exécuter une tâche de type *Faire une omelette*, *Servir à boire* ...

Nous avons modestement contribué au projet COGNIRON, en participant initialement à KE2 dont le LAAS était responsable. La plate-forme mobile JIDO présentée en figure 2.7, a été réalisée pour cela ; de nombreux collègues ont travaillé sur des modules fonctionnels (commande du bras, navigation, localisation...) ou décisionnels (supervision des actions, planification, reprise des erreurs...). Nous avons étudié les fonctions nécessaires pour apprendre le modèle d'un objet 3D, pour le reconnaître et pour le saisir. Pour différentes raisons, nos contributions n'ont pas pu être intégrées sur ce démonstrateur JIDO : nous avons exploité ce robot pour l'acquisition d'images stéréo et pour faire quelques tests préliminaires de saisie d'objets en exploitant les modèles 3D construits depuis ces images.

Avant de décrire nos travaux dans les chapitres suivants, décrivons plus avant la problématique traitée.

2.2 Notre problématique

Après avoir décrit le contexte Robotique dans lequel s'inscrit ce travail, les paragraphes suivantes présentent les différents défis que nous avons abordés.

L'objectif de la thèse était d'intégrer un système complet de reconstruction et de reconnaissance d'objets 3D de forme libre dans un robot manipulateur mobile équipé de caméras stéréo ; par définition, un système robotique est aussi un système actif de perception, dans la mesure où on peut contrôler en ligne certaines modalités, en particulier, la position relative entre l'objet et le capteur stéréo. Notons que notre robot JIDO, dispose de deux capteurs stéréo, un monté sur un mât (la "tête") et un monté sur le

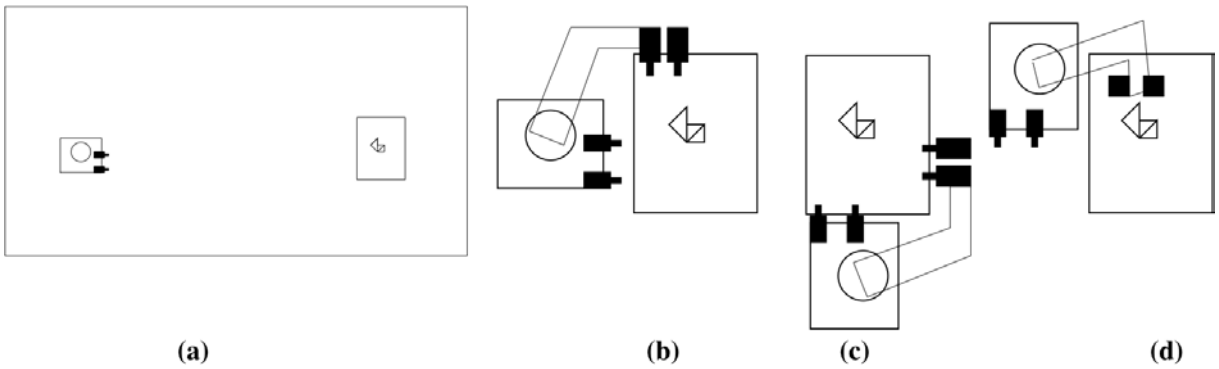


FIGURE 2.6 – Expérimentation sur la reconstruction et la saisie d'un objet 3D posé sur une table.

poignet du manipulateur. Dans nos travaux, nous n'avons exploité que celui qui est sur le poignet.

Les objets d'intérêt sont placés dans un espace de travail réduit ; ils sont posés sur une surface de travail, une table. Le scénario complet est sommairement décrit en figure 2.6 :

- (a) le robot détecte la table à distance, et détecte la présence de quelque chose sur la table : nous dirons dans ce document, qu'un *TRUC* est présent sur la table. Cela peut être un objet isolé, ou un vrac d'objets, les uns posés sur les autres, simplement accolés, ou seulement en recouvrement dans l'image.
- (b1) le robot mobile accoste la table dans une première position. Nous ignorons ici les tâches à intégrer sur notre robot pour reconnaître la table depuis un point de vue lointain, pour détecter la présence de "quelque chose", un *TRUC* posé sur la table, puis pour approcher et accoster cette table en une position adéquate pour percevoir au mieux ce *TRUC*.
- (b2) A l'aide du bras, il déplace le capteur stéréo du poignet pour d'abord comprendre quel est ce *TRUC*, afin de planifier et exécuter une tâche de saisie. Si le *TRUC* n'est pas reconnu comme un objet ou un vrac d'objets connus, il doit être modélisé.
- (b3) le système doit planifier comment prendre un des objets reconnus ou l'objet appris, puis exécuter la tâche de saisie, et emmener l'objet en un autre lieu ou le tendre à un être humain. Cette dernière étape n'était pas considérée dans notre sujet.
- (c) et (d) éventuellement, si le point de vue choisi en (b) ne suffit pas pour la reconnaissance ou pour accéder à une position de prise de l'objet, la plate-forme mobile génère d'autres positions d'accostage afin d'accéder à des points de vue ou des positions de prise non atteignables depuis (b).

Initialement la thèse comprenait seulement la partie active de reconnaissance d'objets 3D mais la thématique s'est étendue pour aussi couvrir la partie de reconstruction d'un objet placé sur une surface. Cette extension avait deux motivations. D'une part, un autre doctorant, A.Restrepo [133], avait développé des fonctions de modélisation 3D à partir de données 3D : nous devions seulement intégrer et exploiter ces fonctionnalités. D'autre part, nous devions extraire du modèle 3D construit en ligne, des indices ou caractéristiques qui seraient ensuite exploités pour la reconnaissance d'objets.

Notre objectif principal dans ce projet était d'exécuter ces fonctions de modélisation et reconstruction en ligne sur un robot plongé dans un environnement humain, par exemple une maison. Ce système aura pour mission, d'aider l'être humain dans ses travaux quotidiens, par exemple d'aider une personne handicapée à avoir un meilleur niveau de vie dans son lieu de vie. "En ligne" implique donc une exécution en temps réel, c'est-à-dire avec des temps de réponse compatibles avec l'acceptabilité d'un tel système par un humain ; notons que nous n'avons pas quantifié le temps de réponse acceptable, et que cette contrainte n'est que qualitative.

L'aire de la vision par ordinateur est un champ de recherche très important : des progrès immenses ont été réalisés dans la dernière décennie, mais il reste encore beaucoup à expérimenter avec de vrais robots, pour intégrer et évaluer les méthodes proposées dans un contexte robotique. Le travail que nous avons réalisé dans cette thèse, est une contribution dans ce domaine encore mal exploité : bien qu'il y ait une grande variété d'articles scientifiques qui parlent tant de la reconstruction d'objets 3D comme de la reconnaissance d'objets 3D, peu aborde le sujet d'objets de forme libre c'est-à-dire des objets d'usage commun dans notre vie quotidienne, et surtout peu d'auteurs en vision intégrent sur des robots agissant dans le monde réel.

Notre objectif est donc d'obtenir un procédé pour la reconstruction et la reconnaissance d'objets 3D sous des conditions normales d'éclairage et en utilisant des objets d'usage commun dans notre vie quotidienne. Le projet est divisé en sujets que nous présentons ci-dessous, en expliquant brièvement l'objectif et la justification de chacun.

2.2.1 Application de nos travaux en robotique.

Comment développer des algorithmes de vision dédiés à des applications Robotique ? Certains algorithmes (recalage, stéréovision...) existaient déjà : nous avons dû comprendre leurs principes, avant d'analyser les codes existants, d'apprendre à les exploiter et de vérifier qu'ils donnaient les bons résultats pour traiter de notre problématique. Cela n'a pas toujours été le cas !!! D'autres algorithmes sont le résultat de nos propres travaux : nous avons généralement prototypé en Matlab, puis porté en C dans un environnement d'intégration propre au LAAS-CNRS, dans lequel on peut appeler des fonctions pour contrôler les actionneurs du robot.

Dès que les algorithmes sont développés, on vérifie leur bon fonctionnement. Cette validation des algorithmes est réalisée en deux étapes. D'abord au moment du prototypage, les algorithmes exploitent une base d'images : elles peuvent être

- calculées par synthèse. Nous avons utilisé pour ce faire *Povray* : on connait complétement la vérité terrain (positions relatives entre les capteurs et les objets), mais les textures sont souvent trop artificielles.
- récupérées dans une des très nombreuses bases existantes sur le Web (citons la base COIL exploitée par tous les chercheurs en reconnaissance par vision),
- ou bien préalablement acquises avec le vrai capteur du robot. L'usage d'une base d'images acquises sur le robot, est très utile puisque cela permet de considérer le vrai espace de travail dans lequel l'algorithme sera effectivement exploité.
- ou encore acquise par un simple appareil photo numérique : c'est ce que nous avons fait lorsque le robot n'était pas disponible.

A ce stade du prototypage, nous avons aussi exploité un environnement virtuel pour simuler les actionneurs afin de déplacer le capteur, cela pour vérifier les phases actives des algorithmes. Cela s'est révélé aussi très utile parce que cela nous a permis d'évaluer la portée et les limitations d'un algorithme donné.

Mais la vraie épreuve pour la validation des algorithmes réalisés dans cette thèse en robotique, c'est leur intégration dans un robot qui va interagir avec le milieu, en particulier dans notre cas, avec un humain. Le robot utilisera nos algorithmes pour interpréter ce qu'il perçoit de son environnement et réagir conformément à cela.

Le système que nous avons utilisé partiellement pour la validation des algorithmes est le démonstrateur JIDO, acheté et équipé au LAAS-CNRS pour réaliser l'expérimentation KE2 du projet COGNIRON. Nous pouvons voir cette plate-forme mobile en figure 2.7. Décrivons les principales composantes de ce robot :

- Un Bras robotique Mitsubishi PA10-6C ;
- Une plate-forme mobile Neobotix ;
- Un mât qui joue le rôle d'une tête, avec une paire stéréo (base 40cm), plutôt dédiée à la perception de l'environnement, et un capteur 3D Swiss Ranger (capteur optique à temps de vol), les deux sont montés sur une platine site et azimuth ;
- Une Paire stéréo compacte VIDERE (base 9cm) montée sur le poignet du bras ;
- Une Pince, équipé de capteurs de force pour détecter quand un objet est tenu ou lâché ;
- Deux télémètres laser Sick : un pivotant en site à l'avant, un fixe à l'arrière.

Ce robot dispose donc de nombreux actionneurs (plate-forme, bras, platine) : nous n'avons exploité que le bras dans nos travaux.

Nous devions intégrer principalement deux fonctions dans le système robotique JIDO, dédiées à la reconstruction et à la reconnaissance d'objets 3D. Dans les deux sections suivantes, nous décrivons rapidement le comportement que doit avoir chaque algorithme une fois intégré sur le robot.

FIGURE 2.7 – Plate-forme mobile avec un bras 6 DOF : JIDO

2.2.2 Un système de Reconstruction 3D

De manière classique, nous appelons *nuage de points 3D*, un ensemble de points 3D inorganisés, et *image 3D ou image de points 3D*, un ensemble de points 3D qui hérite des relations de voisinage inter-pixels des images d'origine. Dans un nuage, ces relations sont soit perdues, soit non complètes (pas de relation entre points acquis depuis des points de vue différents sur la scène).

Implémentation de l'Algorithme de Modélisation d'objets 3D.

L'algorithme de reconstruction une fois intégré sur la plate-forme de développement JIDO, doit exécuter les actions suivantes. On suppose que l'objet à modéliser est posé sur une table.

- Générer une trajectoire sur une demi sphère centrée autour de l'objet à reconstruire, qui passe par un ensemble de points discrétisés P_i.
- Depuis chacune de ces positions P_i,
  - positionner précisément le capteur stéréo sur la demi-sphère en P_i.
  - prendre une paire d'images.
  - par l'algorithme de stéréovision, obtenir une image 3D sur la scène observée.
- Recaler et Fusionner chaque image 3D dans un seul nuage de points 3D.
- Construire un maillage triangulaire à partir du nuage intégrant toutes les vues, afin d'obenir une représentation géométrique 3D générique de l'objet analysé.

Notons qu'avec cette séquence d'actions, nous ne modélisons pas la face support de l'objet avec la table. Il conviendrait de compléter ce processus par une action dédiée pour que le capteur perçoive la face non modélisée et complète le modèle : saisie de l'objet par le robot, dépose sur une autre face pour pouvoir percevoir celle qui était occultée, ou bien présentation de cette face occultée au capteur, l'objet étant toujours dans la pince ou la main du robot.

2.2.3 Un Système de Reconnaissance d'objets 3D de forme libre.

Dans le cas de la reconnaissance d'objets il existe aussi une grande quantité de publications à ce sujet, mais comme dans le cas de la reconstruction d'objets, ces travaux ont souvent des limitations : conditions bien définies d'exploitation, systèmes dédiés à des classes spécifiques d'objet (par exemple, la reconnaissance de visages) ou évalués sur des objets d'apparence ou de forme discriminantes (cannettes de soda...).

Initialement, dans un contexte de vision industrielle, les systèmes de reconnaissance permettaient

d'identifier des objets 2D ou considérés comme tels, objets présentés sur un plan ; les algorithmes exploités pour cela étaient similaires à ceux créés pour reconnaître des caractères ou pour interpréter des cartes. Les approches fondées sur la classification, fondées sur des paramètres globaux extraits sur les formes à reconnaître, ne s'appliquent qu'à des objets complétement vus ; les approches structurelles, fondées sur des algorihmes sur des graphes (recherche de cliques maximales, de sous-graphes isomorphes...), ont permis de reconnaître des objets partiellement occultés. Néammoins ces approches nécessitent une étape robuste d'extraction de primitives depuis des images : jonctions, segments, contours, régions... or cette phase de segmentation est toujours considérée comme très difficile.

Avec l'introduction de la modélisation 3D par des capteurs laser sont apparus des algorithmes de reconnaissance à partir de données 3D, donc fondés sur des appariements de primitives 3D. Des caractéristiques invariantes, définies par des descripteurs de courbure ont été proposées pour rendre ces approches plus robustes.

Avec la stéréovision, on dispose à la fois de données 3D et de données photométriques : on peut donc en théorie combiner des primitives 2D fondées sur l'apparence des objets sur des images, et des primitives 3D fondées sur des propriétés géométriques, afin de reconnaître des objets. Un de nos objectifs consiste à proposer une approche générique qui permette une telle combinaison : dans la mesure où nous traitons à la fois de la construction d'un modèle 3D pour des objets, et de la reconnaissance de ces objets, nous maîtrisons toute la chaîne algorithmique pour extraire de manière robuste des indices 2D et/ou 3D exploitables pour la reconnaissance. Notons qu'un autre avantage de l'approche mixte, est de pouvoir être utilisée aussi bien depuis des données acquises en vision monoculaire qu'en vision stéréo.

Ces dernières années, les systèmes de reconnaissance visuelle intégrés en robotique, exploitent des primitives ponctuelles, appelées *points d'intérêt*, qui peuvent être extraites et appariées de manière robuste : nous tirerons aussi parti de ces points aux propriétés d'invariance bien caractérisées dans la littérature. Mais nous allons aussi profiter du fait que nos capteurs sont montés sur un robot, pour les déplacer pendant le processus de reconnaissance, afin d'améliorer les performances de notre système.

La phase de reconnaissance est réalisée dans deux étapes : d'une part l'apprentissage des caractéristiques sur les objets et d'autre part, la reconnaissance de ces objets à partir de l'acquisition de données sensorielles, de l'extraction des caractéristiques de la scène courante et de leur mise en correspondance avec les caractéristiques préalablement apprises.

Implémentation de l'Algorithme de Reconnaissance d'objets 3D.

Comme dans le cas de la modélisation 3D, dans notre contexte de robotique de service, nous supposerons que les objets à reconnaître sont posés sur une table ; le robot peut déplacer le capteur visuel (mono ou stéréo) au dessus de l'objet. Les deux fonctions d'apprentissage et de reconnaisance sont décrites ci-dessous.

Phase d'apprentissage :

- Générer une trajectoire sur une demi sphère centrée autour de l'objet, qui passe par un ensemble de points discrétisés P_i.
- Depuis chacune de ces positions P_i,
  - positionner précisément le capteur stéréo sur la demi-sphère en P_i.
  - Acquérir des données sensorielles (2D et/ou 3D) depuis P_i.
  - Obtenir les caractéristiques 2D et/ou 3D de l'objet visible depuis cette position P_i.
  - Mémoriser dans une base de données les caractéristiques obtenues.
- Donner un nom symbolique à l'objet appris et l'associer à ces associations vues/caractéristiques dans la base de données.

On peut remarquer que, concernant le robot, cette séquence est identique à celle de reconstruction 3D ; dans la pratique, les deux fonctions d'apprentissage, celle du modèle 3D de l'objet et celle d'un modèle adapté à sa reconnaissance, devront être exécutées en même temps. Par ailleurs, la même restriction s'applique : il conviendrait de compléter la base de données, avec les caractéristiques 2D ou 3D qui peuvent se trouver sur la face d'appui de l'objet sur la table.

Phase de reconnaissance :

- Détecter le *TRUC* à reconnaître dans la scène (objet isolé ou vrac d'objets).
- Positionner une demi sphère centrée autour de ce *TRUC* à reconnaître ; on peut récupérer dans la base de données, l'ensemble de points P_i depuis lesquels les modèles sont appris.
- Choisir dans cet ensemble de points P_i, une position initiale $Preco_0$.
- Puis, jusqu'à ce que toutes les composantes du *TRUC* soient reconnus :
  - positionner précisément le capteur sur la demi-sphère en $Preco_j$.
  - Acquérir des données sensorielles depuis $Preco_j$.
  - Obtenir pour l'actuelle position du capteur, les caractéristiques 2D et/ou 3D de la scène courante.
  - Comparer les caractéristiques extraites de la scène avec celles mémorisées dans la base de données.
  - Utiliser une méthode probabiliste pour déterminer, parmi l'ensemble de points P_i discrétisés sur la demi-sphère, la position optimale $Preco_{j+1}$ du capteur depuis laquelle on obtiendra le plus d'informations pour progresser dans le processus de reconnaissance.
- Sortir le résultat, l'exploiter dans d'autres fonctions...

2.2.4 La saisie d'un objet 3D par le robot

Dans notre cas, l'objectif expérimental final concerne la prise d'un objet posé sur une surface plane, par exemple une table. Nous avons deux cas : la saisie d'un objet connu et la saisie d'un objet inconnu. Cette phase a donc plusieurs étapes :

- Accoster la table sur laquelle il y a un objet.
- Reconnaître l'objet posé sur cette table. Deux cas peuvent survenir.
- *Objet connu* :
  - Localiser l'objet dans l'espace.
  - Choisir la position de prise la plus adaptée vu la position de l'objet et la configuration du robot.
- *Objet inconnu* :
  - Construire un modèle de l'objet : construction 3D et apprentissage du modèle pour de futures reconnaissances.
  - Détermination d'un catalogue de positions de prise de l'objet, en fonction du modèle de la pince ou de la main disponible.
- Finalement, exécuter la prise de l'objet en calculant la trajectoire nécessaire pour emmener la pince ou la main en position de prise.

Les étapes d'accostage par une plate-forme mobile, de planification, de raisonnements géométriques pour sélectionner la position de la pince vis-à-vis de l'objet à saisir, puis de commande du robot pour emmener la pince en position de prendre l'objet, ne sont pas de notre ressort.

2.3 Travaux précédents sur notre problématique.

Parmi les différentes étapes que nous avons décrites dans la dernière section, nous n'avons contribué que dans les phases de reconstruction et de reconnaissance d'objets 3D. L'étape de saisie d'un objet se fonde sur l'intégration entre les modules de reconstruction et de reconnaissance 3D avec les modules de planification de mouvements et de commande du manipulateur, modules qui sont étudiés par ailleurs et dont des versions simplifiées sont déjà sur JIDO. Nous présentons séparément quelques éléments d'état de l'art sur nos domaines d'intérêt.

2.3.1 Reconstruction d'objets 3D

La reconstruction 3D est le processus au moyen duquel, une représentation 3D est créée dans la mémoire d'un ordinateur, afin de décrire les caractéristiques géométriques d'un objet réel (dimensions, volume, forme...). Il existe en perception artificielle, une multitude de techniques de reconstruction 3D, en fonction de la méthode d'acquisition de mesures sur l'objet, et de la représentation 3D finale. On distingue les étapes (1) d'acquisition de données 3D, généralement des images de points 3D, (2) de recalage des données acquises depuis des points de vue différents et (3) de construction d'une représentation surfacique ou volumique à partir des données recalées. Il existe diverses stratégies :

– méthodes incrémentales de type *estimation* : on crée la représentation dès les premières acquisitions, et on fusionne les nouvelles données acquises depuis les points de vue suivants. Ce sont les approches de type SLAM (pour *Simultaneous Localization and Mapping.*
– méthodes globales de type *optimisation* : on recale au fur et à mesure des acquisitions, les données 3D dans un nuage 3D (ensemble de points 3D non organisés), et à la fin, on crée la représentation de l'objet de manière optimale au vue de tous les points disponibles.

Pour des objets de forme libre, on applique généralement une approche globale, car la construction incrémentale de maillage 3D est très complexe. Un algorithme de création d'un maillage 3D à partir d'un nuage de points 3D doit être capable d'effectuer la connexion de l'ensemble de points représentatifs de l'objet sous forme d'éléments de surface plane, que ce soient des triangles, des hexagones, des carrés ou tout autre forme géométrique 2D.

Les algorithmes développés jusqu'à présent, résultent d'un compromis entre le coût de calcul, le volume occupée en mémoire par le modèle, et la qualité du maillage obtenu. Les algorithmes qui travaillent avec des nuages de points, essayent d'obtenir une représentation des relations inter-points, sous la forme d'une matrice de connexions. Cette matrice décrit comment des points de l'ensemble initial sont reliés entre eux. Si nous employons des triangles (méthode assez commune), cette matrice a pour taille, 3 x N (N étant le nombre total de triangles contenus dans le maillage final), c'est-à-dire que chaque ligne de la matrice représente un triangle dans l'espace.

Plusieurs algorithmes permettent de générer un maillage à partir d'un nuages de points 3D : *Marching Cubes*, *Ball Pivoting,Surfaces déformables*, *Alpha shapes*, et de nombreuses méthodes dérivées de la triangulation de Delaunay. L'efficacité de chacune est définie par le temps de calcul pour traiter un nuage de taille donnée, et par la qualité finale du maillage. Cette qualité n'est pas si facile à évaluer : présence ou non de trous dans la surface, distance entre la surface maillée et les points de mesures, robustesse aux points aberrants présents dans le nuage...

Ce dernier critère est essentiel quand on exploite des données stéréo. En effet, la stéréo produit des points erronés. Il existe donc des points dans le nuage, qui risquent de mettre en péril la cohérence du maillage : des points qui sont très proches entre eux, des points faux (artefacts de l'acquisition) ou très bruités, des points redondants... n'offrent aucune information pour la reconstruction. Par exemple, si nous voulons représenter un cube dans l'espace, simplement huit sommets et douze triangles sont suffisants, le reste de l'information est redondante. Dans notre cas, la reconstruction tridimensionnelle est effectuée en partant d'un ensemble de points de grande dimension, dans lequel de très nombreux points n'offrent aucune information. Il vaut mieux les éliminer avant de mailler. Le premier objectif est donc la réduction de l'ensemble de points pour ne garder que les points essentiels.

La réduction du nombre de points dans un nuage de points est une fonction connue sous le nom *décimation*. Elle effectue un sous-échantillonnage sur un nuage de points 3D. Ce sous-échantillonnage permet d'obtenir un nuage moins dense et donc, plus facile à manipuler afin de réaliser le recalage et la construction du maillage d'une façon plus facile et rapide. La décimation n'est pas une opération facile à réaliser, surtout que dans la littérature, la plupart voire tous les travaux sur ce domaine opérent déjà sur un maillage polygonal (des triangles la plupart du temps) et non sur un nuage de points 3D.

2.3.2 Reconnaissance d'objets 3D

La reconnaissance d'objets est une étape clef dans tout processus générique de vision artificielle. Le processus de vision comprend deux étapes : l'apprentissage et la reconnaissance elle-même. En général, dans les modèles des objets, on stocke la plus petite quantité d'informations pour optimiser le processus de reconnaissance [151]. La sélection de ces informations est le choix le plus critique du processus de reconnaissance. Les modèles peuvent être fournis a priori au système par apprentissage supervisé, mais peuvent être aussi acquis automatiquement, par apprentissage non supervisé. Une fois les modèles appris, un objet perçu peut être reconnu par comparaison des caractéristiques extraites de données sensorielles acquises sur cet objet, avec celles stockées dans les modèles ; la sélection de la technique de comparaison entre vues et modèles, est le second choix critique d'une méthode de reconnaissance.

Les premières méthodes de reconnaissance d'objets 3D, exploitaient des modèles structurels, qui représentaient chaque objet par un ensemble de primitives géométriques. Le plus grand problème de ces techniques est que les modèles présentaient une instabilité, du fait qu'un même objet peut être décomposé dans différentes combinaisons de primitives [28] [79].

Les modèles fondés sur l'apparence sont apparus pour faciliter l'apprentissage automatique d'ob-

jets [112]. Cependant, une comparaison directe des images n'est pas possible dans la pratique, puisqu'elle implique une exploitation de trop de ressources informatiques (mémoire, temps de calcul...), et n'est pas souhaitable, parce que les images brutes présentent une information non structurée, très sensible aux bruits ou aux changements d'illumination. Pour cette raison, la majorité des systèmes de reconnaissance se basent sur l'extraction de caractéristiques significatives sur l'ensemble de vues acquises pour apprendre ces modèles. Les caractéristiques choisies doivent être invariantes à des transformations dans les images (échelle, rotation, contraste...) provoquées par les changements de points de vue.

Quelques méthodes extraient l'information significative des images brutes complètes au moyen de différents types de décomposition vectorielle. Ainsi, chaque vue est transformée en un point d'un espace N-Dimensionnel. Un ensemble de vues sur un objet 3D, est représenté par une trajectoire dans cet espace [97]. De cette manière, après avoir observé une scène, le système peut décider quel objets y sont présents. Toutefois, ces méthodes sont très sensibles à l'apparition d'ombres et aux changements d'illumination [134]. En outre, des ambigüités peuvent survenir si deux courbes s'intersectent, ce qui limite l'application de ces méthodes à des ensembles d'objets qui ne présentent pas de vues semblables entre eux [29].

Les méthodes de reconnaissance basées sur l'apparence ont été testées abondamment sur des bases d'images disponibles sur le Web : citons la base COIL, les images du Teddy l'ours... il existe plusieurs défis pour leur application dans des situations réelles. Ainsi, ces méthodes sont très sensibles à la position angulaire entre la caméra et l'objet observé lorsqu'on répète un même essai. En outre, dans la pratique, depuis un point de vue donné, un objet peut être partiellement occulté, spécialement quand il s'agit d'objets volumineux [39].

2.4 Contributions

Les paragraphes suivants décrivent les contributions que nous avons essayées d'obtenir pendant la préparation de cette thèse.

2.4.1 Contribution à la Reconstruction 3D.

La reconstruction d'objets 3D nécessite plusieurs étapes ; nous nous limitons ici à la modélisation exploitant des données visuelles 3D denses. Les étapes sont :

- Acquisition d'images stéréo ;
- Obtention des données 3D par stéréo corrélation.
- Décimation des images de points 3D.
- Recalage dans un nuage de points 3D, des images 3D décimées, obtenues depuis divers points de vue.
- Construction d'un maillage triangulaire à partir du nuage.

Signalons que les étapes mentionnées ci-dessus sont spécifiques à notre choix initial d'exploiter un capteur stéréo pour acquérir des données. Ces étapes seraient différentes, par exemple en considérant les méthodes récentes développées en Vision monoculaire, exploitant l'algorithme *Sparse Bundle Adjustment (SBA)* [85].

Acquisitions 3D.

En relation à l'acquisition d'images 3D, nous avons intégré une fonction qui calcule la discrétisation de la sphère unité (souvent appelée sphère de Gauss), cela pour discrétiser à intervalles réguliers, les points de vue sur une scène à analyser. En réalité, nous n'utilisons qu'un quart de cette sphère du fait des contraintes d'accessibilité des positions discrétisées sur la sphère, par le bras robotique. Comme nous avons une plate-forme mobile, nous pourrions la déplacer afin d'atteindre tous les points de vue souhaités, mais dans un premier temps, le système robotique reste en position fixe et nous ne déplaçons que le bras.

La discrétisation de la demi sphère est obtenue par rapport au centre de gravité de la scène que le robot doit analyser ; cette sphère discrétisée sera exploitée aussi bien pour apprendre le modèle géométrique que pour apprendre la modèle d'apparence. L'algorithme global procède donc de la manière suivante :

- Acquisition d'une première image 3D de la scène.
- Segmentation de la zone d'intérêt.
- Calcul du centre de gravité de cette zone.

– Calcul des points de vue sur la sphère autour du centre de gravité.

Nous supposons que l'objet d'intérêt est posé sur une table, ce qui facilité l'étape de segmentation. Le rayon de la sphère centrée sur le centre de gravité des points 3D acquis sur l'objet, est choisi en fonction du champ de vue des caméras, donc des focales de leurs objectifs : ici, ce rayon est de 60cm.

Le capteur stéréo monté sur l'organe terminal du bras de JIDO comme on peut le voir en figure 2.7, sera donc placé dans des positions discrétisées sur cette sphère. Ces positions d'abord définies dans le repère de la sphère, seront ensuite exprimées via des transformations, dans le repère du bras, afin de vérifier leur accessibilité et de calculer les trajectoires du bras.

Décimation

Les algorithmes de décimation qu'on peut trouver dans la littérature, traitent principalement de la réduction des polygones ou des triangles dans un maillage, et non de la réduction de points dans un nuage de points. Il n'existe pas de travaux sur la décimation des nuages de points, comme cela est indiqué par Volodine *et al.* [153] qui ont abordé ce sujet.

Mais il nous fallait une solution qui soit rapide et efficace ; nous avons donc proposé l'algorithme suivant :

– Nous prenons des points à intervalles réguliers
– Nous calculons les K voisins les plus proches.
– Ensuite, pour chaque point sélectionné et ses K-voisins :
  1. Nous estimons le plan qui passe par le point et ses voisins.
  2. Nous projetons tous ces points sur le plan .
  3. Nous rejetons les voisins qui sont les plus proches du plan.
  4. Nous retournons sur l'espace de l'objet.
  5. On garde les points restants.

Le Recalage entre deux points de vue

Dans le domaine du recalage d'images acquises depuis des points de vue différents, beaucoup de travaux ont visé à résoudre cette problématique [32] [24] [141]. L'algorithme le plus connu et donc, le plus utilisé est appelé **Iterative Closest Points (ICP)**. Au LAAS cet algorithme de recalage a été étudié et développé par A. Restrepo [133] ; cette méthode converge seulement dans les cas où les différences d'angles de vue entre les deux positions du capteur, ne sont pas très grandes, ou quand on dispose d'une estimée initiale de cette différence.

Nous avons implémenté une variante de l'algorithme, variante appelée *ICP Pseudo-couleur* exploitant la couleur pour apparier des points acquis depuis deux positions du capteur. Nous l'avons appelé ainsi car nous assignons à chaque point 3D des nuages à recaler, une information couleur virtuelle, en fonction de leur position dans l'espace. A l'aide de ces informations de pseudo-couleurs, nous utilisons l'algorithme développé par Druon et al. [44], algorithme qui lui, exploite une vraie couleur.

Paramétrisation sphérique pour la construction du maillage.

Un capteur 3D (scanner 3D, stéréovision...) permet d'acquérir une grande quantité de points 3D échantillonnés sur la surface 3D de l'objet que l'on souhaite modéliser en 3D. Ces points sont généralement organisés sous forme d'une image 3D, ce qui donne des informations de voisinage partiel. Malgré cela, après le recalage de plusieurs images 3D, on obtient un nuage de points 3D inorganisés, sans aucune structure exploitable pour la saisie d'un objet. Il faut donc construire une représentation surfacique ; ce sera un maillage pour un objet de forme libre, représentation définie par un ensemble de polygones (généralement des triangles) et par une structure de connectivité entre eux.

Le maillage peut-être obtenu en utilisant les différents algorithmes qui existent actuellement, déjà cités ci-dessus, par exemple *Marching Cubes*. Mais dans notre cas, dans un premier temps nous ne voulons pas avoir une représentation la plus fidèle possible, mais plutôt avoir une représentation grossière de l'objet que nous allons modéliser. Tout cela parce que la fonction de planification des positions de prise sur l'objet, connaissant la géométrie de la pince ou de la main montée sur le manipulateur, n'a pas besoin d'avoir autant de polygones. En plus cette fonction a besoin d'un modèle surfacique complètement fermé pour éviter, au moment de planifier la trajectoire de prise, une mauvaise planification du mouvement

du bras depuis la position initiale jusqu'à la position de prise : or de nombreuses méthodes classiques, *Marching Cubes, Alpha Shapes, Ball Pivoting...*, ne garantissent pas l'absence de trous dans le maillage.

C'est la raison pour laquelle nous avons implémenté un algorithme de reconstruction qui soit en même temps rapide et qui produise un modèle exploitable pour planifier la saisie de l'objet. Cet algorithme est appelé *Paramétrisation Sphérique*. Les différentes étapes sont :

- Estimer la normale en chaque point, par approximation d'un plan local.
- Projeter le point 3D sur une sphère unitaire.
- En utilisant un algorithme de recherche d'enveloppe convexe, trianguler la sphère.
- Récupérer les informations des arêtes.
- En utilisant l'information des points 3D et les arêtes, obtenir le maillage 3D de l'objet.

Cet algorithme est rapide, mais il a un inconvénient important : il travaille sur des objets convexes ou quasi-convexes (pas de repliement de la surface de l'objet, par rapport au centre de la sphère englobante). Cela nous restreint sur la nature des objets que nous pouvons modéliser ; comme la plupart des objets d'usage courant sont convexes, nous pourrons modéliser une large gamme d'objets avec la méthode proposée.

2.4.2 Contribution à la reconnaissance d'objets 3D de forme libre.

Notre contribution dans le domaine de la reconnaissance d'objets est focalisée sur la reconnaissance active d'objets. Nous proposons une approche probabiliste fondée sur la Maximisation de l'Information Mutuelle [41] [40]. Cette approche exploite des histogrammes pour chacune des caractéristiques que l'on veut utiliser pour discriminer des objets : durant un travail préliminaire, nous l'avons d'abord appliquée en exploitant uniquement les histogrammes couleur.

Ensuite, nous avons pris en compte, pour représenter chaque classe d'objets à reconnaître, en plus des caractéristiques colorimétriques, des informations géométriques apprises depuis différents points de vue autour de ces objets. Ces caractéristiques sont apprises hors ligne, sur des séquences d'images dans lesquels l'objet à apprendre est isolé, posé sur une table non texturée et de couleur uniforme. Dans chaque image on extrait la région image de l'objet ; cette région est séparée du fond, par une méthode de contours actifs ; ce contour converge sur la silhouette de l'objet. De cette région, sont extraits les caractéristiques sur la silhouette décrite par la signature polaire, ou *Shape Signature*, sur les contours extraits à l'intérieur de la région, décrits par le descripteur appelé *Shape Context* [21], et sur des points d'intérêt extraits dans cette région par les détecteurs proposés soit par Harris & Stephens, soit par D.Lowe (DOG en multi-résolution), puis caractérisés par les descripteurs SIFT [86].

De plus, pour être robuste aux occultations, nous adoptons une approche hiérarchique pour décrire chaque point de vue sélectionné en phase d'apprentissage sur chaque objet. D'abord au plus haut niveau, l'objet est décrit par la région entière délimitée par sa silhouette ; cette région ne sera reconnue que si l'objet est vu entièrement et est non accolé à un autre objet (la silhouette n'englobe pas plusieurs objets). Puis aux niveaux suivants, l'objet est décrit par des sous-régions obtenus par découpage récursif de cette région entière. En utilisant les points SIFTs nous obtenons un système plus robuste encore : les SIFTs sont également associés aux sous-régions auxquelles ils appartiennent.

Finalement, dans notre base de connaissances, le modèle de chaque classe d'objet est mémorisé, pour chaque point de vue, par un ensemble d'histogrammes différents et par une liste de points dotés de leurs descripteurs SIFT. Le robot dispose donc d'une base de données avec les modèles des objets décrits de manière hiérarchique ; nous proposons une méthode de reconnaissance séquentielle, active, robuste aux occultations.

Initialement, notre robot a donc détecté un ou plusieurs *TRUCs* posés sur la table. A chaque itération, le système acquiert une image et extrait de l'image les silhouettes de ces *TRUCs*, grâce à une méthode de contours actifs ; si rien de significatif n'est extrait, alors le système conclut de suite qu'il n'y a rien sur la table. Sinon il essaye d'identifier chacun des *TRUCs* extraits en comparant leurs caractéristiques avec ce qui a été mémorisé en phase d'apprentissage. S'il ne peut conclure, il choisit le prochain point de vue qui permettra d'avancer vers une décision : cette sélection se fait par maximisation de l'information mutuelle, en fonction de l'état courant du processus de reconnaissance et des modèles appris en phase d'apprentissage pour tous les objets que le robot est susceptible de reconnaître. Le système s'arrête quand tous les *TRUCs* détectés sur la table ont été identifiés avec une probabilité suffisante ; une classe Null permet de détecter qu'un *TRUC* est inconnu.

De nombreuses évaluations ont permis de valider cette approche. Du fait de différents problèmes

avec notre robot JIDO, la plupart de ces validations ont été menées à partir de bases d'images acquises depuis un appareil photo numérique, donc en considérant uniquement une acquisition monoculaire pour la reconnaissance. De ce fait, nous n'avons pas exploité au niveau de la reconnaissance, des caractéristiques 3D qui pourraient être apprises, pour chaque point de vue, depuis des données stéréo : seul un attribut, la densité sphérique, a été proposé et évalué en simulation.

2.5 Conclusion

Nous avons décrit dans ce chapitre, le contexte dans lequel nous avons effectué ce travail : la robotique d'aide à la personne, et plus spécialement, le projet COGNIRON qui a étudié de nombreuses thématiques liées au développement d'un robot compagnon de l'homme. Nous avons ensuite, centré notre propos sur la problématique de la reconstruction 3D et de la reconnaissance active d'objets 3D, en vue de leur saisie ; un bref état de l'art a été présenté, juste pour pouvoir situer nos travaux dans ces deux domaines. Finalement, nous avons décrit sommairement nos contributions, qui vont être décrites plus avant dans les deux chapitres suivants.

Chapitre 3

Reconstruction 3D

3.1 Introduction

Dans ce chapitre nous allons aborder nos contributions dans la thématique Modélisation 3D d'un objet. Il existe de très nombreux travaux en ce domaine, des équipes entières de chercheurs qui consacrent leurs activités à la reconstruction 3D (obtention des données 3D depuis des capteurs visuels), puis à la modélisation 3D (association des données 3D acquises sur un objet depuis différents points de vue). Nous avons rapidement évoqué l'état de l'art dans le chapitre précédent : il s'agit ici de décrire plus précisément chacune des étapes, et ensuite, de présenter les fonctions que nous avons développées pour réaliser la triangulation d'un nuage de point 3D. Rappelons que, par nuage, nous entendons un ensemble de points non structurés, donc qui n'exhibe pas les relations de voisinage inter-points via un tableau 2D (comme dans une image 3D acquise par stéréovision) ou 3D (comme dans une *voxel map*).

Le but de la modélisation 3D d'objets est donc d'obtenir un modèle 3D virtuel d'un objet 3D réel. Cette représentation informatique d'un objet 3D est une abstraction nécessaire pour qu'un système robotique puisse reconnaître cet objet, puis le localiser dans son espace de travail, puis le saisir à l'aide d'un préhenseur monté sur un bras, et enfin le manipuler de manière plus ou moins dextre selon les possibilités du préhenseur et les besoins de l'application. Il existe plusieurs solutions pour obtenir un tel modèle 3D :

- il peut être construit par un opérateur, via un outil de CAO (Autocad, Katia, Robmod...) : mais cette tâche est fastidieuse. En pratique, cela a été fait pour des objets structurés, tels que des objets polyédriques, ou des objets construits par des opérations ensemblistes entre des volumes élémentaires cylindres, sphères ou boîtes (CSG, pour *Constructive Solid Geometry*). Pour être exploité, un tel modèle CAO fourni par l'Homme, doit faire référence à des indices perceptuels que le système robotique pourra détecter avec ses capteurs : sommets, faces ou arêtes dans le cas d'un objet structuré, mais que peut-on exploiter sur des objets de forme libre ? et surtout ces outils CAO ne prennent pas en compte les indices visuels, extraits à partir de l'apparence de l'objet dans des images.
- si on s'intéresse à l'autonomie des machines, le modèle 3D des objets doit être construit de manière automatique par notre robot, qui exécute cette fonction dès qu'il détecte un objet qu'il ne connaît pas. L'avantage est qu'il construit ce modèle à partir de données sensorielles acquises par les mêmes capteurs qui serviront ensuite à reconnaître, localiser ou contrôler une action de saisie ou de manipulation ; le modèle sera donc facilement exploitable. Par contre, durant la phase d'apprentissage, il faut que le robot distingue les indices perceptuels qui font parti de l'objet de ceux qui sont dans le fond. Ce problème de *segmentation* peut s'avérer très complexe.
- dans la pratique et vu l'état de l'art actuel sur la segmentation, la fonction de modélisation 3D est souvent réalisée de manière supervisée : c'est l'opérateur qui met le robot en mode d'apprentissage, qui présente séparément les objets à apprendre selon plusieurs points de vue avec un protocole expérimental qui simplifie la segmentation.

M.Cottret [37] a consacré son travail de thèse au LAAS, au problème de l'apprentissage automatique, donc à la détection des objets inconnus présents dans l'environnement, puis à leur segmentation et à la construction d'un modèle d'apparence de ces objets pour pouvoir ensuite les reconnaître. Outre le fait qu'il n'a pas traité de Modélisation 3D, sa thèse a clairement montré la difficulté actuelle de la détection

et de la segmentation. Nous proposons donc de réaliser hors-ligne, l'apprentissage des modèles 3D des objets que notre robot devra manipuler, avec supervision par un opérateur.

Afin de réaliser cette fonction de reconstruction tridimensionnelle d'un objet, il est nécessaire de passer par différentes étapes :

- L'acquisition des données sensorielles depuis différents points de vue autour de l'objet ; nous exploitons des caméras ; il s'agit donc d'acquisitions d'images.
- L'obtention de données 3D ; nous exploitons un algorithme de stéréovision par corrélation dense des pixels entre les deux images, afin de produire une image de points 3D (points avec une relation de voisinage héritée des images d'origine).
- La décimation des nuages de points 3D reconstruits de chaque point de vue, cela pour limiter la complexité des étapes suivantes.
- Le recalage des images 3D acquises depuis différents points de vue, et la fusion de toutes ces images 3D, dans un seul nuage de points 3D.
- Enfin, la construction d'une représentation 3D à partir de ce nuage. Dans la mesure où nous modélisons des objets de forme libre, il s'agit d'un maillage triangulaire.

Chacune des étapes mentionnées dans cette liste, pourrait faire l'objet d'une thèse en vision par ordinateur : choix des positions d'acquisition, vision 3D, recalage et fusion, construction de maillage... Au début de nos travaux, A.Restrepo Specht [133] traitait de cet unique sujet : nous devions initialement exploiter ses contributions sur le recalage, la fusion et la construction de maillage, et valider ses méthodes avec la stéréovision. Cela s'est avéré délicat du fait des incertitudes des données stéréo : aussi avons nous proposé des méthodes simplifiées pour recalage et construction du maillage, afin de poursuivre vers notre objectif global, la manipulation d'objets 3D par notre robot de service Jido.

Dans ce chapitre nous allons évoquer d'abord chacune des étapes ; puis nous décrirons les méthodes simplifiées que nous avons développées pour modéliser un objet 3D à partir d'un nuage de points 3D acquis par stéréovision dense.

3.2 Positionnement du capteur autour de l'objet.

Pour la construction du modèle 3D d'un objet, il est nécessaire d'avoir de nombreuses images de cet objet, acquises autant que possible, de tous les points de vue autour de celui-ci. Il existe plusieurs protocoles expérimentaux classiques :

- on peut déplacer l'objet devant le capteur maintenu dans une position fixe [148] [111] [110]. Dans le cas le plus simple, l'objet est posé sur une platine, avec un fond uniforme pour faciliter la segmentation objet/platine ; typiquement la platine est contrôlée pour acquérir des données sensorielles tous les N degrés, N choisi en fonction de la résolution nécessaire pour le modèle. Dans des approches plus récentes en Robotique, l'objet est déplacé par l'opérateur, ou par le robot lui-même ; il est alors tenu dans la main ou dans la pince selon différentes positions de prise, afin de montrer tous les points de vue. Se posent alors les problèmes du contrôle des acquisitions (n'acquérir que lorsqu'un nouveau point de vue est présenté au capteur) et de la segmentation objet/main ou objet/pince.
- on déplace le capteur autour de l'objet posé sur un support [133], avec l'aide d'un dispositif de positionnement ou, comme dans notre cas, avec un bras robotique. Reste à traiter la segmentation objet/support.

Dans notre contexte Robotique, nous avons choisi de déplacer automatiquement le capteur stéréo et de l'emmener dans les positions souhaitées pour acquérir les images. La procédure dite du *Next Best View (NBV)*, a pour but de calculer la meilleure position du capteur vis-à-vis de l'objet, en fonction de l'état courant de la construction du modèle de cet objet. La procédure NBV doit (1) réduire la quantité d'images acquises sur l'objet, et (2) garantir d'avoir au final un modèle complet de l'objet, avec une précision ou résolution optimale.

Citons quelques approches proposées pour la problématique NBV : Pito *et al.* [111] [109] [108] [110] définissent une nouvelle représentation, l'espace positionnel, dans lequel il est possible de différencier les directions d'échantillonnage pour observer les portions non encore vues de l'objet, de celles qui permettent d'observer de nouveau des parties déjà connues. En outre l'espace positionnel permet de représenter les

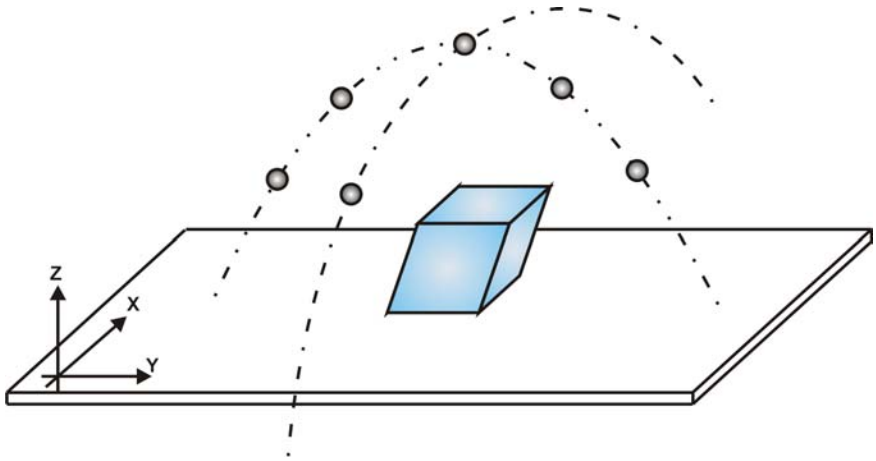


FIGURE 3.1 – Quelques points de vue.

mouvements possibles entre la caméra et l'objet. De son côté Sequeira *et al.* [127] [126] [128] présentent un processus actif de perception exploité pour construire une carte 2D d'un environnement réel en utilisant de multiples vues acquises par un capteur SICK (télémètre laser avec un balayage horizontal uniquement) ; la sélection de la position du capteur pour acquérir la vue suivante se fonde sur la détection des occultations entre la vue courante et les vues précédentes.

Evoquons aussi le développement de Abrams *et al.* [7] qui mettent en oeuvre un système de planification dynamique capable de générer à chaque étape, la situation et les paramètres optiques du capteur afin de modéliser un objet 3D. Sanchiz *et al.* [122] [9] génèrent la position suivante du capteur dans un processus de modélisation, en maximisant des fonctions d'utilité calculées pour chacune des positions possibles du capteur : l'utilité dépend de l'état courant du modèle, du gain hypothétique d'informations, de la facilité de recalage, de la complexité pour atteindre cette position depuis la position courante... Li *et al.* [80] calculent le NBV par deux étapes : la direction dans laquelle orienter le capteur, est d'abord déterminée, puis la position suivante précise du capteur est calculée en optimisant le gain d'informations acquises depuis chaque position et orientation potentielles. De même, Wong *et al.* [157] optimisent une fonction objectif, qui est fondée sur la quantité d'informations non connues pour chaque point de vue potentiel ; ces points de vue potentiels sont distribués de façon uniforme autour de l'objet. Finalement, Mitran et Ferrie [95] montrent que le gradient de l'intensité d'une image peut être utilisé pour induire de manière satisfaisante la direction de mouvement d'un capteur de vision ; cette approche se base sur l'hypothèse que les zones les plus foncées d'une image acquise autour de l'objet, correspondent toujours à des zones encore non explorées par le capteur ; elle n'exploite pas d'autre stratégie pour savoir si une zone a été précédemment vue ou non.

