Handey : un modèle de planificateur pour la programmation automatique des robots

Emmanuel Mazer

THESE

Présentée à

L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

pour obtenir le grade de
DOCTEUR ES SCIENCES
<< Informatique >>

par

Emmanuel François MAZER

◊◊◊◊

HANDEY: UN MODELE DE PLANIFICATEUR POUR LA PROGRAMMATION AUTOMATIQUE DES ROBOTS

◊◊◊◊

Thèse soutenue le 14 Décembre 1987 devant la commission d'examen.

G. VEILLON	Président
G. GIRALT	
Ph. JORRAND	Examinateurs
J.C. LATOMBE	
T. LOZANO-PEREZ	

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Président : Daniel BLOCH

Année 1987

Vice - Présidents : René CARRE
Jean-Marie PIERRARD

Professeurs des Universités

BARIBAUD Michel	ENSERG
BARRAUD Alain	ENSIEG
BAUDELET Bernard	ENSPG
BEAUFILS Jean-Pierre	ENSEEG
BESSON Jean	ENSEEG
BLIMAN Samuel	ENSERG
BLOCH Daniel	ENSPG
BOIS Philippe	ENSHMG
BONNETAIN Lucien	ENSEEG
BOUVARD Maurice	ENSHMG
BRISSONNEAU Pierre	ENSIEG
BRUNET Yves	IUFA
BUYLE-BODIN Maurice	ENSERG
CAILLERIE Denis	ENSHMG
CAVAIGNAC Jean-François	ENSPG
CHARTIER Germain	ENSPG
CHENEVIER Pierre	ENSERG
CHERADAME Hervé	UFR PGP
CHERUY Arlette	ENSIEG
CHIAVERINA Jean	UFR PGP
CHOVET Alain	ENSERG
COHEN Joseph	ENSERG
COUMES André	ENSERG
DARVE Félix	ENSHMG
DELLA-DORA Jean	ENSIMAG
DEPORTES Jacques	ENSPG
DOLMAZON Jean-Mar	ENSERG
DURAND Francis	ENSEEG
DURAND Jean-Louis	ENSIEG
FONLUPTJean	ENSIMAG
FOULARD Claude	ENSIEG
GANDINI Alessandro	UFR PGP
GAUBERT Claude	ENSPG
GENTIL Pierre	ENSERG
GREVEN Hélène	IUFA
GUERIN Bernard	ENSERG
GUYOT Pierre	ENSEEG
IVANES Marcel	ENSIEG
JAUSSAUD Pierre	ENSIEG
JOUBERT Pierre	ENSIEG
JOURDAIN Geneviève	ENSIEG
LACOUME Jean-Louis	ENSIEG
LESIEUR Marcel	ENSHMG
LESPINARD Georges	ENSHMG
LONGEQUEUE Jean-Pierre	ENSPG
LOUCHET François	ENSEEG
MASSE Philippe	ENSIEG
MASSELOT Christian	ENSIEG
MAZARE Guy	ENSIMAG
MOREAU René	ENSHMG
MORET Roger	ENSIEG
MOSSIERE Jacques	ENSIMAG
OBLED Charles	ENSHMG
OZIL Patrick	ENSEEG
PARIAUD Jean-Charles	ENSEEG
PAUTHENET René	ENSIEG
PERRET René	ENSIEG
PERRET Robert	ENSIEG
PIAU Jean-Michel	ENSHMG
POUPOT Christian	ENSERG
SAUCIER Gabrielle	ENSIMAG
SCHLENKER Claire	ENSPG
SCHLENKER Michel	ENSPG
SERMET PIERRE	ENSERG
SILVY Jacques	UFR PGP
SIRIEYS Pierre	ENSHMG
SOHM Jean-Claude	ENSEEG
SOLER Jean-Louis	ENSIMAG
SOUQUET Jean-Louis	ENSEEG
TROMPETTE Philippe	ENSHMG
VEILLON Gérard	ENSIMAG
ZADWORNY François	ENSERG

Professeur Université des Sciences Sociales (Grenoble II)

BOLLIET Louis

Personnes ayant obtenu le diplôme d'HABILITATION A DIRIGER DES RECHERCHES

BECKER Monique
BINDER Zdenek
CHASSERY Jean-Marc
COEY John
COLINET Catherine
COMMAULT Christian
CORNUEJOLS Gérard
DALARD Francis
DANES Florin
DEROO Daniel
DIARD Jean-Paul
DION