Dans notre cas, nous n'avons pas traité de cette problématique NBV. Nous discrétisons un ensemble de points de vue sur une sphère placée autour de l'objet dans l'espace de travail de notre bras, puis nous emmenons notre capteur en chacun de ces points de vue, comme cela est illustré en figure 3.1. L'exploitation du bras manipulateur va nous imposer certaines contraintes pour les déplacements puisque le bras a un espace de travail limité ; les positions atteignables sont définies par les modèles direct et inverse du bras.

La figure 3.2 présente la discrétisation que nous faisons sur une sphère de Gauss positionnée autour de l'objet. Cet objet est supposé posé sur un support plan, typiquement une table ; de ce fait, la figure 3.3 montre que nous n'utilisons que la moitié supérieure de la sphère. En fait, vu les problèmes d'accessibilité des points sur la sphère quand le robot Jido (et donc la base du bras) est dans une position donnée par rapport à la table, nous n'allons nous servir que d'un quart de la discrétisation de la sphère de Gauss pour réaliser l'acquisition d'images de l'objet d'intérêt. Typiquement il conviendrait de déplacer la plateforme Jido, afin de pouvoir positionner notre capteur sur une zone plus grande de la sphère : nous n'avons pas exécuté un tel scénario ; la plateforme Jido est statique durant la phase d'apprentissage.

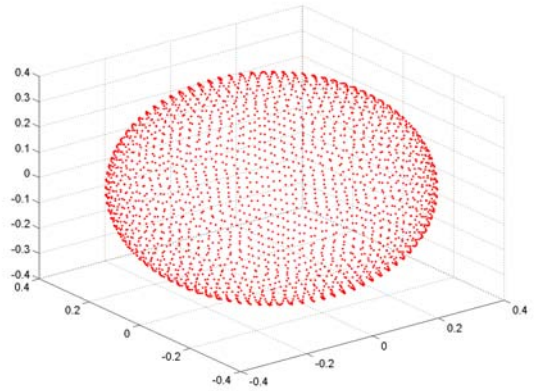


FIGURE 3.2 – Discrétisation de la sphère de Gauss

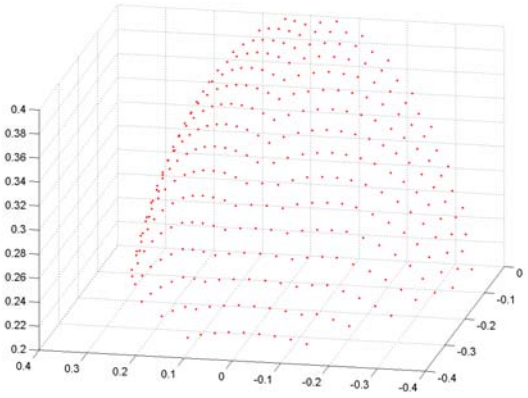


FIGURE 3.3 – Un quart de la discrétisation est seulement atteignable par un bras manipulateur.

Le choix du rayon de la sphère est important ; il dépend de la taille de l'objet, de la précision et résolution souhaitées sur le modèle final, du bras, des focales montées sur nos caméras... on pourrait envisager d'avoir plusieurs sphères concentriques pour adapter les positions choisies pour le capteur en fonction du contexte. Ici le rayon est fixe et égal à 60cm.

Cette sphère sera exploitée aussi bien pour construire le modèle 3D de notre objet, que pour apprendre des caractéristiques discriminantes et les exploiter lors de la reconnaissance (voir chapitre suivant).

3.3 Acquisitions de données 3D sur l'objet.

Dans cette section nous allons décrire les différentes étapes nécessaires pour acquérir des données 3D depuis un point de vue sur l'objet. Ces processus sont :

- Le calibrage hors-ligne des caméras,
- l'acquisition des images,
- la Stéréo corrélation.

3.3.1 Calibrage

Afin d'obtenir une bonne reconstruction 3D, nous devons avoir un bon calibrage de notre banc stéréo. Le terme calibrage est un terme français, construit à partir du terme anglais calibration ; comme V.Garric dit en [54], en mécanique, le mot calibrage désigne l'*Identification des paramètres constitutifs d'un modèle.*

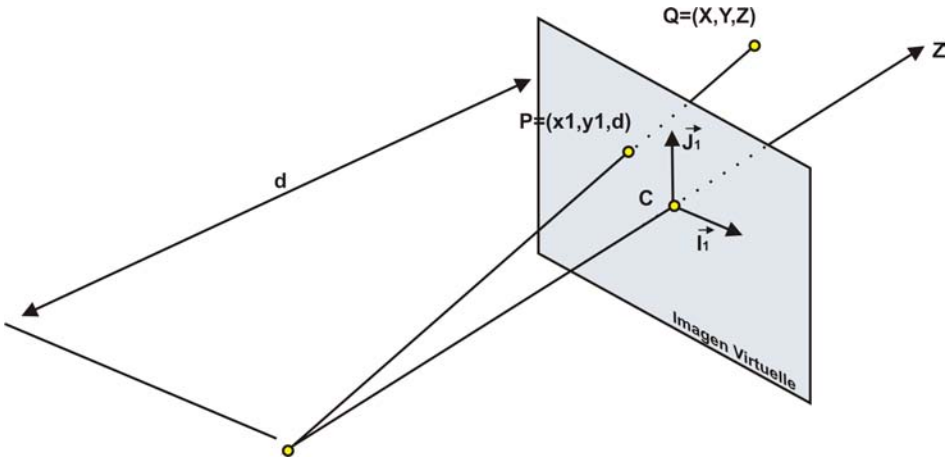


FIGURE 3.4 – Algorithme de calibration

Pour la vision, nous allons plutôt utiliser le terme anglais *calibration* au long de ce manuscrit pour nous référer à la dite procédure. On peut définir la calibration d'un banc stéréo comme la procédure d'estimation des paramètres qui apparaissent dans le modèle du dispositif. Ces paramètres sont liés aux caractéristiques des caméras et à leurs positions dans l'environnement : ils permettront la reconstruction de points 3D.

Il faut d'abord établir le modèle de la caméra, puis d'un capteur stéréo. Un modèle de caméra est donné par tous les paramètres dont dépend la matrice de projection. Cette matrice permet d'associer à un point 3D (x, y, z) connu dans le repère du monde, les coordonnées du pixel (u, v) qui lui correspond sur l'image acquise par la caméra (voir figure 3.4). Ce modèle de caméra dépendra donc de :

- la distance focale d des lentilles de la caméra,
- la taille des pixels,
- la position du point principal dans l'image, correspondant à l'intersection entre l'axe optique et le plan image,
- et la position et l'orientation de la caméra.

On distingue les paramètres intrinsèques qui ne dépendent que de la caméra (distance focale, taille des pixels, position du point principal), et les paramètres extrinsèques (position et orientation de la caméra dans l'environnement).

Paramètres intrinsèques

Nous associons d'abord à la caméra réelle, une caméra virtuelle parfaite. Le repère de cette caméra a son origine C sur le centre optique ; l'axe Z est porté par l'axe optique, dirigé vers la scène observée ; les axes X et Y sont parallèles aux lignes et colonnes du plan image. Ce plan image virtuel est perpendiculaire à l'axe optique, à une distance d du centre optique, vers la scène : le plan image réel est derrière, mais le modèle de la caméra peut s'exprimer plus simplement en exploitant ce plan image virtuel.

Dans ce repère de la caméra, le projeté d'un point réel (X, Y, Z) sur le plan image virtuel a des coordonnées métriques notées (x_1, y_1, z_1). La troisième coordonnée z_1 du projeté est égale à d de part le choix du repère caméra. Les coordonnées x_1 et y_1 sont calculées en fonction de X, Y, Z et de la distance focale d par le théorème de Thalès :

$$\frac{x_1}{d} = \frac{X}{Z} \text{ et } \frac{y_1}{d} = \frac{Y}{Z}$$

En multipliant ces égalités par d, on obtient :

$$x_1 = d\frac{X}{Z} \text{ et } y_1 = d\frac{Y}{Z}$$

En prenant en compte la discrétisation en pixel dans le plan image et le changement des coordonnées métriques (x_1, y_1) en coordonnées pixel (u, v) de type (ligne, colonne), on obtient une relation directe

$$Focal = \begin{bmatrix} 707.303 \\ -707.011 \end{bmatrix}$$
$$Centralpoint = \begin{bmatrix} 392.191 \\ 569.75 \end{bmatrix}$$
$$Distortion = \begin{bmatrix} -0.21154 \\ 0.05401 \end{bmatrix}$$

TABLE 3.1 – Paramètres de la caméra Videre

entre les coordonnées pixel (u, v) et les coordonnées 3D (X, Y, Z) :

$$u = u_0 + \alpha_u \frac{X}{Z} \text{ et } v = v_0 + \alpha_v \frac{Y}{Z}$$

Soit $(C, \vec{i_1}, \vec{j_1})$ le repère orthonormé dans le plan image virtuel d'équation $Z = d$. La projection P de (X, Y, Z) a pour coordonnées (x_1, y_1) dans ce repère, et nous avons donc :

$$P = C + x_1 \vec{i_1} + y_1 \vec{j_1}$$

En raison des imperfections, les axes de la caméra réelle ne sont pas tout à fait orthogonaux ; de même, les pixels ne sont pas tout à fait carrés. Ce sont des parallélogrammes. Soit $(C, \vec{i_2}, \vec{j_2})$ le repère de la caméra réelle sur laquelle sont projetés les pixels. Nous devons introduire l'angle θ entre $\vec{i_2}$ et $\vec{j_2}$, angle θ proche de 90^o mais pas nécessairement égal à 90^o. Si on suppose que $\vec{i_2} = \vec{i_1}$, alors on peut écrire $\vec{j_2} = \vec{i_1} \cos\theta + \vec{j_1} \sin\theta$.

L'ensemble de ces équations font donc apparaître cinq paramètres intrinsèques $(u_0, v_0, \alpha_u, \alpha_v, \theta)$, sachant que presque toujours, on considère que θ est égal à 90°.

Dans la formation de l'image interviennent aussi des coefficients qui permettent de modéliser les distorsions dans le plan image ; il existe plusieurs modèles de distorsions, radiales, tangentielles, prismatiques... nous ne décrirons pas ces coefficients ici.

Paramètres extrinsèques

Jusqu'à maintenant, nous avons fait les calculs dans le repère de la caméra, avec les coordonnées (X, Y, Z) du point 3D dans ce repère. Introduisons maintenant les coordonnées (x, y, z) dans le repère monde. Le changement de repère euclidien 3D entre les deux systèmes de coordonnées, est défini par :

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \Re \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{3.1}$$

La matrice $\Re$ est définie par une rotation et une translation 3D, donc par 6 paramètres indépendants. Dans le cas d'un banc stéréo, nous avons deux caméras, donc deux matrices de ce type. Si l'une des caméras, par exemple la gauche, est choisie pour porter le repère monde, alors, il faut uniquement déterminer lors du calibrage, la position de la caméra droite par rapport à la caméra gauche.

Procédure de calibration.

D'un point de vue opérationnel, le calibrage d'une caméra ou d'un banc stéréo, a été pendant longtemps une boîte de Pandore de publications et de méthodes : la méthode la plus souvent référencée est probablement celle de Tsai [149]. Les différentes étapes du calibrage sont citées dans la littérature en vision par ordinateur, avec les méthodes connues de Tsai, Faugeras et Toscani, Song de Ma, Zhang ... Quelques directives peuvent être fournies quant à ce qui constitue la bonne pratique.

Nous ne parlons ici que des méthodes dites fortes (ou exactes) de calibration, méthodes qui exploitent une mire de modèle connu ; la mire est un objet précis, plan ou volumique, sur lequel on a créé des motifs visuels très discriminants (damier, grille de lignes, grille de disques...). Pour estimer les paramètres des modèles, on va donc exploiter (1) les points 3D de la mire, points de coordonnées réelles connues et

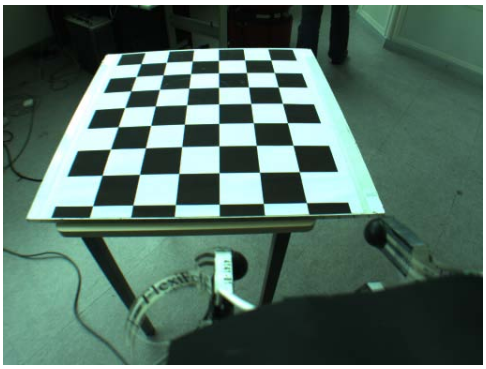


FIGURE 3.5 – Vue gauche d'une mire de calibration

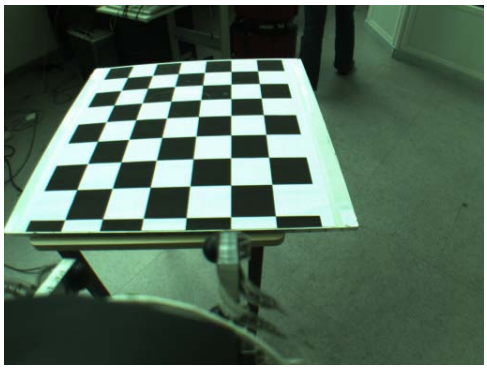


FIGURE 3.6 – Vue droite de la mire.

(2) leurs projections dans des images, dont les coordonnées peuvent être déterminées avec une grande précision (typiquement, le 50ième de pixel, ou mieux).

On acquiert des images sur la mire avec un protocole pour éviter des configurations géométriques dégénérées de l'ensemble des points 3D finalement exploités : coplanarité, orthogonalité, points de fuite etc. Respecter ces restrictions facilitent la calibration sans avoir besoin d'effectuer des mesures auxiliaires. Quelques expériences récentes de calibrage sont décrites dans les travaux de Karras et Mavromati [70], Meng et Hu [90] et de Park et Hong [106].

Pour notre part, nous avons utilisé une approche de calibration avec mires planes : d'abord la procédure créée au LAAS par V.Garric et M.Devy [54], puis la procédure disponible dans la *Camera Calibration Toolbox* de Matlab, créée par JY.Bouguet qui est plus facile à exploiter et qui donne des résultats équivalents. En figure 3.5 et 3.6 nous présentons deux images de la mire utilisée pour faire la calibration de notre banc stéréo : vues obtenues par les caméras gauche et droite en une position du banc. Les paramètres que nous avons obtenus en faisant la calibration de notre banc stéréo Videre, sont montrés dans la table 3.1 ; concernant les extrinsèques, nous retrouvons les propriétés géométriques du banc *Videre*, à savoir les axes optiques quasiment parallèles et une base d'environ 9cm.

3.3.2 Acquisition des images

Après avoir calibré notre système de caméras on peut passer à la phase suivante, l'acquisition d'images. Ces images seront ensuite traitées, par exemple, avec l'extraction des contours, de régions ou de points d'intérêt, le calcul des histogrammes d'intensité... ou tout autre traitement qui produit les différentes caractéristiques liées à l'apparence visuelle d'un objet. Pour la stéréovision, ces caractéristiques seront appariées entre images acquises de différents points de vue, ce qui permettra de les reconstruire par triangulation. Certaines informations 3D seront mémorisées avec le modèle 3D de l'objet, pour une ultérieure utilisation dans le système de reconnaissance, comme nous verrons dans le chapitre suivant.

Nous avons donc exploité le capteur stéréo fourni par la compagnie *Videre*. Ce dispositif a un mode spécifique dans laquelle il rend une image entrelacée ; bien qu'il s'agisse d'une paire stéréo, il ne rend qu'une seule image dans laquelle les deux images droite et gauche sont entrelacées (lignes paires pour la gauche, impaires pour la droite). Notons qu'alors, la résolution verticale des images est divisée par deux. En plus, les images ont une structure type Bayer ; la distribution de l'information colorimétrique est dans un format spécifique qui dépend de la mosaïque mis devant le plan image, format BGGR ou RGGB selon le cas.

Donc à chaque acquisition, nous devons récupérer les images droite et gauche en prenant les lignes paires et impaires de l'image entrelacée, puis appliquer un algorithme de *demosaicing* pour récupérer les informations colorimétriques. Dans nos travaux sur la reconstruction, nous n'avons pas exploité la couleur : aussi l'acquisition rend uniquement deux images N&B.

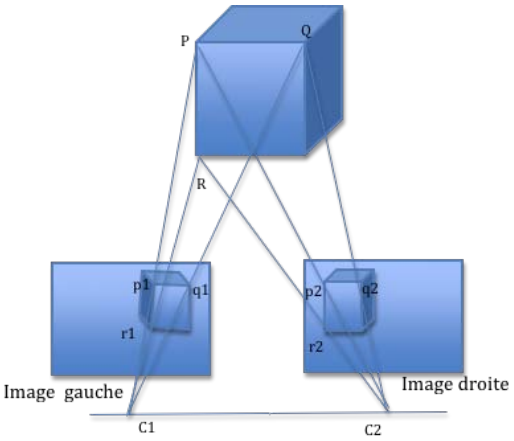


FIGURE 3.7 – Principe de la stéréovision : les positions des points P, Q ou R sont obtenues par triangulation à partir des appariements $(A1, A2)$, $(B1, B2)$ et $(D1, D2)$.

3.3.3 Stéréo corrélation

En vision, le terme **stéréo** est utilisé quand il existe plus d'une vue d'une scène. Utiliser plusieurs vues, acquises par plusieurs caméras calibrées au préalable, permet d'acquérir des caractéristiques tridimensionnelles de la scène analysée.

La géométrie de deux vues est connue comme la Géométrie Epipolaire. Z.Zhang présente une introduction à la géométrie épipolaire et donne un rappel des techniques pour l'estimation de la matrice fondamentale et de son incertitude [170]. La géométrie de deux vues est présentée dans la Figure 3.7 ; le point 3D P (idem pour Q et R) est vu dans les deux images comme des pixels p_1 et p_2. Ces pixels sont obtenus par projection centrale dans l'espace 3D sur les plans image des caméras gauche et droite, par le biais de leurs centres optiques notés C_1 et C_2. A partir d'un pixel seulement, par exemple celui de l'image droite le pixel p_2, on ne peut pas connaître la situation précise de P, puisque dans le processus de projection on a perdu l'information de profondeur. Toutefois, il peut être affirmé que P doit être sur le rayon optique qui naît depuis le centre optique C_2 pour former p_2, c'est-à-dire, P appartient à la droite p_2C_2. Cette situation est montrée dans la Figure 3.8, où plusieurs points P, mais aussi X_1, X_2 et X_3 appartenant à la droite p_2C_2 peuvent être celui qui se projette en p_2 dans l'image droite.

Si à partir du pixel p_2 de l'image droite, on souhaite rechercher le stéréo-correspondant p_1 dans l'image gauche, c'est-à-dire la projection de P à gauche, alors il est nécessaire de projeter dans l'image gauche les possibles points 3D qui peuvent former p_2 à droite ; p_1 est un de ces pixels. A partir du point p_2 on ne peut pas connaître la situation précise de p_1 uniquement par des contraintes géométriques, mais il peut être affirmé que p_1 appartient à la projection de la droite p_2C_2 dans l'image gauche par le biais du centre optique C_1 ; c'est l'intersection du plan défini par les points C_1, C_2 et p_2 avec le plan qui porte l'image gauche. Cette projection montrée en Figure 3.8, notée $D(p_2)$, est appelée ligne épipolaire dans l'image gauche, associée au pixel p_2 de l'image droite.

Par construction géométrique, toutes les droites épipolaires définies dans l'image gauche à partir des pixels de l'image droite, sont concourrantes ; elles forment un faisceau ayant pour origine le point CE_1, centre épipolaire de l'image gauche, défini comme l'intersection entre la droite (C_1C_2) et le plan qui porte l'image gauche. Les images droite et gauche jouent un rôle symétrique, et on peut de même définir les droites épipolaires dans l'image droite des pixels de l'image gauche, droites qui concourrent sur le centre épipolaire de l'image droite.

La contrainte épipolaire [18] est la contrainte exacte la plus forte qui peut être exploitée afin de mettre en correspondance des pixels entre deux images acquises par un banc stéréo. La connaissance des lignes épipolaires ramène le problème de correspondance à une recherche 1D. Cette contrainte est rendue plus facile à exploiter par la procédure de rectification des images acquises par le banc stéréo. N'importe quelle paire d'images peut être transformée "à une géométrie parallèle de caméras" de sorte que les pixels correspondants dans les images gauche et droite rectifiées, se trouvent sur les mêmes lignes dans les deux images, comme cela est présenté en figure 3.7. Dans le banc rectifié, les axes optiques des caméras sont parallèles, orthogonaux à la droite joignant les centres optiques ; les images droite et gauche sont portées

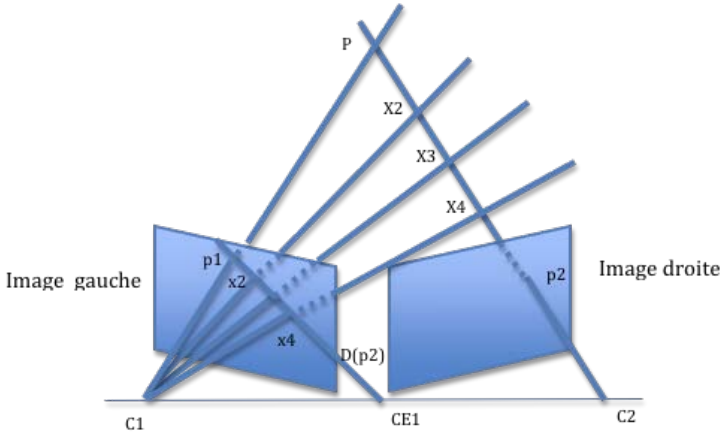


FIGURE 3.8 – Géométrie Epipolaire.

par le même plan, et les lignes de ces images sont portées par des droites parallèles. Les centres épipolaires sont rejetes à l'infini.

On peut appliquer la rectification sans avoir calibré au préalable les caméras en estimant la matrice fondamentale (méthode de Devernay par exemple), mais comme nous voulons reconstruire les pixels appariés en 3D par triangulation, nous appliquons une méthode de rectification fondée sur un calibrage préalable du banc stéréo.

Mentionnons quelques travaux sur la corrélation stéréo. Le modèle stéréo qui est utilisé en vision par ordinateur, est fortement inspiré de la vision humaine : Månsson [100] a proposé un modèle du mécanisme de stéréopsis de l'être humain. Dans ses travaux, au lieu d'établir des appariements potentiels à partir de la détection et la mise en correspondance des caractéristiques, les correspondances se font en comparant les intensités et les contrastes dans des régions delimitées des deux images. Zhou et Wang [171] ont développé une technique similaire de stéréocorrespondance entre pixels, mais en exploitant la couleur.

Dans notre cas, nous utilisons l'algorithme de stéréovision dense développé au LAAS depuis une dizaine d'année [77] : les correspondances se font sur les images rectifiées, en calculant des mesures de ressemblance soit par un score de corrélation ZNCC, soit par la distance de Hamming pour comparer les codes CENSUS associés aux pixels des deux images. Une validation droite-gauche permet de filtrer les mauvais appariements ; un filtrage de l'image de disparité par un algorithme de type *Blob Coloring* permet d'effacer des petits groupes de pixels qui présentent une disparité trop différente de ceux qui les entourent.

3.4 Traitement de l'information 3D : méthodes existantes.

Une fois acquises les données 3D par stéréovision, une fois choisis les points de vue depuis lesquels ces données sont acquises, il reste à recaler ces données pour les exprimer dans un même référentiel, puis à créer une représentation, ici un maillage triangulaire. Nous décrivons ci-dessous les méthodes qui pré-existaient ou qui avaient été évaluées dans le pôle Robotique du LAAS à notre arrivée.

3.4.1 Décimation

Le but d'un algorithme de décimation est de réduire le nombre total de triangles dans un maillage 3D ; dans notre cas, cette fonction est appliquée sur le maillage construit depuis une image 3D ; un tel maillage est construit de manière triviale puisqu'on dispose d'une relation de voisinage entre les points 3D d'une telle image ; il suffit de connecter les points voisins. Par conséquent, décimer ce maillage permet de diminuer aussi la quantité de points dans une image de points 3D, ce qui réduira la complexité des étapes suivantes.

Une méthode de décimation agit donc sur un modèle 3D déjà triangulé : il se doit de préserver la relation topologique originale entre tous les éléments qui constituent le maillage résultant. Schroeder *et al* [125] ont proposé un algorithme itératif pour la décimation de maillages 3D, fondé sur la suppression

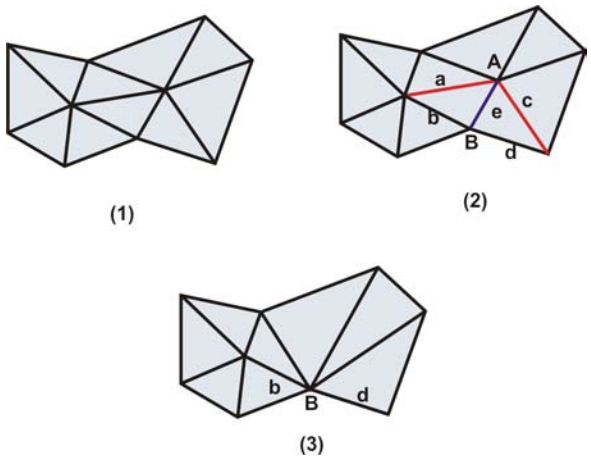


FIGURE 3.9 – Exemple de décimation d'un maillage

itérative de sommets. Les étapes sont les suivantes, le critère de fin pouvant être par exemple la suppression de N% des sommets :

1. Caractériser la géométrie et la topologie locales de chaque sommet.
2. Sélectionner le sommet à supprimer (sommet dans la zone la plus "plane", sommets des triangles les plus dégénérés...)
3. Trianguler le trou résultant.
4. Retour à 2 jusqu'à satisfaire le critère de fin.

Wu *et al.* [158] utilisent une méthode incrémentale pour la décimation massive de maillages basée sur la suppression des arêtes et l'erreur métrique quadratique. Cette dernière méthode a été développée par Garland *et al.* dans [52] [53] [59] : elle consiste à utiliser comme métrique, l'erreur quadratique entre paires de sommets, tout cela en préservant la surface du maillage : supprimer une arête consiste à fusionner ses extrêmités. Melax[89] de son côté, a aussi développé une technique de suppression des triangles ; le coût de la suppression d'une arête dépend de la longueur de cette arête multipliée par un terme de courbure. Ce terme de courbure est déterminé par l'application du produit scalaire entre les normales des triangles adjacents à une arête en cours d'analyse. Pour finir, mentionnons Talton [140] qui a fait une comparaison entre différentes techniques de décimation de maillages.

Donnons quelques exemples de décimation appliquée à une image de points 3D, issue d'une acquisition 3D. La figure 3.9 présente un exemple de décimation en 2D. En partant du maillage en (1), supprimons l'arête (e), ce qui revient à ramener le sommet A vers B dans le maillage (2). De ce fait les arêtes en rouge (a) et (c) se fusionnent avec (b) et (d) ; le maillage obtenu en (3) a deux triangles de moins que le maillage initial en (1).

Donnons un autre exemple, mais cette fois ci en utilisant un modèle 3D ; les figures 3.11, 3.12 et 3.13 sont des maillages décimés obtenus à partir du maillage initial présenté en figure 3.10. Les taux de réduction sont respectivement égaux à 25%, 50% et 75% du nombre initial de sommets. Ce résultat a été obtenu par l'outil *Qslim* développé par Garland [52] [53] [59]. On observe qu'au fur et à mesure de la suppression des triangles, un maillage commence à perdre sa topologie, c'est-à-dire, il finit par perdre sa forme originale comme on peut l'observer en figure 3.13. Il existe donc une limite pour décimer un maillage tout en gardant sa topologie d'origine.

Ces méthodes de décimation, s'appliquent donc à un maillage déjà construit. Dans notre cas, elles peuvent intervenir soit sur les images 3D qui peuvent être maillées très facilement, soit sur le maillage final construit à partir du nuage de points 3D résultant de la fusion de toutes ces images 3D.

3.4.2 Recalage

Comment fusionner en une seul nuage de points 3D, toutes les images 3D acquises depuis les différents points de vue ? Ce processus s'appelle recalage, ou *registration* en anglais. Ce n'est pas une tâche facile et beaucoup de méthodes ont été développées à ce sujet. L'algorithme le plus utilisé est celui appelé ICP

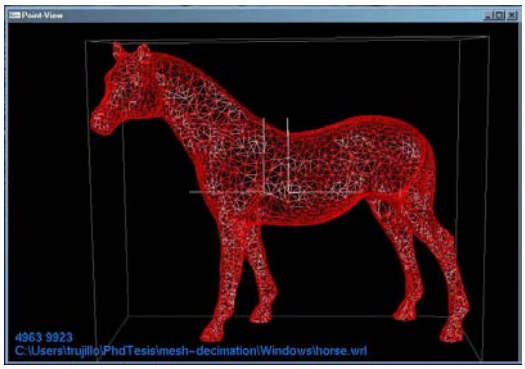


FIGURE 3.10 – Modèle du cheval avec 100% des sommets d'origine.

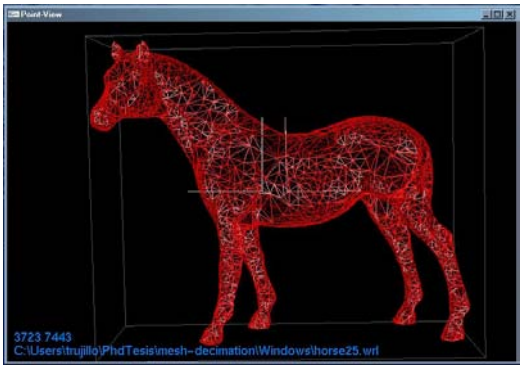


FIGURE 3.11 – Modèle du cheval décimé à 25%.

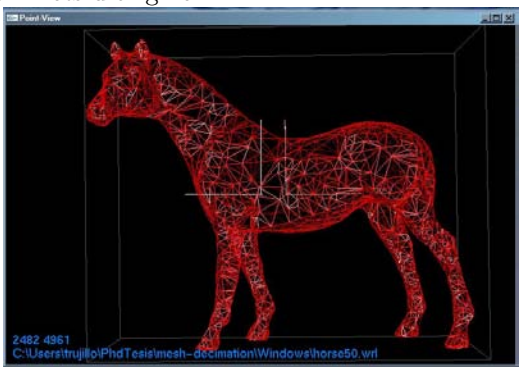


FIGURE 3.12 – Modèle du cheval décimé à 50%.

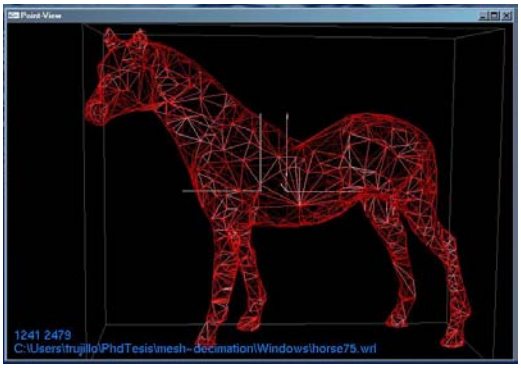


FIGURE 3.13 – Modèle du cheval décimé à 75%.

pour *Iterative Closest Point*. Cet algorithme fut développé par Chen et Medioni [32] : Besl et Mckay [24] ont proposé une variante et ce sont eux qui lui ont donné le nom d'ICP.

L'algorithme ICP fonctionne de la façon suivante. Il prend en entrée deux ensembles de points M et P, comme ceux de la figure 3.14 donnés sous la forme de deux lignes. Ensuite il commence à calculer la distance Euclidienne afin de trouver pour chaque point de l'ensemble M le point le plus proche dans l'ensemble P. La figure 3.15 montre cette relation entre les deux lignes. La distance Euclidienne entre deux points r_1 et r_2 se calcule comme :

$$d(r_1, r_2) = \|r_1 - r_2\| = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \tag{3.2}$$

La distance entre un point m de M et l'ensemble P peut s'exprimer comme la distance Euclidienne minimale entre m et les points de P :

$$d(r_1, P) = \min_{i \in 1..n} d(m, p_i) \tag{3.3}$$

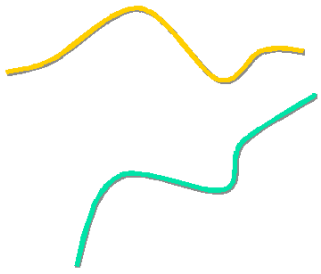


FIGURE 3.14 – Deux lignes à recaler

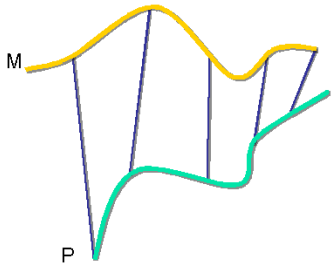


FIGURE 3.15 – Appariements entre deux lignes.

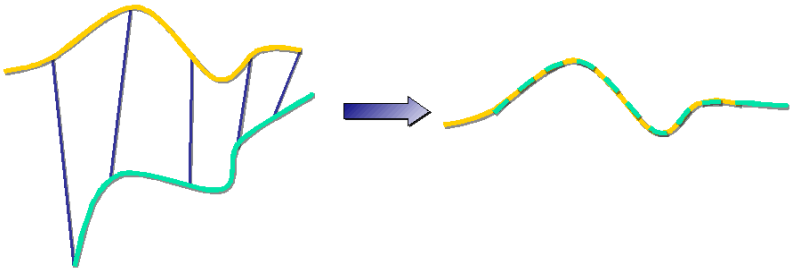


FIGURE 3.16 – Recalage entre deux lignes.

Dans ce cas il y aura toujours une correspondance entre un point de M et un autre point de P. Une fois trouvées ces correspondances, et en supprimant les doublons (si deux points de M sont appariés au même point de P, on ne garde que celui qui a la distance la plus faible), on peut estimer une transformation $(Rot, Trans)$ qui permet de minimiser l'erreur entre les points appariés (m_i, p_i), en minimisant le critère :

$$f(Rot, Trans) = \frac{1}{N_P} \sum_{i=1}^{N_P} \|m_i - Rot(p_i) - Trans\|^2 \tag{3.4}$$

où N_P représente le nombre de points appariés.

La figure 3.16 montre le résultat du recalage entre les deux lignes. Des zones sont partagées entre les deux lignes initiales : ce sont les zones où l'algorithme ICP a trouvé les points appariés.

Pour qu'à la fin, cet algorithme puisse converger sur un minimum quasiment nul pour la distance entre les points appariés, il est nécessaire de lui donner en paramètres d'entrée, une estimée initiale des rotation et translation entre les deux nuages de points à recaler. Selon le protocole choisi pour l'acquisition, cette transformation initiale provient de mesures de déplacement relatif entre l'objet et le capteur ; plus précises sont ses mesures, plus vite va converger l'algorithme ICP. Citons quelques exempels :

- l'objet est sur une platine, le capteur est fixe : la transformation initiale est calculée à partir du codeur de la platine, et de la transformation rigide platine-capteur.
- l'objet est fixe, le capteur est déplacé sur un robot : la transformation initiale est calculée à partir des mesures de déplacement du robot (codeurs sur les articulations d'un manipulateur, codeurs sur les roues d'un robot mobile), et de la transformation rigide capteur-robot.
- l'objet est fixe, le capteur est déplacé à la main. L'estimée peut venir de capteurs inertiels montés sur le capteur (par exemple sur le capteur *HandyScan* de la société *Creaform*) et/ou d'un traitement sur les données sensorielles, visant à mesurer le mouvement du capteur (SLAM visuel par exemple).

L'algorithme ICP présente des caractéristiques qui font de lui un outil indispensable pour la fusion de nuages de points 3D. C'est un algorithme efficace mais qui ne converge pas toujours. Ces problèmes de convergence surviennent si la transformation initiale entre les deux nuages est trop imprécise, en particulier si la rotation est trop mal estimée. D'autres problèmes surviennent si l'objet observé présente des symétries, ou a une forme trop régulière, sans discontinuité d'orientation des normales : impossible de recaler deux nuages de points extraits sur une seul plan, ou une seule sphère ...

C'est à cause de ces limitations, que de très nombreuses variantes d'ICP ont été proposées. Nous allons présenter une classification de ces différentes variantes d'ICP. Cette classification n'essaie pas

d'être exhaustive mais présente à notre façon de voir, les principaux domaines dans lesquels on cherche à améliorer ou maximiser l'algorithme d'ICP. Le lecteur pourra trouver un état de l'art exhaustif sur ICP, dans le travail de Rusimkiewick et Levoy[117] ; ils ont fait une énumération et une évaluation des méthodes ICP existantes à l'époque. Une comparaison est proposée sur la base du temps d'exécution et de l'erreur finale d'alignement.

Améliorations d'ICP : meilleure estimation de la transformée initiale

Quelques variantes cherchent à obtenir une transformation initiale adéquate entre les nuages à recaler. Citons les travaux sur l'exploitation de l'analyse en composantes principales (PCA) : Salamanca *et al.* [121] utilisent les *eigenvalues* issus de la PCA pour vérifier quelles sont les zones dans lesquelles il existe une plus grande probabilité d'apparier deux points de vue différents d'un même objet ; ils utilisent les *eigenvectors* comme un moyen d'obtenir la transformation initiale entre ces points de vue. Sameh *et al.* [166] [164] [165] exploitent une signature de surface invariante au point de vue afin de représenter l'information de courbure de la surface ; ils engendrent ainsi des images de ressemblance entre les nuages de points 3D.

Certains auteurs font usage, aussi, de la programmation floue, fondée sur la logique floue (*fuzzy logic* en anglais) afin de mieux estimer la transformation initiale entre les deux points de vue. Mentionnons le développement de Tarel *et al.* [142] qui utilise la logique diffuse pour trouver les paramètres nécessaires au recalage de deux nuages de points 3D. Citons quelques travaux réalisés en utilisant des algorithmes génétiques comme celui de Lomosonov *et al.* [82] qui utilisent les algorithmes génétiques pour réaliser un pre-recalage des données ; la matrice de transformation ainsi obtenue est exploitée comme transformation initiale pour la méthode LTS aussi proposée par Chetverikov *et al.* [33] (voir ci après). Chow *et al.* [34] introduisent un opérateur de mutation dynamique qui permet d'éviter les convergences prématurées et de minimiser l'erreur de recalage. Robertson *et al.* [116] utilisent quatre opérateurs différents de mutation et deux autres opérateurs pour le croisement ; à chaque étape, un de ces opérateurs est sélectionné selon une probabilité d'activation propre à chacun d'eux, et adaptée au fur et à mesure du recalage.

Améliorations d'ICP : méthodes robustes d'optimisation.

Si les points 3D sont issus de la vision, nous pouvons citer plusieurs approches proposées comme alternatives à ce type d'algorithmes, comme l'algorithme de *Sparse Bundle Adjustment* (SBA) [146] [85] qui utilise la méthode d'optimisation non linéaire de Levenberg-Marquardt, pour affiner à la fois la position de points 3D perçus depuis plusieurs positions du capteur, et les positions de ces points de vue. Cet algorithme ne fusionne pas vraiment des nuages de point, mais résoud de manière unique, à la fois le problème de la reconstruction de points 3D et du recalage des points de vue. Signalons la complexité importante de SBA lorsqu'on l'applique à un grand nombre de points ; signalons aussi la sensibilité au choix des estimées initiales pour les positions des points 3D et du capteur ; plusieurs auteurs proposent aujourd'hui des méthodes d'optimisation globale (donc sans estimée initiale) pour éviter le problème des minimas locaux.

Mentionnons aussi les travaux qui cherchent à améliorer la méthode d'optimisation exploitée pour trouver la transformation entre les deux nuages à recaler. Chetverikov *et al.* [33] utilisent une variante de la minimisation aux Moindres carrés (*Least Squares Method*) appelée *Least Trimmed Squares* (LTS) ; le but est de ne conserver que les meilleurs appariements entre points des deux nuages, pour estimer la transformation. Estépar *et al.* [45] présentent une généralisation de la formulation des moindres carrés, qu'ils appelent GLTS pour *Generalized Total Least Squares* ; cette variante de LTS est plus robuste puisqu'elle élimine aussi les appariements trop incertains. Citons encore les développements de Mian *et al.* [91], qui exploitent les tenseurs vectoriels.

Améliorations d'ICP : exploitation de l'apparence des objets.

En fait, utiliser seulement l'information qui est obtenue par un capteur 3D (stéréo, télémétrie laser...) n'est pas suffisante pour obtenir un bon recalage entre deux nuages de points 3D. Exploiter des caractéristiques inhérentes à l'objet que l'on souhaite modéliser, augmente les chances de bien recaler les images acquises sur cet objet depuis plusieurs points de vue. Mais que sont ces caractéristiques intrinsèques aux objets ? En exploitant toujours des données 3D, A.Restrepo [133] utilise l'extraction ded points de discontinuités (profondeur et orientation) afin de maximiser le processus d'appariement de points ; d'un côté

cela nécessite d'exécuter une procédure en plus, coûteuse en temps de calcul ; mais à la fin, le recalage exploite moins de points 3D et surtout, n'exploite pas les points non discriminants (ceux qui sont sur les surfaces uniformes).

Les caractéristiques de couleur et de texture, sont les plus utilisées puisque tous les objets possèdent une texture et une couleur ; on peut donc affirmer que ces caractéristiques seront toujours exploitables par le processus d'appariement de points 2D ou 3D. Plusieurs auteurs ont utilisé la texture ; citons les travaux de Weik [155] ou la méthode développée par Johnson [66]. Ils proposent l'usage de la texture comme caractéristique inhérente à tous les objets, et l'exploitent pour le recalage de deux nuages de points.

La couleur est une caractéristique qui est beaucoup exploitée en vision par ordinateur, mais c'est aussi une caractéristique très faible, à cause de sa sensibilité aux conditions d'illumination (pas d'invariance des couleurs) et aussi de son pouvoir discriminant très variable en fonction des objets traités : un système de reconnaissance basé sur la couleur fera beaucoup d'erreurs si deux objets ont des spectres chromatiques trop similaires. Concernant le recalage de nuages 3D en utilisant comme caractéristique discriminante la couleur pour trouver des appariements de points entre les nuages, les travaux de Kim *et al.* [73] et Pulli [113], prouvent qu'on peut utiliser la couleur pour obtenir un recalage de données 3D de façon efficace. Citons aussi Druon *et al.* [44], qui utilisent la couleur après une transformation de l'espace RGB vers l'espace HSV : ils apparient les points dans cet espace par classification, en considérant un nombre limité de classes.

Dans notre contribution, nous allons nous inspirer de ces travaux sur la couleur, sauf que dans notre cas, elle n'est pas disponible : nous allons donc proposer d'exploiter une Pseudo-Couleur.

3.4.3 Construction d'un maillage

Le but de la construction d'un maillage est d'obtenir un modèle surfacique 3D d'un objet réel à partir d'un nuage de points, nuage obtenu par recalage d'images 3D acquises depuis divers points de vue autour de cet objet.

Comme dans le cas du recalage, il existe plusieurs algorithmes qui ont été développés pour cette tâche. Parmi eux, mentionnons :

- Marching cubes.
- Ball Pivoting.
- Surfaces déformables.
- Approches fondées sur la Triangulation de Delaunay.

Ici nous proposons uniquement de construire le maillage à partir du nuage recalé. Certains auteurs ont proposé des méthodes incrémentales afin de mettre à jour successivement un tel maillage dès qu'une nouvelle image est acquise. Mais ces méthodes sont très complexes et peu robustes.

Dans la suite, nous allons décrire de façon brève chacune des méthodes citées ci-dessus. Tout cela pour donner un cadre de référence pour notre travail.

Marching Cubes

Les « marching cubes » désignent un algorithme d'infographie publié en 1987 par Lorensen et Cline. Cet algorithme permet de générer un maillage triangulaire à partir d'une discrétisation du volume de travail 3D englobant le nuage de points à représenter par un maillage. Ce volume est discrétisé en un tableau à 3 dimensions de cubes élémentaires appelés voxels : la taille des voxels est un paramètre essentiel de l'algorithme ; plus elle est grande, plus grossière sera la représentation ; plus elle est petite, plus grande sera la probabilité d'avoir des trous dans le maillage.

Cet algorithme est le pendant 3D de l'algorithme des "marching squares". Il agit de la manière suivante. L'étape la plus longue consiste à associer un champ scalaire à la carte de voxels ; chaque sommet des voxels sera étiqueté '1' si ce point est intérieur à l'objet, '0' s'il est extérieur. Cet étiquetage nécessite une analyse locale du nuage , des calculs de normales ... On ne s'intéresse ensuite qu'aux côtés de voxels qui ont une extrêmité à '0' et une extrêmité à '1' : en effet ces arêtes traversent la surface de l'objet, et vont donc porter un sommet du maillage à construire.

Ensuite on boucle sur le tableau des voxels et de voxel en voxel on construit le maillage ; la méthode garantit qu'au final, ce maillage est cohérent, i.e. les triangles créés entre deux voxels voisins partagent

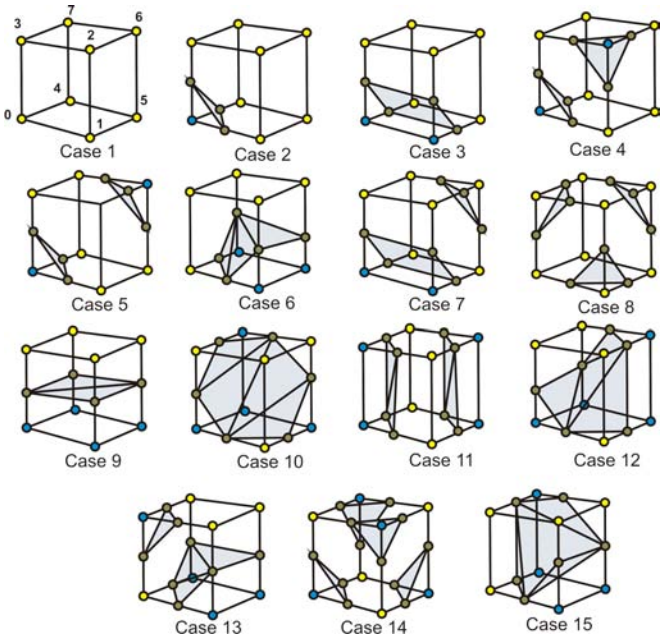


FIGURE 3.17 – Marching cubes : les 15 configurations pour créer les mailles.

les sommets portés par les côtés communs et les arêtes portées sur la face commune. Chaque voxel est analysé ; on lui associe un index compris entre 0 et 256, simplement en concaténant les '0' et '1' du champ scalaire correspondants à ses sommets ($2^8 = 256$). Cet index pointe sur un tableau précalculé des 256 configurations possibles pour créer un maillage dans un voxel. Ces 256 configurations de triangles associés à un voxel sont précalculées par réflexion et symétrie à partir de 15 cas possibles seulement, présentés en figure 3.17 ; les extrêmités sont codées en bleu si elles sont externes à l'objet, en jaune sinon ; les triangles sont grisés. Chaque sommet de triangle est placé à sa position finale le long du côté du voxel, par interpolation, en fonction de la position des points du nuage dans le voisinage de ce côté.

A partir de cet algorithme ont été développées d'autres variantes comme celle dite des *dual marching cubes* , realisée par Nielson [104] ou celle developpée par Schaefer et Warren [123] dans laquelle les auteurs proposent l'utilisation d'une grille topologique duale qui va servir pour faire un maillage plus dense, donc plus fin et en conséquence avec moins d'erreurs dans le maillage. La variante Marching tetrahedra [145] utilise une discrétisation en tétraèdres plutôt que celle en cubes ; le principal avantage est la réduction de l'ambiguïté de la version standard de Marching Cubes au moment de créer la surface.

Citons aussi l'implémentation de Marching triangles, developpée par Hilton *et al.* [62] [61] qui utilisent une modification locale des contraintes 3D de la triangulation de Delaunay. Akkouche et Galin [8] font des améliorations à la méthode des Marching triangles ; les triangles qui vont former le maillage ont des tailles différentes, à la différence des triangles équilatéraux utilisés par Hilton dans son implémentation.

Ball Pivoting

L'algorithme *Ball-Pivoting* a été développé pas Bernardini, Taubin *et al* [23] dans les laboratoires IBM. C'est un algorithme glouton (*greedy*) qui crée un maillage triangulaire à partir d'un nuage de points 3D. Il procède par propagation : les triangles d'un maillage sont construits progressivement de proche en proche.

Le principe du BPA est très simple. Le maillage doit être tel que, pour tout triangle, une boule de rayon ρ qui passe par ses trois sommets, ne contient aucun autre point ; le rayon ρ est défini par l'utilisateur en fonction de la résolution souhaitée pour le maillage. Notons que les *Alpha Shapes* sont définis à partir de cette même contrainte. Comment construire un tel maillage ?

1. Initialement, trois points A, B et C du nuage sont choisis pour former un triangle souche ;
2. une boule de rayon ρ est positionnée sur ce triangle ABC ;

3. puis, elle pivote autour de l'arête AB, i.e. elle tourne autour de l'arête, en gardant le contact avec ses extrêmités, jusqu'à ce qu'elle touche un autre point C'. En ce cas, on crée un triangle ABC' et on le rajoute à la liste des triangles à traiter. Si aucun point n'est atteint par la boule quand elle pivote autour de AB, alors cette arête sera une frontière de la surface.
4. faire de même avec les arêtes BC et AC, pour éventuellement créer deux nouveaux triangles A'BC et AB'C ; les rajouter à la liste des triangles à traiter.
5. Tant que cette liste n'est pas vide, revenir en (2) pour traiter le triangle en tête de liste.
6. S'il reste des points non inclus dans le maillage, revenir en (1) pour sélectionner un nouveau triangle souche parmi les points encore libres.

Bernardini *et al* ont appliqué la méthode BPA à des nuages contenant plusieurs millions de points. La quantité réduite de mémoire exigée par cette méthode, son efficacité en temps d'exécution, et la qualité des résultats obtenus rivalisent favorablement avec des techniques existantes. Par contre, le maillage créé peut contenir des trous si le rayon ρ est choisi trop petit, et surtout, si les points sont bruités, il faut rajouter des étapes intermédiaires, coûteuses en temps de calcul.

Surfaces Déformables

Une surface déformable est un objet composé d'une surface , et d'une loi d'évolution permettant la déformation de cette surface afin qu'elle représente au mieux un nuage de points 3D. Un aspect principal des modèles déformable concerne leur nature géométrique. Fondamentalement, deux critères peuvent être associés à la géométrie de ces modèles :

- **Description de la Forme**. Une surface déformable peut être définie par une fonction explicite afin de représenter des formes simples (par exemple juste un ellipsoïde qui s'adapte au nuage à représenter) ou des formes de topologie restreinte (par exemple descripteurs sphériques de Fourier). Par contre, si on veut représenter des formes quelconques indépendamment de leur topologie, la surface est un maillage triangulaire rendu actif, par exemple avec une masse positionnée en chaque sommet, et un ressort à la place de chaque arête.
- **Description des Déformations**. La complexité de la déformation d'une surface déformable est complétement liée à sa définition. Par exemple, pour coller au mieux sur un nuage de points une surface de type ellipsoïde, on a seulement 9 degrés de liberté (translation, rotation et trois rayons) tandis que la déformation d'une surface décrite par un maillage masse-ressort a autant de degrés de liberté que d'arêtes.

Ces définitions s'appliquent en particulier, aux surfaces déformables définies par des superquadriques. Ces formes geométriques représentées de manière analytique, sont fermées (on parle souvent de représentations volumiques) ; elles s'apparentent aux ellipsoïdes ou aux autres fonctions quadriques. L'utilisation des superquadriques comme une surface déformable, se justifie car ces formes ont des capacités de déformation bien plus importantes du fait du nombre de paramètres dans leurs descriptions paramétriques analytiques ; par exemple, on a 4 degrés de liberté pour déformer une sphère (la position et le rayon), alors qu'on dispose de 7 à 11 degrés de liberté pour déformer une sphéroïde. Notons néammoins qu'on ne peut pas avec une superquadrique, coller à toutes les formes : la flexibilité d'un maillage déformable est beaucoup plus importante.

Citons quelques travaux sur les superquadriques : Terzopoulos et Metaxas [144] ont utilisé les premiers une méthode pour recaler les points 3D d'un nuage à ceux d'une superquadrique en appliquant des déformations sur les paramètres de celle-ci par des techniques d'optimisation non linéaire. Bardinet *et al.* [19] ont aussi déformé une superquadrique pour la faire coincider avec un nuage de points 3D mais en appliquant des déformations aléatoires et en les validant seulement a posteriori. Saito *et al.* [119] ont utilisé les algorithmes génétiques pour estimer la déformation que doit subir une superquadrique afin de coller au mieux à un nuage de points 3D. Enfin, citons les travaux originaux de Lui *et al.* [81] qui ont développé une approche de reconstruction 3D en utilisant des superquadriques, et un ensemble d'images 2D acquises sur l'objet à reconstruire ; ils proposent un critère calculé dans une image 2D, en faisant coincider la silhouette de la superquadrique en cours de déformation, avec les contours extraits des images 2D.

Pour modéliser des formes complexes, la surface déformable est constituée par un maillage : initialement, ce maillage est souvent une discrétisation d'une sphère à l'aide d'un icosaèdre, comme cela est montré en figure 3.18 ; la résolution de ce maillage initial est choisie en fonction des spécifications de

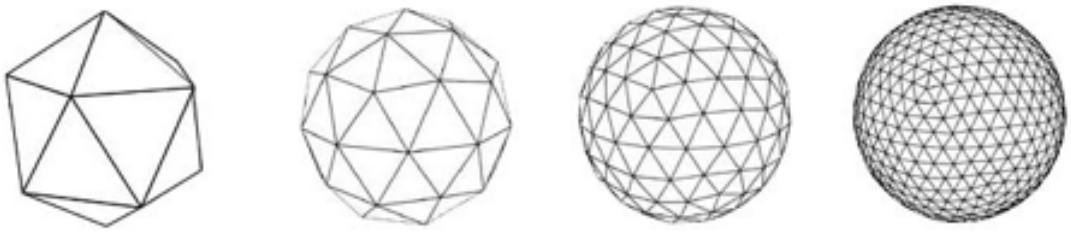


FIGURE 3.18 – Surface déformable initiale : une sphère discrétisée.

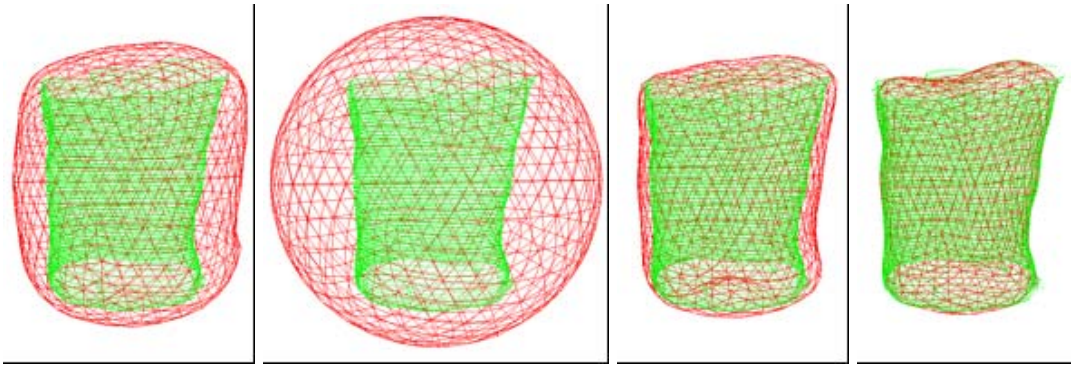


FIGURE 3.19 – Exemple de convergence d'une surface déformable sur un nuage de points 3D, acquise sur le torse d'un homme [76].

l'application. Cette surface initiale est placée autour du nuage de points à partir duquel on veut crer un maillage : un exemple sur la modélisation corporelle à partir de données acquises par profilométrie [76], est présenté en figure 3.19. La méthode est itérative ; à chaque étape, on calcule des forces externes et internes qui s'appliquent à chaque sommet du maillage, forces externes qui tirent la surface déformable vers la forme à modéliser, forces internes pour que cette surface reste continue et pour limiter les fortes discontinuités de courbure.

Il est connu que la modélisation par surface déformable, donne de bons résultats, mais est chère en temps de calcul. De même, il faut rajouter des étapes lors de la construction d'un tel maillage, afin de prendre en compte les propriétés topologiques de la forme. Par exemple le maillage doit être coupé en deux, si on peut détecter que la forme a deux composantes connexes. Par contre, cette méthode s'impose dès lors qu'une forme non rigide doit être modélisée depuis une séquence d'images : la surface déformable est adaptée sur chaque image selon les évolutions de la forme.

Approches issues de la triangulation de Delaunay

Cette méthode de triangulation porte le nom du mathématicien russe Boris Delone (1890 - 1980), francisé en Delaunay. La triangulation de Delaunay d'un ensemble de N points 2D est l'unique triangulation telle qu'un cercle passant par les trois sommets d'un triangle ne contienne aucun autre point. Cette notion peut être généralisée à n'importe quelle dimension : en particulier, en 3D, la triangulation produit des tétraèdres (au lieu de triangles), tels que la sphère (au lieu du cercle) qui passe par les 4 sommets de chaque tétraèdre, ne contient aucun autre point.

La triangulation de Delaunay est le dual du diagramme de Voronoï. On peut donc définir la triangulation de Delaunay à partir de la construction (préalable) du diagramme de Voronoï des N mêmes points : cette triangulation est formée par l'ensemble des segments qui relient 2 des N points à condition qu'ils soient les centres de deux polygones adjacents du diagramme de Voronoï.

Fang et Piegl [47] [48] ont développé une variante qui améliore la façon de chercher les points qui vont former les triangles du maillage. Dans [47], ils traitent uniquement la triangulation d'un ensemble de points 2D en utilisant cette approche tandis qu'en [48], ils l'utilisent pour résoudre la triangulation de nuages de points 3D.

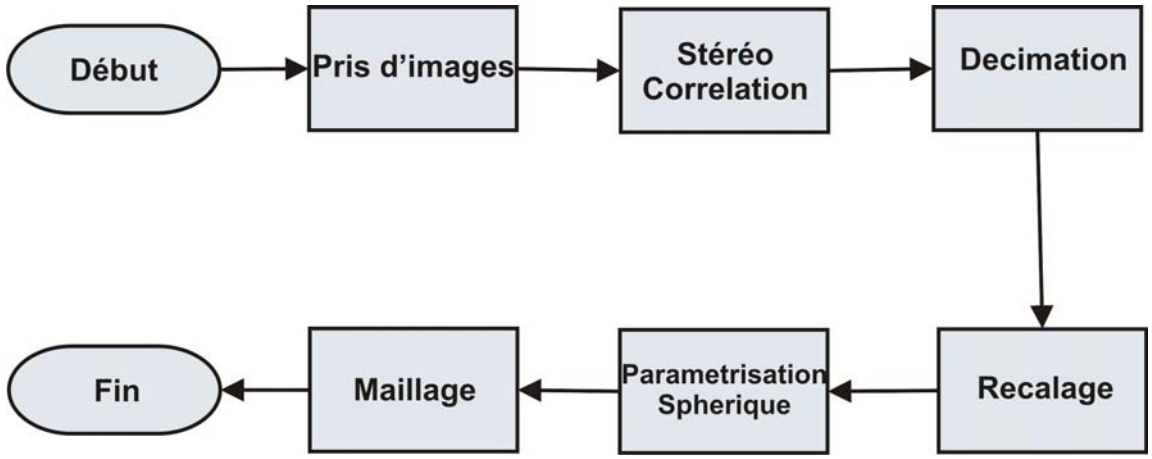


FIGURE 3.20 – Les différentes étapes de la reconstruction 3D.

Notre implémentation peut se situer parmi ces approches inspirées par Delaunay puisqu'elle va se servir de l'algorithme de *convex hull* pour obtenir un maillage complet d'un nuage de points 3D. Dans la section suivante nous allons présenter et décrire les différentes étapes de la méthode de triangulation réalisée par notre algorithme.

3.5 Traitement de l'information 3D : nos contributions.

Nous avons décrit différentes méthodes proposées dans la littérature pour le traitement de données 3D, à savoir le recalage d'images 3D dans un seul nuage de points 3D, puis la construction d'un maillage à partir de ce nuage. Plusieurs de ces méthodes avaient été implémentées dans notre équipe ; nos propres évaluations avec des données acquises par stéréovision, n'ont pas donné des résultats satisfaisants. C'est la raison pour laquelle nous avons proposé des contributions :

- pour le recalage, en exploitant une autre distance que la distance euclidienne afin de trouver les appariements entre points des ensembles à recaler.
- pour la construction de maillage, en mettant en oeuvre une approche simplifiée, la paramétrisation sphérique.

Le diagramme en figure 3.20 résume le processus complet que nous avons réalisé pour modéliser un objet 3D

3.5.1 Recalage des données 3D : ICP PseudoCouleur

Présentation de la méthode.

Si un système d'acquisition ne fournit pas l'information de couleur avec chaque point acquis, on ne peut pas exploiter les méthodes de recalage tirant parti de cette information. Or nous avons vu que l'exploitation de la couleur par ICP pour le recalage de deux nuages de points 3D non structurés, présente des avantages tels que rapidité et robustesse. Nous proposons de trouver une façon de créer une information de Pseudo-Couleur en chaque point 3D de nos nuages à recaler ; il s'agit d'associer à chaque point une information de couleur RGB en fonction de sa position XYZ dans le nuage. Dans les paragraphes suivants nous allons décrire comment affecter une couleur en chaque point, et comment l'exploiter par ICP.

Soit deux nuages à recaler M et P. Nous assignons à chaque point de ces nuages, une pseudo-couleur (R, G, B) definie par :

$$R = \frac{X-\hat{X}}{Max_X}; G = \frac{Y-\hat{Y}}{Max_Y}; B = \frac{Z-\hat{Z}}{Max_Z}$$

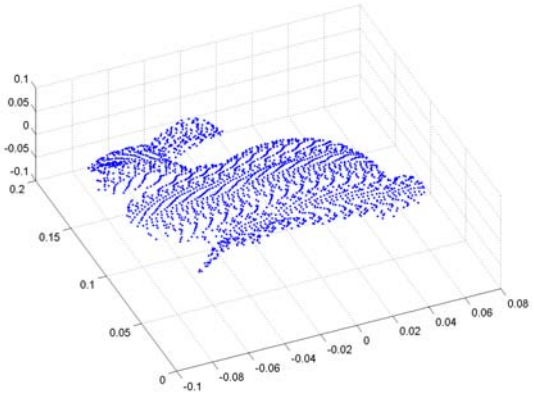


FIGURE 3.21 – Une vue du Bunny.

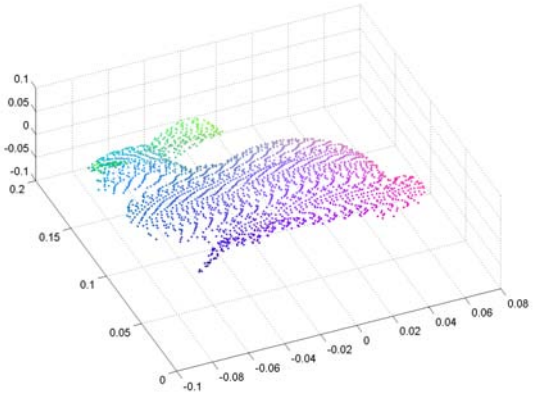


FIGURE 3.22 – Une vue du Bunny en pseudo-couleur.

où $(\hat{X}, \hat{Y}, \hat{Z})$ est barycentre du nuage global $M + P$, et (Max_X, Max_Y, Max_Z) sont les valeurs maximales et minimales des coordonnées sur les trois axes.

A l'itération k, le nuage P est mis à jour avec la dernière transformation calculée T_{k-1}. L'appariement entre deux points de M et P est validé si et seulement si la distance euclidienne entre ces points est inférieure à un seuil ϵ_d et si la différence entre leurs pseudo-couleurs est inférieure à un seuil ϵ_c. Nous montrons ci-dessous sous la forme d'un pseudo-code, notre implémentation de l'algorithme ICP Pseudo-Couleur

Algorithme de ICP Pseudo-couleur.

```
Function Icp PseudoCouleur
Begin
 /* Lit données */
   M=read_data (nuage1);
   P=read_data (nuage2);

  /* Procédure de recalage itératif entre deux nuages de points M et P */
   Recalage:

  /* Assignation de la couleur aux points 3D */
        Calcule les moyennes, minimums et maximums des X,Y,Z pour M et P
   Pour chaque point  dans les nuages M et P,
```

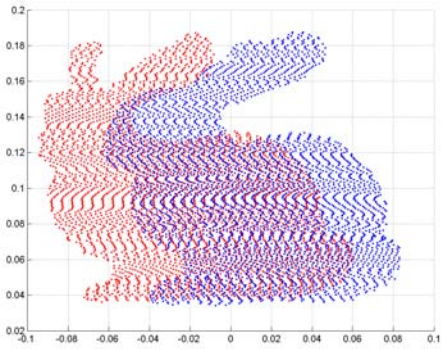


FIGURE 3.23 – Position initiale des deux vues de Bunny : en rouge la vue de référence ; en bleu, une vue avec une rotation de 45deg par rapport à la référence.

```
            calculer l'attribut Couleur

  /* Recalage par ICP */
        Trouver pour tout point i de M, le point j
            le plus proche dans P, en minimisant d(M[i], P[j])
   Si (d < epsilon-d ) et (Couleur(M[i])-Couleur(P[j] < epsilon-c))
   Alors
            garde_matching (M[i], P[j]);

  /* Calculer la matrice de transformation de P vers M
        Estimer par SVD, la matrice T qui recale les points appariés.
   Si (Resultat_Estimation (T) < erreur )
   Alors
            Appliquer T sur les points de P
            Fusionner le nuage P dans le nuage M

   sinon
            Appliquer T sur les points de P
            Aller en Recalage
End.
```

Comme l'a montré A.Restrepo, une adaptation en ligne du seuil $epsilon_d$ permet d'améliorer la stratégie de recalage, de même que d'exploiter une méthode plus robuste telle que RANSAC pour estimer la transformation T_k entre M et P à l'itération k.

Notons que nous avons évalué la méthode *Least Trimmed Square* pour cette estimation, sans constater une amélioration flagrante avec un simple SVD ...

Résultats sur des données virtuelles.

Nous allons évaluer et comparer notre variante de l'algorithme ICP avec l'algorithme ICP classique, d'abord avec des données issus du Web. Les essais ont été effectués sur un PC AMD Athlon à 1GH et l'algorithme a été mis en oeuvre en Matlab 6.

Nous avons fait nos évaluations sur différents modèles 3D récupérés sur le Web à l'université de Stanford [1]. Nous montrons ici des résultats obtenus avec l'objet *Bunny*. En ce cas, les points 3D ont été acquis avec un scanner Cyberware 3030 MS, sur l'objet posé sur une platine ; les différentes vues sont obtenues en faisant tourner l'objet devant le capteur. Après reconstruction, le modèle complet, obtenu à partir de 10 vues différentes, possède 35947 sommets et 69451 triangles. Dans les résultats que nous montrons, nous utilisons seulement deux des 10 vues de *Bunny* disponibles sur le site.

La figure 3.23 montre la position initiale pour deux vues différentes de *Bunny* ; la vue en rouge sert de référence ; la vue en bleu a été acquise après une rotation de 45° de la platine. À partir de cette

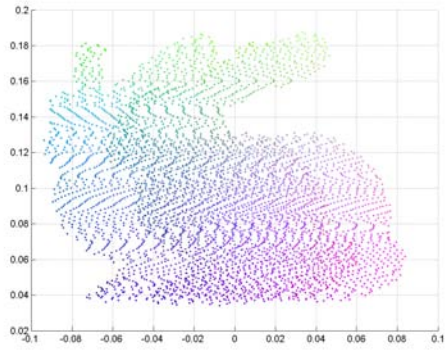


FIGURE 3.24 – Les deux vues de Bunny, en Pseudo-Couleur.

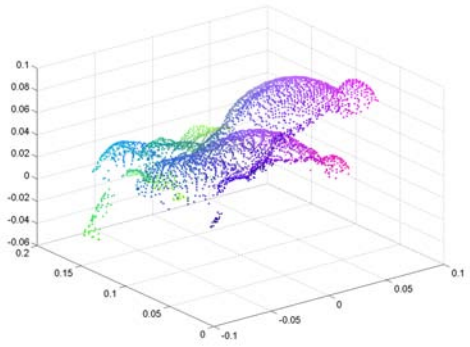


FIGURE 3.25 – Une autre vue des deux ensembles de points acquis sur Bunny, avec leurs Pseudo-Couleurs ; on peut apprécier la rotation inter-vues et la similitude des Pseudo-Couleurs.

configuration nous allons évaluer notre algorithme et le comparer avec l'algorithme ICP classique, mis en oeuvre par A.Restrepo [133]. Cette version est une implémentation standard de ICP ; il ne fait des appariements entre les points 3D des nuages à recaler, qu'avec un critère de distance ; différentes méthodes de filtrage permettent de ne garder que les points appariés les plus sûrs avant d'estimer la transformation entre les deux nuages par SVD.

En figure 3.24 nous pouvons observer aussi les deux mêmes vues, mais avec l'information en Pseudo-Couleur qui a été assignée à chacun des points dans les deux nuages acquis sur *Bunny*. Cette information en couleur a été assignée en suivant le processus mentionné dans la section précédente. La figure 3.25 présente un autre angle de vue des mêmes données en Pseudo-Couleur : nous pouvons observer la similitude des couleurs qui existent entre deux zones semblables dans les deux nuages. Grâce à cette similitude nous pouvons appliquer notre algorithme en utilisant la Pseudo-Couleur comme attribut discriminant des points. Il peut être observé qu'il y a quelques zones des deux modèles qui ne s'accordent pas précisément en couleur ; ce sont seulement quelques petites zones et par conséquent ces différences locales n'affectent pas le résultat final quand on applique la méthode à l'ensemble des points des deux nuages à recaler.

En appliquant l'algorithme d'ICP classique à ces deux vues, nous obtenons le résultat montré en figure 3.26. Les deux nuages sont recalés mais pas de manière adéquate ; il soit peut-être suffisant de relancer l'algorithme en utilisant la matrice obtenue comme transformation initiale afin d'obtenir un meilleur recalage, à la condition de ne pas tomber dans un minimum local lors de l'estimation de la transformation T_k, vu que nous n'exploitons pas ici de méthode d'optimisation globale.

En appliquant notre variante de l'algorithme ICP on obtient les résultats montrés en figure 3.27. Cette figure montre qu'en utilisant la variante ICP Pseudo-Couleur, nous obtenons de meilleurs résultats qu'avec l'algorithme classique d'ICP.

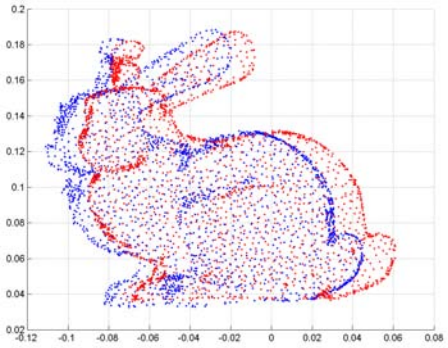


FIGURE 3.26 – Recalage final des deux vues de Bunny par l'algorithme ICP classique.

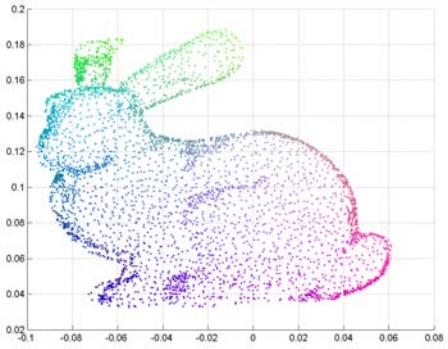


FIGURE 3.27 – Recalage final des deux vues de Bunny par l'algorithme ICP Pseudo-Couleur

FIGURE 3.28 – Image d'un des objets exploités pour évaluer nos travaux : une boîte.

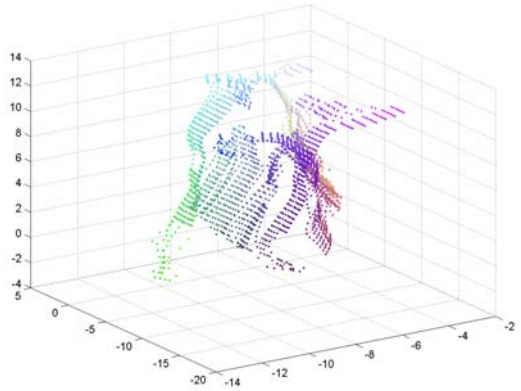


FIGURE 3.29 – Deux vues de notre objet, une qui sert de référence, l'autre qui est obtenue après une rotation de 15deg par rapport à la référence ; les deux sont présentés en Pseudo-Couleur.

Résultats sur des données réelles.

Nous allons à présent comparer les deux méthodes, dans les même conditions, sur des données réelles obtenues à l'aide d'une paire stéréo.

La figure 3.28 présente une image de l'objet réel, une petite boîte que nous avons utilisé pour valider nos méthodes de modélisation 3D. Cet objet, même s'il paraît simple, puisqu'il s'agit d'un parallélogramme, ne l'est pas en réalité pour ICP vu sa symétrie. Nous allons vérifier un des problèmes d'ICP quand il existe une symétrie dans les données à recaler : en effet, l'algorithme ICP classique tend à superposer deux vues symétriques d'un objet.

On a acquis 8 vues différentes pour modéliser l'objet montré en figure 3.28 ; les points sont acquis par stéréovision à partir d'images acquises par le capteur stéréo Videre. On a cherché à exploiter un minimum de vues de l'objet pour en construire le modèle. Ceci permet de gagner du temps, d'abord en exécutant moins souvent la stéréovision, ensuite en fusionnant moins d'images 3D dans le nuage complet, et enfin en générant un maillage sur un nuage moins dense. Par contre, ce choix de minimiser le nombre de vues, rend plus complexe le recalage, puisque la transformation inter-vues est plus grande.

Notons que ce compromis à trouver pour choisir le nombre de vues, est évité dans des approches plus récentes de la modélisation 3D, dans lesquelles deux ensembles de vues sont exploitées : d'une part peu de vues 3D pour acquérir des données 3D denses, d'autre part des vues acquises en continu lors du mouvement du capteur autour de l'objet, sur lesquelles peu de points 3D sont extraits, uniquement pour le recalage.

Pour réduire la complexité, nous effectuons une décimation sur chaque image 3D, ceci afin de manipuler

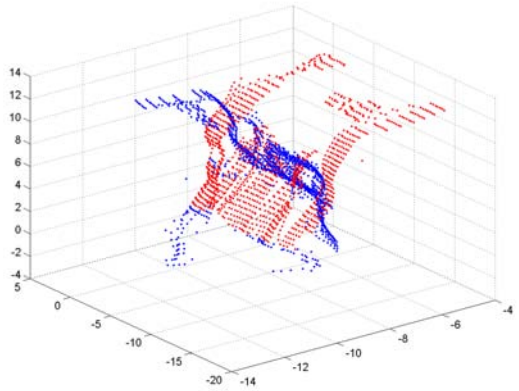


FIGURE 3.30 – Recalage final des deux vues de la boîte par l'algorithme ICP Classique.

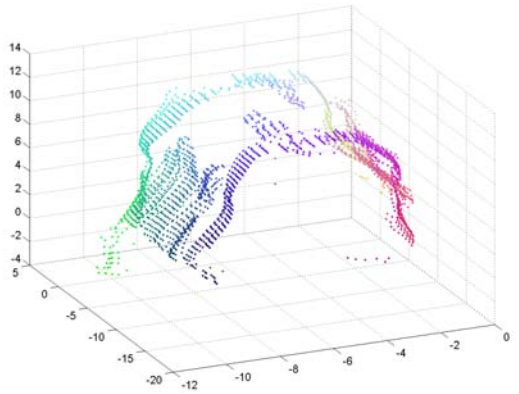


FIGURE 3.31 – Recalage final des deux vues de la boîte par l'algorithme ICP PseudoCouleur.

moins de points 3D au moment d'effectuer le recalage, la fusion et la construction du maillage : nous ne gardons que 1200 points environ par vue de l'objet. La figure 3.29 montre l'objet avant d'effectuer le recalage entre les vues ; elle présente déjà l'information en Pseudo-Couleur assignée à chacun des points dans les deux vues. Il peut être observé que, du fait des occultations et des positions du capteur par rapport à la boîte, il existe une grande quantité de points qui ne correspondent pas dans les deux ensembles de points. C'est seulement la partie supérieure de la boîte qui est commune aux deux vues ; c'est cette zone qui nous intéresse pour effectuer l'alignement et obtenir la matrice de transformation nécessaire pour fusionner les deux images 3D dans un seul nuage de points 3D.

L'algorithme d'ICP classique fournit le résultat que nous montrons en figure 3.30 : il peut être observé que malheureusement l'algorithme ICP ne converge pas puisque les zones opposées de l'objet ne sont pas superposées à la fin. Quand nous appliquons notre algorithme ICP PseudoCouleur, ce problème est corrigé, comme cela est montré en figure 3.31.

TABLE 3.2 – Tableau comparatif des temps d'exécution entre les algorithmes ICP Classique et la variante ICP Pseudo-Couleur. Les temps sont donnés en s.

Algorithme	Modèle virtuel	Modèle Réel
ICP Classique	127	45
ICP Pseudocouleur	141	58

TABLE 3.3 – Tableau comparatif sur le nombre d'itérations nécessaires pour les algorithmes ICP Classique et la variante ICP PseudoCouleur.

Algorithme	Modèle virtuel	Modèle Réel
ICP Classique	22	20
ICP Pseudocouleur	35	28

Finalement dans les tableaux 3.2 et 3.3 nous donnons le temps d'exécution pris par chacun des algorithmes, ainsi que le nombre d'itérations nécessaires pour converger. On peut observer que notre mise en oeuvre utilise davantage de temps et effectue davantage d'itérations que l'algorithme d'ICP classique. Ceci vient du temps pris dans notre méthode, par le processus d'assignation de la Pseudo-couleur aux points 3D des deux nuages à recaler.

3.5.2 Maillage des points 3D : Paramétrisation Sphérique

Présentation de la méthode.

Il existe divers articles qui parlent de la paramétrisation sphérique [55] [129]. Mais comme dans le cas de la décimation, tous les travaux appliquent la paramétrisation sphérique aux sommets d'un maillage et non d'un nuage de points 3D. Nous avons developpé une méthode similaire à celles mentionnées par Gu [55] et Sheffer [129] mais cette fois ci orientée pour traiter un nuage de points 3D.

Définissons d'abord une sphère, qui enferme complétement le nuage. Le centre $\vec{C}$ est sur le centre de gravité du nuage de points 3D de l'objet ; le rayon R est la distance entre $\vec{C}$ et le point le plus éloigné du nuage, plus un petit incrément ε.

Une fois déterminée le centre et le rayon de la sphère, chacun des points du nuage est projeté vers la sphère. Il doit y avoir un seul point sur la sphère pour un point dans le nuage 3D, ceci afin d'éviter des ambiguïtés. La projection s'obtient de la façon suivante :

soit une droite $\vec{L}$ qui passe par le centre $\vec{C}$ et par un point $\vec{N}$ du nuage à trianguler. On calcule la distance la plus petite entre le point $\vec{N}$ et la sphère.

Pour réaliser ce calcul, il est nécessaire de trouver l'intersection entre la droite $\vec{L}$ et la sphère définie par $\vec{C}$ et R. Considérons en premier lieu la paramétrisation de la droite $\vec{L}$ comme :

$$\vec{L}(t) = \vec{N} + t\vec{D} \tag{3.5}$$

où

$$\vec{D} = \vec{N} - \vec{C} \tag{3.6}$$

et la sphère comme

$$|\vec{L} - \vec{C}|^2 = R^2 \tag{3.7}$$

Le point $\vec{L}$ est à la fois sur le cercle (équation 3.7) et sur la droite (équation 3.5) ; nous obtenons donc une équation du second degré en t. Nous gardons uniquement la racine positive pour obtenir l'intersection de la droite et de la sphère. Avec ceci on obtient la projection de $\vec{N}$ sur la sphère. La figure 3.32 illustre cette projection ; le point P_i est projeté sur le point S_i de la sphère.

Après avoir projeté tous les points vers la sphère nous procédons à la triangulation des points 3D afin de générer un maillage triangulaire. Le maillage s'effectue sur la sphère, par l'algorithme de *Delaunay-convex hull*. Du fait que tous les points sont sur la sphère, nous obtenons une surface uniforme. Une fois finie la triangulation sur la sphère, on a les relations sommets/arêtes et arêtes/triangles : pour générer le modèle final de l'objet, il suffit de remplacer chaque sommet sur la sphère, par le point du nuage qui s'y est projeté.

Dans les figures 3.33 et 3.34 nous pouvons observer le nuage de points 3D de l'objet *Mannequin*, disponible sur le web, et sa projection sur la sphère.

Finalement, la figure 3.35 montre le résultat de l'application de la triangulation en utilisant la Paramétrisation Sphérique.

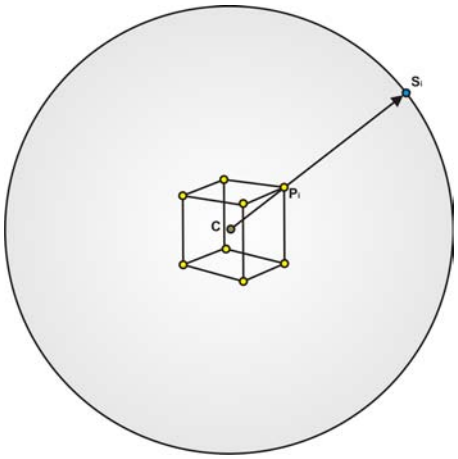


FIGURE 3.32 – Projection d'un point 3D vers la sphère.

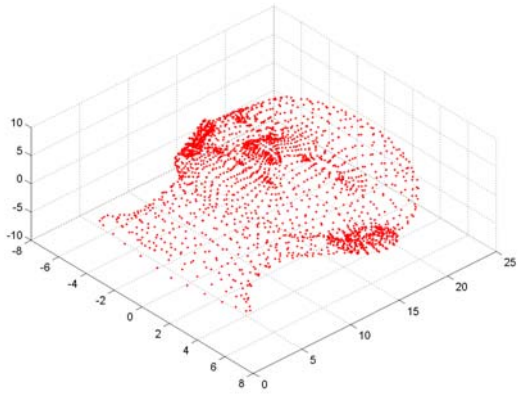


FIGURE 3.33 – Une vue du nuage de points sur le Mannequin.

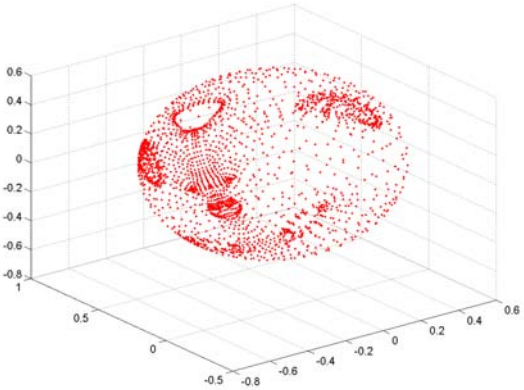


FIGURE 3.34 – Une vue de la projection Sphérique du Mannequin.

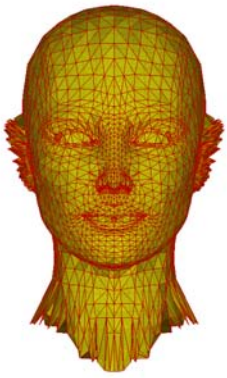


FIGURE 3.35 – Une vue du maillage obtenu par l'algorithme de Paramétrisation sphérique.

Résultats sur des données virtuelles.

Dans les sections suivantes nous allons montrer les résultats obtenus en utilisant l'algorithme de Marching Cubes, celui de Ball Pivoting et notre développement sur la Paramétrisation Sphérique. Les résultats qui exploitent l'algorithme de Marching Cubes ont été obtenus en utilisant une implémentation réalisée en VTK (Visual ToolKit) [2] [74]. Ceux qui exploitent l'algorithme de Ball Pivoting, ont été obtenus en utilisant l'implémentation développée par A.Restrepo [133].

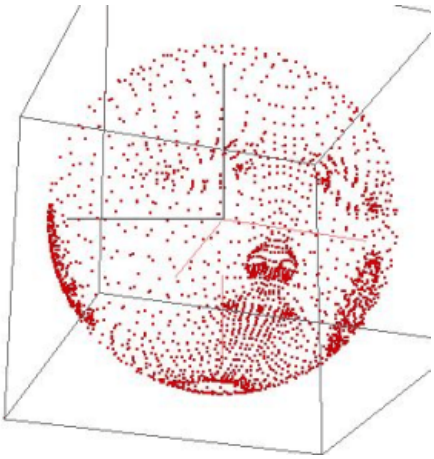


FIGURE 3.36 – Paramétrisation sphérique du mannequin

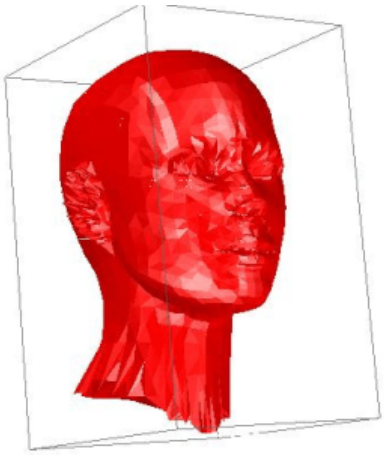


FIGURE 3.37 – Reconstruction par la paramétrisation sphérique

Nous allons commencer notre analyse sur l'objet *Mannequin*. La figure 3.36 montre la paramétrisation exploitée initialement par notre algorithme ; les points 3D acquis sur l'objet sont d'abord projetés sur une sphère. Le maillage construit en triangulant les points sur la sphère est présenté en figure 3.37).

Les figures 3.38 et 3.39 présentent la triangulation faite en exploitant respectivement les algorithmes de Marching cubes, puis de Ball Pivoting. On peut observer que la triangulation réalisée par notre méthode simple, a donné de meilleurs résultats que les autres algorithmes classiques de triangulation.

La table 3.4 montre les temps d'exécution pris par chacun des algorithmes. Bien que notre algorithme soit un peu plus lent, il rend de meilleurs résultats que celui de Marching Cubes. L'exécution de l'algoritmhe de Ball Pivoting a été arrêtée avant de finir ; c'est la raison pour laquelle le maillage obtenu par cet algorithme n'est pas complet.

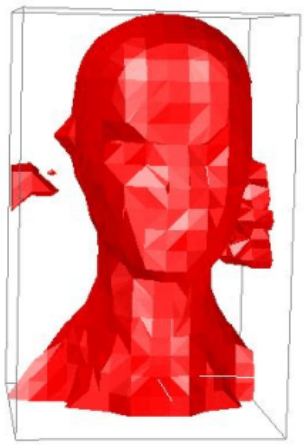


FIGURE 3.38 – Reconstruction par *Marching cubes*

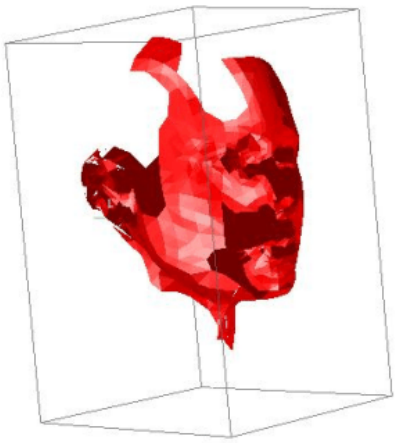


FIGURE 3.39 – Reconstruction par *Ball-Pivoting*

TABLE 3.4 – Tableau comparatif des temps d'exécution entre les algorithmes de triangulation. Les temps sont donnés en s.

Algorithme	M.C.	B.P.	P.S.
Mannequin	3.82	540.13	5.87
Boîte	2.51	360.32	3.74

Résultat sur des données réelles.

Dans les mêmes conditions, nous allons à présent évaluer notre méthode sur un nuage de points obtenus à partir de nos propres images stéréo. L'objet modélisé est toujours une boîte : bien que cela soit un objet simple, il est quand même significatif de la complexité qui apparaît dès que l'on souhaite construire un maillage à partir de données 3D acquises sur un objet quelconque. La boîte est présentée en figure 3.28, nous l'avons déjà utilisée pour évaluer la fonction de recalage.

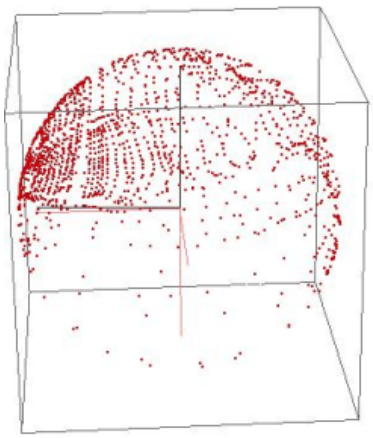


FIGURE 3.40 – Paramétrisation sphérique des points de la boîte

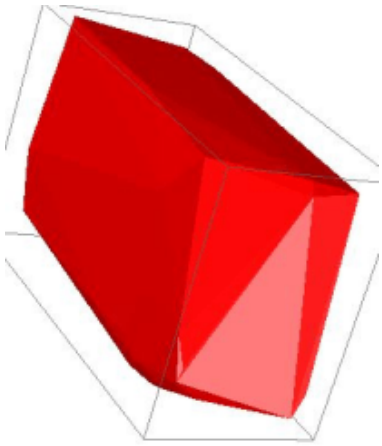


FIGURE 3.41 – Reconstruction par la paramétrisation sphérique

La projection des points 3D de la boîte sur la sphère peut être observée en figure 3.40 ; le maillage résultat est montré en figure 3.41. La reconstruction a été faite en 3.74 secondes ; dans la table 3.4, nous pouvons observer que le temps pris par notre implémentation pour réaliser le maillage de la boîte, n'a pas été le plus rapide, mais on peut constater en regardant les images 3.42 et 3.43, que le résultat est

meilleur.

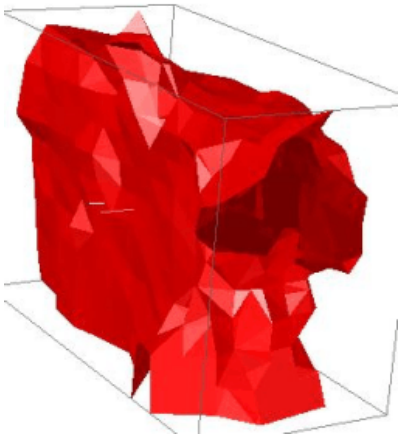


FIGURE 3.42 – Reconstruction par *Marching cubes*

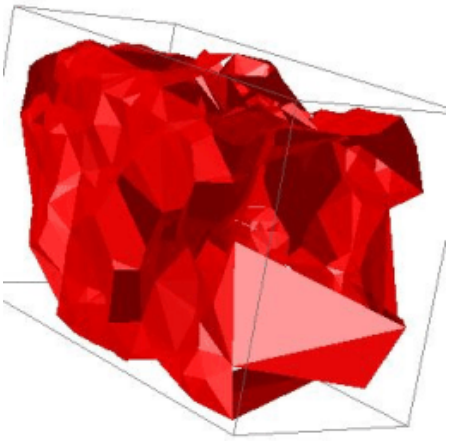


FIGURE 3.43 – Reconstruction par *Ball-Pivoting*

Les maillages triangulaires obtenus tant par l'algorithme de Marching Cubes que par l'algorithme de Ball Pivoting, ne sont pas satisfaisants à cause des trous sur le maillage. Marching Cubes a été le plus rapide ; Ball Pivoting prend beaucoup de temps pour obtenir un maillage qui en plus présente le même problème de trous.

3.6 Conclusion sur la reconstruction.

Dans ce chapitre nous avons montré la méthode que nous avons proposée pour accomplir en ligne, la tâche de reconstruction tridimentionnelle d'un objet posé sur une table. Nous avons montré que les procédures réalisées tant pour l'algorithme ICP que pour celui de paramétrisation sphérique, rendent les résultats souhaités.

Cependant, les modèles 3D, de type maillage triangulaire, créés par ces procédures, n'ont pas pu être exploités par le module de planification des positions de prise qui doit, également en ligne, trouver les positions optimales des mors d'une pince ou des doigts d'une main, afin de saisir l'objet. Nous avons pronostiqué que cette erreur vient :

- de la stéréovision qui donne des données 3D trop imprécises pour construire un maillage lisse,
- et de la méthode d'intégration des nuages de points 3D acquis de la stéréo par ICP : les techniques d'optimisation fondées sur l'ajustement des faisceaux, ou d'estimation de type SLAM, semblent plus prometteuses.

Par ailleurs, les méthodes proposées ont des limitations : comme précisé en section 3.5.2, l'algorithme de paramétrisation sphérique est valide seulement pour les objets convexes. Une première amélioration serait la prise en compte de la dite limitation : la méthode par elle-même ne peut pas être améliorée. Il existe plusieurs solutions possibles :

- rechercher ou produire une implémentation robuste des *Marching Cubes*, celle de VTK n'ayant pas donné satisfaction.
- optimiser le code de la procédure *Ball Pivoting* qui a été testée avec succès par A.Restrepo, mais (1) cette méthode est intrinséquement complexe, et il sera difficile d'approcher le temps réel pour l'exécuter en ligne sur le robot, et (2) les tests effectués au LAAS ont été faits presque exclusivement sur des données laser : cette méthode est peu robuste lorsque les données 3D sont très bruitées.
- enfin exploiter une autre méthode. Depuis peu, la procédure des *Alpha Shape* de la librairie CGAL a été testée avec succès au LAAS. Des tests plus nombreux restent à faire pour évaluer sa robustesse. On pourrait aussi exploiter les *Surfaces Déformables*, avec une implémentation déjà existante au LAAS, mais à "dérouiller" pour l'intégrer avec les codes existants.

Nous n'avons pas poursuivi sur ce sujet, préférant nous consacrer à notre sujet initial, la reconnaissance

active d'objets 3D de forme libre, appris en ligne, sujet que nous allons aborder dans le chapitre suivant.

Pour cette tâche de reconnaissance, nous proposons d'exploiter différents descripteurs afin d'avoir une représentation qui permet de reconnaître l'objet même s'il est partiellement occulté. Ces descripteurs devaient être fondés soit sur des informations photométriques (dites *d'apparence*) extraites d'images 2D , soit sur des informations 3D extraites de maillages. Les limitations sur notre procédure de construction de maillage, nous ont conduit à privilégier l'apparence plutôt que la géométrie.

Chapitre 4

Reconnaissance d'objets 3D de forme libre

4.1 Introduction

Ce chapitre va traiter la problématique de la reconnaissance d'objets 3D de forme libre, c'est-à-dire un objet qui n'a pas une forme structurée précise comme une boîte ou une sphère. C'est le type d'objets que nous utilisons dans notre vie quotidienne, comme, par exemple, un verre, une tasse, un *mug*, une agrafeuse, une télécommande, un briquet...

La reconnaissance d'objets est essentielle pour des systèmes robotisés, qui doivent savoir quels sont les objets présents dans l'environnement dans lequel ils doivent exécuter des tâches et à quels endroits ils sont placés. Par exemple, en robotique mobile un tel système de reconnaissance pourrait permettre à un robot de se localiser dans son environnement, en connaissant la position respective d'objets particuliers présents dans le monde où il se déplace : c'est la navigation sur amers visuels. Rappelons en particulier les travaux effectués au LAAS sur les environnements intérieurs : détection et reconnaissance d'amers quadrangulaires plans (posters, portes...) [58][16], planification optimale de la trajectoire en fonction de la perception sur des amers visuels [147][15]... Dans notre cas, nous cherchons à développer plutôt un système robotisé qui soit capable de modéliser, puis de reconnaître des objets présents dans le milieu où le robot exécute certaines tâches demandées par l'Homme, du type *Va chercher Objet X dans Lieu Y*, *Ramasse Objet au Sol*... Nous laissons de côté les fonctions *Navigation autonome* ou *Localisation du robot*, bien que notre méthode de reconnaissance pourrait aussi être exploitée pour la reconnaissance d'amers.

Nous désirons que le système soit capable de reconnaître les objets tridimensionnels qui sont dans le champ de vue de notre robot, à l'aide de caractéristiques 2D et 3D apprises sur ces objets : 2D car

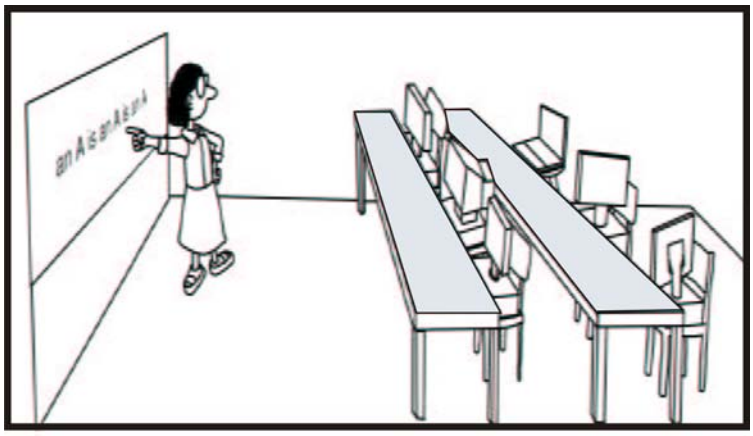


FIGURE 4.1 – Apprentissage Machine

obtenues à partir d'informations photométriques extraites des images acquises sur l'objet, et 3D obtenues à partir des propriétés géométriques extraites des modèles 3D construits au préalable par la méthode décrite dans le chapitre précédent. Ces caractéristiques sont mémorisées dans une base de données construite incrémentalement par le système lors d'une phase d'apprentissage, généralement supervisée par un opérateur. Cette base de données est exploitée de manière autonome par le système, lors de la phase de reconnaissance.

Dans la section suivante, nous décrirons quelques méthodes qui ont retenu notre attention dans la littérature : nous ne pourrons pas être exhaustif, vu l'énorme quantité de travaux en ce domaine. Ensuite nous présenterons les différentes méthodes et techniques qui ont été développées pour représenter la connaissance acquise lors d'un apprentissage artificiel.

Enfin, nous présenterons nos travaux sur la reconnaissance active fondée sur la maximisation de l'information mutuelle. Nous décrirons d'abord une première mise en oeuvre simple que nous avions développée au Mexique, en ne considérant que la couleur pour discriminer les objets entre eux. Puis, nous présenterons la méthode complète pour prendre en compte un ensemble quelconque d'attributs 2D ou 3D dans les modèles des objets à reconnaître ; c'est une généralisation de notre première approche :

- elle peut prendre en compte un grand nombre de caractéristiques, présentées sous la forme d'histogrammes ;
- elle est active dans le sens où le système calcule les positions successives de la caméra vis-à-vis de la scène en cours d'analyse, jusqu'à étiquetage de tous les objets présents dans cette scène ;
- une étiquette *NULL* est associée aux objets présents dans la scène, non encore appris.

Cette méthode est évaluée sur un grand nombre de séquences d'images de synthèse ou d'images réelles, acquises sur des scènes de complexité croissante (objet isolé, plusieurs objets disjoints, objets en vrac, objets inconnus présents dans la scène...) ; finalement nous commentons les résultats obtenus pour discuter des avantages et des limitations de l'approche proposée.

4.2 Etat de l'art en Reconnaissance d'objets

Dans la littérature, nous trouvons une grande quantité d'articles écrits à ce sujet ; nous évoquons plusieurs classes de méthodes que nous avons analysées avant de développer notre propre approche.

Afin de pouvoir réaliser une bonne reconnaissance d'objets il est nécessaire d'avoir (1) une représentation des objets qui vont être recherchés dans les données acquises par les capteurs, (2) des traitements sur ces données permettant d'extraire et de comparer des caractéristiques discriminantes, et (3) une stratégie d'analyse d'une scène donnée. La représentation doit être choisie en accord avec les caractéristiques que l'on sait extraire de ces données, et avec le type de traitements que nous voulons appliquer pour la reconnaissance. La représentation influe sur les limitations de l'analyse : par exemple, une représentation globale en terme de régions, reconnues par leur périmètre, leur surface... est très sensible aux occultations, tandis qu'une représentation en terme d'un ensemble de primitives locales (points d'intérêt, lignes...) permettra de reconnaître un objet à partir d'une vue partielle.

Approches 3D vs Approches apparence.

Il existe plusieurs façons de représenter un objet. Les modèles fondés sur la **forme** de l'objet, sont construits à partir de caractéristiques géométriques 3D, typiquement des primitives surfaciques extraites par des méthodes de segmentation. Les méthodes fondées uniquement sur ces modèles 3D, suivent une **approche structurelle** : l'objet est représenté par une hiérarchie de sous-composants (faces, arêtes, sommets, mais aussi, faces incluses dans une autre : trous ou excroissances...). Cette décomposition conduit à représenter un objet par un graphe ou un arbre. Un objet dans une scène pourra alors être reconnu à partir des éléments qui sont référencés dans les noeuds de ce graphe : des Points (sommets de maillage, points de courbure maximale...), des Lignes (arêtes sur un objet polyédrique, limbes d'un cylindre...), des Faces Planes et Polygonales (faces d'un polyèdre...). L'analyse de la scène sera guidée par les graphes, modèles des objets qui peuvent éventuellement s'y trouver.

Cette approche a été très utilisée pour décrire des objets structurés (par exemple, polyédriques), à partir de modèles 3D obtenus par CAO ou construits lors d'une phase préalable de modélisation à partir des données sensorielles. C'est une approche bien adaptée pour des objets artificiels (construits

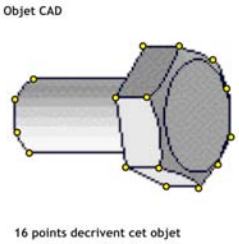


FIGURE 4.2 – Description d'un objet structuré par seulement 16 points.

par l'homme) comme celui de la figure 4.2. L'avantage de ces modèles est la facilité d'utilisation, et les attributs géométriques invariants qui leurs sont attachés. Par exemple, si nous voulons décrire une pièce mécanique on peut exploiter la longueur des arêtes, la surface des faces, la quantité de sommets qui les définissent... Par contre, ces méthodes nécessitent d'exploiter un capteur 3D aussi bien lors de l'apprentissage que lors de la reconnaissance : la simplicité de la phase de reconnaissance est compensée par la complexité d'extraction des attributs 3D à partir des données sensorielles. Par ailleurs, la robustesse de la reconnaissance dépend complétement de la robustesse des méthodes d'extraction de ces attributs 3D, robustesse difficile à obtenir.

Ce constat, plus une tendance toujours forte dans nos milieux à imiter le monde vivant, ont conduit à privilégier depuis une décennie, les approches fondées sur **l'apparence** des objets dans des données visuelles. En effet, les êtres biologiques (dont l'Homme) reconnaissent les objets qui les entourent principalement à l'aide du sens visuel. Les méthodes fondées sur l'apparence représentent les objets par des caractéristiques telles que la Couleur et/ou la Texture rattachées à des régions, ou telles que des primitives visuelles invariantes de type points ou courbes.

Certaines méthodes exploitent les deux types de représentations, **forme et apparence**, en tirant parti de la fusion de données 3D et de primitives visuelles. La stéréovision est un capteur 3D, très présent dans le monde vivant, qui facilite cette fusion : une face extraite par stéréovision peut avoir pour attributs, une couleur, une texture, une taille, une forme (disque d'un rayon donné, polygone avec nombre de sommets ...), des primitives visuelles (contours ou points d'intérêt). La méthode que nous proposons s'inscrit dans cette dernière classe, car elle peut exploiter un nombre quelconque de descripteurs 2D ou 3D sur les objets.

Dans la suite, nous décrivons succinctement différentes approches, celles qui, à notre avis, sont les plus significatives dans l'état de l'art.

Approches fondées sur PCA ou sur les *eigenfaces*

Commençons par les travaux qui utilisent les caractéristiques obtenues à partir des valeurs et vecteurs propres de régions prises sur les images : on les appelle *eigenfeatures*, ou bien *eigenimages* quand on les calcule sur des images, et plus spécifiquement *eigenfaces* si on les utilise pour reconnaître des visages. Turk et Pentland [150] utilisent les *eigenfeatures* pour localiser, reconnaître et suivre un visage. Swets *et al.* [139] [138] font la sélection automatique des caractéristiques à partir d'un ensemble d'images d'apprentissage en utilisant les théories d'analyse discriminante sur des attributs multidimensionnels. Abdi *et al.* [6] suggèrent que les caractéristiques agissent de manière flexible et qu'elles peuvent être exploitées pour la reconnaissance de visages en fonction de l'origine ethnique. Leonardis et Skočaj [78] [131] pour leur part se servent des avantages des *eigenfeatures* pour réaliser des reconnaissances d'objets plus robustes en résolvant le problème des pixels perdues, du bruit et des occultations.

Citons aussi dans le même registre, les travaux qui utilisent l'Analyse en Composantes Principales ou PCA (*Principal Component Analysis*). Salamanca *et al.*[120] utilisent l'information obtenue par PCA sur l'image, pour reconnaître un objet à partir de vues partielles. De leur côté Vicente *et al.*[152] appliquent la technique de PCA sur les images de contours et obtiennent ainsi une meilleure performance que les méthodes qui appliquent PCA sur les images directement. Ke et Sukthankar [72] exploitent aussi une image de contours comme dans [152], mais ils utilisent une région autour d'un point d'intérêt et c'est à cette région qu'ils vont appliquer la méthode de PCA : les résultats obtenus ont été meilleurs et plus

rapides qu'avec une approche globale sur l'image. Nagabhushan *et al.* [102] proposent une technique appelée *Principal Component Analysis en deux dimensions* pour reconnaître des objets depuis des données 3D ; cette technique est fondée sur des matrices à deux dimensions plutôt que sur des vecteurs unidimensionnels ; de cette manière une image n'a pas besoin d'être transformée en un vecteur afin d'obtenir ses caractéristiques.

Approches fondées sur les points d'intérêt

Depuis une dizaine d'années, de nombreuses thématiques en Vision par Ordinateur ont connu de grands progrès grâce aux travaux qui exploitent des points d'intérêt comme caractéristiques photométriques invariantes. Tout d'abord mentionnons les travaux précurseurs de Smith et Brady [132] sur le détecteur Susan, puis ceux de Harris et Stephens [57] qui ont développé la méthode de détection de coins et de points d'intérêt appelés souvent *Harris points*. De nombreux auteurs ont exploité les points de Harris pour recaler des images, pour reconnaître des objets, pour estimer le mouvement d'une caméra mobile ... Citons en particulier au LAAS, la méthode d'appariement de points de Harris proposée par I.K.Jung [68], méthode exploitée ensuite pour l'odométrie optique, pour la reconnaissance d'objets 3D ...

Les points de Harris sont toujours exploités, car ils sont très rapides à détecter ; par contre, ils ont peu de caractéristiques, d'où les difficultés pour les mettre en correspondance. D.Lowe [86] [87] [88] a popularisé une technique multiéchelle d'extraction de points d'intérêt, basée sur les différences de gaussiennes (DOG), et surtout, il a proposé de caractériser chaque point ainsi extrait par un descripteur invariant à l'échelle et dans une moindre mesure, aux rotations. Par extension, on appelle point SIFT (pour *Scale Invariant Feature Transform*) ou simplement SIFT, un point doté d'un tel descripteur de la région qui l'entoure ; une telle région est souvent appelée *patch*. Dans cette même catégorie on retrouve les points SURF (pour *Speeded Up Robust Features*) ; de la même façon qu'un SIFT, un SURF est caractérisé par un descripteur sur une région d'intérêt ou *patch* centré sur lui. Les SURFs ont été développés par Bay *et al*[20] et, comme leur nom l'indique, présentent la qualité d'être plus rapides à extraire, bien que cette extraction se base sur la Hessienne, donc les dérivées d'ordre 2. Ils sont réputés pour être plus robustes que des SIFTs. Dans les deux cas, les descripteurs servent à mettre en correspondance par classification, les points extraits sur des images différentes.

D'autres approches ont été proposées pour extraire des points invariants. Citons sans les détailler les *Scaled Salient Patches* de Kadir et Brady [69] exploités à Oxford (UK) pour l'identification d'amers visuels par un robot, mais aussi au LAAS, par M.Cottret [37] pour apprendre et reconnaître des objets par une approche passive. Citons en France les travaux de K.Mikolajczyk et C.Schmid [92], connus pour les études comparatives sur l'invariance des points d'intérêt extraits par diverses méthodes. Citons enfin Stein et Hebert [135] qui ont développé une variante des SIFTs appelée BSIFT (pour *Background and Scale Invariant Feature Transform*) qui incorpore l'information locale des limites de l'objet ; avec cette variante ils rendent le descripteur invariant aux changements de l'arrière-plan.

Approches fondées sur des représentations 3D.

La plupart des travaux cités précédemment décrivent des approches uniquement fondées sur l'apparence des objets dans des images, c'est-à-dire, qui négligent l'aspect tridimensionnel des objets ou des scènes qu'ils veulent analyser. Il existe un grand nombre d'auteurs aussi qui ont proposé des approches de reconnaissance par comparaison d'images 3D avec des modèles 3D. Pour ce faire il faut exploiter des invariants 3D : dans de très nombreux travaux sur la reconnaissance d'objets de forme libre, c'est l'orientation des normales (dérivées d'ordre 1) ou de la courbure (dérivées d'ordre 2) qui serviront pour représenter et discriminer des objets [168]. Les normales suffisent pour des objets artificiels polyédriques : elles servent généralement à segmenter des images 3D en facettes planes, qui sont à la base de nombreuses méthodes de reconnaissance.

Concernant les objets de forme libre, mentionnons le travail de Hebert *et al.* [65] [64] [63] qui ont développé une approche appelée *Spin images*. La *Spin Image* est un descripteur de forme de niveau hiérarchique, qui est utilisé pour faire la comparaison entre maillages. C'est aussi une description des objets 3D invariante en rotation et translation.

Toujours en exploitant le 3D, nous retrouvons l'utilisation de *Point Signature* développée par Chua et Jarvis [35] ; cette approche permet de décrire le voisinage d'un point 3D d'une meilleure manière qu'en utilisant des coordonnées 3D du point et de ses voisins ; en plus, ce descripteur est invariant à la rotation

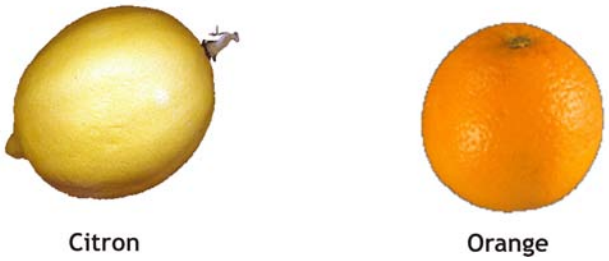


FIGURE 4.3 – Description d'un objet en utilisant la couleur.

et à la translation. De la même façon on peut citer la méthode qui exploite des *Surface Signatures* [166] [164] [165] ; cette manière de décrire un objet 3D capture l'information de la courbure 3D de n'importe quel objet de forme libre et la codifie en une image 2D qui décrit jusqu'à un certain point, les variations de courbure sur la surface de l'objet.

Approches fondées sur la couleur.

L'extraction des caractéristiques telles que la couleur ou la texture, permet souvent de discriminer des objets entre eux, comme cela peut être apprécié dans la figure 4.3. Cette figure présente deux objets similaires puisqu'ils ont une forme plus ou moins circulaire tous les deux, mais en ajoutant la couleur on peut faire la différence entre le citron et l'orange sans difficulté. De même de très nombreux travaux en vision, ont été consacrés à la classification des textures, à partir de divers descripteurs : citons les matrices de cooccurences (Haralick) et les histogrammes de sommes et différences (Unser) ; citons au LAAS les travaux de G.Aviña [14], sur l'exploitation de descripteurs couleur-texture, pour identifier la nature de régions extraites dans une image acquise en milieu naturel (herbe, arbre, terre, cailloux, ciel...).

Le principal problème de ces descripteurs couleur-texture, tient à leur faible invariance vis-à-vis des conditions d'illumination pour la couleur, et de la profondeur (on parle plus d'échelle ou *scale*) ou de la résolution des images pour la texture. Ces caractéristiques sont donc peu robustes ; cela peut provoquer des ambiguïtés sur la nature des objets présents dans une scène déterminée, par exemple de trouver que éclairés d'une certaine manière, notre citron et notre orange ont la même couleur. De ce fait, ces attributs, que nous, humains, avons l'impression d'exploiter en permanence dans nos propres processus d'interprétation d'images, sont assez peu utilisés dans les fonctions visuelles des robots intervenant dans un environnement non contrôlé (éclairage naturel) ; avec les SIFTs et leurs variantes, les chercheurs ont développé d'autres approches pour décrire un objet avec des caractéristiques invariantes à des facteurs comme l'échelle, l´illumination, la rotation ou la translation.

Si nous voulons exploiter la couleur pour reconnaître un objet, il convient de maîtriser parfaitement l'acquisition, donc d'appliquer des traitements de base tels que le démosaïquage qui permet de construire une image couleur à partir d'une mosaïque Bayer, puis la balance des blancs qui permet de supprimer une couleur dominante propre à chaque capteur (phénomène appelé *color casting*). Puis il faut exploiter une représentation de la couleur qui limite les effets de sa non invariance, donc transformer l'image acquise en RGB, dans un espace de couleur qui soit moins sensible aux changements de la lumière, par exemple l'espace de couleur Cie Lab [1] qui a de meilleures propriétés d'invariance vis-à-vis des variations d'illumination.

Approches actives.

Rappelons que le paradigme de la perception active, est d'agir pour mieux percevoir ; cette notion a été introduite par de nombreux auteurs à la fin des années 1980, provoquant de nombreux débats dans la communauté. Citons Y.Aloimonos [10], R.Bajcsy [17] et J.K.Tsotsos [156]. Une synthèse de ces travaux peut être trouvé dans [30].

Il existe différentes étapes d'un processus perceptuel pendant lesquelles on peut agir. Tsotsos en identifie quatre :

1. En 1976, la CIE a proposé l'espace de couleur CIELAB, aussi appelé espace de couleur CIE 1976 comme une approximation d'un espace de couleur uniforme (Munsell est la référence standard).

- le choix des fonctions perceptuelles : va t'on reconnaître un objet en exploitant des SIFTs, des segments, des régions, un groupement perceptuel... ?
- le choix des paramètres des fonctions perceptuelles : quels seuils du gradient pour extraire des segments, quelle surface pour supprimer les petites régions.... ... ?
- le choix de la zone de l'image sur laquelle appliquer ces fonctions.
- le choix du point de vue sur la scène à analyser : où positionner le capteur pour acquérir l'information la plus discriminante à un stade donné du processus de reconnaissance ?

Pour agir pendant un processus perceptuel, il faut rajouter une étape de décision pour sélectionner la meilleure action, les meilleurs paramètres, la meilleure position du capteur... en fonction de l'état du processus. Comme l'interprétation d'images est par nature incertaine, les approches proposées sont généralement probabilistes ; citons les deux formalismes classiques en ce domaine, les réseaux bayésiens [107, 115] ou *Bayesian Belief Network* avec leurs nombreuses variantes (DBN...) et les Modèles Décisionnels de Markov [114] (MDP, avec les variantes *Partially Observable* MDP, *Hidden Markov Model*...).

Il existe quleques travaux sur la reconnaissance active : citons les travaux récents de N.Trujillo Morales [96] qui propose une représentation des objets par un ensemble de points (ni point de Harris, ni SIFT), et une recherche active de ce modèle dans des images, en sélectionnant les zones successives dans lesquelles rechercher des éléments du modèle. Cette approche a été validée essentiellement dans un contexte vidéo-surveillance, donc avec des capteurs fixes.

Quand on traite de la reconnaissance d'objets pour des applications Robotique, les capteurs sont généralement embarqués sur des robots, bras manipulateurs ou plates-formes mobiles : il est donc possible de les déplacer pendant le processus de reconnaissance, cela afin d'optimiser le point de vue. Citons les travaux de F.Dornaika [43] qui propose une stratégie active afin de contrôler une tâche de prise d'objet par asservissement visuel : un système stéréo est déplacé au fur et à mesure que s'accomplit la tâche afin de suivre au mieux le préhenseur fixé en bout du bras qui doit saisir un objet.

Dans un contexte plus lié à la reconnaissance, citons les travaux fondés sur les réseaux bayésiens, en particulier sur la bibliothèque Hugin très populaire il y a une dizaine d'années : D.Djian et P.Rives [42] d'abord, puis au LAAS, S.Jonquières [67] ont proposé des approches de reconnaissance active contrôlées par des BBN préalablement appris ou construits. Les écueils habituels de ces approches concernent

- la taille du réseau bayésien nécessaire pour représenter toutes les situations, tous les critères à prendre en compte pour conduire à la bonne décision ;
- l'initialisation des tables de probabilités conditionnelles exploitées dans ce formalisme pour mettre à jour les probabilités de tous les événements par la loi de Bayes, dès lors qu'un noeud est modifié par un événement externe (une primitive a été extraite, un appariement scène-modèle a été trouvé, un objet a été reconnu, une information contextuelle peut être exploitée ...).

Concernant ces tables, les approches proposées sont généralement empiriques, du type essais-erreurs successifs. Concernant la taille du réseau, signalons le découpage proposé par S.Jonquières, qui a conçu une approche active de reconnaissance d'objets structurés par vision monoculaire. Le processus de reconnaissance a été décrit par 4 réseaux relativement découplés, les noeuds de sortie des uns servant d'entrée aux autres. Ces réseaux permettent respectivement (1) de sélectionner quel groupement perceptuel extraire de l'image courante, (2) de décider de la mise en correspondance entre les groupements extraits et ceux mémorisés dans le modèle des objets appris, (3) de décrire l'état du processus de reconnaissance. et enfin (4) de sélectionner la prochaine position de la caméra en fonction de graphes d'aspects des objets appris et de l'état du processus. Notons que cette approche n'a été que partiellement implémentée et validée.

Conclusion sur l'état de l'art.

Nous avons ci-dessus présenté quelques travaux sur la reconnaissance. On pourrait encore citer les travaux fondés sur la programmation flexible, comme la logique floue [98] [118] ou les réseaux de neurones [137] souvent exploités uniquement pour réaliser la classification de descripteurs extraits d'une image, parmi ceux appris sur différents objets. Ces approches donnent des résultats intéressants pour des applications en milieu contrôlé : généralement elles sont peu robustes aux variations dans l'environnement ou à l'introduction de nouveaux objets dans la base de connaissances.

Pour ces raisons, nous avons privilégié les approches probabilistes dans notre contribution. Mais au lieu d'exploiter les réseaux bayésiens, qui conduisent à des approches complexes et qu'il est difficile de

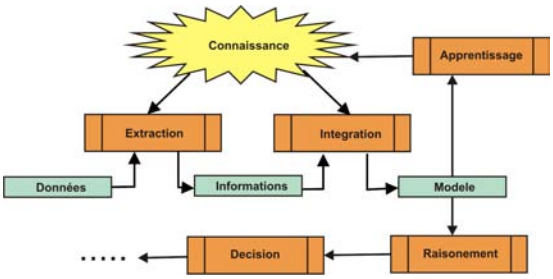


FIGURE 4.4 – Connaissance

faire évoluer en ligne pour introduire de nouveaux objets, nous nous sommes inspirés des travaux de J.Denzler et C.Brown [41, 40] qui se fondent sur la théorie de l'information, plus exactement sur la notion d'information mutuelle, afin de sélectionner la position d'une caméra pour mieux estimer l'état d'une scène statique. Evoquons également d'autres approches que nous allons exploiter dans notre propre méthode : les travaux qui font usage de la couleur [99] ou de la forme des contours des objets dans l'image [22].

Mais, avant de décrire nos propres travaux, évoquons une question fondamentale : comment représenter les connaissances nécessaires pour interpréter une image, pour reconnaître des objets ?

4.3 Représentations des connaissances

Qu'est que la connaissance ? comment une machine peut-elle apprendre ? Pour répondre à ces questions, nous allons commencer par définir la connaissance tel que l'être humain la perçoit.

4.3.1 Systèmes à base de connaissances.

La connaissance, en son sens le plus général, est constituée de multiples données interdépendantes ; chacune, par elle-même, représente la plus petite valeur qualitative d'information sur le champ d'intérêt à décrire. Pour les connaissances sur le monde réel qui nous entoure, cela se concrétise par un modèle ou une représentation de la réalité, mémorisé à plusieurs niveaux. La construction d'une telle représentation commence par les sens qui acquièrent des données sur le monde physique, les fusionnent et les filtrent (acquisition du *primal sketch*) ; le modèle multisensoriel passe à une couche perceptive, exploitant l'ensemble des connaissances disponibles, afin d'interpréter, de comprendre ; enfin, le modèle perceptif du monde est exploité par la *raison* afin de prendre des décisions.

Il y a une utilisation simplement formelle des représentations produites par les étapes de compréhension et de raison, c'est-à-dire une utilisation logique puisque ces représentations sont une abstraction du monde physique (symboles). Le savoir est l'ensemble des connaissances qui permettent à l'être humain de produire une pensée continue tout au long du temps.

Intelligence Naturelle.

Afin de définir un processus cognitif sur une machine, le premier pas consiste à identifier les caractéristiques d'un être humain intelligent, définies comme ses capacités de réponse devant certaines circonstances. Cette capacité dépend de :

- L'expérience.
- Les règles de comportement.
- La facilité d'acquérir une information externe.
- La rapidité pour enchaîner des pensées.
- La logique pour interpréter (comprendre) et prendre des décisions.
- Certains éléments non définissables.

La faculté d'acquérir, de manipuler des connaissances, et de les exploiter dans des raisonnements, est une autre façon de considérer le concept d'intelligence naturelle. Un être humain intelligent se caractérise par les facultés suivantes :

1. Il acquiert des informations sur le monde qui l'entoure, spécialement à travers les sens de la vue et de l'audition, et les stocke dans sa mémoire.
2. Il raisonne et déduit. Il manipule les connaissances et l'information qu'il possède, en appliquant une série de règles apprises par l'expérience, règles parfois formelles et parfois intuitives qui permettent de résoudre les problèmes qui se présentent par déductions successives.
3. Il agit en fonction des solutions élaborées. Soit il communique ses déductions à l'extérieur au moyen d'un langage, soit il exécute des mouvements avec ses organes moteurs afin de manipuler ou de se déplacer.
4. Enfin, il produit des émotions. La joie, la tristesse, y compris l'amour, sont des sentiments qui se détachent des autres raisonnements dits "intelligents".

Connaître, comprendre et raisonner sont donc les termes qui sont le plus en rapport avec l'intelligence ; il existe évidemment plusieurs degrés ou niveaux d'intelligence. Ce n'est pas un attribut exclusif des hommes, puisque d'autres êtres vivants peuvent aussi se considérer intelligents, dans la mesure où ils savent extraire des informations du monde extérieur, mémoriser des expériences et raisonner pour réagir face à de nouvelles situations.

Un comportement intelligent se caractérise aussi par la diversité des solutions mises en oeuvre face à une situation donnée. Selon les circonstances et les facteurs objectifs et subjectifs, devant une même situation, les êtres intelligents ne prennent pas la même décision. Signalons par exemple, l'importance qu'ont dans les jeux, les diverses attitudes qui sont adoptées par des joueurs placés dans la même situation, selon les diverses phases du jeu ou du comportement des rivaux.

Intelligence Artificielle.

Ces modèles cognitifs des êtres naturels dits "intelligents", ont été exploités par l'Homme pour concevoir des artefacts ou machines munies d'une Intelligence Artificielle.

L'Intelligence Artificielle a été créée par l'Homme pour copier, dans une certaine mesure, l'intelligence naturelle. Pour ce faire, l'Homme exploite l'ordinateur, machine capable de traiter une grande quantité de données en très peu de temps, selon des directives définies dans un programme. De nos jours on considére encore la plupart des ordinateurs comme des machines idiotes, qui n'exécutent que ce qui a été préalablement programmé. Cette manière d'opérer élimine une des qualités principales de l'intelligence, le raisonnement afin d'adopter un comportement qui n'a pas été programmé de manière explicite, mais qui est planifié suite à une manipulation intelligente de connaissances et d'expériences, comme l'être humain le fait si bien : devant les même faits, son expérience lui donne en permanence, plusieurs voies pour raisonner et pour réagir

Une machine pourra être considérée comme intelligente quand elle remplira les conditions suivantes :

A) Elle est capable de percevoir à partir de diverses données sensorielles (vision, audio, haptique...) les objets qui l'entourent et de reconnaître leurs caractéristiques (forme, apparence, comportement s'il est dynamique...).

B) Elle « comprend » le langage naturel, parlé, écrit ou sur support multi-média, et elle produit des réponses dans ce langage.

C) Elle planifie des actions en accord avec les conditions changeantes de l'environnement et les exécute au moyen de ces organes physiques (actionneurs : roues, pattes, bras manipulateur...)..

D) Enfin, elle peut stocker de l'information et des connaissances qu'elle peut ensuite manipuler au moyen de règles et d'algorithmes pour trouver des solutions aux problèmes qui lui sont posés.

Comme un être vivant, un artefact (robot, système informatique...) doit donc être doté de facultés comme :

- Percevoir et manipuler des informations issues du monde réel.
- Acquérir, appliquer des connaissances et comprendre le langage naturel.
- Raisonner et résoudre des problèmes.

Considérons quelques éléments physiques et logiques nécessaires pour une machine dite intelligente :

1. **Yeux, ou capteurs visuels**. Les caméras sont intégrées avec un système de vision qui permet d'extraire les caractéristiques liées à l'apparence des objets de l'environnement.

2. **Ouïes, ou capteurs auditifs**. Un capteur sonore (composé de un ou plusieurs microphones) permet d'acquérir le signal sonore. Dans une première étape, ce signal est segmenté par des méthodes de séparation de sources ; ces sources peuvent être localisées en orientation autour du capteur. Ensuite chaque source peut êre identifiée : voix d'un Homme interagissant avec le robot, musique... des outils de reconnaissance de la parole peuvent être intégrés. Une des contributions les plus avancées de l'Intelligence Artificielle est la possibilité de reconnaître le langage naturel avec lequel s'expriment les êtres humains ; cette reconnaissance permet (ou permettra) à toute personne d'accéder au contrôle de machines intelligentes.
3. **Autres sens, autres capteurs**. Même si vision et audition sont les sens privilégiés par l'Homme, le toucher (capteur haptique, mesure de force...), l'odorat (capteurs chimiques...) peuvent aussi être considérés.
4. **Mémoire**. Il s'agit d'une grande base de données dans laquelle on stocke faits et informations.
5. **Cerveau**. C'est le module décisionnel du système, chargé des missions suivantes :
   - Analyse et interprétation des données sensorielles à partir des connaissances stockées dans la mémoire.
   - Raisonnement, qui emploient l'information de la base de données pour obtenir davantage d'informations.
   - Procédures de planification, qui construisent des plans pour satisfaire les missions confiées au robot.
   - Apprentissage, donc génération de nouvelles connaissances, à partir de l'expérience, de démonstrations faites par des tuteurs...
6. **Organes Moteurs**, qui vont permettre à la machine d'interagir avec l'environnement ou avec d'autres entités douées d'intelligence (autres robots, humains), avec lesquels elle exécute les plans générés pour satisfaire une mission donnée. Ces organes peuvent aussi permettre à la machine d'exprimer des émotions : citons à ce sujet, les expérimentations faites par C.Breazeal avec le robot Kismet au MIT Media Lab.

L'état actuel de l'Intelligence Artificielle est très loin d'atteindre l'harmonie fonctionnelle excellente des êtres vivants Intelligents et, encore plus loin, de fabriquer des machines capables de se comporter sur de longues périodes, de manière "intelligente".

Bien sûr il existe déjà des machines spécialisées qui ont un certain degré d'intelligence : quel Homme n'a pas été vaincu par une machine électronique de jeux, comme pour les échecs ? Mais dès lors qu'un robot est confronté à des situations complexes, dès lors qu'il doit exécuter des actions dans ce monde, les résultats sont très décevants. Il convient d'indiquer que l'Homme ne souhaite pas toujours que les machines suivent le plus fidèlement possible la manière d'agir des êtres humains ; parfois, la machine est conçue comme un compromis entre son efficacité pour exécuter les tâches pour laquelle elle a été créée (nettoyage, logistique, exploration, compagnon de l'homme...), son coût (capteurs ou moyens de calcul dont elle a été dotée) et son acceptabilité (son apparence, son comportement compatibles avec les attentes de l'Homme).

4.3.2 Apprentissage automatique.

Pour des raisons pratiques et surtout pour faciliter la tâche du robot, nous allons définir la connaissance comme un ensemble de données ou d'informations stockées durant une phase d'apprentissage ; nous allons considérer donc que le robot *apprend* d'abord, puis qu'il *exploite* les représentations apprises. C'est une simplification par rapport à une capacité continue d'apprentissage (on parle en anglais de *open-ended learning process*).

Il faut donc définir ce concept d'apprentissage. Dans le monde vivant, c'est d'abord des changements relativement permanents dans le répertoire comportemental d'un sujet soumis à l'expérience, qui vont ensuite impliquer des changements neurophysiologiques.

Dans le cas d'ordinateurs, de robots, de machines quelconques..., l'Apprentissage Automatique est une branche de l'Intelligence Artificielle dont l'objectif est de développer des techniques qui permettent à ces artefacts construits par l'homme, d'apprendre de manière plus ou moins supervisée ou autonome. De manière plus concrète, il s'agit de créer des programmes capables de généraliser des comportements

à partir d'informations non structurées fournies sous forme d'exemples. C'est par conséquent, un processus d'induction de la connaissance. Pour beaucoup, le domaine de l'Apprentissage Automatique dépend des concepts et outils développés en Statistique, puisque les deux disciplines se basent sur l'analyse de données ; toutefois, l'Apprentissage Automatique met plus l'accent sur l'étude de la *Complexité Computationnelle* des problèmes ; en effet, beaucoup de problèmes sont classés NP-hard (*non-deterministic polynomial-time hard)*. C'est pourquoi une grande partie de la recherche effectuée en Apprentissage Automatique est focalisée sur la conception de solutions possibles pour résoudre ces problèmes.

Quelques systèmes d'Apprentissage Automatique essayent d'éliminer l'intervention d'un opérateur humain expert des processus d'analyse de données dans le domaine considéré, tandis que d'autres essayent d'établir un cadre de collaboration entre l'expert et l'ordinateur. Dans tous les cas, l'intervention humaine ne peut pas être remplacée dans sa totalité, puisque le concepteur du système doit spécifier (1) les représentations pour les connaissances à apprendre à partir des données (typiquement, des modèles) et (2) les méthodes de manipulation et de caractérisation de ces connaissances et données (liens entre modèles et observations).

Les différents algorithmes d'Apprentissage Automatique sont groupés dans une taxonomie en fonction de leurs sorties. Quelques types d'algorithmes sont :

- *Apprentissage Supervisé.* L'algorithme produit une fonction qui établit une correspondance entre les entrées et les sorties souhaitées du système. Un exemple de ce type d'algorithme est un classifieur : bayésien, K plus proches voisins, SVN, AdaBoost... Le système apprend comment étiqueter (classer) un vecteur d'entrées (observations) entre plusieurs catégories (classes). La base de connaissance du système est formée par des exemples d'étiquetages (couples observation-classe) généralement obtenus par annotation d'un expert humain.
- *Apprentissage Non Supervisé.* Ce processus exploite seulement un ensemble d'exemples (ou d'échantillons) donnés comme entrées au système, sans aucune information sur les catégories (ou classes) qui correspondent à ces exemples. Le système effectue lui-même la catégorisation des exemples en classes : on parle de *clustering.*
- *Apprentissage par Renforcement.* L'algorithme apprend en observant le monde qui l'entoure. Son information d'entrée est le feedback (ou rétro alimentation) du monde extérieur comme réponse à ses actions.
- *Apprentissage par Transduction.* Semblable à l'apprentissage supervisé, mais il ne construit pas de manière explicite une fonction. Il essaye de prédire les catégories des futurs exemples en se basant sur les exemples déjà connus et leurs catégories respectives et sur les exemples nouveaux fournis au système. On parle souvent de *open-ended learning*, ou méthode capable d'améliorer la base de connaissances dans le temps, donc d'apprendre en continu.
- *Apprentissage Multitâches.* Méthodes d'apprentissage qui utilisent des connaissances préalablement apprises par le système en vue de faire face à des problèmes semblables à ce qu'il a déjà vu.

L'analyse de la complexité et des performances des algorithmes d'apprentissage automatique est une branche de la statistique connue comme théorie computationnelle de l'apprentissage.

C'est presque impossible de décrire toutes les méthodes d'apprentissage qui existent : par ailleurs, ce n'est pas le but de notre travail. Contentons nous de mentionner deux méthodes essentielles :

- *Apprentissage bayésien*, par nature supervisé, fondé sur une représentation des incertitudes par des distributions de probabilités : connaissances apprises (probabilité a priori de percevoir telle caractéristique sur tel objet) et interprétation du monde à partir des données sensorielles (probabilité a posteriori de reconnaître tel objet à partir de telle caractéristique).
- *Apprentissage par clustering* qui vise à caractériser des classes d'objet à partir de la distribution des données et des relations entre variables. Citons l'algorithme K-moyennes ou *K-means*, qui cherche à découvrir les modes principaux d'une distribution, les "prototypes", les catégories principales, etc...

Nous allons nous centrer sur la méthode que nous allons exploiter ensuite dans notre système d'apprentissage et de reconnaissance d'objets depuis des données visuelles. Dans les sections suivantes nous présentons ce système : nous montrerons les différentes étapes réalisées afin de le rendre stable et robuste. On va commencer par illustrer notre algorithme en utilisant un exemple simpliste de reconnaissance d'objets uniquement caractérisés par la couleur ; bien que le pouvoir discriminant de cette propriété est faible, cela va nous permettre de présenter les bases des méthodes fondées sur *la Maximisation de l'Information Mutuelle* pour la reconnaissance active d'objets.

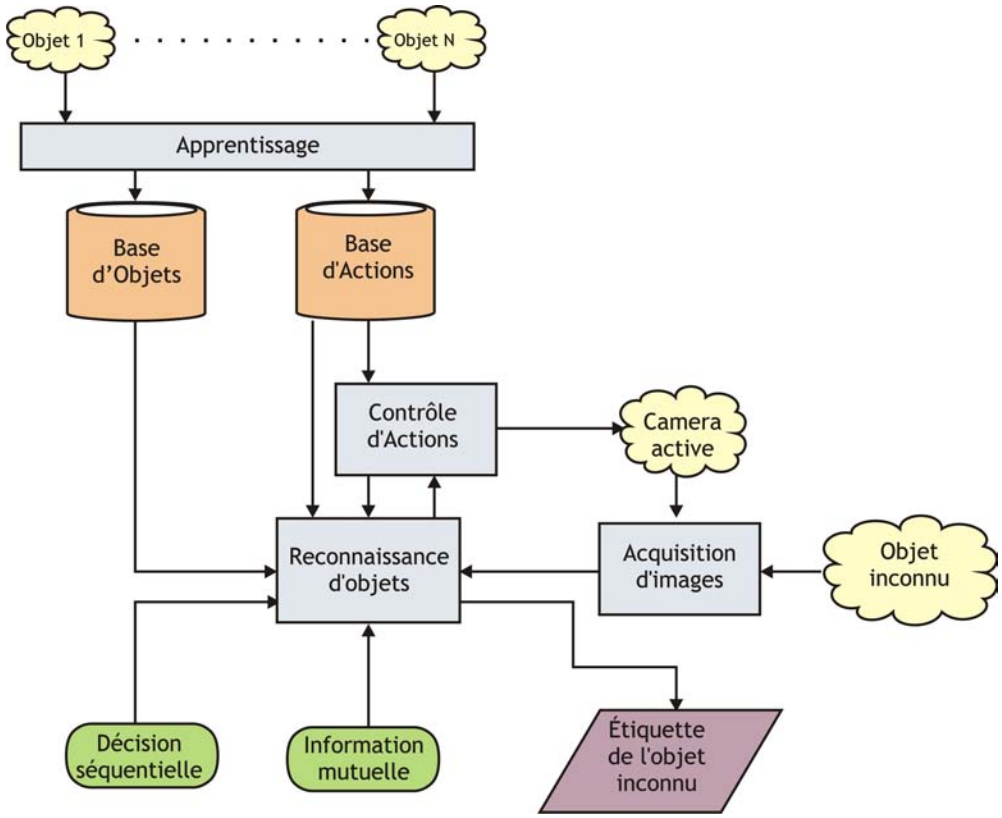


FIGURE 4.5 – Système actif de reconnaissance d'objets

4.4 Reconnaissance active d'objets représentés par la Couleur.

Un système de reconnaissance d'objets exploite des traitements pour l'extraction des caractéristiques sur chaque image ; elles seront associées à une classe et enregistrées dans une base de données pendant la phase d'apprentissage, et elles seront comparées avec celles déjà enregistrées pendant la phase de reconnaissance. Ces caractéristiques peuvent être la couleur, la taille, la forme, la texture, etc. Dans cette section nous allons fixer les bases de ce qui sera notre méthode de reconnaissance active, la Maximisation de l'Information Mutuelle, et nous allons montrer un exemple en utilisant seulement la couleur pour caractériser les objets à reconnaître. La figure 4.5 montre l'organigramme pour un système actif de reconnaissance d'objets.

Nous allons pour ce faire, utiliser un capteur actif, une caméra PTZ avec contrôle de ces différentes modalités ; il est possible de changer ses paramètres comme le zoom et l'orientation de l'axe optique. De manière plus générale, un capteur actif pourrait aussi être capable de se déplacer et de choisir sa position en fonction de l'information qu'il extrait de son environnement ; c'est la généralisation que nous proposerons dans la section suivante.

Une caméra permet d'extraire des informations discriminantes pour reconnaître un objet. C'est la raison pour laquelle, Denzler et Brown [40] [41] ont proposé d'exploiter une caméra comme un capteur actif avec une stratégie pour la sélection successive des configurations possibles du capteur afin de reconnaître des objets appris dans la base de données. De la même façon, nous allons exploiter cette méthode de Maximisation de l'Information Mutuelle pour contrôler le capteur actif et pour minimiser l'incertitude de la tâche de reconnaissance.

4.4.1 La Maximisation de l'Information Mutuelle

Soit un problème de reconnaissance sur Ω_k classes pour $k \in 1, n$. Durant une phase d'apprentissage, le système a mémorisé des probabilités a priori d'observations *obs*, sur chacun des objets, et pour chaque configuration $conf$ d'un capteur actif. Il a donc mémorisé les distributions $p(obs|obj, conf)$.

Pendant la phase de reconnaissance, le système analyse une scène supposée statique de manière séquentielle, en sélectionnant à chaque itération t, une action a_t qui lui permet d'avancer dans la tâche de reconnaissance. Soit x_t l'état estimé de reconnaissance sur les Ω_k classes pour $k \in 1, n$, après t itérations. Il s'agit en fait d'une distribution discrète des n probabilités d'avoir un objet de la classe i présent dans la scène analysée :

$$p(x_t) = (....p(obj_i|o_1, ..., o_{t-1})...)^T \text{ avec } \Sigma_{i=1}^{n} p(obj_i|o_1, ..., o_{t-1}) = 1 \tag{4.1}$$

Nous voulons estimer l'état vrai de la reconnaissance étant donné une observation o_t en chaque acquisition. La méthode est fondée sur une boucle qui comporte les étapes suivantes :

- *Choix d'une action* a_t ; il s'agit de mettre le capteur dans la configuration qui maximise l'information sur l'état x_t, donc le gain attendu après exécution de a_t. ;
- *Exécution de l'action* : on met le capteur dans la configuration optimale ;
- *Extraire l'observation* ; on extrait l'observation o_t depuis cette configuration ;
- *Mettre à jour l'état* ; on met à jour l'état x de la reconnaissance par la loi de Bayes pour calculer x_{t+1}.

Pour le choix de l'action, le gain maximal d'information sera donné par une action a_t qui optimise l'information mutuelle, ou la *transinformation* entre l'état courant et l'état qui sera obtenue après ajout de l'observation prédite o_t dans la configuration du capteur atteinte par a_t. Nous pouvons définir l'Information Mutuelle comme suit :

$$I(x_t; a_t|o_t) = H(x_t) - H(x_t|o_t, a_t) \tag{4.2}$$

où $H(\cdot)$ dénote l'entropie de la distribution de probabilité. Rappelons que l'entropie d'une distribution donne son incertitude. Une distribution uniforme, par exemple un état x_t avec toutes ces composantes $p(obj_i) = \frac{1}{n}$ aura une entropie maximale (toute observation ajoutera de l'information sur l'état de la reconnaissance), tandis que cette entropie sera minimale pour une distribution centrée sur une classe (on sait quel objet est dans la scène ; on ne va plus rien apprendre). Sur une distribution continue X, l'entropie est

$$H(x_t) = -\int_{x_t} p(x_t) \log p(x_t) dx_t \tag{4.3}$$

et

$$I(x_t; a_t|o_t) = \int_{x_t} \int_{o_t} p(x_t) p(o_t|x_t, a_t) log \frac{p(o_t|x_t, a_t)}{p(o_t, a_t)} do_t dx_t \tag{4.4}$$

Une action optimale a_t^* qui maximise l'information mutuelle, est donc donnée par :

$$a_t^* = max_{a_t} I(x_t; a_t|o_t) \tag{4.5}$$

Après exécution de cette action et extraction de l'observation o_t depuis la nouvelle configuration du capteur, on applique la règle de Bayes pour mettre à jour l'état a posteriori de la reconnaissance en fonction des connaissances a priori sur les observations, apprises pendant la phase d'apprentissage :

$$p(x_{t+1}) = p(x_t|o_1, ..., o_{t-1}, o_t) = \frac{p(o_t|x_t, a_t) p(x_t)}{p(o_t|a_t)} \tag{4.6}$$

dans laquelle le terme de normalisation $p(o_t|a_t)$ est la probabilité d'obtenir l'observation o_t pour tous les états possibles, donc pour tous les objets possibles.

Comme nous l'avons déjà mentionné, la procédure de reconnaissance d'objets se fait en deux étapes : l'apprentissage et la reconnaissance proprement dit. Dans les paragraphes qui suivent nous allons montrer comment se réalise chacune de ces étapes en utilisant la méthode d'information mutuelle dans un exemple simple.

FIGURE 4.6 – Quatre des huit objets utilisés dans cette validation simple de la méthode de Maximisation de l'Information Mutuelle

4.4.2 Apprentissage d'objets par la couleur

Pendant cette étape on crée une base de données, dans laquelle chaque objet qui a été appris par plusieurs vues (typiquement plusieurs centaines), est représenté par un ensemble de fonctions de densité de probabilité conditionnelles (PDF :probability density functions). Ces fonctions de probabilité représentent donc les probabilités d'obtenir une information donnée *obs* depuis différentes configurations du capteur actif $conf$ sur les différents objets obj de la base de données. Donc ce qui est appris, ce sont les distributions $p(obs|conf, obj)$; par exemple pour 100 configurations du capteur et 8 classes, nous aurons 800 distributions de ce type.

Au moment de la reconnaissance, une action va correspondre à mettre le capteur dans une des configurations $conf$; donc $conf$ et a_t sont équivalents. Dans cet exemple simplifié, nous avons considéré un ensemble de trois différentes actions : contrôler l'Orientation θ, l'Inclinaison ϕ et le Zoom (ou la focale) f de notre caméra active. Comme chaque paramètre peut prendre différentes valeurs on peut diviser chaque action configurable en un ensemble de valeurs discrètes. Chaque action a_t est donc définie comme :

$$a_t = (\theta_{k,t}, \phi_{k,t}, f_{k,t})^T \tag{4.7}$$

où $\theta_k, k \in 1, n_p$ est la valeur de l'orientation, $\phi_k, k \in 1, n_t$ est la valeur de l'inclinaison et $f_k, k \in 1, n_z$ est la valeur pour le zoom, avec n_θ, n_ϕ, n_f, nombres des valeurs discrètes pour les configurations possibles de l'orientation, l'inclinaison et le zoom respectivement.

Nous pouvons obtenir différentes caractéristiques de nos objets mais nous allons nous contenter ici de la couleur ; avec cette caractéristique nous allons construire une base de données pour l'ensemble des objets montrés en figure 4.6. La couleur a été modélisée par une fonction de densité de probabilité gaussienne. Ici nous avons obtenu trois différentes fonctions de densité de probabilité, une pour chaque Couleur RGB (voir figure 4.7). Ces distributions gaussiennes prennent en compte des variations d'illumination pendant la phase d'apprentissage.

L'équation 4.8 représente le vecteur des composantes RGB moyennes. Nous utilisons ces valeurs pour caractériser un objet dans la base de données pour une configuration du capteur actif donnée. Afin d'obtenir les paramètres de la distribution en prenant en compte des variations d'illumination, nous calculons la moyenne et la variance pour plusieurs observations du même objet dans une même configuration du capteur, mais en faisant varier l'illumination sur la scène.

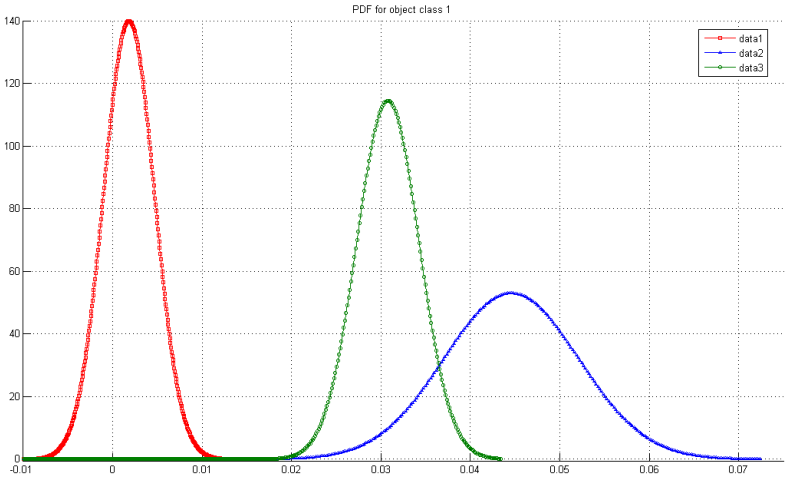


FIGURE 4.7 – PDF de la couleur pour un objet de la classe 1

$$I_p = \begin{pmatrix} I_r(p) \\ I_g(p) \\ I_b(p) \end{pmatrix} = \frac{1}{n} \sum_{i=0}^{n-1} \frac{1}{I_i} \begin{pmatrix} I_r(i) \\ I_g(i) \\ I_b(i) \end{pmatrix} \tag{4.8}$$

Dans l'équation 4.8, I_p représente la couleur moyenne en chaque image. I_i represente l'intensité pour chaque pixel de l'image et n est la quantité totale de pixels dans l'image. Avec ces valeurs nous pouvons calculer la probabilité d'observer la couleur d'un objet lorsque le capteur a été placé dans une configuration déterminée.

Nous avons utilisé 8 objets différents avec des propriétés similaires. Nous montrons quatre des huit objets utilisés en figure 4.6. Pour chaque objet on a obtenu un ensemble d'images différentes. Nous avons utilisé 30 valeurs différentes pour l'orientation, 5 valeurs pour l'inclinaison et 3 valeurs pour le zoom. Au total nous avons donc pris 450 images différentes pour chaque position de l'objet. Le processus a été répété 8 fois à chaque fois pour une position différente de l'objet devant le capteur : dans cette expérimentation simple, l'objet est posé sur une platine en azimut, afin de présenter ses différentes faces au capteur.

4.4.3 Reconnaissance d'objets par la couleur

Dans cette section nous allons montrer l'étape de reconnaissance en utilisant la Maximisation de l'Information Mutuelle pour décider des déplacements du capteur, et la Couleur comme caractéristique observée sur l'objet à reconnaître. L'objectif principal de cette section sera d'illustrer l'utilisation d'un système actif de reconnaissance.

Lorsqu'on présente un objet inconnu k au système de reconnaissance, muni de sa base de connaissances sur les objets appris, nous entamons un processus séquentiel. Au début on suppose des probabilités P_k égales pour toutes les classes d'objets apprises dans notre base de données.

Commençons par instancier les expressions sur l'entropie et l'information mutuelle, pour nos distributions discrètes. Tout d'abord, l'information mutuelle est calculée comme :

$$I_1(\Omega, c|a) = \sum_{k=1}^{K} e_k(a) P_k \tag{4.9}$$

où l'entropie e_k est définie de la manière suivante :

$$e_k(a) = \sum_{c_i} P(c_i|\Omega_k, a) \log \frac{P(c_i|\Omega_k, a)}{P(c_i|a)} \tag{4.10}$$

FIGURE 4.8 – Objet présenté au système : objet 1

Par la Maximisation de l'Information Mutuelle, on obtient l'action, ou la configuration du capteur, qui optimise le gain d'informations à partir de l'état courant de la reconnaissance et l'observation prédite en cette configuration. L'action optimale a_1^* doit donc maximiser l'information mutuelle :

$$a_1^* = \max_a I_1(\Omega, c|a) \tag{4.11}$$

Le système exécute cette action et met à jour les probabilités P_k pour chaque possible classe, en fortifiant les probabilités des classes pour lesquelles l'observation courante correspond à l'observation apprise sur les objets de cette classe pour cette configuration du capteur pendant la phase d'apprentissage, et en diminuant les probabilités pour les classes où ces observations ne sont pas similaires.

$$P_k = \frac{P(c_0|\Omega_k, a_1)}{P(c_0|a_1)} \tag{4.12}$$

Cette procédure est itérée d'une façon séquentielle jusqu'à ce que la probabilité de la classe la plus probable excède un seuil donné.

Au moment où on montre un objet au système il commence à faire le processus de reconnaissance en changeant les modalités du capteur afin d'obtenir le meilleur appariement possible avec les objets de sa base de données. Par exemple si nous présentons au système, l'objet de la figure 4.8 le résultat est présenté en figure 4.9 sous la forme de l'évolution de la distribution x_t, donnée par les probabilités de présence d'un objet de chaque classe apprise dans la scène analysée, en fonction des itérations. On observe qu'en seulement trois actions, le système a reconnu l'objet qui lui était présenté au départ : initialement, pour $t = 1$, les classes sont équiprobables ; dès l'intégration de la première observation o_1 acquise depuis la configuration initiale du capteur, la probabilité de présence d'un objet de la classe 1 est à 0,85, mais il existe une possibilité d'ambiguïté avec la classe 8 ; une action est sélectionnée, le capteur mis dans la configuration correspondante, l'observation o_2 est extraite, et l'état mis à jour par la règle de Bayes. A $t = 3$, il n'y a plus d'ambiguïté : le système est sûr qu'un objet de la classe 1 est devant le capteur.

Mais, qu'est ce qui se passe lorsqu'on présente à notre système, un objet qu'il n'a pas appris ou qui est différent de celui avec lequel il a été entraîné ? Si maintenant nous présentons au système l'objet de la figure 4.10, qui est l'objet 1 altéré, donc avec des attributs de couleur perturbés, nous avons le résultat de la reconnaissance en figure 4.12

Dans l'évolution de l'état x_t montrée en figure 4.12, on peut observer que le système oscille entre deux classes pour étiqueter l'objet qui lui est présenté : la réponse correcte, la classe 1 auquel l'objet présenté appartient effectivement, et la classe 8 dont les objets ont des caractéristiques similaires à celles de l'objet 1 occulté. On observe que le système donne une grande probabilité à la classe 1 au départ pour finir à incliner en faveur de la classe 8. Nous avons exécuté ce test plusieurs fois et approximativement, dans 50% des cas, l'objet fut reconnu comme appartenant à la classe 8. C'est important de dire que le système oscille seulement entre la classe 1 et la classe 8.

Donc, comme nous avons pu le constater, l'utilisation d'une seule caractéristique ne suffit pas pour réaliser la reconnaissance d'un objet de manière robuste. En plus, dans nos environnements très monochromatiques d'intérieur, l'utilisation d'une caractéristique aussi peu discriminante que la couleur n'est pas suffisante pour décrire un objet dans une scène. C'est la raison pour laquelle il a été nécessaire de chercher d'autres propriétés des objets qui nous permettent de bien décrire une classe d'objets et par conséquence, de reconnaître sans problème des objets appartenant aux classes apprises. Dans les sections

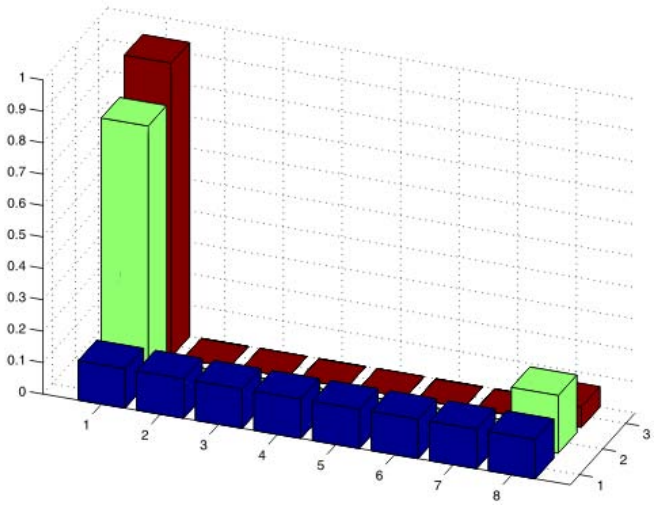


FIGURE 4.9 – Résultat de la reconnaissance sur l'objet 1

FIGURE 4.10 – Objet 1 partiellement occulté par un objet de couleur jaune.

FIGURE 4.11 – Objet 8 avec PDF de couleur similaire à l'objet 1 occulté.

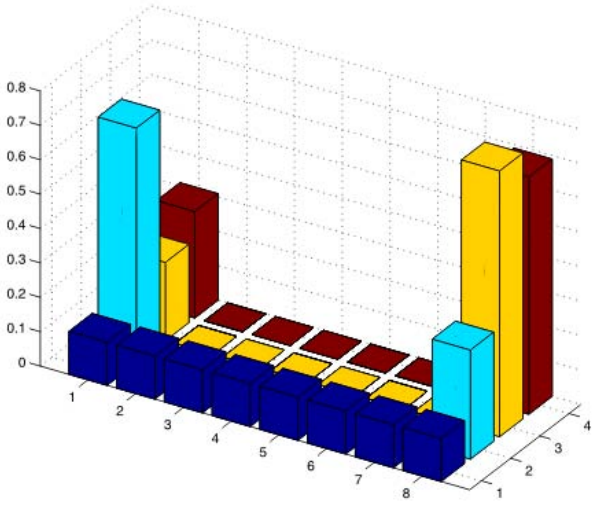


FIGURE 4.12 – Reconnaissance de l'objet 1 occulté : ambiguïté avec l'objet 8.

FIGURE 4.13 – Image à segmenter

FIGURE 4.14 – Image dans laquelle le ciel et le chemin ont été segmentés.

suivantes nous allons aborder cette thématique en ajoutant des caractéristiques, comme l'extraction de contours, afin d'apprendre des representations plus riches des objets, et donc, une reconnaissance plus robuste. Nous allons aussi considérer des configurations de la caméra, plus adaptées à notre contexte robotique.

4.5 Extraction d'un ensemble de caractéristiques.

Nous décrivons dans cette section, notre méthode de reconnaissance active d'objets 3D à partir d'un ensemble de caractéristiques, principalement liées à l'apparence visuelle de ces objets. Nous présentons les méthodes mises en oeuvre pour l'extraction des caractéristiques choisies, puis les phases d'apprentissage et de reconnaissance. Comme nous avons pu le constater dans les images utilisées pour réaliser le test précédemment citée il n'y avait pas d'arrière-plan. Mais, dans un contexte réaliste de reconnaissance, pour extraire des caractéristiques à partir d'une vue d'un objet, le premier problème concerne la segmentation : comment isoler l'objet du fond dans l'image ?

4.5.1 Segmentation de la région d'intérêt

La segmentation d'une scène est aussi un thème très abordé par les chercheurs dans le domaine de la vision par ordinateur. L'utilité de réaliser une segmentation avant d'analyser une scène est qu'on peut isoler les éléments qui forment la dite scène, comme en figure 4.14 dans laquelle on peut observer différentes zones isolées en fonction de la couleur de ses pixels, dans cet exemple on a segmenté les régions correspondant au ciel et au chemin. Citons la contribution de G.Aviña [14] qui a utilisé la segmentation en couleur et en texture, pour réaliser l'identification des éléments qui se trouvent dans une scène naturelle, en particulier pour identifier la région *Chemin* qui pourra servir pour contrôler les mouvements d'une machine mobile autonome.

L'image segmentée de la figure 4.14 est un bon exemple de segmentation utilisant l'intensité de l'image, mais que se passerait-il si on voulait juste enlever la maison ou le chemin ? Afin qu'un système automatisé puisse le faire, il est nécessaire qu'il dispose de l'information a priori de ce qu'il doit prendre ou laisser dans cette image. Nous avons cherché une autre alternative ; dans notre cas nous allons nous servir des contours actifs afin d'arriver à notre objectif, segmenter un ou plusieurs *TRUCs* posés sur une table de couleur et texture uniforme.

Les méthodes fondées sur les contours actifs ou les *Snakes* ont été largement utilisées en vision par ordinateur pour différentes applications, comme la reconstruction 3D à partir d'images médicales, mais aussi pour la segmentation d'objets d'intérêt dans des images. Citons parmi les travaux qui utilisent les contours actifs pour la segmentation en imagerie médicale, ceux de Tauber *et al.* [143] qui font la segmentation de cavités cardiaques dans les images échographiques. Mais il y a bien d'autres travaux qui font usage des contours actifs comme celui de Wang et Ghosh [154] qui ont proposé une nouvelle géométrie pour le modèle de contours actifs. Le dit modèle fut implémenté en utilisant la méthode multi-échelle dite de *Level Set*. Nous avons aussi celui de Yuan *et al.* [167] qui ont proposé l'utilisation d'une nouvelle classe de force externe appelé SSEF pour *Simulated Static Electric Field*, définie par analogie avec des charges d'électricité statiques disposées en chaque point de contrôle suivi sur un contour : les forces de répulsion entre ces charges, permettent de conserver une distribution uniforme de ces points le long du contour pendant le déplacement du contour vers la silhouette de l'objet à segmenter.

FIGURE 4.15 – Initialisation du snake pour segmenter un objet dans une scène

Définissons donc un Snake. Un snake n'est pas autre chose qu'une courbe spline contrôlée par des forces qui sont présentes dans une image, mais aussi par des forces internes et externes qui délimitent l'action du Snake. Ces forces internes calculées à partir de la forme même du snake, servent à imposer des contraintes sur la souplesse de la courbe par morceaux. Les forces présentes dans l'image poussent le snake vers les discontinuités présentes dans l'image comme les contours ; les forces externes permettent d'emmener le snake près du minimum local souhaité.

En accord avec [71], on peut représenter la position d'un snake de façon paramétrique par $\mathbf{v}(s) = (x(s), y((s))$, où s est l'abscisse curviligne le long du contour. Alors, on peut écrire sa fonction d'énergie comme :

$$E_{Snake}^{*} = \int_0^1 E_{Snake}(\mathbf{v}(s))ds = \int_0^1 E_{int}(\mathbf{v}(s)) + E_{image}(\mathbf{v}(s)) + E_{ext}(\mathbf{v}(s))ds \tag{4.13}$$

où E_{int} représente l'énergie interne qui empêche le recourbement de la spline, E_{image} est l'énergie due à l'image et E_{ext} représente la contrainte due aux forces externes.

Dans les images 4.15 et 4.16 on peut observer la phase d'initialisation et le résultat final de l'utilisation des contours actifs pour la segmentation d'un objet dans une scène. La figure 4.16 montre le résultat, dans lequel l'objet doit être extrait du fond ; on peut observer que la segmentation n'est pas parfaite. Malgré tout cette segmentation peut être suffisante pour segmenter la région d'intérêt dans l'image et ensuite, pour réaliser les traitements nécessaires uniquement sur cette région selon les besoins. Mais ce que nous cherchons est d'avoir l'objet le mieux segmenté possible. Pour cette raison nous calculons d'abord l'image de contours de la scène (voir figure 4.17) avant d'appliquer l'algorithme de contours actifs.

Les images qu'on peut voir dans les figures 4.18 et 4.19, présentent l'initialisation et le résultat final de l'extraction du snake sur l'image de contours de la scène montrée sur la figure 4.15.

Nous pouvons constater que le résultat obtenu sur l'image des contours construite avant d'extraire le snake, est meilleur que juste l'extraction du snake sur l'image brute comme cela était proposé auparavant. Comme ce qui nous intéresse est la segmentation de l'objet de la scène, on peut voir dans la figure 4.20 l'objet tout seul dans l'image. Maintenant les processus que nous allons appliquer pour apprendre ou reconnaître des objets depuis des images, seront plus efficaces puisqu'on a enlevé le bruit qui pourrait provenir des informations présentes dans l'arrière-plan de la scène.

L'utilisation d'image des contours comme entrée de l'algorithme du Snake présente un autre avantage : le temps d'exécution de l'algorithme. La table 4.1 montre les temps d'exécution ainsi que le nombre d'itérations pour chacune des procédures.

L'algorithme de contours actifs que nous avons utilisé pour développer ces exemples, est celui développé par Xu *et al.* [160] [162] [161] [163]. Xu *et al.* ont proposé l'utilisation d'une force externe nouvelle appelée

FIGURE 4.16 – Segmentation finale de la boîte dans la scène.

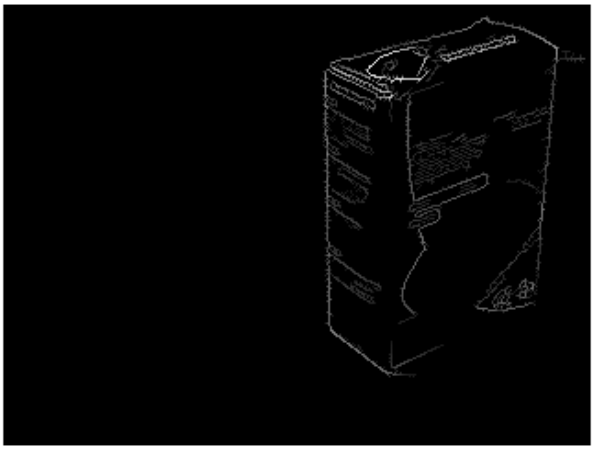


FIGURE 4.17 – Image des contours de la scène

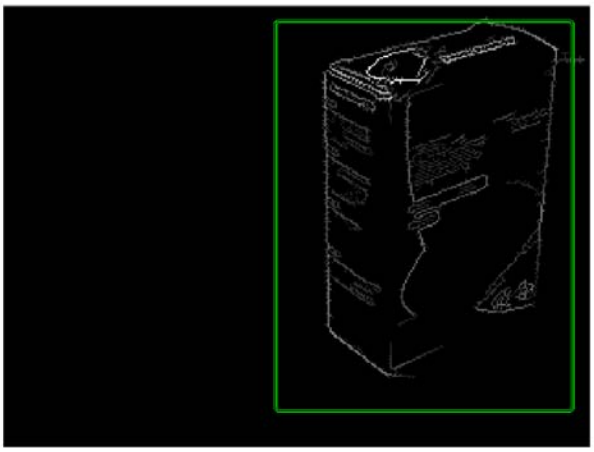


FIGURE 4.18 – Initialisation de l'algorithme de contours actifs, sur l'image des contours.

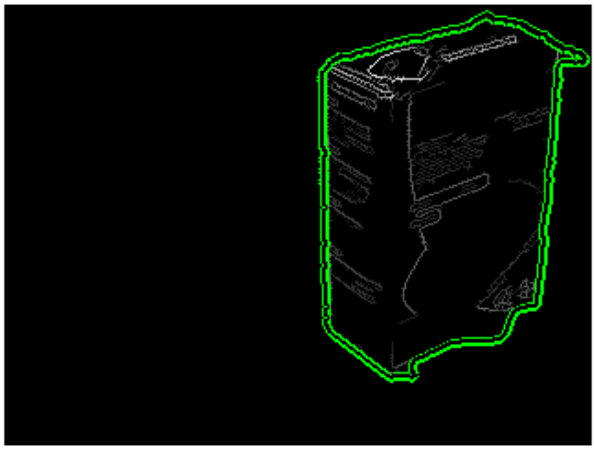


FIGURE 4.19 – Résultat de la segmentation par un snake, à partir d'une image des contours.

FIGURE 4.20 – L'objet tout seul après la segmentation.

TABLE 4.1 – Table comparative des temps d'exécution (en secondes) de l'algorithme Snake.

Cas	Itérations	Temps
Sans Contours	1400	214.75
Avec Contours	300	9.46

GVF pour *Gradient Vector Flow*. Cette force est calculée comme une diffusion du vecteur de gradient à partir des niveaux de gris ou de la carte binaire des contours extraits de l'image. L'avantage de cette méthode est que les forces externes peuvent emmener le contour actif vers des régions concaves.

En plus nous avons complété l'algorithme présenté en [160] puisque cet algorithme ne peut pas gérer les images avec plusieurs objets et les segmenter d'une façon appropriée. Etant donné cet inconvénient, nous avons utilisé le travail de Xingfei et Jie [159] qui font une modification de la procédure présentée par Xu et Jie pour détecter plusieurs points de divergence et pour incorporer une méthode afin de séparer les contours de régions disjointes dans l'image : ceci sera utile dans notre contexte si plusieurs objets sont posés sur la table, et que notre robot doit les détecter et reconnaître séparément.

Les figures 4.21, 4.22, 4.23, 4.24 et 4.25 présentent un résultat sur l'extraction de plusieurs composantes dans une image, ici des pièces posées sur une table : on peut constater que les 4 pièces sont extraites. Les figures 4.26, 4.27, 4.28, 4.29 et 4.30 montrent un exemple en utilisant une image de synthèse. Dans ces images, à la différence de l'exemple précédent ou de l'image avec la boîte, la table sur laquelle l'objet est posé, a une texture non uniforme.

On peut observer (fig. 4.30) que la segmentation n'est pas parfaite mais c'est beaucoup mieux de perdre un peu d'information que de traiter toute l'image pour la reconnaissance, vu les inconvénients que

FIGURE 4.21 – Image à segmenter.

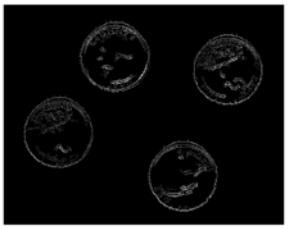


FIGURE 4.22 – Image des contours.

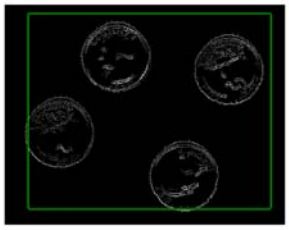


FIGURE 4.23 – Initialisation du snake.

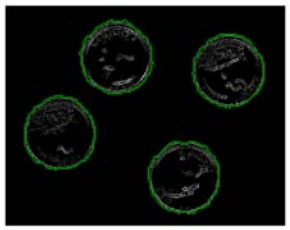


FIGURE 4.24 – Image finale de la segmentation.

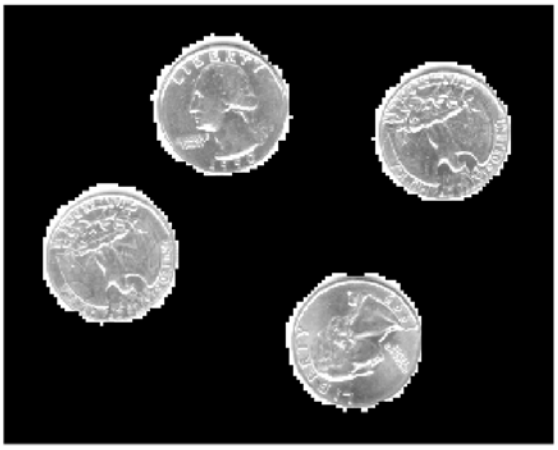


FIGURE 4.25 – Les objets après la segmentation.

FIGURE 4.26 – Image à segmenter.

FIGURE 4.27 – Image des contours.

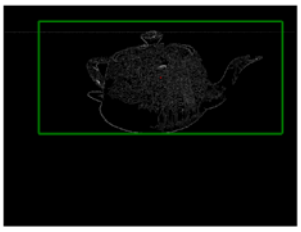


FIGURE 4.28 – Initialisation du snake.

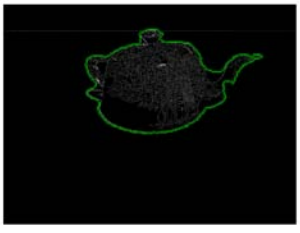


FIGURE 4.29 – Image finale de la segmentation.

FIGURE 4.30 – L'objet tout seul après la segmentation.

cela entraîne.

Une fois finie la segmentation on peut réaliser toutes les procédures nécessaires pour créer une base de données plus précise et plus fiable qu'en utilisant toute l'information de la scène. Cette procédure peut aussi détecter l'absence d'informations dans une image ; dans notre cas, le fait que rien n'est posé sur la table ; en ce cas l'image résultat de la segmentation sera toute noire, ou bien le snake va s'accrocher sur du bruit facile à filtrer.

Elimination des ombres

Les ombres qui se produisent au moment de prendre les images sont normalement indésirables puisqu'elles vont produire du bruit pendant les processus que nous voulons appliquer aux dites images ; cela sera surtout sensible pendant la phase de segmentation de l'objet présent sur l'image par la méthode des contours actifs. Alors on doit, soit apprendre à vivre avec elles, soit les enlever de l'image.

Cette procédure d'élimination des ombres a été abordée par plusieurs chercheurs parmi lesquels G.Aviña *et al.* [13] : ils font disparaître les ombres des images réelles. En prenant la même procédure nous avons enlevé les ombres des images exploitées ensuite pour faire la reconstruction et la reconnaissance des objets.

FIGURE 4.31 – Elimination des ombres : résultat sur une image de synthèse de la tasse.

Dans les images que nous montrons dans les figures 4.31 nous pouvons apprécier dans l'image de droite, ce qui reste de l'image de gauche après avoir enlevé l'ombre.

Dans tous nos traitements décrits ci-après, nous allons donc segmenter l'image en gardant seulement la partie contenant l'information utile pour notre analyse. Un exemple de cela est donné en figure 4.32 ; cette image est l'extraction de l'information utile sur l'image présentée en figure 4.20.

FIGURE 4.32 – Boîte délimitant l'objet.

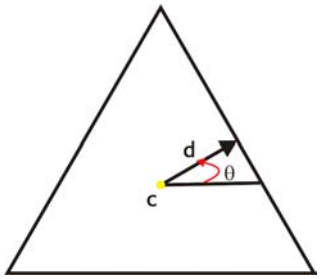


FIGURE 4.33 – Objet pour lequel on calcule la signature.

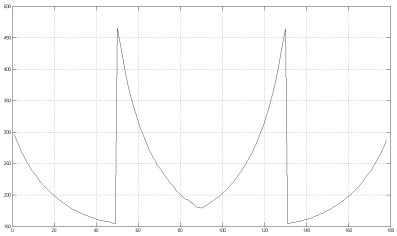


FIGURE 4.34 – *Shape signature* du triangle

4.5.2 Extraction des caractéristiques

Comme nous l'avons déjà dit en sections précédentes l'utilisation de la couleur ne suffit pas pour faire un système de reconnaissance d'objets qui soit robuste, et en conséquence, efficace. L'exploitatoin d'autres caractéristiques de l'objet s'est imposée. Dans les paragraphes qui suivent nous allons décrire les différentes caractéristiques que nous avons utilisées pour développer notre système de reconnaissance active d'objets.

Signature de la forme : Shape signature.

La signature de la forme donne une représentation de la limite de la région projection de l'objet dans l'image, exprimée par rapport au centre de gravité de cette région. La limite de la projection de l'objet est donnée par sa silhouette. Les signatures de la forme sont classiquement utilisées comme un mécanisme rapide d'indexation de directions pour la récupération de la forme. Un des travaux référence pour la *Shape signature* est celui de Fujimura et Sako [50], sauf qu'ils proposent une variante fondée sur la déformation de la forme. Ils encapsulent l'objet, normalement un polygone, dans un autre polygone et mesurent la déformation que doit subir le polygone extérieur pour atteindre le contour du polygone intérieur.

Dans notre cas, la signature d'un objet est calculée à partir de la mesure de la distance entre le centre de gravité et la silhouette de l'objet. Dans la figure 4.34 on peut observer la signature d'un triangle.

Nous obtenons la silhouette de l'objet par la segmentation fondée sur un contour actif. Nous pouvons ensuite directement extraire la *Shape signature*. La signature pour l'objet de la figure 4.32 est montrée dans la figure 4.35.

Puisqu'un objet sera appris à partir de plusieurs images, seulement une signature approximative est extraite de la silhouette de l'objet en chacune des vues ; elle peut être prélevée, avec un rayon régularisé

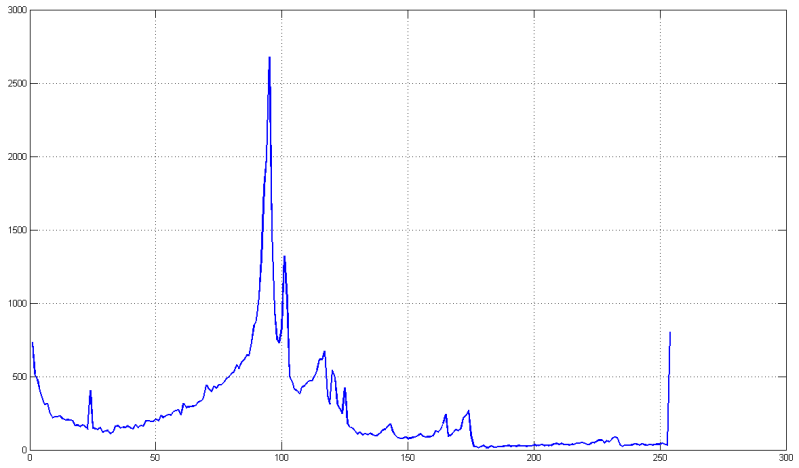


FIGURE 4.35 – Signature de l'objet de la figure 4.32

pour 0 deg, 10 deg, 20 deg ..., produisant une signature de l'objet sous la forme d'un vecteur de 36 éléments.

Signature des contours : Shape Context

L'autre descripteur que nous allons utiliser est celui de *Shape Context*. Selon Belongie *et al.* [22], le descripteur *Shape Context* est la distribution relative des points de contours exytraits de l'image, relativement à chaque point sur la forme. Dans notre cas, afin de limiter le temps de calcul, la distance et les histogrammes d'orientation sont établis seulement par rapport au centre de gravité de tous les points de contour à l'intérieur de la silhouette de l'objet. Nous prenons trois cellules (*bins*) pour le rayon et huit pour les orientations, produisant un descripteur *Shape Context* de 24 éléments.

La figure 4.36 est l'histogramme obtenu à partir de l'image de contours de la figure 4.17.

Points d'intérêt

L'utilisation des points d'intérêt comme descripteur d'une scène n'est pas du tout nouvelle, comme on peut observer dans les publications. Citons Nagao et Grimsom [103] qui ont présenté un algorithme pour la reconnaissance d'objets dans laquelle ils combinent des caractéristiques invariantes tant photométriques que géométriques ; ils utilisent la mise en correspondance entre les centroïdes de groupes de points d'intérêt. Pour leur partie Schmid *et al.*[124] montrèrent que les points d'intérêt sont géométriquement stables sous différentes transformations et ont un contenu élevé d'information. Chen et Bhanu [31] créent des patches surfaciques locaux en utilisant les point d'interêt obtenus à partir des images de profondeur des objets. Grabner et Bischof[56] suivent les trajectoires des points d'intérêt dans une séquence d'images et extraient la description d'objets 3D en utilisant cette information spatio-temporelle. On peut mentionner aussi D.Lowe [86] [87] [88], Harris [57], Bay [20] dont les travaux ont déjà été commentés en sections précédentes.

Nous allons utiliser les points de Harris et les SIFTs comme descripteurs de la scène. Les SIFTs définissent des patches pour décrire une zone de l'objet ; ils sont créés à partir de points d'intérêt détectés par Différences de Gaussiennes, dont les caractéristiques photométriques sont invariantes en échelle et orientation. Utiliser les descripteurs locaux attachés à des points d'intérêt va permettre de reconnaître un objet partiellement occulté, puisque le calcul de ce descripteur ne dépend que du voisinage local de chaque point.

Les figures 4.37 et 4.38 sont un exemple de l'extraction de points d'intérêt respectivement, des points de Harris (non orientés) et des points extraits par DOG, représentés par le descripteur SIFT (qui exploite l'orientation du gradient en chaque point).

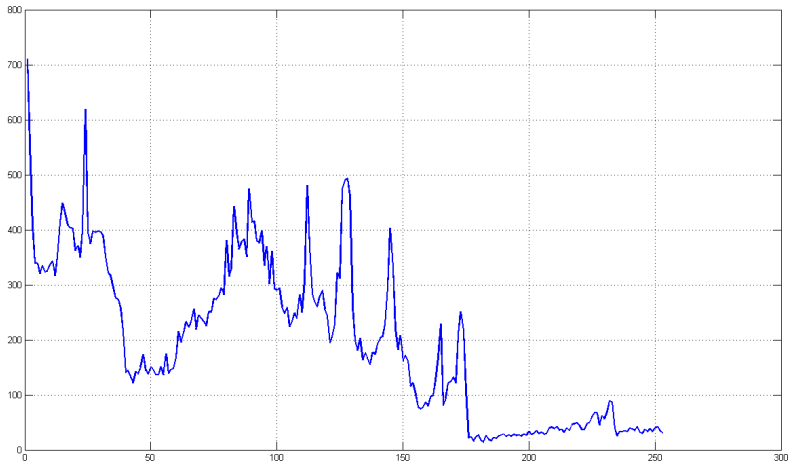


FIGURE 4.36 – Descripteur de type Shape Context de l'objet de la figure 4.32

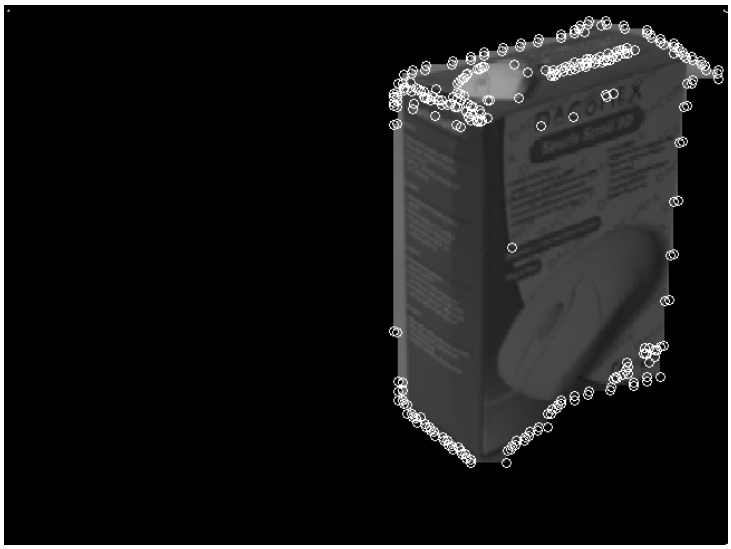


FIGURE 4.37 – Points de Harris extraits sur l'image de l'objet en figure 4.32

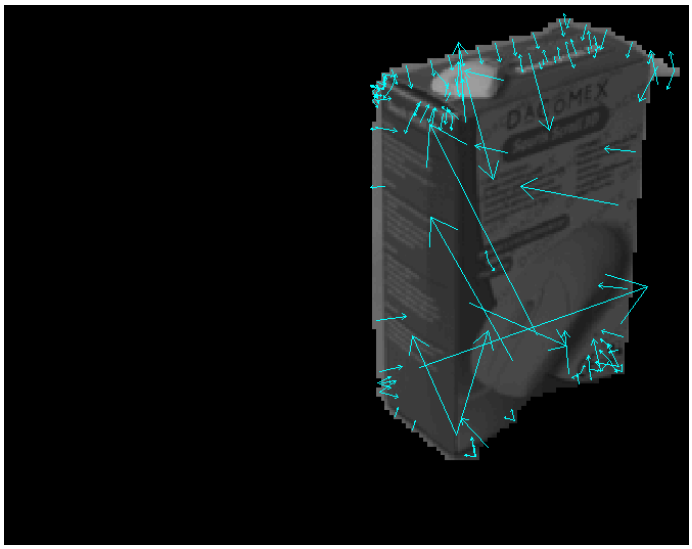


FIGURE 4.38 – Points SIFT extraits sur l'image de l'objet en figure 4.32

Densité sphérique

Les descripteurs mentionnés précédemment sont des descripteurs 2D, extraits d'images 2D uniquement. On ne peut pas dire qu'ils sont des descripteurs d'objets 3D ; ils caractérisent l'apparence de l'objet. Pour cette raison nous avons aussi proposé un descripteur d'objets 3D, la Densité Sphérique.

La densité sphérique est un descripteur qui utilise la distribution des points 3D d'un nuage de points par rapport à une sphère de rayon r. Le rayon r est calculé comme la distance la plus éloignée entre le centre de gravité du nuage et les points eux-mêmes, plus un paramètre δ. Ce descripteur a été utilisé par Oirrak *et al.* [105] qui, dans leurs travaux, l'ont défini comme la distribution des distances du rayon r mesurés entre le centre de gravité et les points du nuage. Ils en dérivent une probabilité ou densité par :

$$P(r) = n(r)/N \tag{4.14}$$

où $n(r)$ est le nombre de points qui ont r comme rayon sphérique et N le nombre total de points.

Les figures 4.39 et 4.40 montrent respectivement le nuage de points 3D pour une image 3D de l'objet Bunny [1] et son descripteur utilisant la densité sphérique. Dans la figure 4.41 on peut observer la même image 3D que dans la figure 4.39 mais cette fois ci après avoir réalisé une décimation de 50% des points. Le graphique résultat de son descripteur est présenté en figure 4.42. En regardant les figures 4.44 et 4.45, qui montrent les densités sphériques de Bunny mais cette fois ci avec une rotation de 35° et 60° respectivement, on peut constater que les descripteurs obtenus pour les deux modèles sont les mêmes.

On peut donc affirmer que le descripteur est invariable à la rotation et à l'échelle. Pour le rendre invariable à la quantité des points 3D présents dans le nuage de points, donc à la résolution de cette image qui dépend largement de la distance objet-capteur 3D, il suffit de décimer les nuages de points 3D acquis sur l'objet à un nombre donnée de points N_{min} (et de ne pas considérer les nuages qui ont moins que ce nombre de points).

4.6 Reconnaissance active d'objets représentés par un ensemble de caractéristiques.

Cette section présente l'évaluation de notre méthode. Nous utilisons 8 objets différents, ces objets sont montrés dans la figure 4.6. Les modèles 3D ont été obtenus du site web TurboSquid [3].

Pendant la phase d'apprentissage, le système acquiert un ensemble d'images. Ces images sont acquises en déplaçant (virtuellement en synthèse) notre capteur sur la surface d'une demi-sphère centrée sur le centre de gravité de l'objet. Pour apprendre nos objets, nous avons acquis autour de 320 images (environ 40 par objets) pour évaluer notre approche de la reconnaissance active en utilisant l'Information Mutuelle.

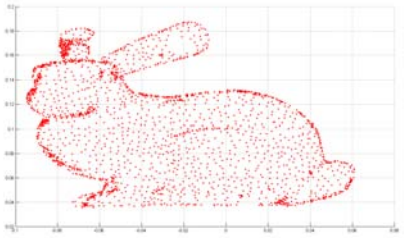


FIGURE 4.39 – Objet Bunny

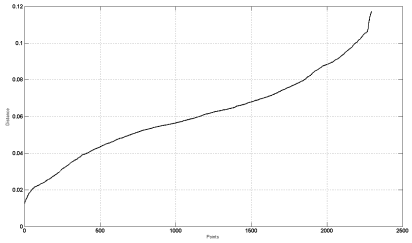


FIGURE 4.40 – Densité sphérique pour objet Bunny

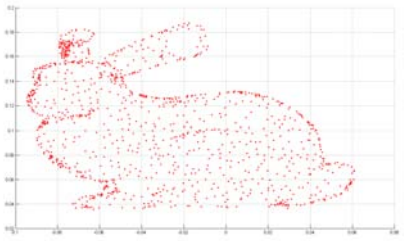


FIGURE 4.41 – Objet Bunny décimé (1500 points)

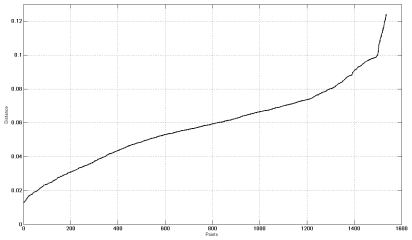


FIGURE 4.42 – D ensité sphérique pour l'objet Bunny décimé

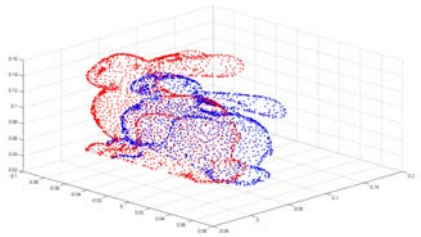


FIGURE 4.43 – Objet Bunny à 35 deg (blue) et 60 deg (rouge) par rapport à l'original

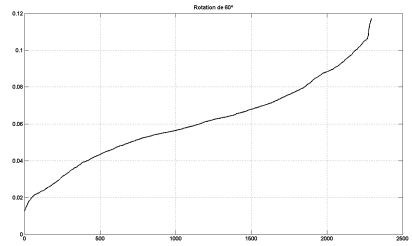


FIGURE 4.44 – Distances pour l'objet Bunny tourné de35°

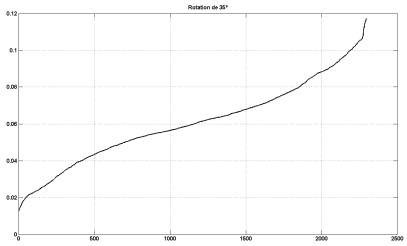


FIGURE 4.45 – Distances pour l'objet Bunny tourné à 60°

4.6.1 Phase de l'apprentissage des modèles des objets.

Dans la phase d'apprentissage, le système doit apprendre les différents objets à partir de leurs caractéristiques. Le résultat est une base de données des objets appris, base qui sera exploitée pour d'ultérieures phases de reconnaissance.

Le méthode d'apprentissage que nous allons utiliser, est identique à celle que nous avons montrée dans la section 4.4. elle vise à exploiter l'Information Mutuelle lors de la reconnaissance. Dans la section 4.4 nous avons montré un exemple du fonctionnement de cet algorithme en utilisant uniquement la couleur. Maintenant nous allons nous servir de tous les descripteurs cités dans les sections précédentes mais aussi de la couleur, comme caractéristiques qui vont décrire nos objets dans la base de données et qui, ensuite, seront exploitées pour les reconnaître.

Le concept d'Information Mutuelle a déjà été défini dans la section 4.4. Nous allons décrire la façon avec laquelle nous allons utiliser les différents descripteurs qui ont été mentionnés précédemment.

Etant donné que nous allons chercher à reconnaître des objets parmi l'ensemble des objets montrés dans la figure 4.46, il faut apprendre pour chaque objet, la probabilité $p(o_t|a_t)$, d'observer une caractéristique o_t de l'objet lorsque le capteur se trouve dans une configuration donnée a_t, autrement dit :

$$p(o_t|a_t) = \int_{x_t} p(o_t|x_t, a_t)p(x_t)dx_t \tag{4.15}$$

Le processus de création de la base de données pour un objet donné peut se résumer comme suit :

1. Prendre une image de l'objet isolé.
2. Obtenir les vecteurs de caractéristiques, un pour chaque caractéristique.
3. À partir des vecteurs trouvés, mettre à jour moyenne et variance de chaque caractéristique.
4. Garder les vecteurs caractéristiques, leurs moyennes et variances et le numéro de l'action.
5. Si il n'y a pas plus d'image sur l'objet, aller en (6), dans le cas contraire aller en (1)
6. Calculer les valeurs marginales des moyennes et variances

Les images vont être acquises en plaçant le capteur dans les différentes positions obtenues lors de la discrétisation d'une demi-sphère centrée sur l'objet. Cela a été déjà présenté dans le chapitre 3. Cette demi-sphère sera placée autour de l'objet de la façon suivante :

- On acquiert les images gauche et droite de la scène ;
- Par stéréo corrélation on récupère les points 3d de la scène ; on segmente l'objet d'intérêt, celui que l'on veut apprendre ;
- Avec les points 3D segmenté sur l'objet, on calcule le centre de gravité de l'objet ;
- On applique la transformation nécessaire afin que le centre de gravité de la scène soit le centre de la demi-sphère.

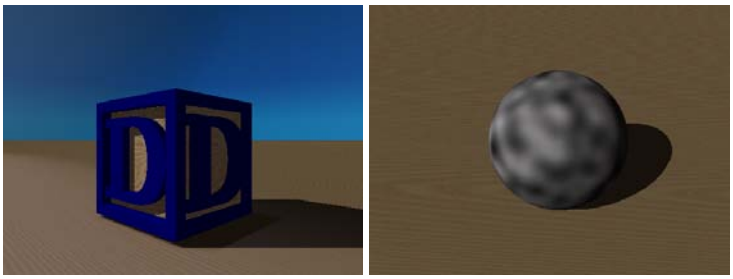
Les images montrées dans la figure 4.47 sont quelques unes des 320 images acquises pour un objet.

Actions

Les actions que va pouvoir exécuter le capteur, sont les mouvements de la caméra vers les différentes positions dans lesquelles il a été placé en phase d'apprentissage. Pour chaque position sur la demi-sphère définie autour de l'objet, pour tenir compte des occultations, on va calculer diviser l'image en 16 sections différentes sur lesquelles sont calculées les caractéristiques. Finalement nous aurons donc une quantité d'actions égale à 16 fois le nombre de positions sur la demi-sphère. Comme nous avons discretisé la demi-sphere en 320 points donc nous avons 16 fois 320, soit 5120 actions différentes. c'est-à-dire :

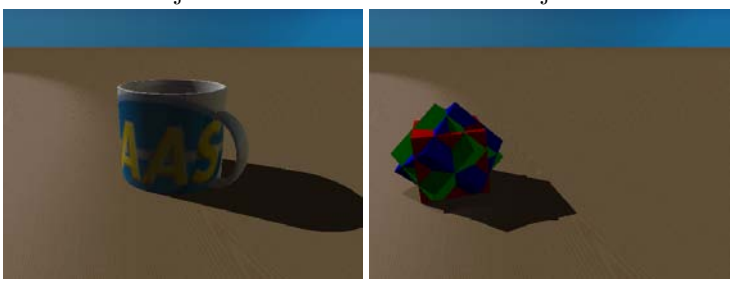
$$a_t = (s_{k,t}, i_{k,t})^T \tag{4.16}$$

où $s_k, k \in (1, n_s)$ est un indice pour la position du capteur sur la demi-sphère, $i_k, k \in (1, n_i)$ est un indice pour une section de l'image, avec n_s nombre des positions discrétisées sur la demi-sphère, et n_i nombre de sections considérées dans chaque image.



Objet 1 Objet 2

Objet 3 Objet 4



Objet 5 Objet 6

Objet 7 Objet 8

FIGURE 4.46 – Objets appris dans notre base de donnée exploitée pour valider nos travaux sur la reconnaissance active avec des images de synthèse.

FIGURE 4.47 – Quelques images de l'objet 5 exploitées en phase d'apprentissage.

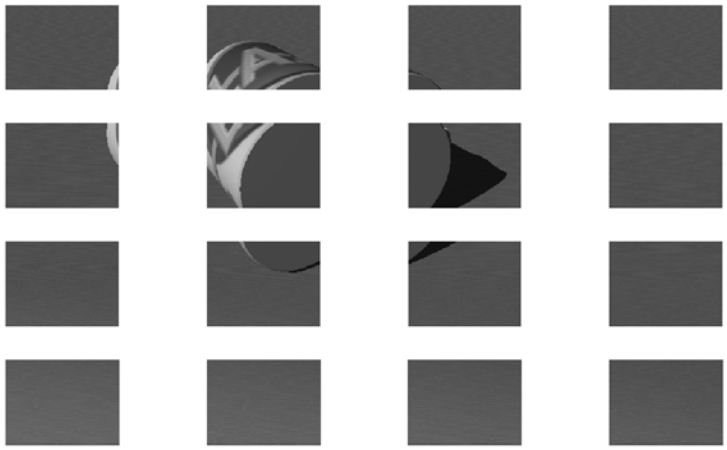


FIGURE 4.48 – Division d'une image en 16 sections

Division en sections

Avec la finalité de pouvoir reconnaître un objet même s'il est partiellement caché ou s'il se trouve dans une position différente de celle dans laquelle le système l'a appris, il est nécessaire d'utiliser des descripteurs correspondants à une vue partielle, donc à des régions plus petites de l'image.

On pourrait se contenter de diviser une image en seize sections de mêmes surfaces, comme cela est montré en figure 4.48. Mais l'intérêt est de d'abord segmenté l'objet dans l'image et de diviser la région segmentée ; le principal avantage de diviser après segmentation, est quelquesoit la taille apparente de l'objet (donc la distance objet-caméra), on a le même nombre de sections et, en conséquence, on peut analyser les mêmes régions depuis des images acquises depuis des points de vue proches. La figure 4.49 montre la division obtenue pour le même point de vue de la tasse, mais après avoir réalisé la segmentation de l'objet.

Le différentes mesures de caractéristiques vont être realisées sur chacune des sections de l'image afin d'obtenir un vecteur pour chacune d'elles.

Constructions des représentations des objets.

Une fois que nous avons acquis une image depuis une position donnée, on extrait les vecteurs de caractéristiques. Les caractéristiques que nous avons retenues, sont donc : la couleur, la Shape Signature, le Shape context, les points de harris, les points SIFT et la densité sphèrique. Ces caractéristiques ont été déjà présentées aux sections précédents.

Nous avons 6 caractéristiques différentes avec lesquelles on espère obtenir un système de reconnaissance plus robuste que celui qui n'exploite rien d'autre que la couleur. La section suivante présentera les performances de cet algorithme. Pour l'instant nous allons montrer la façon de construire la représentation d'un objet de notre base de donnée. La procédure est décrite dans la suite :

1. Acquisition de l'image i en la position s_i
2. Segmentation de la scène pour extraire chaque objet ou amas d'objets posés sur la table
3. Division de l'image résultat.
4. Pour chaque sous-image $j_{i,n}$ obtenir :
   - **A** les histogrammes en couleur
   - **B** Les points de Harris
   - **C** Les points SIFT
   - **D** En utilisant l'image des contours obtenir
     - La Shape signature et

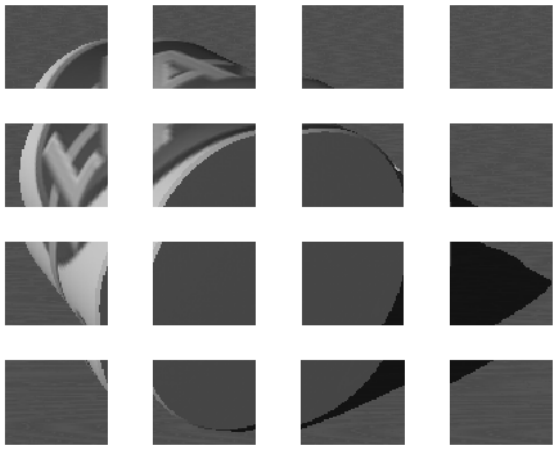


FIGURE 4.49 – Division de l'image montrée dans la fig. 4.48 après segmentation.

– le Shape context

E Finalement, en utlisant les points 3D provenant de la stéréo corrélation (si elle est disponible), extraire la densité sphérique.

5. Générer les vecteurs de caractéristiques, et obtenir la moyenne et la variance de ces vecteurs.

Dans les figures suivantes (figure 4.50) nous illustrons ce calcul des caractéristiques pour une image correspondant à l'objet 5 dans notre base de connaissances, placée dans une position exploitée lors de l'apprentissage.

On doit répéter le processus pour chaque point de vue différent et pour chaque objet afin d'obtenir la base de connaissance qui va nous servir pour réaliser la reconnaissance active d'objets.

4.6.2 Phase de la reconnaissance d'objets.

Maintenant nous allons décrire la phase de reconnaissance proprement dit. Dans cette phase le système d'abord va prendre une image d'une scène, dans laquelle un ou plusieurs objets peuvent être présents dans des positions quelconques ; il segmente les objets ou amas d'objets ; ensuite il extrait les vecteurs de caractéristiques afin de pouvoir les comparer avec les vecteurs stockés dans la base de données que le système a construite pendant la phase d'apprentissage.

Au début tous les objets observés ont la même probabilité d'appartenir à une des classes apprises. Mais cette probabilité change au fur et à mesure que le système évolue en cherchant la classe qui explique le mieux les caractéristiques de l'objet ou des objets observés. Dans la figure 4.51 (extraite des travaux de Denzler et Brown [40]), on peut observer comment évolue la probabilité qu'un objet soit présent dans la scène perçue, en fonction des différentes actions que le système a sélectionnées jusqu'à parvenir à la décision souhaitée, à savoir la reconnaissance de l'objet présent dans la scène observée.

La manière avec laquelle le système met à jour ces probabilités est décrite ci-dessous :

1. Prendre une image depuis la position initiale du capteur.
2. Obtenir les vecteurs de caractéristiques en suivant la méthode présentée en section 4.6.1
3. Comparer ces vecteurs avec ceux de la base de données ; sélectionner la classe la plus proche.
4. Sélectionner l'action suivante du système en fonction de l'état courant de la reconnaissance.
5. Exécuter cette action et
   (a) aller en (1) si l'action était un déplacement du capteur ;
   (b) aller en (2) si l'action concernait le choix des traitements (ou des paramètres) appliqués sur l'image.

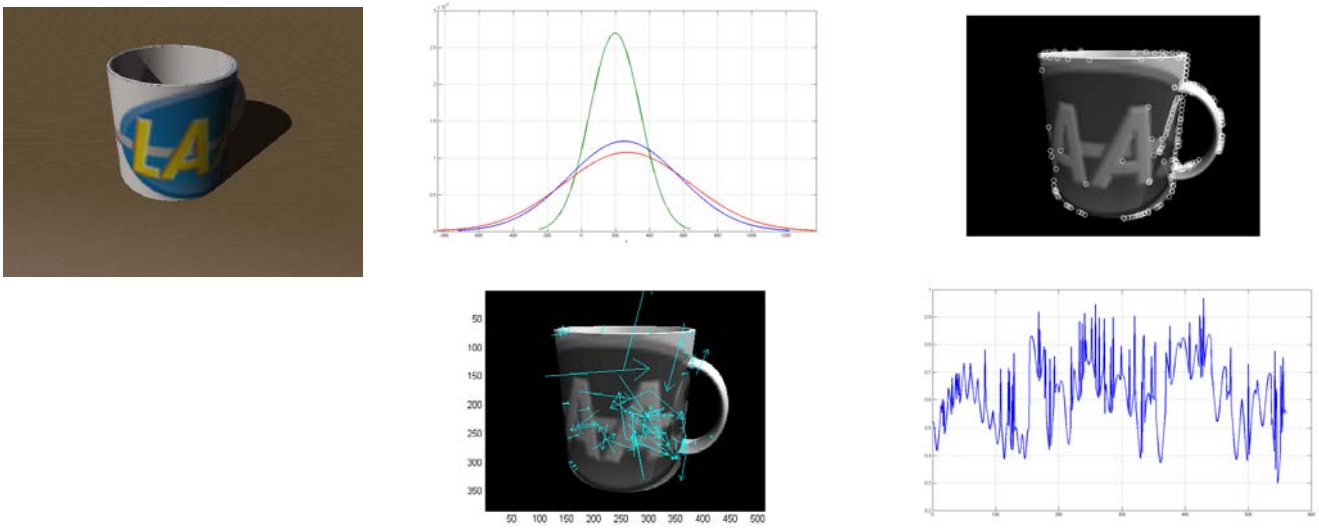


FIGURE 4.50 – Extraction des caractéristiques depuis un point de vue de l'objet 5.

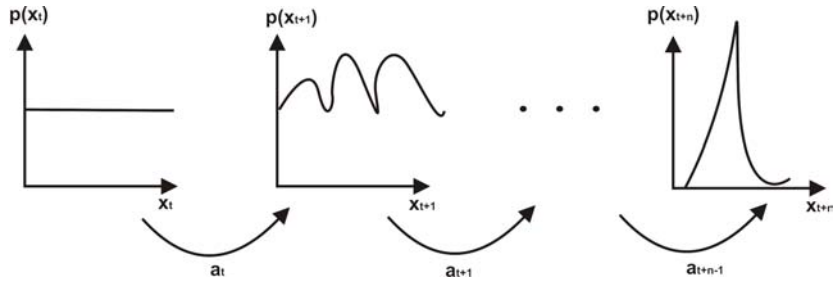


FIGURE 4.51 – Evolution des probabilités en fonction des actions réalisées.

FIGURE 4.52 – Test1 : trois images nécessaires pour converger.

6. Le système itère jusqu'à ce qu'il parvienne à trouver un objet avec une probabilité supérieure à un seuil donné.

En bref, pour réaliser la reconnaissance, on présente un objet, en principe inconnu, au système ; on suit la procédure décrite précédemment pour arriver à trouver l'objet de la base d'apprentissage, qui soit le plus proche de l'objet en cours d'analyse, au sens de la distance entre les histogrammes liées à chaque caractéristique. Le système rend comme réponse l'étiquette de cet objet le plus proche. Cette procédure doit pouvoir s'appliquer aussi pour reconnaître plusieurs objets présents dans une même scène, séparés ou en amas.

4.7 Evaluation de la méthode de reconnaissance active.

Nous avons réalisé plusieurs tests afin de vérifier la robustesse de notre méthode. Nous allons montrer plusieurs des tests réalisés, qui correspondent à des configurations différentes d'une scène ; ces tests ont d'abord été faits sur des images de synthèse construites sur des objets virtuels, puis sur des images acquises sur des objets réels.

4.7.1 Test sur des images de synthèse avec un objet isolé.

Ces tests sont significatifs puisqu'ils représentent tous les cas qui peuvent se présenter lorsqu'un robot doit rechercher un objet dans le monde réel. Les cas suivants ont été considérés :

- test1 : l'objet est présenté en une position dans laquelle il a été appris et sans occultation.
- test2 : ce même objet est présenté dans une position différente de celle dans laquelle il a été appris.
- test3 : l'objet est présenté en une position dans laquelle il a été appris mais cette fois ci, avec une occultation.

Ce test a été effectué sur des images synthétisées sur la tasse, objet 5 de la base présentée en figure 4.46, objet appris sur les images présentées en figure 4.47.

Les figures 4.52 et 4.53 montrent les images exploitées pour le test 1, et l'évolution de l'état de la reconnaissance. Trois images suffisent pour conclure ; l'ambiguïté entre les classes 1 et 5 est rapidement levée.

Les figures 4.54 et 4.55 montrent les images exploitées pour le test 2, et l'évolution des probabilités. Il faut en ce cas 8 images pour converger vers la bonne interprétation. Le système initialement, donne la même probabilité qu'un objet de la classe 2 ou de la classe 5 soit présent dans la scène. A partir de la 4ième image, le motif inscrit sur la tasse est mieux visible, et la classe 5 se détache sans problème, pour être la seule possible après 8 itérations.

Enfin, les figures 4.56 et 4.57 montrent les images exploitées pour le test 3, et l'évolution des probabilités. Dans la scène virtuelle, nous avons rajouté une pièce occultante, avec une apparence de bois, donc une couleur différente de celle de la tasse. Le système converge au bout de cinq itérations.

Nous avons réalisé ce test pour chacun des objets afin de valider la robustesse de notre algorithme. On a repeté l'expérience du test1 trente fois pour chaque objet et nous avons obtenu la matrice de confusion présentée dans la table 4.2. Rappelons que la matrice de confusion est un outil servant à mesurer la qualité d'un système de classification qui exploite un apprentissage supervisé. Chaque colonne i montre

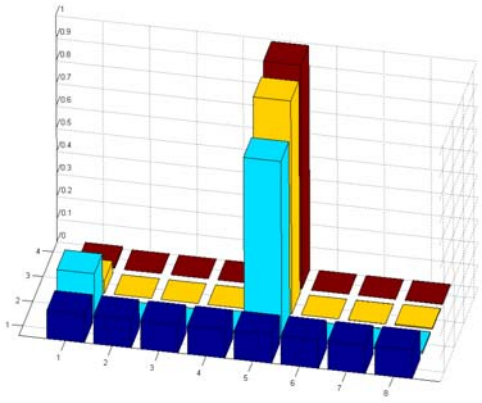


FIGURE 4.53 – Test1 : évolution des probabilités pendant les trois itérations.

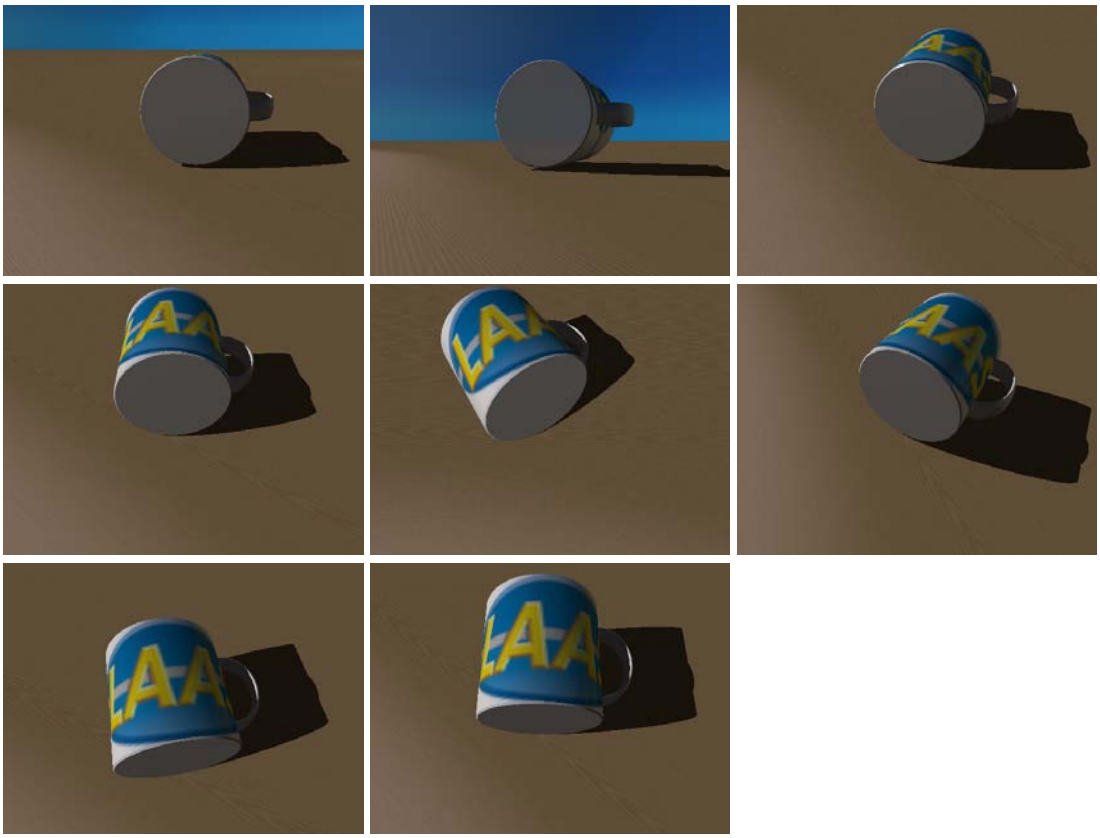


FIGURE 4.54 – Test2 : les huit images nécessaires pour converger.

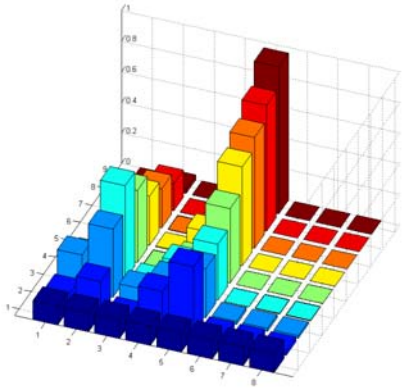


FIGURE 4.55 – Test2 : évolution des probabilités pendant les huit itérations.

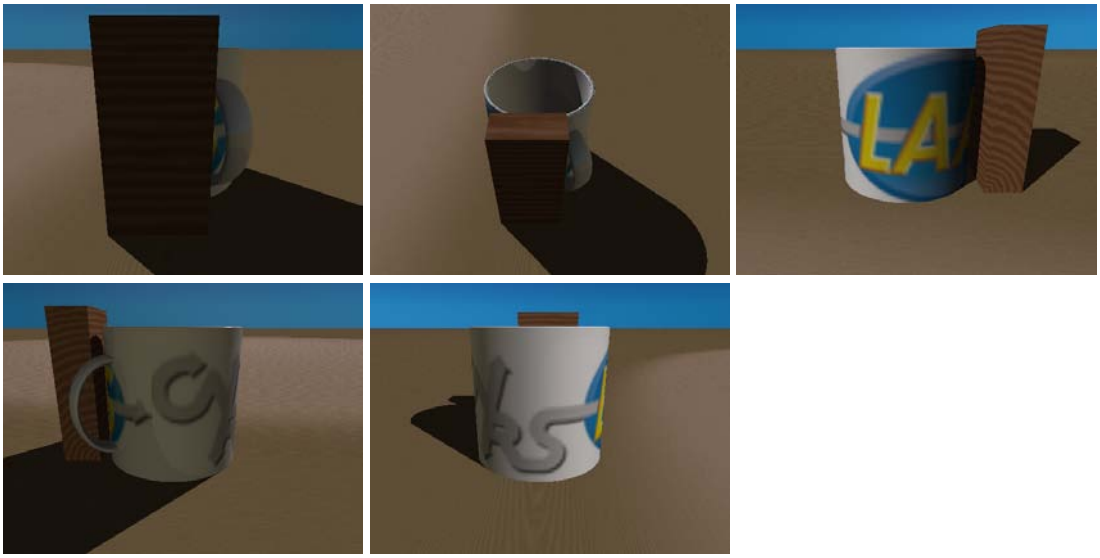


FIGURE 4.56 – Test3 : cinq images nécessaires pour converger.

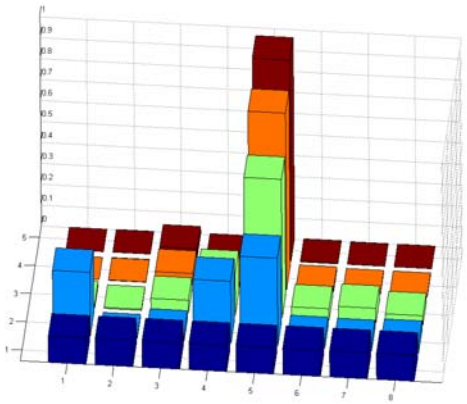


FIGURE 4.57 – Test3 : évolution des probabilités pendant les cinq itérations.

	objet 1	objet 2	objet 3	objet 4	objet 5	objet 6	objet 7	objet 8
objet 1	28	0	0	1	0	0	1	0
objet 2	0	30	0	0	0	0	0	0
objet 3	0	0	29	0	0	0	0	1
objet 4	0	0	0	27	2	1	0	0
objet 5	0	0	0	0	30	0	0	0
objet 6	0	0	0	0	0	30	0	0
objet 7	0	0	0	1	0	0	29	0
objet 8	0	0	1	0	1	1	0	27

TABLE 4.2 – Reconnaissance sur des images de synthèse : matrice de confusion.

le nombre de fois où un objet de la classe C_i est reconnu : si $mat_{ki}, k \neq i$ est non nul, c'est un faux positif pour C_i. Chaque ligne j montre le résultat de la reconnaissance quand un objet de la classe C_j est présenté : si $mat_{jl}, j \neq l$ est non nul, c'est un faux négatif pour C_j.

L'idéal est que, pour toute classe C_i, tout objet présenté appartenant à cette classe, soit classifié C_i (pas de faux négatif), et que tout objet classifié C_i, appartienne réellement à cette classe (pas de faux positif) ; la matrice de confusion en ce cas, est diagonale. Un des intérêts de la matrice de confusion est qu'elle montre rapidement si le système parvient à classifier correctement. On voit par exemple que l'objet de la classe 5 (tasse) est toujours reconnu : donc pas de faux négatif. Par contre, il existe trois faux positifs avec des objets d'apparence cylindrique (abat-jour et théière).

Finalement nous avons réalisé aussi une série de tests dans lesquels nous avons montré différents objets au système ; certains sont des objets des classes apprises ; d'autres n'ont pas été appris, mais peuvent éventuellement ressembler aux objets appris (cubes avec différentes lettres sur les faces). Nous limitons le nombre d'itérations ; si aucune probabilité d'appartenance de l'objet présenté à une des huit classes apprises, n'atteint le seuil fixé, au bout de dix itérations, alors, l'objet est étiqueté avec la classe $NULL$.

Les résultats obtenus sont montrés dans la figure 4.58 ; on peut observer que les objets appartenant aux classes apprises sont toujours reconnus, que pour les objets inconnus, soit le système parfois n'arrive pas à converger, soit l'objet est assigné à la classe la plus proche en fonctions de ses caractéristiques. On peut constater que la forme est prépondérante ici ; les objets cylindriques (resp. parallélépipédiques) sont classés *tasse* (resp. *boîte*).

4.7.2 Test sur des images réelles.

Dans cette section nous allons tester notre algorithme de reconnaissance en utilisant des images et des objets réels. La figure 4.59 montre les objets que nous allons utiliser pour les expériences suivantes. Nous allons nous servir de ces objets afin de confirmer les résultats obtenus en simulation ; pour simplifier, nous désignons un objet de la classe i, par objet i.

Les images de la figure 4.61 montrent les différents points de vue de l'objet 1 utilisés pendant un test de la procédure de reconnaissance. On peut vérifier que la position de l'objet a changé par rapport aux images exploitées lors de l'apprentissage, images présentées en figure 4.60.

Le graphique presenté en figure 4.62 est le résultat obtenu de l'exécution de notre algorithme de reconnaissance. Sur ce graphique on peut voir que le système après la première image, génère trois hypothèses, correspondant aux classes 1, 4 et 5. Mais grâce aux caractéristiques trouvées en analysant les images suivantes, il se décide sans ambiguïté pour la classe 1.

Ensuite nous présentons au système une situation dans laquelle nous avons deux objets, un objet connu et l'autre inconnu. Si on regarde les images de la figure 4.63 on constate que l'objet inconnu provoque des occultations sur l'objet connu, qui, dans cet exemple, appartient à la classe 1.

Lorsqu'on démarre le processus de reconnaissance sur cette scène, on obtient le résultat montré dans le graphique présenté en figure 4.64 ; on voit que malgré les hésitations du système, il arrive à reconnaître à la fin l'objet de la classe 1. Ces hésitations sont dues à la présence de l'objet inconnu dans la scène.

Dans la situation suivante, nous allons présenter au système deux objetsde classes différentes, mais cette fois ci, les deux objets ont été appris. Nous pouvons prévoir trois résultats possibles pour la phase

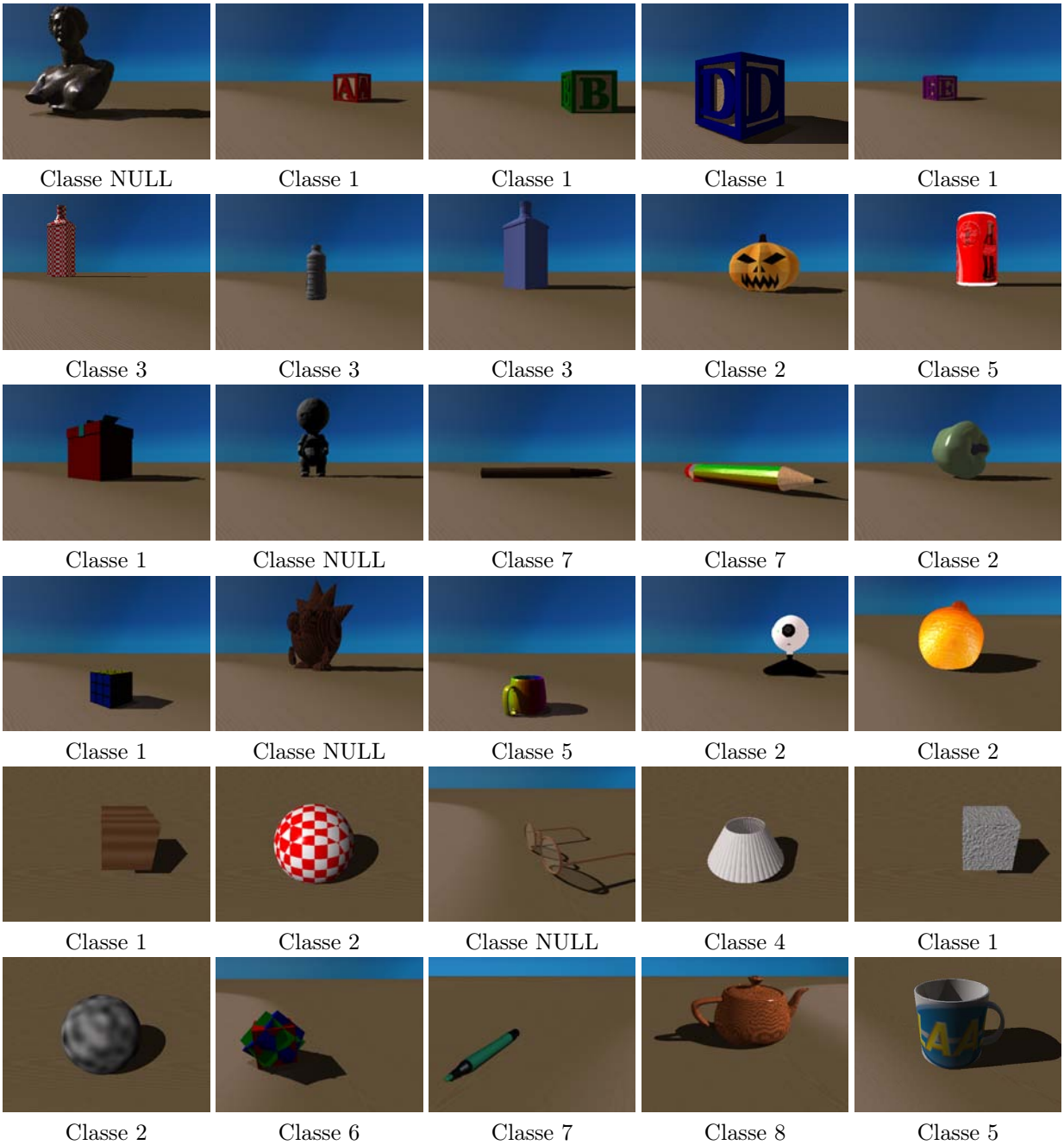


FIGURE 4.58 – Reconnaissance de divers objets et assignation à une classe particulière.

FIGURE 4.59 – Base d'objets réels utilisés pour nos évaluations de notre méthode de reconnaissance active ; ils seront appelés objet 1, 2 . . . 7,8.

FIGURE 4.60 – Quelques images de l'objet 1 en phase d'apprentissage.

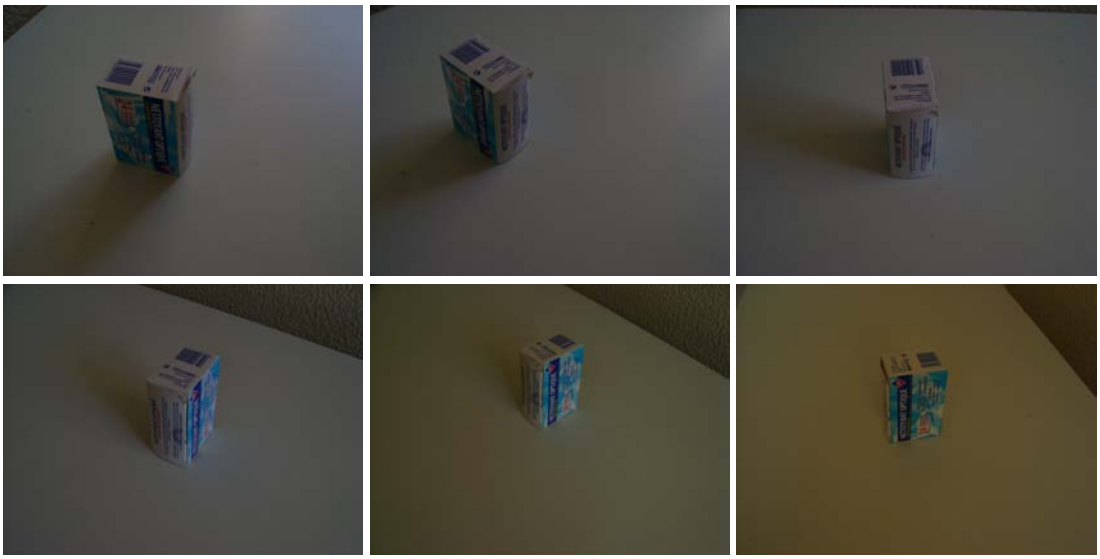


FIGURE 4.61 – Images nécessaires pour reconnaître l'objet 1 isolé sur la table.

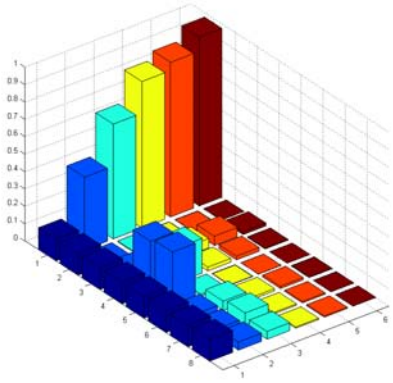


FIGURE 4.62 – Evolution des probabilités pendant la reconnaissance de l'objet 1 isolé.

FIGURE 4.63 – Images nécessaires pour reconnaître l'objet 1, occulté par un objet inconnu.

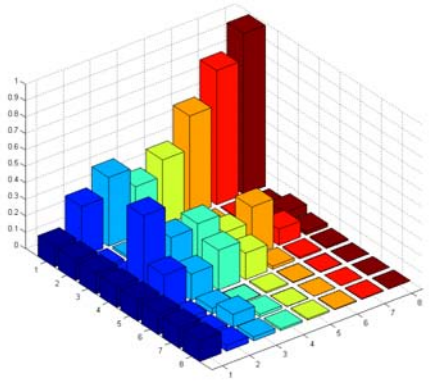


FIGURE 4.64 – Evolution des probabilités pendant la reconnaissance de l'objet 1 occulté.

FIGURE 4.65 – Images sur la scène avec deux objets connus ; quatre vues qui font converger sur objet 1.

de reconnaissance : soit le système reconnaît les deux objets, soit il reconnaît l'un, mais pas l'autre, soit il ne reconnaît ni l'un ni l'autre. Ces cas vont nous permettre d'apprécier comment notre stratégie de reconnaissance se comporte lorsqu'on lui présente deux objets connus.

En fait, le résultat va dépendre de l'image initiale. Les images des figures 4.65 et 4.67 montrent les images acquises successivement, dans deux cas où le processus de reconnaissance converge sur l'objet 1 et sur l'objet 5 respectivement. On peut voir l'évolution des probabilités dans les deux cas en figures et 4.68. Le système oscille entre un des deux objets possibles, mais à la fin, il arrive à reconnaître un seul objet sur les deux, car il sélectionne les actions suivantes (donc les points de vue sur la scène), en fonction de l'hypothèse la plus probable ; il va donc renforcer celle qui prend le dessus, ce qui dépend de l'image de départ. Il existe des configurations qui conduisent à une oscillation ; en figure 4.69, après 12 itérations, le processus n'a pas convergé.

Après avoir realisé cette expérience une trentaine de fois nous avons construit la table 4.3, pour montrer la performance du système. On peut constater que le système a une forte tendance à reconnaître un des objets presentés : le nombre d'occurences dans lesquelles il ne peut pas décider est plus bas que le nombre d'occurences dans lesquelles il arrive à reconnaître soit l'un, soit l'autre.

Finalement, nous avons réalisé le processus de reconnaissance trente fois pour chacun des objets de la figure 4.59, avec l'objet isolé. Nous avons obtenu la matrice de confusion présentée dans la table 4.4. On peut voir qu'on a un pourcentage de reconnaissance correcte dans 86,6% des cas. Cela paraît acceptable

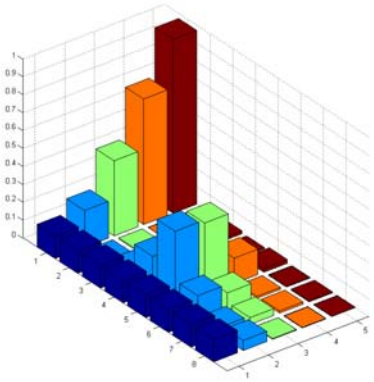


FIGURE 4.66 – Evolution des probabilités pour traiter le cas de deux objets connus, avec convergence sur objet 1.

FIGURE 4.67 – Images sur la scène avec deux objets connus ; cinq vues qui font converger sur objet 5.

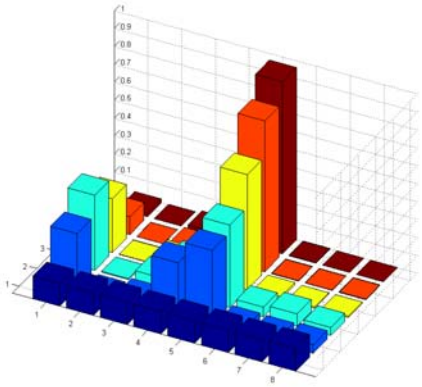


FIGURE 4.68 – Evolution des probabilités pour traiter le cas de deux objets connus, avec convergence sur objet 5.

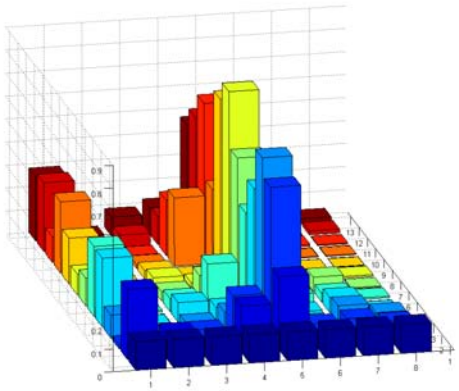


FIGURE 4.69 – Evolution des probabilités pour traiter le cas de deux objets connus : pas de convergence.

	objet 1	objet 5	Aucun
Nombre de tests	11	13	6

TABLE 4.3 – Résultat de la reconnaissance sur une scène avec deux objets connus.

s'agissant d'expérimentations sur des objets réels, si on prend en compte toutes les conditions telles que les variations d'illumination.

Comme nous l'avons vu ci-dessus, lorsqu'on montre au système une situation ambigüe, comme dans le cas de la scène avec deux objets connus, la performance du système n'est pas optimale : on désire qu'il soit capable de gérer des situations de ce genre. Dans la section suivante, nous allons proposer quelques adaptations à notre stratégie de reconnaissance, en particulier pour qu'il reconnaisse plusieurs objets présents dans une même scène de manière plus robuste. Cela est rendu possible avec l'inclusion d'une nouvelle classe : la classe *NULL*.

4.7.3 Traitement de la classe Inconnu

Dans cette section nous allons étudier la possibilité de détecter la présence d'un objet inconnu dans la scène en cours d'analyse. Comme notre système est conçu afin de faire évoluer la probabilité des objets qui ont été appris pendant la phase d'apprentissage, il est nécessaire d'inclure une classe qui puisse être sélectionnée par notre système, dans les cas où il n'y a pas d'objet sur la table ou quand l'objet présenté est inconnu. Nous appelerons cette classe, classe *NULL*. Mais l'inclusion de cette classe n'est pas une tâche facile puisqu'on doit analyser les caractéristiques d'un objet qui n'ont pas été apprises, afin de pouvoir mettre à jour sa probabilité au fur et à mesure que le système évolue dans la phase de reconnaissance.

L'inclusion de cette classe va nous permettre de gérer différentes situations comme celles citées ci-dessous

1. un objet inconnu,

	objet 1	objet 2	objet 3	objet 4	objet 5	objet 6	objet 7	objet 8
objet 1	27	0	1	0	1	1	0	0
objet 2	1	25	0	0	2	0	1	1
objet 3	0	2	20	2	1	0	5	0
objet 4	0	2	1	26	0	0	1	0
objet 5	0	1	0	0	28	0	1	0
objet 6	0	0	0	0	0	30	0	0
objet 7	2	3	2	0	1	0	22	0
objet 8	0	0	0	0	0	0	0	30

TABLE 4.4 – Reconnaissance d'objets réels : matrice de confusion.

FIGURE 4.70 – Images pour tester l'algorithme sans aucun objet sur la table.

2. le manque d'objet sur la scène, et
3. la possibilité de reconnaître un, deux ou plusieurs objets sur la même scène.

La motivation principale d'associer un objet à la classe *NULL*, est la détection de la présence d'objets inconnus ainsi que celle d'absence d'objets, donc de la table vide. Notre système est developpé pour qu'il puisse reconnaître un objet posé sur une surface plane, typiquement une table. Quelque soit la nature du ou des *TRUCs* détectés sur la dite table, que ce soit des objets connus ou inconnus, notre système va essayer pour chaque *TRUC*, de trouver l'objet le plus proche dans sa base de données en fonction des descripteurs obtenus à partir des images acquises sur la scène ; si l'objet est inconnu, il sera assimilé à un connu, le plus proche dans l'espace des caractéristiques, certainement avec une probabilité de présence faible si la ressemblance est relative. En rajoutant une classe *NULL*, l'objectif des de pouvoir décider explicitement la présence d'un objet inconnu, ou l'absence d'objet.

D'autre part, avec cette classe *NULL*, nous avons pu intégrer des mécanismes spécifiques, soit de focalisation, soit d'inhibition : notre système cherche à reconnaître explicitement des objets d'une classe, ou qu'au contraire, il ignore une classe donnée, typiquement celle d'un objet dont il a déjà détecté la présence dans la scène. Le système peut ainsi déterminer s'il y a un autre objet connu ou inconnu dans la scène en cours d'analyse.

La création de cette nouvelle classe n'est pas une tâche facile à mettre en oeuvre, puisqu'il faut prendre en compte plusieurs éléments tels que la manière avec laquelle on va traiter la mise à jour des probabilités de présence d'objets des différentes classes lors de la phase de reconnaissance. Nous allons commenter ces diverses possibilités dans les paragraphes suivants.

Manque d'objet sur la scène

Le premier problème qui vient à notre esprit est : que se passe t'il s'il n'y a pas d'objet dans la scène à analyser, mais que, pour une raison quelqonque, notre méthode de segmentation trouve un *TRUC* à analyser ? Cela survient en particulier en cas de reflet ou d'ombre sur la table. Dans ce cas nous souhaitons que le système rende la réponse *Aucun objet connu présent dans la scène*. Cela est posible seulement si on incorpore dans le système une classe *NULL*. Une fois que le système démarre il va essayer de trouver une ressemblance avec les objets de sa base de données ; l'objet NULL agit comme un filtre entre le système et le monde réel afin d'éviter que le système nous rende une fausse réponse et nous dise qu'il y a un objet alors qu'il n'y en a pas.

Les images montrées dans la figure 4.70 ont été utilisées pour tester le processus de reconnaissance mis face à la situation de ne pas avoir d'objet dans la scène en cours d'analyse. La figure 4.71 présente la façon dans laquelle l'algorithme fait évoluer les probabilités afin de parvenir à donner une réponse en accord avec ce qu'il est en train d'analyser. Dans ce graphique la classe numéro 9 est la classe *NULL* ; une affectation dans cette classe signifie qu'il y a un objet inconnu dans la scène ou que la table est vide. Comme on peut le constater, la case qui obtient la probabilité la plus haute, correspond précisément à la classe *NULL*.

Un objet inconnu dans la scène.

Maintenant, nous présentons un objet inconnu au système et nous demarrons le processus de reconnaissance. Le système commence à segmenter l'objet, puis à extraire les caractéristiques de l'objet. Ensuite

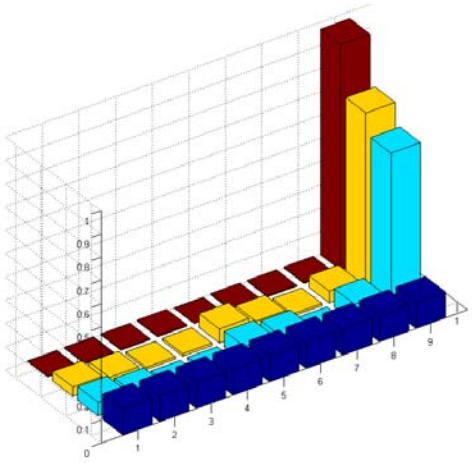


FIGURE 4.71 – Evolution des probabilités pour la reconnaissance sans aucun objet sur la table.

il cherche dans sa base de données, la classe la plus ressemblante dans l'espace de ces caractéristiques. S'il ne connait pas l'objet présent dans la scène, il va rendre une réponse aléatoire.

Grâce à l'utilisation de la classe *NULL*, le système est capable de faire la différence entre un objet appris, donc référencé dans sa base de données, et un autre qui lui ressemble. Dans ce cas là le système va traiter la requête de reconnaissance comme une recherche nulle et va nous dire qu'il n'y pas d'objet connu dans la scène.

Le graphique de la figure 4.73 montre la manière avec laquelle le système met à jour les probabilités des différentes classes qui ont été apprises. Nous pouvons voir que dans un premier temps, le système hésite entre la classe *NULL* et les classes 3, 5 et 6 pour finalement se rendre à l'évidence que l'objet observé ne correspond à aucune classe apprise. Ainsi le système a la possibilité de détecter la présence d'un objet inconnu.

Reconnaissance de plusieurs objets connus sur une scène.

Enfin reprenons le test déjà effectué en plaçant dans la scène à analyser, plusieurs objets connus. Nous avons vu que notre méthode donne un résultat qui dépend de la configuration initiale : au meilleur des cas, il reconnaît un seul des objets présents.

Recommençons ce test, en considérant la possibilité de la classe *NULL*. Dans un premier temps le système réagit comme attendu : il va commencer à osciller entre plusieurs hypothèses sur la présence d'objets dans la scène ; dans certains cas, il finit par converger sur l'une d'elles, mais cela dépend du point de vue depuis lequel il calcule initialement les caractéristiques. Mais si on considère qu'il existe une classe *NULL*, le système peut avoir un comportement différent de celui décrit précédemment ; l'inclusion de cette classe *NULL* va permettre de focaliser explicitement la reconnaissance sur un seul des objets, et en conséquence, on provoque une inhibition de la reconnaissance pour tous les autres objets présents dans la scène. Ainsi le système est capable de se focaliser sur un objet determiné en ignorant tout le reste de la scène ; en pratique, on cherchera un objet précis et tout le reste sera affecté à la classe *NULL*.

Donc dans notre situation avec plusieurs objets connus dans la scène, à un moment donné du processus de reconnaissance, le système gardera en memoire la quantité d'objets qui peuvent être présents dans la scène, ainsi que les classes auxquelles ces objets peuvent appartenir. A partir de là, en exploitant le mécanisme de focalisation/inhibition, il va concentrer son effort successivement sur chacune de ces hypothèses, chaque fois en reprenant sa recherche depuis l'image depuis laquelle il a commencé cette focalisation.

Le graphique montré en figure 4.75, permet d'apprécier la manière avec laquelle le système évolue dans le temps. Après deux images, le système peut faire la supposition qu'il existe trois objets dans la scène, objets qui appartiennent aux classes 1, 5 et 7. A partir de cette supposition le système garde en mémoire la position DEPART depuis laquelle il a vu ou il a cru voir les different objets qui sont sur la

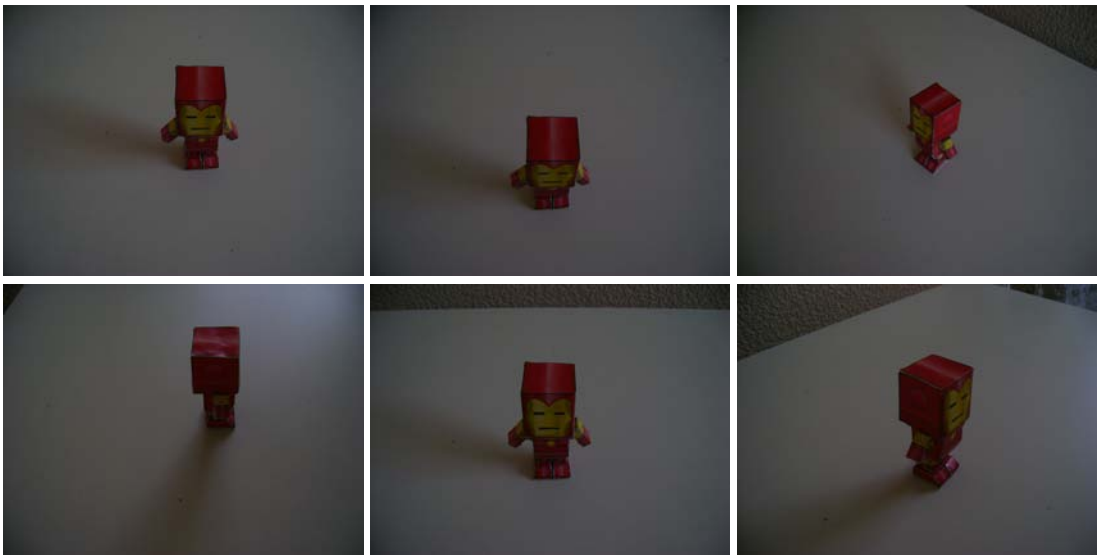


FIGURE 4.72 – Images pour tester l'algorithme avec un objet inconnu sur la table.

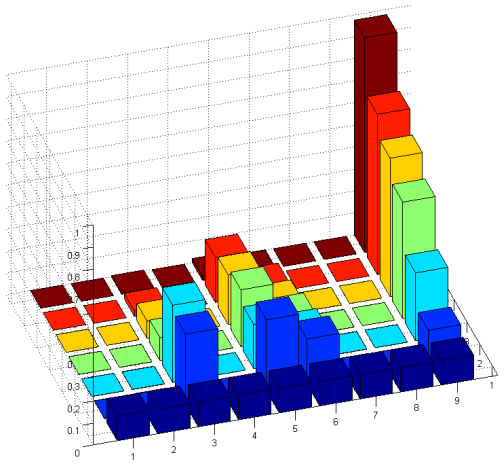


FIGURE 4.73 – Evolution des probabilités pour la reconnaissance avec un objet inconnu sur la table.

FIGURE 4.74 – Images pour tester l'algorithme avec trois objets connus dans la scène.

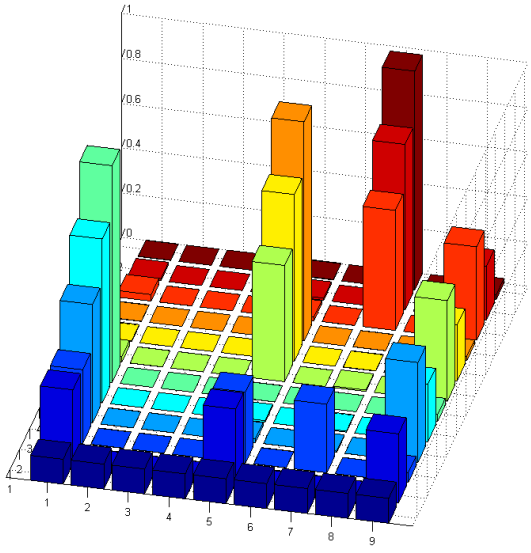


FIGURE 4.75 – Evolution des probabilités lors de la reconnaissance avec trois objets connus dans la scène.

table.

Après il focalise sa recherche sur l'objet avec la probabilité la plus élévée, dans ce cas l'objet de la classe 1 ; il n'y a plus que deux probabilités mises à jour, correspondantes aux classes 1 et *NULL* ; les autres sont forcées à 0. Une fois que le système a validé l'hypothèse qu'un objet de la classe 1 est dans la scène, alors il retourne (virtuellement) à la position DEPART pour sélectionner une action lui permettant de valider la présence de l'objet suivant. Il lui faudra trois positions pour vérifier qu'un objet de la classe 5 est bien dans la scène. Finalement, de la même manière en sélectionnant une action adaptée depuis la position DEPART, il va essayer de valider sa troisième hypothèse concernant la présence d'un objet de la classe 7. Comme on peut le constater en figure 4.75, le système a bien validé toutes les hypothèses qu'il avait émis initialement par la recherche multi-classes.

4.8 Discussions sur quelques scénarios spécifiques.

Nous avons testé différentes situations en plus de celles que nous avons montrées précédemment. Mais malheureusement les résultats obtenus n'ont pas été ceux que nous attendions. Dans la suite nous faisons une description de chacune de ces situations ainsi que les résultats obtenus afin de donner une idée plus claire des problèmes rencontrés. Nous proposons aussi une solution possible pour chacun des cas étudiés.

Les situations étudiées sont les suivantes :

- Deux objets identiques dans la scène.
- Deux objets avec un seul motif distintif.
- Un objet qui disparaît pendant la phase de reconnaissance.

Scénario1 : deux objets de la même classe dans l'image

Lorsqu'on fait analyser à notre système, une scène contenant deux objets identiques, de la même classe, on voudrait qu'il détecte la présence de deux objets de la même classe. Ce qui arrive, c'est que le système trouve un objet qui appartient à la bonne classe et ensuite il arrête le processus de reconnaissance à cause de l'inhibition des caractéristiques de la classe de l'objet déjà trouvé. Pourtant il a juste reconnu un des deux objets ; il va oublier celui qui reste.

Il faut noter que notre système serait capable de reconnaître ce second objet dans une telle scène, mais pour cela, il faudrait qu'il puisse savoir la quantité d'objets de chaque classe qu'il doit essayer de trouver pendant la phase de reconnaissance.

La solution à ce problème serait qu'au moment de faire la segmentation des objets qui sont dans la scène, on puisse récupérer le nombre d'objets présents dans la scène. A cet effet on pourrait faire appel à l'algorithme de segmentation par les contours (snake), ou par les régions (segmentation chromatique, Mean Shift, Graph cut...). Pour l'instant, nous sommes capables de segmenter sur une image, plusieurs régions d'intérêt, correspondant dans notre contexte, à plusieurs *TRUCs* présents sur la table. Néammoins nous traitons chaque région segmentée comme s'il n'y avait qu'un seul objet à identifier : or il se pourrait que dans une des images prises ensuite, cette région puisse se découper en deux régions ; en quel cas, le système saurait qu'il y a au moins deux objets dans la scène. C'est ce qui survient dans nos séquences en figure 4.63 : l'objet 1 est occulté par l'objet A : à la 4ième image de la séquence, le système peut séparer les deux régions, et donc, savoir qu'il existe deux objets dans la scène.

Pour conclure sur ce problème, disons que la segmentation apparaît, comme souvent en vision, comme la fonction la plus critique de toute la phase de reconnaissance.

Scénario2 : deux objets avec un seul motif distinctif.

Dans ce cas, le résultat obtenu serait similaire à celui montré avec deux objets de la même classe. La seule différence existante est quand le motif qui fait la différence entre les deux instances d'objet, est visible au moment de la reconnaissance. Dans ce cas le système est capable de discriminer les deux objets sans aucun problème. Mais si le motif est caché dans un des objets ou dans les deux, le système va réagir comme dans le cas de deux objets identiques : il reconnaîtra l'un de deux et ne parviendra pas à reconnaître l'autre à cause du mécanisme d'inhibition.

La seule solution possible pour résoudre ce cas est d'avoir plus d'éléments qui puissent servir pour faire la différence entre les deux objets. Sinon nous pourrions procéder comme dans le cas précédent et utiliser la phase de segmentation pour indiquer au système qu'il doit reconnaître deux objets, mais en considèrant que ces deux objets puissent appartenir à la même classe.

Scénario3 : un objet qui disparaît pendant la reconnaissance

Comme on peut l'imaginer, un tel scénario va poser problème. Notre système a été developpé pour reconnaître un ou plusieurs objets posés sur une table ; par conséquent, il n'est pas capable de gérer les changements qui pourraient arriver pendant la phase de reconnaissance. Alors, si pendant la phase de reconnaissance d'un objet, celui-ci disparaît soit par magie, soit parce qu'un être humain a pris l'objet, notre système devrait tout simplement rendre la réponse qu'il n'y a plus d'objet sur la scène.

Pour résoudre ce cas il faudra avoir un mécanisme capable de détecter quand la scène change, et d'agir en conséquence à un tel changement. Cela nécessiterait d'intégrer dans notre système, des méthodes visuelles (détection de mouvement par exemple), qui sortent du cadre de cette étude ; néammoins, dans un vrai robot personnel qui doit interagir avec l'Homme, un tel scénario est possible, et le robot devrait pouvoir le traiter.

Autres considérations

Il reste quelques améliorations possibles pour notre système. Par exemple, le système incorpore la classe *NULL* pour traiter différentes situations, mais il peut rendre la même réponse pour deux situations pourtant différentes : il n'y a pas d'objet sur la table et l'objet presenté est un objet inconnu. Ces deux situations ne peuvent être discriminées dans la phase de reconnaissance, donc il faut un mécanisme qui permette de savoir que la table est vide, tout cela avant de commencer la phase de reconnaissance. La phase de segmentation, avec un filtrage appropriée des petites régions d'intérêt, devrait s'en charger.

L'autre amélioration que nous devrions prendre en compte est l'exploitation de données 3D acquises par stéréovision afin de rendre plus robuste, cette phase de segmentation des objets. En utilisant les données 3D, nous allons pouvoir enlever de la scène toutes les régions de l'image qui sont à des profondeurs trop grandes ; donc l'exploitation des discontinuités de profondeur devrait permettre de segmenter l'objet d'intérêt du fond.

Il existe probablement de nombreuses autres améliorations à réaliser, mais celles proposées dans cette section nous paraissent les plus urgentes à mettre en oeuvre, et les plus faciles à obtenir.

4.9 Conclusion sur la reconnaissance.

Nous avons presenté une méthode qui peut reconnaître des objets usuels, posés sur une table. Ces objets doivent d'abord être appris de manière supervisée : chaque objet est présenté au robot, isolé sur la table. Ensuite, notre méthode est capable de reconnaître un objet, même quand il est partiellement occulté. Elle peut aussi reconnaître plusieurs objets qui sont posés ensemble sur la table. Elle incorpore une classe d'objet appelé *NULL* : notre système est capable de discriminer un objet dont il n'a pas de modèle dans sa base de connaissance en l'associant à cette classe *NULL*.

Nous avons montré par de nombreux exemples, que notre système est assez robuste. Mais il a aussi quelques défauts que nous avons commentés dans la section précédente : il n'est pas capable de reconnaître deux objets identiques posés ensemble sur la table puisqu'il va reconnaître le premier perçu, mais il va laisser de côté le deuxième. Une possible amélioration serait de prendre en compte la partie 3D pour améliorer la segmentation et réaliser une focalisation différente de celle que nous avons présentée dans ce chapitre.

Dans le chapitre qui suit, nous allons appliquer les méthodes décrites précédemment pour reconstruire et reconnaître un objet 3D, cela pour traiter d'une tâche particulière en Robotique : la saisie d'objets.

Chapitre 5

Intégration : vers la saisie d'un objet 3D modélisé et reconnu par vision

Après avoir conçu, développé et évalué hors-ligne plusieurs fonctions perceptuelles, il reste à décrire une application pratique dans un contexte robotique, des algorithmes de reconstruction et de reconnaissance 3D développés dans cette thèse. C'est la raison pour laquelle nous allons évoquer dans ce chapitre, l'intégration de ces fonctions dans un système robotique complet, capable de saisir des objets.

Essentiellement il s'agit de l'intégration entre les modules de reconnaissance et de modélisation 3D, avec les modules qui font bouger le bras et qui traitent de planification de trajectoires. Ces modules sont nécessaires pour mener à bien une tâche de saisie et manipulation d'un objet.

Dans un premier temps nous allons montrer l'intégration de tous ces modules dans un environnement de simulation afin de pouvoir les tester, retirer les erreurs conceptuelles et de programmation (déboguer) avant leurs intégrations finales sur le système robotique réel. En fait, nous présentons essentiellement des résultats de validation du système intégré, obtenus dans cet environnement de simulation : il s'agit d'un environnement que nous avons réalisé nous-même à partir de logiciels Open Source (affichage 3D OpenGL, définition des modèles des robots ...).

Donc dans la section suivante, nous évoquons quelques travaux sur la manipulation intégrée dans un contexte Robotique Mobile. Nous finissons par rappeler quel est le scénario dans lequel s'inscrivent nos fonctions perceptuelles, scénario initialement défini dans le cadre du *Key Experiment 2* du projet COGNIRON. Pour finir, nous montrons des résultats de simulation sur trois scénarios de saisie d'objects connus ou inconnus.

5.1 Contexte : manipulation par un robot de service.

La nécessité chaque fois plus grande d'augmenter la productivité et d'obtenir des produits finis d'une qualité uniforme, a conduit l'industrie à une automatisation poussée, donc à l'exploitation de robots contrôlés par des ordinateurs. Le manque de flexibilité et généralement le haut coût de ces machines, souvent appelés systèmes d'automatisation durs, ont généré un intérêt croissant pour la conception et le développement de robots capables d'effectuer une variété de fonctions de fabrication dans un environnement de travail plus flexible et à un plus petit coût de production.

Mais les robots mobiles sans bras sont limités à se déplacer (robots de service à roues, ou à pattes ..), à surveiller ou explorer un environnement par la perception des événements qui arrivent dans son entourage (robots de surveillance)..., mais pas plus. Le robot ne peut pas atteindre ni toucher les objets et ne peut pas par conséquent manipuler son environnement. Les robots les plus sophistiqués de nos jours exploités essentiellement pour la Recherche et le Développement, sont des robots mobiles dotés d'au moins un bras pour tenir, réorienter ou déplacer des objets : citons les fauteuils robotisés intégrant le bras MANUS pour l'aide aux personnes handicapées, les robots mobiles à deux bras ou torses sur roues développés en Allemagne, Justin au DLR (figure 5.1, gauche) et Armar à FZI (figure 5.1, milieu), ou encore le robot Humanoïde HRP2 utilisé au LAAS-CNRS (figure 5.1, droite). Dans notre cas, nous avons travaillé sur le démonstrateur JIDO, une plate-forme Neobotix dotée d'un seul bras Mitsubishi PA10.

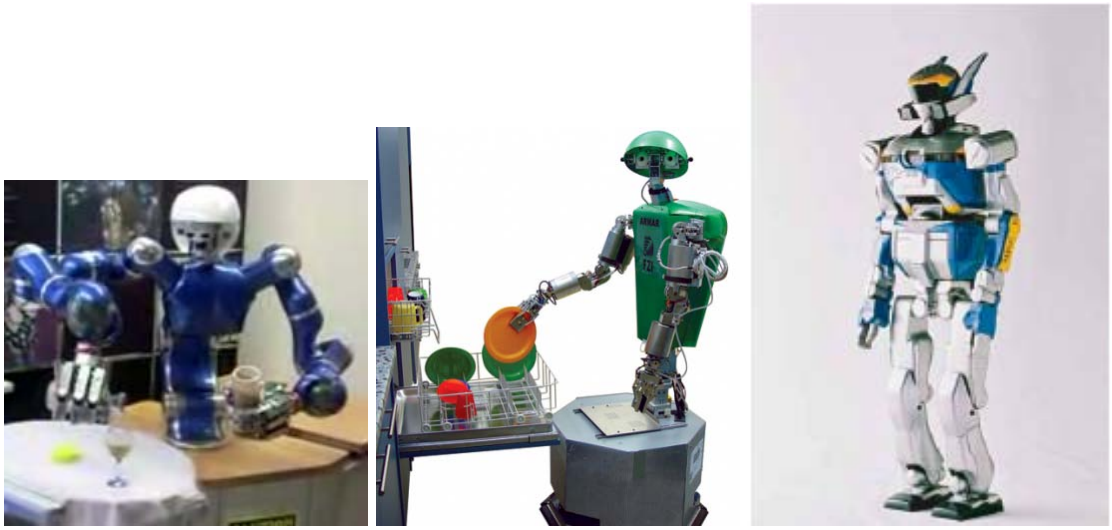


FIGURE 5.1 – Exemples de robots mobiles équipés de deux bras : Justin (gauche), Armar (milieu) et HRP2 (droite)

Les bras, donc, étendent la portée des Robots et ils les rendent plus semblables aux hommes. Étant donné toutes les capacités supplémentaires qu'ils fournissent à un robot, il est intéressant que les bras ne soient pas difficiles à construire et ensuite, à exploiter pour des applications robotiques. Les bras existants peuvent être employés comme robots stationnaires du type de ceux utilisés dans les usines, généralement dans un milieu très contrôlé (ateliers) ; ils peuvent être placés comme un appendice sur un robot mobile, exploité pour des tâches de service dans un milieu non contrôlé (domicile, lieu public ...). Quand nous parlerons de bras généralement nous voulons dire seulement le mécanisme du bras, en ne prenant pas en compte la main.

5.1.1 Le bras humain

Observons nos propres bras pour un moment (figure 5.2, à gauche). D'abord, nos bras sont des mécanismes énormément adaptables. Ils sont capables de manoeuvrer dans un espace de travail très grand. Pour cela, ils ont deux articulations principales : l'épaule et le coude : pour le poignet, on va considèrer ici que c'est une partie du mécanisme de la main.

- L'épaule peut se déplacer sur deux plans, en haut et vers le bas, vers derrière et vers l'avant. Si les muscles de l'épaule se déplacent vers le haut le bras complet est levé en étant séparé du corps. Si les muscles de l'épaule se déplacent vers l'avant le bras complet se déplace vers l'avant.
- L'articulation du coude est capable de se déplacer sur deux plans : en arrière et en avant, en haut et vers le bas.

Les articulations du bras humain fournissent donc quatre degrés de liberté. L'épaule offre deux degrés : rotation et la flexion de l'épaule. L'articulation du coude ajoute un troisième et un quatrième degrés de liberté : la flexion et la rotation du coude.

5.1.2 Types de bras artificiel

Les bras robotiques ont aussi plusieurs degrés de liberté. Cependant au lieu de muscles, tendons, rotules et os, les bras robotiques sont faits en métal, de matière plastique, bois, moteurs, électro-aimants, pignons, poulies et autres composants mécaniques. Quelques bras robotiques fournissent seulement un degré de liberté ; d'autres fournissent trois, quatre, cinq, jusqu'à six degrés de liberté pour le bras PA10 qui est monté sur notre démonstrateur JIDO (figure 5.2, à droite).

Les bras robotiques sont classés par la forme de l'aire que l'extrémité du bras (où on place la pince, ou la main robotisée) peut atteindre. Cette aire accessible est appelé "enveloppante de travail". Considérons pour simplifier, que l'enveloppante de travail ne tient pas compte du mouvement du corps du robot mais seulement des degrés de liberté du bras.

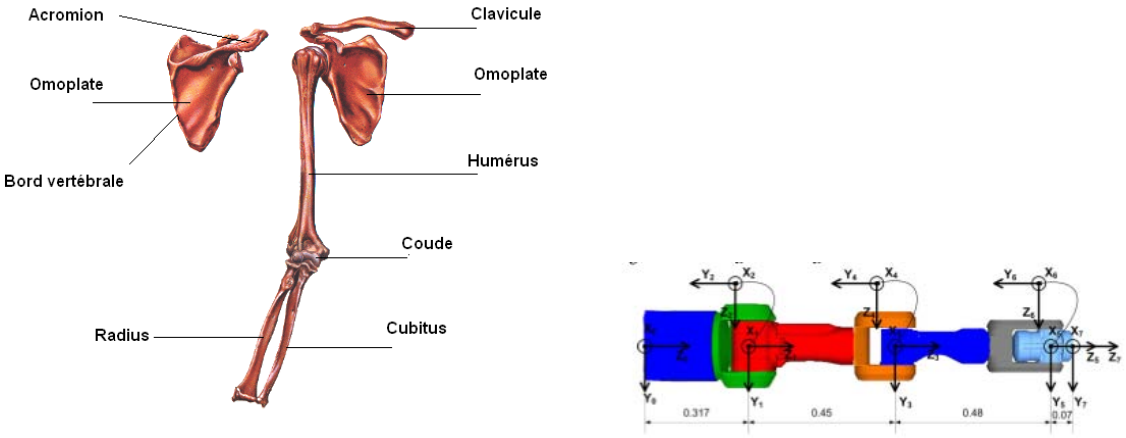


FIGURE 5.2 – A gauche, système squelettique du bras humain ; à droite bras robotique PA10 (from [101])

Le bras humain a une enveloppante de travail presque sphérique. Nous pouvons atteindre presque toute chose dans la portée de la longueur du bras, approximativement dans trois quarts d'une sphère (Imaginons que l'on est dans une écorce d'orange vide : si nous sommes placés dans son centre, nous pouvons toucher la peau de l'orange dans ses trois quarts).

On dit d'un bras robotique capable d'avoir une enveloppante sphérique, qu'il a des coordonnées de révolution. Les autres importants types de bras robotiques ont des coordonnées polaires ou cylindriques et des coordonnées cartésiennes ou rectangulaires. Toujours en excluant le poignet, qui généralement, donne par lui-même trois degrés supplémentaires, on observera qu'il y a trois degrés de liberté dans ces quatre types de bras robotiques : regardons d'un peu plus près chacun d'eux.

Coordonnées de révolution Les bras avec des coordonnées de révolution sont conçus à partir du bras humain, de sorte qu'ils héritent beaucoup de leurs capacités. La conception typique est quelque peu différente, en raison de la complexité de l'articulation de l'épaule humaine. Les trois degrés de liberté sont en rotation : le premier permet de faire tourner tout le bras autour de sa base (généralement entre π et $3\pi/2$) ; les deux autres rotations correspondent plus ou moins aux rotations de l'épaule et du coude, mais sans aucune flexion. Les bras de coordonnées de révolution fournissent beaucoup de flexibilité : ils s'apparentent à des bras réels.

Coordonnées polaires ou cylindriques L'enveloppante de travail du bras de coordonnées polaires a une forme d'une hémisphère ou de cylindre selon le type du deuxième degré de liberté. Les bras de coordonnées polaires ont une conception proche de ceux qui ont des coordonnées de révolution, mais ils sont plus flexibles au sens où ils peuvent prendre une grande variété d'objets dispersés autour du robot. Comme pour les bras avec des coordonnées de révolution, le premier degré permet une rotation du bras complet autour de sa base. Son second degré de liberté est l'articulation du coude, qui déplace l'avant-bras vers le haut et vers le bas, en rotation ou en translation. Le troisième degré de liberté est obtenu en faisant varier la portée de l'avant-bras : il étend ou rétracte un avant-bras intérieur pour amener la pince plus ou moins loin du robot. Ce type de bras est utilisé souvent dans des robots industriels, donc à poste fixe.

Coordonnées rectangulaires L'enveloppante de travail du bras de coordonnées cartésiennes (appelé aussi souvent table XYZ) ressemble à une boîte ; c'est le bras le plus différent du bras humain, puisqu'il n'a pas de degré en rotation. Ils sont construits en "empilant" des axes linéaires de plus ou moins grande portée : un premier axe pour déplacer les deux autres le long d'un axe X, un second axe qui déplace le troisième le long d'un axe Y, et enfin un troisième axe qui déplace la charge utile (éventuellement un poignet qui donne trois autres degrés en rotation) selon l'axe Z. Il est clair qu'un tel robot ne peut pas être embarqué sur une plate-forme mobile : c'est un type de bras qui est exploité à poste fixe.

Dans notre cas, nous avons exploité le bras Mitsubishi PA10, qui est du type coordonnées de révolution. Il nous a permis de positionner nos caméras sur une demi-sphère centrée sur l'objet à modéliser ou à reconnaître.

5.2 Objectifs : la saisie d'objets

Nous avons décrit dans le chapitre 2, dans quel contexte nous avons effectué ces travaux sur la modélisation et la reconnaissance d'objets 3D en vue de les manipuler. Nous rappelons ci-dessous nos objectifs en terme de démonstration Robotique : saisir un objet qui est au préalable, modélisé, reconnu et localisé par le système embarqué sur le robot.

Etat de l'art sur la saisie

Commençons par citer quelques travaux qui ont été effectués dans ce domaine. Miller *et al.* [94] ont réalisé la saisie d'un objet modélisé par des primitives 3D, telles que des sphères ou des boîtes et en disposant pour chacun, d'un catalogue de positions de prise déjà apprises ou données par l'opérateur. La méthode proposée en [94] est très similaire à celle développée au LAAS-CNRS par López-Damian [83] qui a aussi exploité les primitives des objets afin de réaliser la saisie de l'objet de manière rapide et efficace. Strandberg et Wahlberg [136] ont présenté une méthode fondée sur la capacité du robot de rejeter des forces de perturbation au moment de la saisie : la procédure prenait en considération la géométrie de l'objet ainsi que sa position et son orientation dans la scène.

Au LAAS-CNRS, la vision a été exploitée par V.Garric [54] pour effectuer des opérations de saisie précises d'un objet avec une stratégie de type *Static Look and Move*, c'est-à-dire que les images acquises depuis une caméra montée sur le poignet du manipulateur, sont exploitées pour localiser l'objet plusieurs fois sur la trajectoire de saisie. Vu le temps nécessaire pour ce traitement de localisation, le mouvement est stoppé à chaque prise d'images.

De nombreuses contributions ont traité de la saisie par asservissement visuel, donc par une stratégie *Dynamic Look and Move* (ou asservissement 3D, via des localisations successives) ou *Image-Based servoing* (ou asservissement 2D, via des lois de commande définies à partir de consignes exprimées dans l'image) : citons les travaux de F.Dornaika [43] qui contrôlait une opération de saisie faite par un manipulateur A depuis un système stéréo monté sur un manipulateur B : une fois localisé l'objet à saisir depuis les images acquises par B, on peut déterminer où sera en fin de saisie, la pince de A, et plus précisément où devront être des amers visuels placés sur la pince dans l'image finale acquise par B avant de fermer cette pince. Le contrôle était effectué en asservissement visuel 2D, en plaçant de manière optimale les caméras montées sur B pour garder à tout moment la visibilité sur la pince de A.

Expérimentation générique de saisie

En collaboration avec des collègues du pôle Robotique et IA du LAAS-CNRS, nous avons participé à la définition et la préparation d'une expérimentation réelle, démonstration qui rendait nécessaire de porter nos algorithmes sur la plate-forme robotique JIDO déjà présentée en chapitre 2. Afin d'accomplir cette tâche, nous avons exploité différents modules qui étaient déjà embarqués sur le robot JIDO : commande du bras manipulateur (thèses de Ignacio Herrera [60] et Xavier Broquère [27]), planification de trajectoires (thèse de E.Lopez Damian [83])... sans évoquer les autres modules liés à l'interaction avec l'homme :

- I.Herrera [60] a réalisé sa thèse sur la commande souple de bras manipulateurs, commande nécessaire quand de tels bras doivent être exploités en milieu humain.
- E.López-Damian[83] a fait sa thèse sur la planification d'une trajectoire pour un bras robotique afin de pouvoir saisir un objet posé sur une table ; il s'est surtout intéressé à la recherche des positions de prise optimales.

Nous ne présentons pas ici l'intégralité du scénario du KE2 Cogniron, scénario rappelé au chapitre 2 : il inclue des séquences d'Interaction Homme-Robot sur lesquelles nous n'avons pas travaillé. Notre objectif va se centrer dans la prise d'un objet connu ou inconnu, placé sur une table. Le but final est que le système soit capable d'exécuter la séquence d'actions suivantes :

- dans un premier temps, détection d'un objet ou d'un amas d'objets (un *TRUC*) posé sur une table ;
- planification de la meilleure position d'accostage autour de la table pour pouvoir modéliser et/ou reconnaître le *TRUC* ;
- accostage le long de la table et positionnement de la caméra sur la demi-sphère centrée approximativement sur le *TRUC* ;
- tentative de reconnaissance de manière active à partir des images acquises en plusieurs positions autour de cet objet ;
- s'il n'est pas reconnu,

  - sélection de N positions pour le capteur stéréo sur la demi-sphère autour du *TRUC* ;
  - acquisition d'un nuage de points 3D sur le *TRUC* par recalage des N images de points 3D acquises par stéréo depuis les N positions du capteur ;
  - modélisation 3D du *TRUC* par triangulation du nuage de points 3D ;
  - sélection des positions de prise du *TRUC*, sélection d'une position et génération d'un mouvement pour aller en cette position (planification) ;
  - exécution de ce mouvement en boucle ouverte (commande).
- s'il est reconnu : le *TRUC* est en fait un objet de la classe C_i
  - localisation de l'objet dans un repère lié à la plate-forme robotique,
  - sélection de la position de prise la plus adéquate étant donnée la position courante de l'objet, génération d'un mouvement pour aller en cette position (planification) ;
  - exécution de ce mouvement en boucle ouverte (commande) ;

Il est supposé (1) que la scène n'est pas encombrée (en fait le *TRUC* est toujours un objet isolé) et (2) que le manipulateur est précis (ce qui est vrai) et que la tolérance sur la précision de la localisation est assez grande vis-à-vis de la précision de positionnement de la pince. De ce fait, le mouvement de positionnement de la pince se fait en boucle ouverte : en principe, il conviendrait de le faire en asservissement visuel sur des points suivis au fur et à mesure des déplacements de la pince.

Dans l'expérimentation compléte, une fois l'objet saisi, le robot devrait l'apporter à l'Homme, lui tendre, surveiller les mouvements de l'Homme et lâcher l'objet quand il est pris par l'Homme. L'ensemble de ce scénario a été exécuté pour le projet COGNIRON, mais sans traiter de la modélisation en ligne de l'objet 3D, et sans notre méthode de reconnaissance non finalisée à temps. Une méthode plus élémentaire a été intégrée sur JIDO pour reconnaître et localiser un objet 3D (module Ublob, fondé sur la détection d'un *blob* couleur pour reconnaître et localiser une bouteille) ; JIDO ne saisit que des bouteilles colorés aujourd'hui.

Pour saisir un objet, plusieurs points sont à considérer : comme c'est le seul objet présent sur la table, est-il possible que le bras puisse le prendre ? Rappelons qu'un bras manipulateur a des points faibles : il doit éviter les singularités des moteurs, à chaque fois qu'il lui est demandé de se déplacer d'un point à un autre. Par exemple, la possibilité qu'il passe dans un de ces points lorsqu'on lui demande de se déplacer d'un extrême à l'autre de la demi-sphère, est relativement haut : c'est exactement ce type de mouvement dont nous avons besoin, aussi bien dans la phase de modélisation que dans celle de reconnaissance.

Citons quelques raisons de notre renoncement à valoriser nos fonctions perceptuelles dans cette expérimentation, et de manière plus générale, à finaliser des expérimentations en ligne pour l'instant :

- le modèle géométrique du PA10 étant assez imprécis, son enveloppante est mal connue, et nous avons eu de nombreux problèmes pour estimer quelles étaient les positions atteignables sur la demi-sphère centrée sur l'objet.
- notre méthode de reconnaissance a été prototypée en Matlab, en exploitant le ToolKit Image : la réécriture en C ou C++ n'est pas achevée.

Environnement de simulation

Les environnements de simulation sont très utilisés dans le monde de la robotique puisqu'ils permettent d'analyser simplement comment un système évolue pendant le déroulement d'une tâche. Cela est possible grâce à la simulation de la physique et du comportement des capteurs ainsi que des éléments du monde physique qui vont interagir avec le système robotique. De cette manière nous pouvons simuler un environnement en considérant des collisions et l'action de la force de gravité sur les objets et sur le ou les robots.

Mentionnons quelques exemples de ce type de simulateur : Miller [93] a réalisé la saisie des objets en exploitant les primitives 3D qui composent l'objet. Au LAAS, le groupe de robotique a développé un environnement de simulation appelé Move3D [130], qui est surtout focalisé sur la planification des trajectoires libres de collisions ; il ne traite pas des effets dynamiques. Microsoft a aussi développé un environnement de simulation connu comme *Robotic Studio* [4] : cet environnement permet d'interagir avec des robots virtuels pour simuler leurs fonctionnements et valider les modules fonctionnels avant de les exporter sur le robot réel. Le trio Player/Stage/Gazebo [5] est un simulateur créé par l'Université of South California (USC), qui a l'avantage de fonctionner sous Linux ; il permet de simuler un très grand nombre de matériels y compris les capteurs (surtout la télémétrie laser ; les caméras sont simulées de

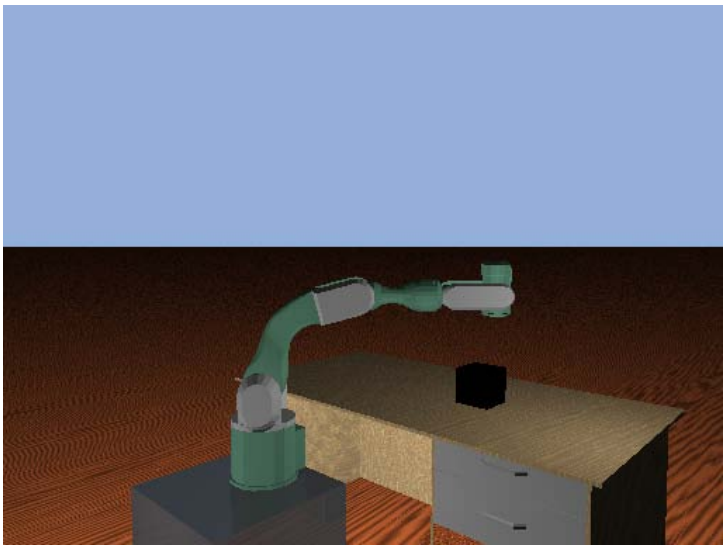


FIGURE 5.3 – Environnement de simulation

manière trop sommaire).

Nous avons développé notre propre environnement de simulation en Opengl (voir figure 5.3) en utilisant les modèles de la plate-forme JIDO et du bras PA10 existants au laboratoire. Nous avons simulé aussi la physique (force de gravité, rebond des objets...) afin d'avoir des collisions de robot manipulateur avec son environnement, mais nous n'avons pas traité les contraintes venant des singularités du bras. Nous simulons aussi la paire stéréo de caméras afin d'obtenir des images de points 3D nécessaires pour l'extraction des caractéristiques géométriques de l'objet.

Dans les sections suivantes nous allons montrer quelques tests de prise d'un objet. Nous présentons deux cas possibles : (1) l'objet est connu du système, c'est-à-dire l'objet a été déjà appris par le système et (2) on ne connaît pas l'objet avant de décider de le saisir.

5.3 Saisie d'un objet connu

Dans ce scénario le système va essayer de reconnaître l'objet qu'il voit. Comme le système ne sait pas s'il connaît ou ne connaît pas l'objet, il active en premier la phase de reconnaissance. S'il reconnaît l'objet, il le localise (typiquement avec les points d'intérêt appariés) et va le saisir.

Le première simulation a donc été la saisie d'un objet connu par le système robotique virtuel. Nous présentons les images en figure 5.4.

Dans ces images nous pouvons voir que le bras manipulateur s'approche de l'objet afin d'atteindre la position de prise.

5.4 Saisie d'un objet inconnu

Quand nous posons un objet inconnu devant le système, il va réaliser les traitements suivants :

- Essai de le reconnaître,
- si le système considère que l'objet a une forte probabilité d'être un objet présent dans sa base de données, il va essayer de le saisir comme dans le cas précédent : il y a bien sûr un grand risque d'échec.
- Dans le cas normal, où le système ne reconnaît pas l'objet, il passe en mode apprentissage afin de capturer ses caractéristiques d'apparence (pour le reconnaître la prochaine fois) et afin de reconstruire son modèle (afin de rechercher des positions de prise).
- une fois acquis toutes les caractéristiques, il sélectionne une position de prise, planifie une trajectoire dans l'espace libre vers cette position et procéde à la saisie de l'objet.

Dans les images de la figure 5.5, l'objet présenté au système est un objet qu'il n'a pas appris, mais qui a de fortes similarités avec une instance d'objet de la classe 1. C'est la raison pour laquelle le système l'a

FIGURE 5.4 – Reconnaissance de l'objet 5 et sa saisie par le bras manipulateur.

reconnu comme un objet de cette classe : et il va le saisir comme s'il s'agissait d'un objet de cette classe. En ce cas, la saisie se passe bien.

Dans le cas présenté en figure 5.6, le système reconnaît l'objet presenté comme s'il s'agissait d'un objet de la classe 2 : une sphère. Selon la position initiale du manipulateur et de l'objet, la saisie peut bien se passer ou non. En ce cas il a bien saisi l'objet, mais ce n'est pas toujours le cas comme on peut le voir en figure 5.7 dans laquelle la saisie de l'objet a échoué puisque le bras a fait tomber l'objet.

La figure 5.8 montre un objet que le système ne connaît pas ; en conséquence, il ne peut pas trouver des similarités avec les objets qui sont dans sa base de données. Donc il va apprendre le modèle afin de le rajouter dans cette base. La figure 5.9 montre comment les probabilités évoluent pendant la phase de reconnaissance active de cet objet : à la fin, il conclue à un échec de reconnaissance.

Une fois que le système sait qu'il ne connaît pas cet objet, il doit apprendre ses caractéristiques pour pouvoir le saisir. En figure 5.10, on montre différents descripteurs pour seulement un point de vue de l'objet inconnu pendant sa phase d'apprentissage. En même temps, le système peut aussi générer le modèle 3D du dit objet : les mêmes vues, acquises en stéréovision sur la demi-sphère centrée sur l'objet posé sur la table, sont exploitées à la fois pour apprendre la forme et l'apparence de l'objet. La figure 5.11 est le résultat de la reconstruction tridimentionnelle obtenue à partir de tous les points de vue sur l'objet. Finalement,les images de la figure 5.12 montrent la meilleure approche vers la position de prise de l'objet

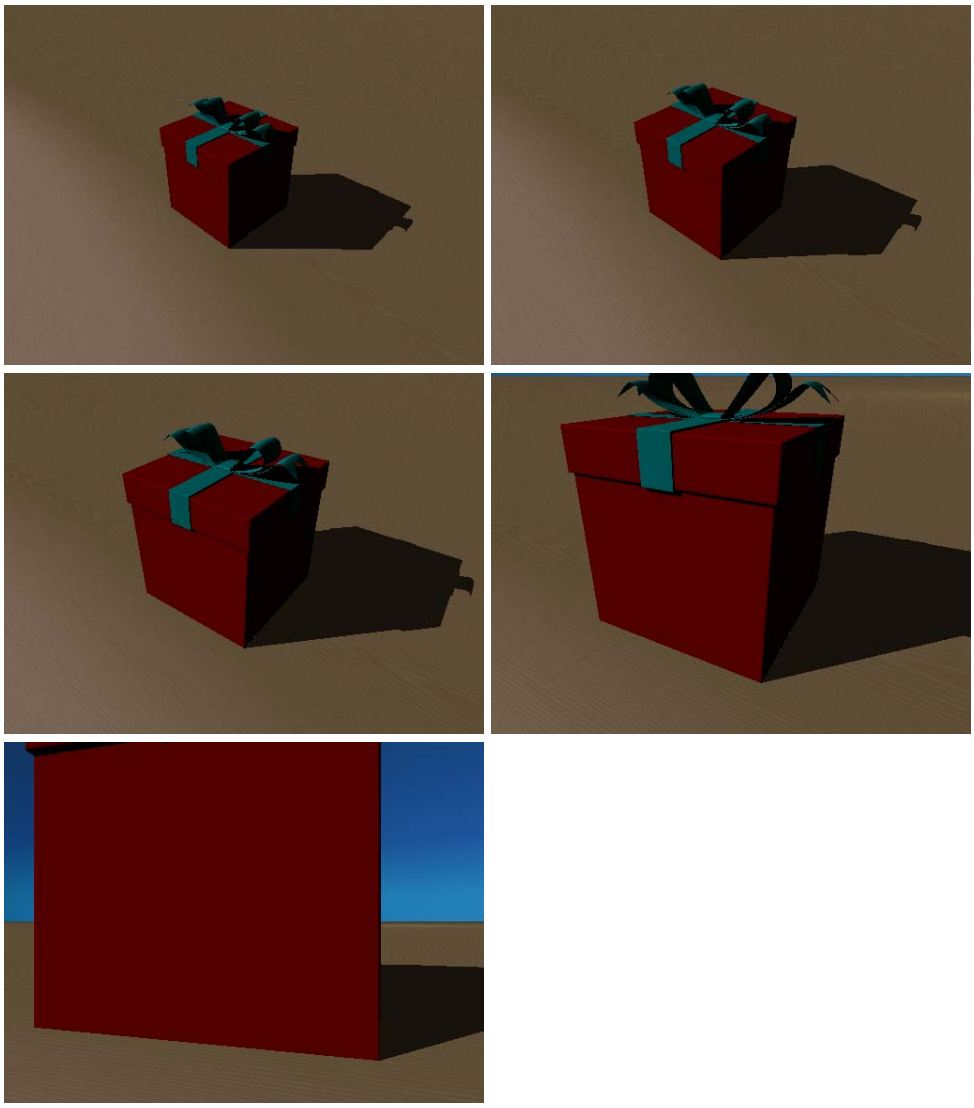


FIGURE 5.5 – Objet inconnu mais qui a été reconnu comme l'objet 1

qui vient d'être appris.

5.5 Conclusions.

Nous avons présenté dans ce chapitre, nos amorces de travaux sur la saisie d'objets, au préalable modélisés, reconnus et localisés par les approches que nous avons développées dans les chapitres précédents. Nous sommes conscients des limites de ces travaux, que nous n'avons pas pu mener à leur terme du fait de différents problèmes technologiques.

Il reste donc de nombreuses perspectives pour cette thématique, le but étant l'intégration d'un système de Robotique complet, capable d'exécuter des tâches complexes pour offrir de nouveaux services ; dans notre cas, nos travaux concernait le Robot Compagnon de l'Homme, capable d'assister l'Homme à son domicile ou dans des lieux publics.

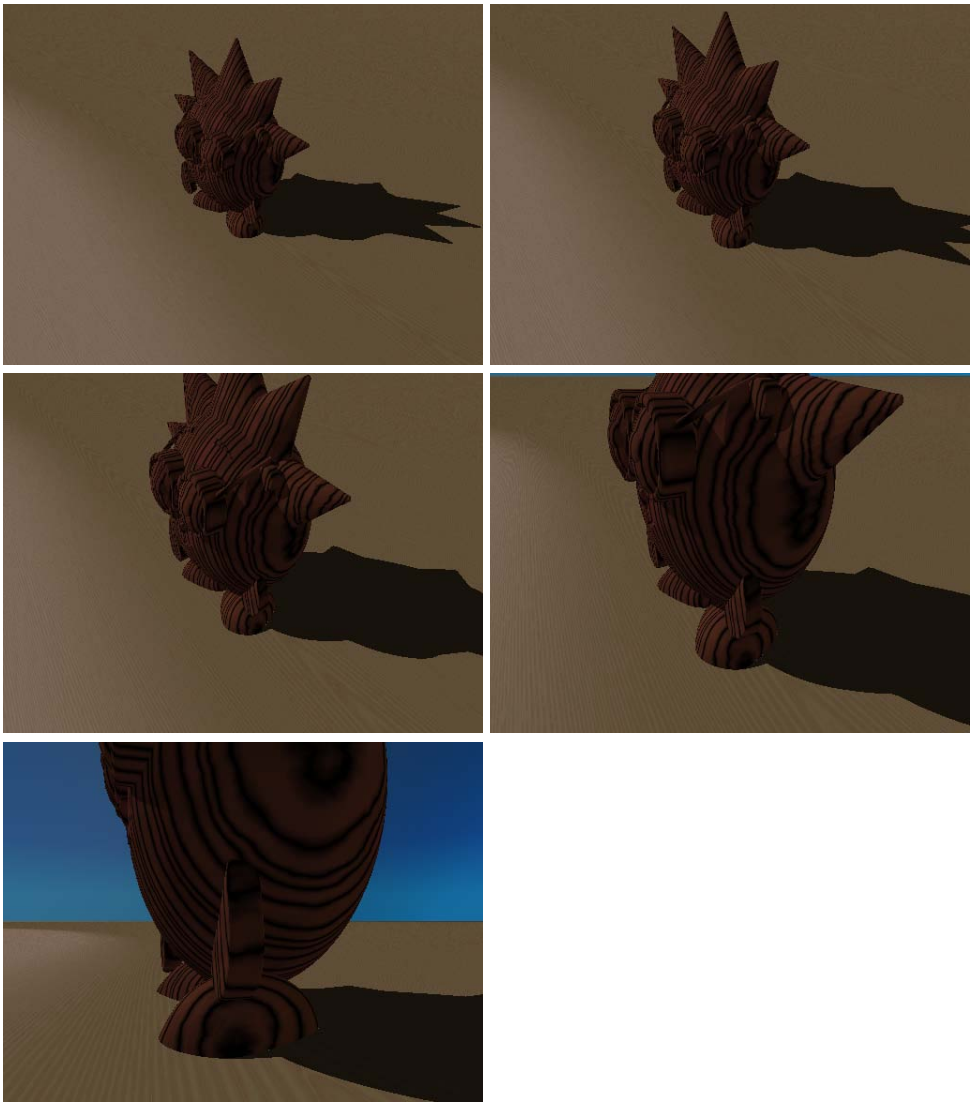


FIGURE 5.6 – Objet inconnu mais similaire à l'objet 2 : la saisie a réussi.

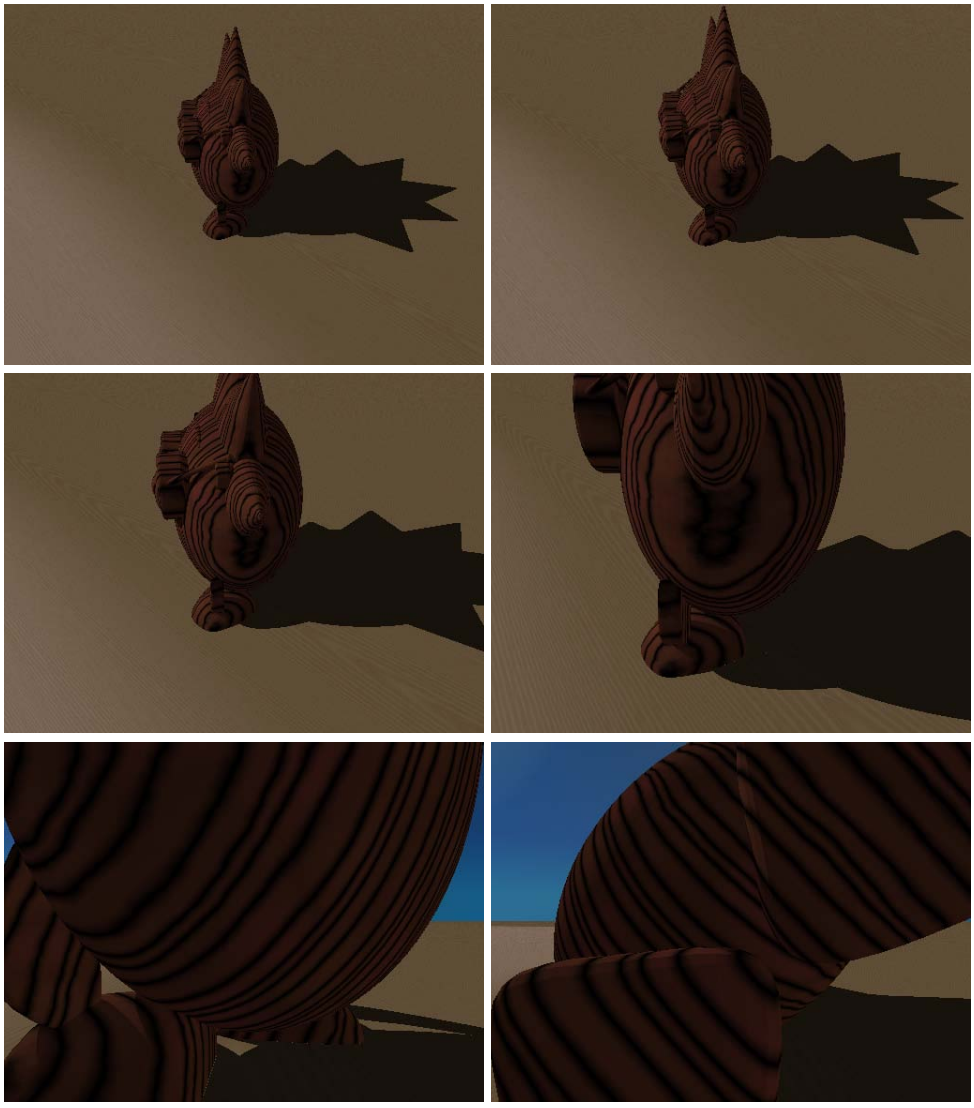


FIGURE 5.7 – Objet inconnu mais similaire à l'objet 2 : la saisie a échoué.

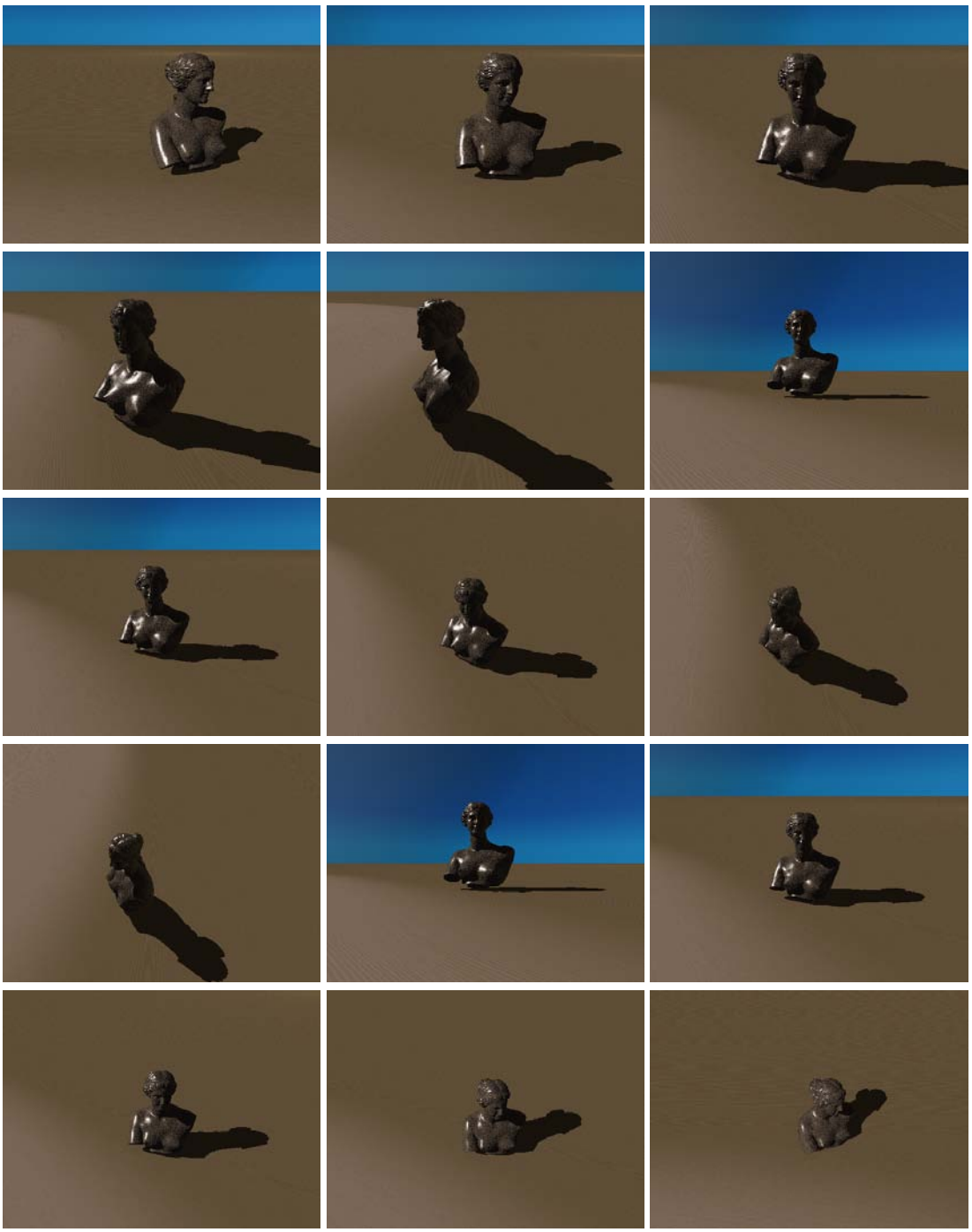


FIGURE 5.8 – Essai de reconnaissance de l'objet présenté.

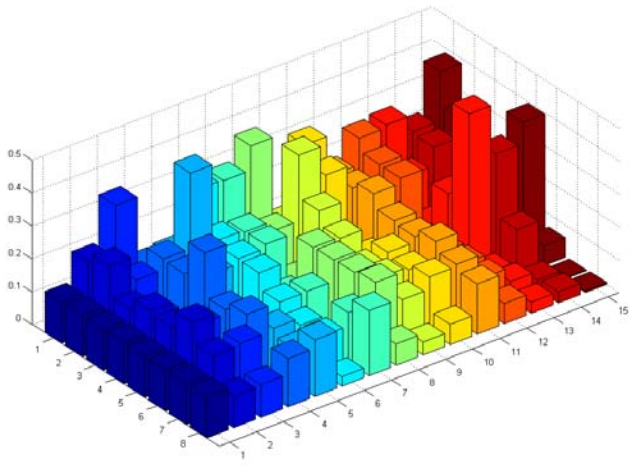


FIGURE 5.9 – Evolution des probabilités pendant la phase de reconnaissance de l'objet inconnu.

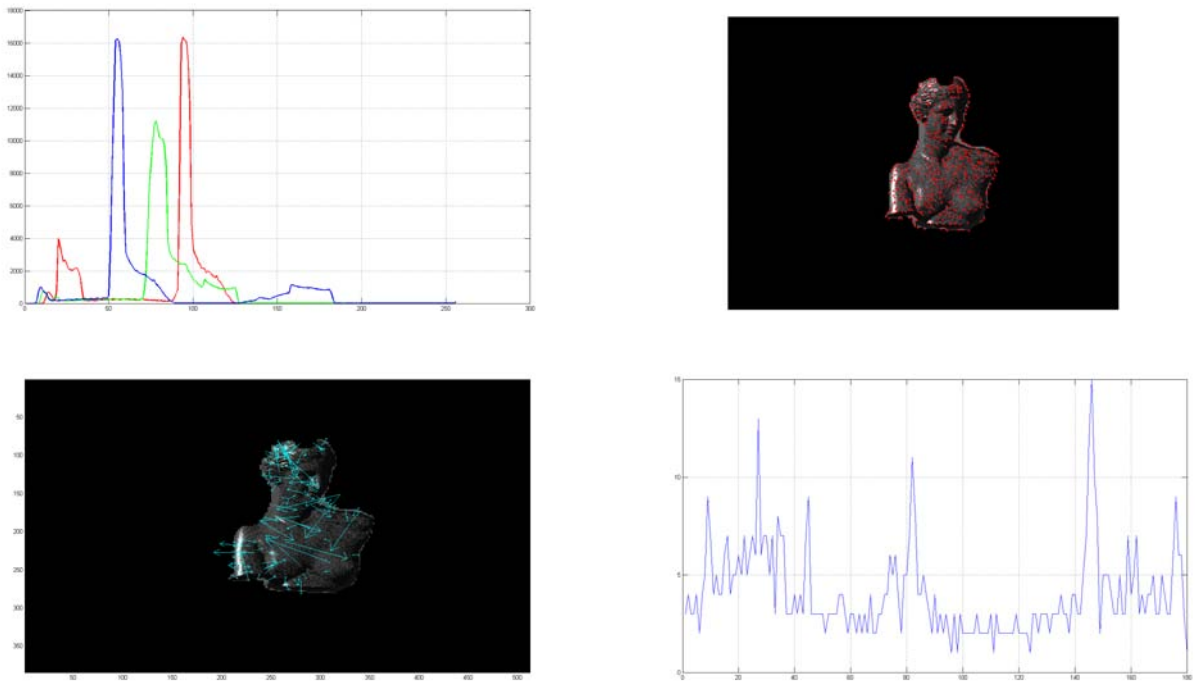


FIGURE 5.10 – Quelques caractéristiques de l'objet inconnu depuis un point de vue.

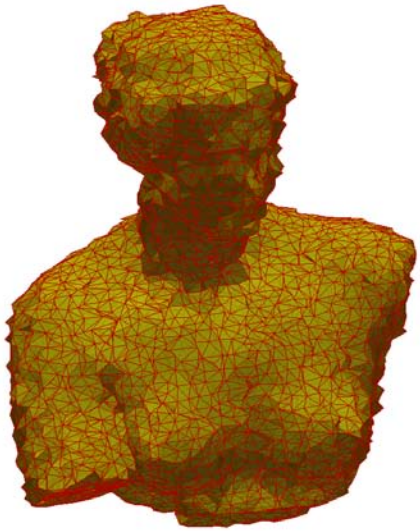


FIGURE 5.11 – Objet 3D reconstruit à partir des différentes images 3D acquises des différents points de vue de l'objet.

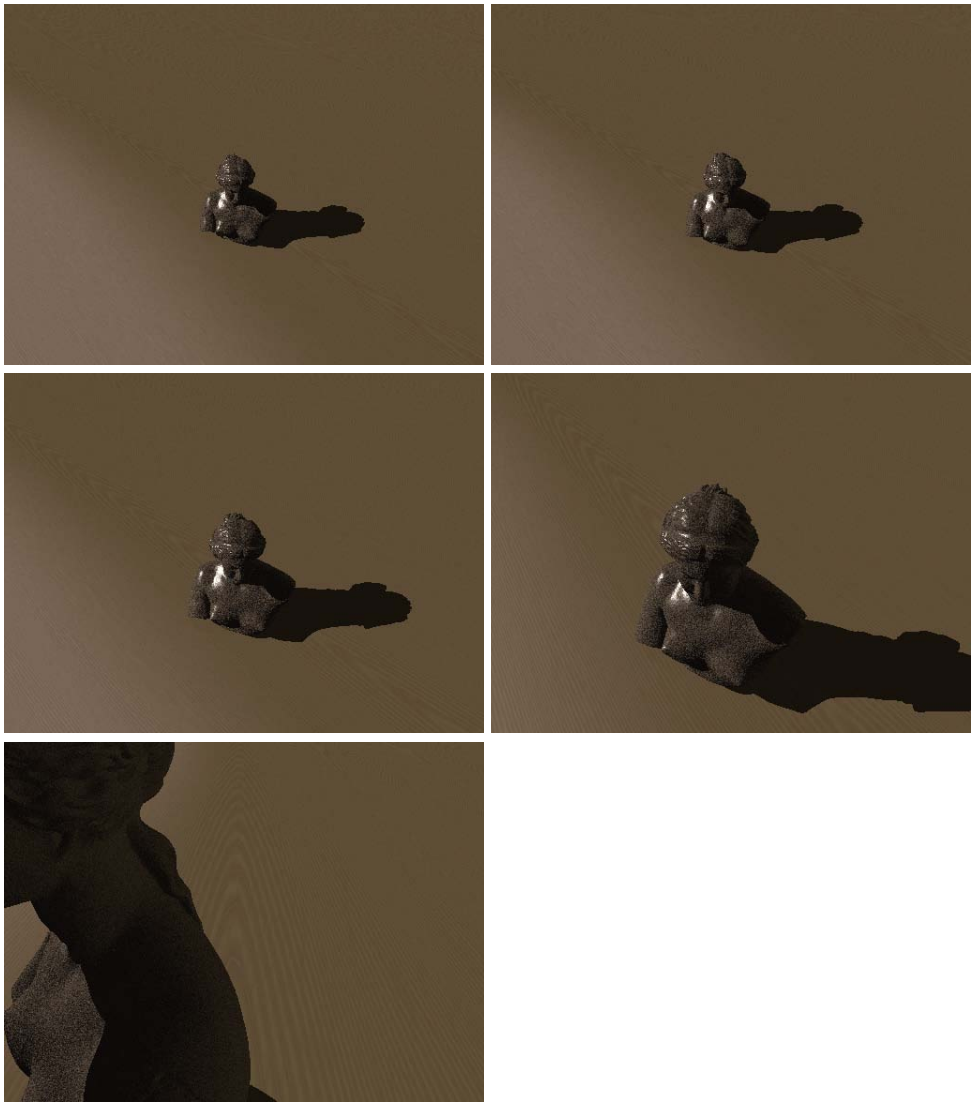


FIGURE 5.12 – Saisie d'un objet inconnu après l'avoir appris.

Chapitre 6

Conclusion

Dans ce chapitre nous allons d'abord rappeler quelles ont été nos objectifs et approts scientifiques, discuter plus avant de notre contribution sur la reconnaissance, puis finalement, indiquer les possibles améliorations ainsi que les perspectives pour la continuation de travaux sur ce sujet de travail.

6.1 Bilan du travail réalisé

Dans ce document, nous avons étudié les fonctions visuelles à intégrer sur un robot personnel, afin qu'il puisse manipuler des objets usuels, ceux que tout humain exploite à son domicile dans la vie courante. Nous supposons donc que ces objets sont de petite taille, et que le robot va les percevoir quand ils sont posés sur un plan, typiquement une table.

Deux tâches ont été considérées dans notre travail. D'abord nous avons étudié la fonction d'apprentissage d'un modèle 3D pour un tel objet, modèle construit depuis des données 3D acquises lorsque l'objet est perçu, isolé sur la table. Un tel modèle est requis pour la manipulation, afin de sélectionner les positions de prise en fonction de la géométrie du ou des préhenseurs. Ensuite, nous avons considéré la fonction de reconnaissance de ces objets, à partir de leurs propriétés photométriques. En exploitant ces caractéristiques liées à l'apparence, la vision monoculaire est suffisante pour reconnaître un objet préalablement appris.

Concernant la modélisation géométrique d'un objet 3D, nous avons presenté un système capable de reconstruire un objet réel, depuis des informations acquises en ligne par stéréovision. Nous devions initialement adapter et intégrer des fonctions existantes dans l'équipe, issues des travaux de A.Restrepo [133] : la tâche s'est révélée plus complexe que prévue, et nous avons dû proposer des contributions sur deux fonctions :

- L'ICP Pseudo-Couleur, car le recalage par l'algorithme ICP classique, ne donnait pas de bons résultats sur les données stéréo.
- La construction d'un maillage triangulaire en utilisant la paramétrisation sphérique, car de même, la méthode *Ball Pivoting* qui donnait de superbes modèles sur des données laser, s'est révélée très difficile à paramétrer avec des données stéréo. Par ailleurs cette méthode est très complexe, et une phase d'optimisation du codage devrait être envisagée avant toute exploitation en robotique.

Notre principale contribution concerne la fonction de reconnaissance. Fort de notre expérience acquise lors de travaux précédents au Mexique, nous avons proposé une méthode de reconnaissance active fondée sur la Maximisation de l'Information Mutuelle, et sur un ensemble de descripteurs attachés aux différents points de vue d'un objet 3D. Ces descripteurs sont appris en phase d'apprentissage, sur des images de l'objet isolé sur une table ; à terme le modèle d'apparence pourrait être construit en même temps que le modèle 3D. Ces descripteurs sont obtenus pour chaque point de vue, à partir des caractéristiques photométriques et géométriques extraites, qui sont : la couleur, la *Shape signature* ou signature polaire calculée sur la silhouette de l'objet depuis le point de vue considéré, le *Shape context* ou histogramme polaire des points de contours contenus à l'intérieur de cette silhouette, des points d'intérêt décrits par leurs descripteurs SIFT , et la densité sphérique, calculée à partir des données stéréo.

Il est certain que d'autres descripteurs pourraient être intégrés dans notre approche : descripteurs

sur la courbure, sur la texture ... En exploitant ces caractéristiques apprises sur les objets à reconnaître, nous avons proposé une stratégie active de reconnaissance, capable d'étiquetter tous les objets posés sur une table, à partir de déplacements successifs du capteur.

Finalement, nous avons produit un ensemble de fonctionnalités visuelles nécessaires pour :

1. Obtenir le modèle 3D d'un objet posé sur une table,
2. Apprendre les caractéristiques discriminantes sur le dit objet, perçu depuis un grand nombre de points de vue,
3. Reconnaître tous les objets appris que le robot est susceptible de trouver sur la table, y compris détecter que certains objets sont inconnus,
4. et enfin, Saisir un objet après l'avoir reconnu et localisé.

Bien que nous n'ayons pas achevé l'intégration de ces fonctionnalités sur le robot, nous avons montré par le biais de la simulation et en traitant des images réelles acquises par ailleurs, que les algorithmes fonctionnent.

6.2 Discussions sur la reconnaissance

L'algorithme que nous devions développer pour reconnaître un objet, devait exploiter à la fois des caractéristiques photométriques et géométriques, cela afin d'obtenir un système de reconnaissance robuste et stable : finalement, nous avons exploité seulement des caractéristiques photométriques dans nos évaluations, car il était plus facile de travailler en monoculaire.

Le système implémenté a montré sa capacité de généralisation, c'est-à-dire qu'il peut reconnaître des objets 3D même s'ils n'ont jamais été appris, dans la mesure où ils ont des caractéristiques similaires à ceux qui ont été appris et qui sont dans la base de données du système.

Les caractéristiques que nous avons utilisées pour modéliser un objet 3D de forme libre (couleur, forme, les points d'intérêt...) sont toujours présentes dans les objets de notre vie commune. En plus, l'utilisation de descripteurs globaux et locaux en même temps, et l'approche hiérarchique qui essaie d'abord la reconnaissance du modèle complet avant de reconnaître uniquement des sous-modèles de plus en plus petits, rendent possibles de détecter dans une scène, un objet même s'il est partiellement caché, comme on peut le constater dans nos résultats expérimentaux du chapitre 4.

Nous avons montré aussi bien avec des images de synthèse qu'avec des images réelles, que le système est capable d'abord de reconnaître un objet posé sur une table, puis de réaliser sa saisie à partir de l'information reçue des images acquises pendant le processus de reconnaissance : nous nous servons des dites images pour la localisation de l'objet, en exploitant l'algorithme ICP afin de faire coïncider les ensembles de points 3D acquis par la stéréovision, avec le modèle complet mémorisé dans la base de connaissances. Ainsi, nous pouvons obtenir la position dont nous avons besoin pour planifier et exécuter la tâche de saisie avec le manipulateur.

Malheureusement la simulation donne juste un aperçu des possibilités des fonctionnalités développées pour des applications de manipulation, une fois qu'elles seront intégrées avec des modules complémentaires qui traiteront de la recherche des positions de prise en fonction du modèle de la pince, la planification de la trajectoire jusqu'à cette position, le contrôle d'exécution de cette trajectoire...

Nous n'avons pas réussi à intégrer les fonctionnalités développées dans cette thèse sur le robot réel, partiellement à cause de divers problèmes avec le bras robotique : modèles direct et inverse du bras trop approximatifs, mauvaises détections des singularités des articulations ou des limites de l'espace de travail...

Le robot nous a seulement servi à acquérir des bases de données images, avec lesquelles nous avons évalué tant l'apprentissage que la reconnaissance active ; la procédure de reconnaissance n'a pas été exécutée en temps réel, mais nous avons néanmoins validé un système actif, en "piquant" dans notre base d'images, les images acquises depuis telle ou telle position de la caméra. Tout cela montre que notre algorithme devrait pouvoir rapidement être intégré dans le système embarqué d'un robot réel (Jido ou HRP2) en temps réel ; cela restera un point à réaliser dans les futurs travaux.

6.3 Perspectives

Les perspectives pour continuer notre travail sur cette thématique, concernent la fonction de reconnaissance d'une part, et l'intégration sur un robot réel d'autre part. Nous avons indiqué quelques pistes concernant la reconstruction 3D, en fin du chapitre deux. La perspective principale est de pouvoir intégrer dans leur totalité, nos algorithmes dans un système embarqué avec un robot manipulateur, type robot personnel (robot mobile équipé d'un bras) ou robot humanoïde. Notons que plusieurs de nos fonctions ont été seulement prototypées en Matlab : il conviendra de les programmer en C/C++.

Notre système de reconnaissance s'est révélé robuste avec l'ensemble des descripteurs que nous avons proposés afin de décrire un objet, perçu isolé depuis plusieurs points de vue en phase d'apprentissage. Afin d'obtenir la robustesse aux occultations, ces descripteurs sont appliqués de manière hiérarchique à la vue complète de l'objet, puis à des vues partielles obtenues par un découpage récursif ; cette exploitation d'un modèle hiérarchisée n'est pas optimale dans l'implémentation actuelle ; il conviendrait de l'améliorer.

Nous avons cité les travaux de D.Lowe, qui n'exploite qu'un seul type de caractéristiques pour la reconnaissance : les points d'intérêt dotés de descripteurs SIFT. Dans le cas de données sensorielles mixtes 2D et 3D, il resterait à trouver un descripteur mixte, qui permette d'avoir une description globale et locale d'un objet réel afin de pouvoir ainsi le numériser, puis le reconnaître a posteriori.

Concernant les descripteurs sur la forme 3D, nous n'avons considéré que la densité sphérique, alors que de nombreux descripteurs liés à la courbure ont été proposés dans la littérature. Pour apprendre et exploiter de telles caractéristiques, une perspective importante serait de choisir un autre capteur pour l'acquisition de données 3D, capteur exploitant d'autres technologies que la stéréovision, et qui soit réaliste pour un robot personnel. S'il est bien vrai qu'un capteur stéréo est similaire aux senseurs biologiques (nos yeux par exemple) générés par la nature pour acquérir des données 3D, la stéréo temps réel, bien que fonctionnelle, n'est pas robuste. Le démonstrateur Jido du LAAS par exemple, est équipé avec un capteur optique à temps de vol développé par CSEM en Suisse, le Swiss Ranger : nous n'avons pas eu le temps de l'exploiter.

Dans tous les systèmes de reconnaissance d'objets ou de classe d'objets, l'apprentissage aujourd'hui, se fait de manière supervisé : un opérateur humain montre les objets au robot, généralement isolés et sur un fond uniforme ; il associe un nom symbolique à ces objets. M.Cottret [37] a proposé une approche préliminaire sur l'apprentissage automatique des objets présents dans un environnement humain par un robot personnel équipé uniquement de caméras. Dans une phase pré-attentive, le robot détecte des structures locales, l'équivalent de nos *TRUCs* posés sur des tables ; puis dans une phase attentive, le robot déplace sa caméra autour de ces structures pour générer un modèle d'apparence, dans son cas, un ensemble de points d'intérêt associés à des descripteurs SIFT. Même si les résultats obtenus étaient très préliminaires, une intégration avec nos propres travaux sur la reconnaissance, permettrait d'améliorer la représentation et la reconnaissance des structures locales.

Dans ce contexte de l'apprentissage automatique, le Robot pourrait-il se passer de l'Homme pour déterminer si un *TRUC* est un objet isolé ou un ensemble d'objets en vrac ? En exploitant des raisonnements géométriques sur le modèle construit pour un *TRUC*, puis en exécutant des mouvements adaptés avec le manipulateur, le robot pourrait-il déterminer à quoi il a affaire ? dans le cas d'objets en vrac, pourrait-il isoler les objets [1] et les apprendre ou les reconnaître les uns après les autres ?

1. On évitera d'expérimenter avec des objets fragiles au début !!!

Bibliographie

[1] http://graphics.stanford.edu/data/3Dscanrep/.

[2] http://www.vtk.org/index.php.

[3] http://www.turbosquid.com/.

[4] http://msdn.microsoft.com/en-us/robotics/default.aspx.

[5] http://playerstage.sourceforge.net/gazebo/gazebo.html/.

[6] H. Abdi, D. Valentin, and B. Edelman. Eigenfeatures as intermediate level representations : the case for pca models. *Brain and Behavioral Sciences*, 21 :175–177, 1998.

[7] S. Abrams, P. Allen, and K. Tarabanis. Computing camera viewpoints in an active robot workcell. *Int. Journal of Robotics Research (IJRR)*, 18(3) :267–285, March 1999.

[8] S. Akkouche and E. Galin. Adaptive implicit surface polygonization using marching triangles. *Comput. Graph. Forum*, 20(2) :67–80, 2001.

[9] M.T. Lozano Albalate, M. Devy, and J.M. Sanchiz-Marti. Perception planning for an exploration task of a 3d environment. In *Proc. Int. Conf. on Pattern Recognition (ICPR), Québec, Canada*, August 2002.

[10] J. Aloimonos. Purposive and qualitative vision. In *Proc. 1st European Conf. on Computer Vision (ECCV)*, May 1990.

[11] N. Amenta and M.W. Bern. Surface reconstruction by voronoi filtering. *Discrete & Computational Geometry*, 22(4) :481–504, 1999.

[12] N. Amenta, S. Choi, and R.K. Kolluri. The power crust. In *Proc. Symposium on Solid Modeling and Applications*, pages 249–266, 2001.

[13] J.G. Aviña-Cervantes, L. Martínez-Jiménez, M. Devy, A. Hernández-Gutierrez, D.L. Almanza, and M.A. Ibarra. Shadows attenuation for robust object recognition. In *Proc. Mexican Int. Conf. on Artificial Intelligence (MICAI)*, pages 650–659, 2007.

[14] J.G. Aviña-Cervantes. *Navigation visuelle d'un robot mobile dans un environnement d'extérieur semi-structuré*. PhD thesis, LAAS-CNRS, Université de Toulouse, February 2005.

[15] V. Ayala-Ramírez and M. Devy. Active selection and tracking of multiple landmarks for visual navigation. In *Proc. 2nd Int. Symp. on Robotics and Automation (ISRA)*, pages 557–562, Monterrey, Mexico, November 2000.

[16] V. Ayala-Ramírez, J.B. Hayet, F. Lerasle, and M. Devy. Visual localization of a mobile robot in indoor environnements using planar landmarks. In *Proc. IEEE/RSJ Int. Conf on Intelligent Robots and Systems (IROS)*, volume 1, pages 275–280, Takamatsu, Japan, November 2000. IEEE Press.

[17] R. Bajcsy and M. Campos. Active and exploratory perception. In *Proc. CVGIP Image Understanding*, July 1992.

[18] D.H. Ballard and C.M. Brown. *Computer Vision*. Prentice-Hall Inc., 1982.

[19] E. Bardinet, L.D. Cohen, and N. Ayache. Superquadrics and free-form deformations : A global model to fit and track 3d medical data. In Nicholas Ayache, editor, *Proc. 1st Int. Conf. Computer Vision, Virtual Reality and Robotics in Medicine (CVRMed)*, volume 905 of *Lecture Notes in Computer Science*, pages 319–326. Springer, 1995.

[20] H. Bay, T. Tuytelaars, and L. Van Gool. Surf : Speeded up robust features. In *Proc. 9th European Conf. on Computer Vision (ECCV), Austria*, May 2006.

[21] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts, 2001.

[22] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 24(4) :509–522, 2002.

[23] F. Bernardini, J. Mittleman, H.E. Rushmeier, C.T. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Trans. Vis. Comput. Graph.*, 5(4) :349–359, 1999.

[24] P.J. Besl and N.D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 14(2) :239–256, 1992.

[25] C. Breaeal. Towars sociable robots. In *Proceedings Workshop Robots as Parterns at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Switzerland, September 2002. EPFL Lausanne.

[26] C. Breazeal and J. Velasquez. Toward teaching a robot infant using emotive communications acts. Technical report, MIT report, Cambrige, USA, 2000.

[27] X. Broquere, D. Sidobre, and I. Herrera Aguilar. Soft motion trajectory planner for service manipulator robot. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Nice (France)*, 2008.

[28] H.H. Buelthoff and S. Edelman. Psychophysical support for a 2d view interpolation theory of object recognition. *Proc. National Academy of Science*, 89 :60–64., 1992.

[29] R. Campbell and P. Flynn. Eigenshapes for 3d object recognition in range data. In *Proc. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 505–510, Fort Collins - Colorado, USA., 1999.

[30] F. Chausse. *Un système de perception attentionnelle pour les robots mobiles en environnement extérieur*. PhD thesis, Habilitation à Diriger des Recherches, LASMEA, Université Blaise Pascal de Clermont-Ferrand, May 2008.

[31] H. Chen and B. Bhanu. 3d free-form object recognition in range images using local surface patches. In *Proc. Int. Conf. on Pattern Recognition (ICPR 3)*, pages 136–139, 2004.

[32] Y. Chen and G.G. Medioni. Object modelling by registration of multiple range images. *Image Vision Comput. (IVC)*, 10(3) :145–155, 1992.

[33] D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek. The trimmed iterative closest point algorithm. In *Proc. 16th Int. Conf. on Pattern Recognition (ICPR), Quebec City, Canada*, pages 545–548, 2002.

[34] C. Chow, H. Tsui, and T. Lee. Surface registration using a dynamic genetic algorithm. *Pattern Recognition*, 37 :105–117, 2004.

[35] C.S. Chua and R. Jarvis. Point signatures : A new representation for 3d object recognition. *Int. Journal Computer Vision (IJCV)*, 25(1) :63–85, 1997.

[36] Microsoft Co. Microsoft encarta 2005 encyclopedia. CD. Microsoft Home, 2005.

[37] M. Cottret. *Exploration visuelle d'environnement intérieur par détection et modélisation d'objets saillants*. PhD thesis, LAAS-CNRS, Institut National Polytechnique de Toulouse, October 2007.

[38] R. Cowie. Emotion recognition in human-computer interaction. *IEEE Signal Processing Magazine*, January 2001.

[39] C. de Trazegnies, J. Bandera, C. Urdiales, and F. Sandoval. A real 3d object recognition algorithm based on virtual training. In *Proc. IASTED Conf. on signal processing, pattern recognition and applications, (SSPRA)*, pages 342–347., 2003.

[40] J. Denzler and C. Brown. Optimal selection of camera parameters for state estimation of static systems : An information theoretic approach. Technical Report 732, The University of Rochester, New York, August 2000.

[41] J. Denzler, C. Brown, and H. Niemann. *Pattern Recognition – 23rd DAGM Symposium*, chapter Optimal Camera Parameters for State Estimation with Applications in Object Recognition, pages 305–312. Springer, Berlin, September 2001.

[42] D. Djian and P. Rives. Reconnaissance d'objets 3d par vision active : Apprentissage de réseaux bayésiens. In *Proc. 11ième Congrés AFRIF/AFIA Reconnaissance des Formes et Intellignece Artificielle (RFIA), Clermont Ferrand, France*, January 1998.

[43] F. Dornaika. *Contributions à l'intégration Vision-Robotique : calibrage, localisation et Asservissement.* PhD thesis, Institut National Polytechnique de Grenoble, 1995.

[44] S. Druon, A. Crosnier, and M.J. Aldon. Recalage de masses de données 3d/couleur non structurées. In *Proc. Journées COmpression et REprésentation des Signaux Audiovisuels (CORESA)*, November 2006.

[45] R. San José Estépar, A. Brun, and C.F. Westin. Robust generalized total least squares iterative closest point registration. In *Proc. Mexican Int. Conf. on Artificial Intelligence (MICAI 1)*, pages 234–241, 2004.

[46] Projet européen COGNIRON (2004-2008). http ://www.cogniron.org.

[47] T.P. Fang and L.A. Piegl. Delaunay triangulation using a uniform grid. *IEEE Computer Graphics and Applications*, 13(3) :36–47, May 1993.

[48] T.P. Fang and L.A. Piegl. Delaunay triangulation in three dimensions. *IEEE Computer Graphics and Applications*, 15(5) :62–69, September 1995.

[49] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition*, volume 2, pages 264–271, June 2003.

[50] K. Fujimura and Y´. Sako. Shape signature by deformation. In *Proc. IEEE Int. Conf. on Shape Modeling and Applications*, pages 225–. IEEE Computer Society, 1999.

[51] S. Gao and H.Q. Lu. A fast algorithm for delaunay based surface reconstruction. In *Proc. 11th Int. Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision*, February 2003.

[52] M. Garland and P.S. Heckbert. Surface simplification using quadric error metrics. In *Proc. annual Conf. on Computer graphics and interactive techniques (SIGGRAPH)*, pages 209–216, 1997.

[53] M. Garland and P.S. Heckbert. Simplifying surfaces with color and texture using quadric error metrics. In *Proc. IEEE Conf. Visualization*, pages 263–269, 1998.

[54] V. Garric. *Vision pour la robotique de manipulation : Calibration, localisation et saisie d'objets.* PhD thesis, LAAS-CNRS, Université de Toulouse, May 1996.

[55] C. Gotsman, X. Gu, and A. Sheffer. Fundamentals of spherical parameterization for 3d meshes. *ACM Trans. on Graphics*, 22(3) :358–363, July 2003.

[56] M. Grabner and H. Bischof. Object recognition based on local feature trajectories. In *Proc. 1st Cognitive Vision Workshop, OCG Oesterreichische Computer Gesellschaft*, 2005.

[57] C. Harris and M. Stephens. A combined corner and edge detection. Technical report, Plessey Research Roke Manor, UK, 1988.

[58] J.B. Hayet, F. Lerasle, and M. Devy. A visual landmark framework for mobile robot navigation. *Image and Vision Computing*, 25(8) :1341–1351, 2007.

[59] P.S. Heckbert and M. Garland. Optimal triangulation and quadric-based surface simplification. *Comput. Geom.*, 14(1-3) :49–65, 1999.

[60] I. Herrera-Aguilar. *Commande des bras manipulateurs et retour visuel pour des applications à la robotique de service.* PhD thesis, LAAS-CNRS, Université de Toulouse, September 2007.

[61] A. Hilton and J. Illingworth. Marching triangles : Delaunay implicit surface triangulation. Technical Report CVSSP 01, University of Surrey, January 1997.

[62] A. Hilton, A. Stoddart, J. Illingworth, and T. Windeatt. Marching triangles : Range image fusion for complex object modelling. In *Proc. IEEE Int. Conf. on Image Processing (ICIP)*, volume 2, pages 381–384, Laussane, 1996.

[63] A. Johnson and M. Hebert. Recognizing objects by matching oriented points. Technical Report CMU-RI-TR-96-04, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 1996.

[64] A. Johnson and M. Hebert. Control of polygonal mesh resolution for 3-d computer vision. Technical Report CMU-RI-TR-96-20, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, April 1997.

[65] A.E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 21(5) :433–449, 1999.

[66] A.E. Johnson and S.B. Kang. Registration and integration of textured 3d data. *Image Vision Comput. (IVC)*, 17(2) :135–147, 1999.

[67] S. Jonquières. *Application des reseaux bayésiens à la reconnaissance active d'objets 3D : Contribution à la saisie*. PhD thesis, LAAS-CNRS, Université de Toulouse, January 2000.

[68] I.K. Jung and S. Lacroix. A robust interest point matching algorithm. In *Proc. Int. Conf. on Computer Vision (ICCV)*, Vancouver (Canada), July 2001.

[69] T. Kadir and M. Brady. Scale, saliency and image description. *Int. Journal on Computer Vision (IJCV)*, 2001.

[70] G. Karras and D. Mavrommati. Simple calibration techniques for non-metric cameras. In *Proc. Int. CIPA Symposium*, septembre 2001.

[71] M. Kass, A. Witkin, and D. Terzopoulos. Snakes : Active contour models. *Int. Journal of Computer Vision (IJCV)*, 1(4) :321–331, 1988.

[72] Y. Ke and R. Sukthankar. Pca-sift : A more distinctive representation for local image descriptors. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[73] S. Kim, K. Kim, and W. Woo. 3d registration for image-based virtual environment generation using color and depth information. In *Proc. IEEE Int. Conf. on Image Processing (ICIP), Singapore*, 2004.

[74] Kitware. *VTK User's Guide : The Visualization Toolkit User's Guide*. ISBN 1-930934-13-0. Kitware, Inc publishers, 28 Corporate Drive, Clifton Park, NY 12065 USA.

[75] R. Klette, K. Schlüns, and A. Koschan. *Computer Vision : Three-Dimensional Data from Images*. Springer, 1^{st} reprint 2001 edition, 1998.

[76] J.M. Lagarde, M. Devy, L. Moreno, P. Lacroix, P. Le Coutaller, and M. Briot. Projet body scan : modélisation corporelle par profilométrie. In *Proc. 14ème Congrès Francophone AFRIF-AFIA de Reconnaissance des Formes & Intelligence Artificielle (RFIA), Toulouse (France), Vol.2, pp.883-892, Rapport LAAS N.04653*, January 2004.

[77] P. Lasserre and P. Grandjean. Stereo Vision Improvments. In *Proc. IEEE Int. Conf. on Advanced Robotics (ICAR)*, pages 679–685, Barcelona (Spain), September 1995.

[78] A. Leonardis and H. Bischof. Robust recognition using eigenimages. *Computer Vision and Image Understanding (CVIU)*, 78(1) :99–118, 2000.

[79] A. Leonardis, A. Jaklic, and F. Solina. Superquadrics for segmenting and modeling range data. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 19(11) :1289–1295., 1997.

[80] Y.F. Li, B. Hea, and P. Bao. Automatic view planning with self-termination in 3d object reconstructions. *Elsevier Sensors and Actuators A : Physical*, 122(2) :335–344, August 2005.

[81] W. Liu, C. Zhang, and B. Yuan. Superquadric-based reconstruction from 2d images. In *Proc. 6th Int. Conf. on Signal Processing*, volume 1, pages 668–671, August 2002.

[82] E. Lomonosov, D. Chetverikov, and A. Ekárt. Pre-registration of arbitrarily oriented 3d surfaces using a genetic algorithm. *Pattern Recognition Letters*, 27 :1201–1208, 2006.

[83] E. Lopez-Damian. *Grasp planning for object manipulation by an autonomous robot*. PhD thesis, LAAS-CNRS, Université de Toulouse, July 2006.

[84] W.E. Lorensen and H.E. Cline. Marching cubes : A high resolution 3d surface construction algorithm. In *Proc. 14th annual Conf. on Computer graphics and interactive techniques (SIGGRAPH)*, pages 163–169, New York, NY, USA, July 1987. ACM Press.

[85] M.I.A. Lourakis and A.A. Argyros. The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm. Technical Report 340, Institute of Computer Science - FORTH, Heraklion, Crete, Greece, Aug. 2004. Available from `http://www.ics.forth.gr/~lourakis/sba`.

[86] D.G. Lowe. Object recognition from local scale-invariant features. In *Proc. Int. Conf. on Computer Vision (ICCV)*, pages 1150–1157, Corfu, Greece, September 1999.

[87] D.G. Lowe. Local feature view clustering for 3d object recognition. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 682–688, Kauai, Hawaii, December 2001.

[88] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision (IJCV)*, 60(2) :91–110, 2004.

[89] S. Melax. A simple, fast and effective polygon reduction algorithm. Game Developer Magazine, November 1998.

[90] X. Meng and Z. Hu. A new easy camera calibration technique based on circular points. *Pattern Recognition*, 36(5) :1155–1164., 2003.

[91] A.S. Mian, M. Bennamoun, and R.A. Owens. A novel representation and feature matching algorithm for automatic pairwise registration of range images. *Int. Journal of Computer Vision (IJCV)*, 66(1) :19–40, 2006.

[92] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *Proc. Int. Conf. on Computer Vision (ICCV)*, 2001.

[93] A.T. Miller. *GraspIt ! : A Versatile Simulator for Robotic Grasping.* PhD thesis, Department of Computer Science, Columbia University, June 2001.

[94] A.T. Miller, S. Knoop, P.K. Allen, and H.I. Christensen. Automatic grasp planning using shape primitives. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1824–1829, September 2003.

[95] M. Mitran and F.P. Ferrie. Active surface reconstruction using the gradient strategy. In A. Heyden et al., editor, *Proc. European Conf. on Computer Vision (ECCV)*, number 2353 in LNCS, pages 267–281, 2002.

[96] N. Trujillo Morales. *Stratégie de perception pour la compréhension de scènes par une approche focalisante , appplication à la reconnaissance d'objets.* PhD thesis, LASMEA, Université Blaise Pascal de Clermont-Ferrand, December 2007.

[97] H. Murase and S.K. Nayar. Visual learning and recognition of 3d objects from appearance. *Int. Journal of Computer Vision (IJCV)*, 14 :5–24., 1995.

[98] A.A.Y. Mustafa. Fuzzy shape matching with boundary signatures. *Pattern Recognition Letters*, 23(12) :1473–1482, 2002.

[99] A.A.Y. Mustafa, L.G. Shapiro, and M.A. Ganter. 3d object identification with color and curvature signatures. *Pattern Recognition*, 32(3) :339–355, 1999.

[100] J. Månsson. Stereovision : A model of human stereopsis. Technical Report 8453, Lund University Cognitive Studies, Kungshuset, Lundagård S-222 22 Lund, Sweden, 1998.

[101] A.S. Oikonomopoulos N.A. Bompos, P.K. Artemiadis and K.J. Kyriakopoulos. Modeling, full identi ?cation and control of the mitsubishi pa-10 robot arm. In *Proc. IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics (AIM), Switzerland*, 2007.

[102] P. Nagabhushan, D.S. Guru, and B.H. Shekar. Visual learning and recognition of 3d objects using two-dimensional principal component analysis : A robust and an efficient approach. *Pattern Recognition*, 39(4) :721–725, 2006.

[103] K. Nagao and W.E.L. Grimson. Using photometric invariants for 3d object recognition. *Computer Vision and Image Understanding (CVIU)*, 71(1) :74–93, 1998.

[104] G.M. Nielson. Dual marching cubes. In *Proc. Conf. on Visualization (VIS '04)*, pages 489–496, Washington, DC, USA, 2004. IEEE Computer Society.

[105] A. El Oirrak, D. Aboutajdine, and M. Daoudi. Discrimination d'objets 2d ou 3d utilisant des histogrammes polaires ou sphériques. In *Proc. 9èmes Journées COmpression et Représentation des SIgnaux Audiovisuels (CORESA)*, Lille, France, May 2004.

[106] S.W. Park and K.S. Hong. Practical ways to calculate camera lens distortion for real time camera. *Pattern Recognition*, 34(6) :1199–1206., 2001.

[107] J. Pearl. *Probabilistic Reasoning in intelligent systems : Network of Plausible inference.* Morgan Kauffman, 1988.

[108] R. Pito. *automated surface acquisition using range cameras.* PhD thesis, University of Pennsylvania, 1996.

[109] R. Pito. A sensor based solution to the next best view problem. In *Proc. Int. Conf. on Pattern Recognition (ICPR)*, 1996.

[110] R. Pito. A solution to the next best view problem for automated surface acquisition. *IEEE Trans.Pattern Analysis and Machine Intelligence (PAMI)*, 21(10) :1016–1030, 1999.

[111] R. Pito and R. Bajcsy. A solution to the next best view problem for automated cad model acquisition of free-form objects using range cameras. Technical Report MS-CIS-95-23, Department of Computer and Information Science, University of Pennsylvania, Levine Hall, 3330 Walnut Street, Philadelphia, PA 19104-6389, 1995.

[112] T. Poggio and S. Edelman. A network that learns to recognize threedimensional objects. *Nature*, 343 :263–266., 1990.

[113] K. Pulli, H. Abi-Rached, T. Duchamp, L.G. Shapiro, and W. Stuetzle. Acquisition and visualization of colored 3d objects. In *Proc. 14th Int. Conf. on Pattern Recognition (ICPR), Brisbane, Australia*, pages 11–15, August 1998.

[114] R.D. Rimey and C.M. Brown. Controlling eye movements with hidden markov models. *Int. Journal on Computer Vision (IJCV)*, 1991.

[115] R.D. Rimey and C.M. Brown. Control of selective perception usig bayes nets and decision theory. *Int. Journal on Computer Vision (IJCV)*, 1994.

[116] C. Robertson and R. Fisher. Parallel evolutionary registration of range data. *Computer Vision and Image Understanding (CVIU)*, 87 :39–50, 2002.

[117] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Proc. 3rd Int. Conf. on 3D Digital Imaging and Modeling (3DIM)*, June 2001.

[118] A. Sadiq, R. Oulad Haj Thami, M. Daoudi, and J.P. Vandeborre. Classification des objets 3d basée sur la logique floue. In *Proc. 9èmes Journées COmpression et Représentation des SIgnaux Audiovisuels (CORESA)*, Lille, France, May 2004.

[119] H. Saito and M. Kimura. Superquadric modeling of multiple objects from shading images using genetic algorithms. *Industrial Electronics, Control, and Instrumentation*, 3 :1589–1593, August 1996.

[120] S. Salamanca, C. Cerrada, A. Adán, and M. Adán. Partial views matching using a method based on principal components. In H. Araújo, A. Vieira, J. Braz, B. Encarnação, and M. Carvalho, editors, *Proc. 1st Int. Conf. on Informatics in Control, Automation and Robotics (ICINCO)*, pages 3–10. INSTICC Press, 2004.

[121] S. Salamanca, C. Cerrada, A. Adán, P. Merchán, and M. Adán. Matching de vistas parciales basado en el cálculo de componentes principales. XXIV Jornadas de Automática, September 2003.

[122] J. Sanchiz and R. Fisher. A next-best-view algorithm for 3d scene recovery with 5 degrees of freedom. In *Proc. British machine vision conference (BMVC), Nottingham, UK*, 1999.

[123] S. Schaefer and J.D. Warren. Dual marching cubes : Primal contouring of dual grids. In *Proc. Pacific Conf. on Computer Graphics and Applications*, pages 70–76. IEEE Computer Society, 2004.

[124] C. Schmid, R. Mohr, and C. Bauckhage. Comparing and evaluating interest points. In *Proc. Int. Conf. on Computer Vision (ICCV)*, pages 230–235, 1998.

[125] W.J. Schroeder, J.A. Zarge, and W.E. Lorensen. Decimation of triangle meshes. In *Proc. annual Conf. on Computer graphics and interactive techniques (SIGGRAPH)*, pages 65–70. ACM, 1992.

[126] V. Sequeira, J.G.M. Gonçalves, and M.I. Ribeiro. 3d reconstruction of indoor environments. In *Proc. IEEE Int. Conf. on Image Processing (ICIP)*, pages 405–408, September 1996.

[127] V. Sequeira, J.G.M. Gonçalves, and M.I. Ribeiro. Active view selection for efficient 3d scene reconstruction. In *Proc. 13th Int. Conf. on Recognition (ICPR)*, pages 815–819, August 1996.

[128] V. Sequeira, K. Ng, E. Wolfart, J.G.M. Gonçalves, and D. Hogg. Automated 3d reconstruction of interiors with multiple scan-views. In *SPIE Proc. Vol. 3641-Videometrics VI, SPIE-Photonics West - Electronic Imaging'99 Int. Symposium*, pages 106–117, January 1999.

[129] A. Sheffer, C. Gotsman, and N. Dyn. Robust spherical parameterization of triangular meshes. *Computing*, 72(1-2) :185–193, 2004.

[130] T. Simeon, J.P. Laumond, and F. Lamiraux. Move3d : A generic platform for path planning. In *Proc. IEEE Int. Symposium on Assembly and Task Planning*, pages 25–30, 2001.

[131] D. Skocaj and A. Leonardis. Robust recognition and pose determination of 3-d objects using range images in eigenspace. In *Proc. 3rd Int. Conf. on 3D Digital Imaging and Modeling (3DIM)*, pages 171–178. IEEE Computer Society, 2001.

[132] S.M. Smith and J.M. Brady. Susan – a new approach to low level image processing. Technical Report TR95SMS1c, Chertsey, Surrey, UK, 1995.

[133] J.A. Restrepo Specht. *Modélisation d'objets 3D par construction incrémentale d'un maillage triangulaire, dans un contexte robotique*. PhD thesis, LAAS-CNRS, Université Paul Sabatier de Toulouse, January 2005.

[134] S. Startchik, R. Milanse, and T. Pun. Projective and illumination invariant representation of disjoint shapes. In *Proc. 5th European Conf. on Computer Vision (ECCV)*, 1998.

[135] A. Stein and M. Hebert. Incorporating background invariance into feature-based object recognition. In *Proc. 7th IEEE Workshop on Applications of Computer Vision / IEEE Workshop on Motion and Video Computing (WACV/MOTION)*, pages 37–44. IEEE Computer Society, 2005.

[136] M. Strandberg and B. Wahlberg. A method for grasp evaluation based on disturbance force rejection. *IEEE Trans. on Robotics (ITRO)*, 22(3) :461 – 469, June 2006.

[137] Z. Sun, X. Yuan, G. Bebis, and S. Louis. Neural-network-based gender classification using genetic eigen-feature extraction. In *Proc. IEEE Int. Joint Conf. on Neural Networks*, pages 2433–2438, Honoloulu, Hawaii, May 2002.

[138] D.L. Swets and J. Weng. Discriminant analysis and eigenspace partition tree for face and object recognition from views. In *FG*, pages 192–197. IEEE Computer Society, 1996.

[139] D.L. Swets and J. Weng. Using discriminant eigenfeatures for image retrieval. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 18(8) :831–836, 1996.

[140] J.O. Talton. A short survey of mesh simplification algorithms. Course notes for CS 592 MJG, October 2004.

[141] J.P. Tarel and N. Boujemaa. Une approche floue du recalage 3D : généricité et robustesse. Rapport de recherche 2716, INRIA, 1995.

[142] J.P. Tarel and N. Boujemaa. Une approche floue du recalage 3D : généricité et robustesse. Rapport de recherche 2716, INRIA, 1995.

[143] C. Tauber, H. Batatia, and A. Ayache. Contours actifs basés sur trois énergies : détection des cavités cardiaques. In *Proc. 15ème Congrès Francophone AFRIF-AFIA Reconnaissance des Formes et Intelligence Artificielle (RFIA)*, pages 2433–2438, Tours, France, January 2006.

[144] D. Terzopoulos and D.N. Metaxas. Dynamic 3d models with local and global deformations : Deformable superquadrics. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 13(7) :703–714, 1991.

[145] G.M. Treece, R.W. Prager, and A.H. Gee. Regularised marching tetrahedra : improved iso-surface extraction. *Computers and Graphics*, 23 :583–598, 1999.

[146] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – a modern synthesis. In B. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms : Theory and Practice*, volume 1883 of *Lecture Notes in Computer Science*, pages 298–372. Springer-Verlag, 2000.

[147] F. Trujillo-Romero, V. Ayala-Ramírez, A. Marín Hernández, and M. Devy. Control de modalidades en un modelo de funcionalidad visual : aplicación a una tarea de localización de un robot móvil. In *Proc. 13th Int. Conf. on Electronics, Communications, and Computers (CONIELECOMP)*, pages 206–211, UDLAP, Puebla, México, February 2003.

[148] F. Trujillo-Romero, V. Ayala-Ramírez, A. Marín-Hernández, and M. Devy. Active object recognition using mutual information. In *Proc. 3rd Mexican Int. Conf. on Artificial Intelligence (MICAI) : Advances in Artificial Intelligence*, volume 2972 of *Lecture Notes in Computer Science*, pages 672–678. Springer, 2004.

[149] R.Y. Tsai. An efficient and accurate camera calibration techniqu e for 3d machine vision. In *Proc. IEEE Computer Vision and Pattern Recognition*. IEEE, 1987.

[150] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1) :71–86, 1991.

[151] S. Ullman. *High-level vision : Object recognition and visual cognition.* Cambridge - MA, USA : MIT Press., 1996.

[152] M.A. Vicente, C. Fernández Peris, O. Reinoso, and L. Payá. 3d object recognition from appearance : Pca versus ica approaches. In Aurélio C. Campilho and Mohamed S. Kamel, editors, *Proc. Int. Conf. Image Analysis and Recognition (ICIA)*, volume 3212 of *Lecture Notes in Computer Science*, pages 547–555. Springer, 2004.

[153] T. Volodine, D. Roose, and D. Vanderstraeten. Efficient triangulation of point clouds using floater parameterization. In *Proc. 8th SIAM Conf. on Geometric Design and Computing*, pages 523–536, Seattle, November 2003.

[154] H. Wang and B.K. Ghosh. Geometric deformable model and segmentation. In *Proc. IEEE Int. Conf. on Image Processing (ICIP)*, pages 328–332, 1998.

[155] S. Weik. Registration of 3-d partial surface models using luminance and depth information. In *Proc. Int. Conf. on 3D Digital Imaging and Modeling (3DIM)*, pages 93–100. IEEE Computer Society, 1997.

[156] D. Wilkes and J.K. Tsotsos. Active object recognition. In *Proc. IEEE Int.Conf. on Vision and Pattern Recognition (CVPR), Urbana, Illinois*, June 1992.

[157] L.M. Wong, C. Dumont, and M.A. Abidi. Next best view system in a 3-d object modeling task. In *Proc. IEEE Int. Symposium on Computational Intelligence in Robotics and Automation*, pages 306–311, November 1999.

[158] J. Wu and L. Kobbelt. A stream algorithm for the decimation of massive meshes. In *Proc. Graphics Interface*, pages 185–192, 2003.

[159] G. Xingfei and T. Jie. An automatic active contour model for multiple objects. In *Proc. Int. Conf. on Pattern Recognition (ICPR 2)*, pages 881–884, 2002.

[160] C. Xu and J.L. Prince. Gradient vector flow : A new external force for snakes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 66–. IEEE Computer Society, 1997.

[161] C. Xu and J.L. Prince. Generalized gradient vector flow external forces for active contours. *Int. Journal Signal Processing*, 71(2) :131–139, December 1998.

[162] C. Xu and J.L. Prince. Snakes, shapes, and gradient vector flow. *IEEE Trans. on Image Processing*, 7(3) :359–369, 1998.

[163] C. Xu and J.L. Prince. Global optimality of gradient vector flow. In *Proc. 34th Annual Conf. on Information Sciences and Systems (CISS'00)*, March 2000.

[164] S.M. Yamany and A.A. Farag. Free-form surface registration using surface signatures. In *Proc. Int. Conf. on Computer Vision (ICCV)*, pages 1098–1104, 1999.

[165] S.M. Yamany and A.A. Farag. Surfacing signatures : An orientation independent free-form surface representation scheme for the purpose of objects registration and matching. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 24(8) :1105–1120, 2002.

[166] S.M. Yamany, A.A. Farag, and A.R. El-Bialy. Free-form object recognition and registration using surface signatures. In *Proc. IEEE Int. Conf. on Image Processing (ICIP 2)*, pages 457–461, 1999.

[167] D. Yuan and S. Lu. Simulated static electric field (ssef) snake for deformable models. In *Proc. Int. Conf. on Pattern Recognition (ICPR 1)*, pages 83–86, 2002.

[168] T. Zaharia and F. Prêteux. Indexation de maillages 3d par descripteur de forme. In *Proc. Congrès AFRIF-AFIA de Reconnaissance des Formes et Intelligence Artificialle (RFIA)*, Angers, France, January 2002.

[169] D. Zhang and M. Hebert. Multi-scale classification of 3-d objects. Technical Report CMU-RI-TR-96-39, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, November 1996.

[170] Z. Zhang. Determining the epipolar geometry and its uncertainty : A review. Technical Report 2927, Sophia-Antipolis Cedex, France, 1996.

[171] X. Zhou and R. Wang. Stereo matching based on color and disparity segmentation by belief propagation. *Optical Engineering*, 46(4) :043203–1 – 043203–8, April 2007.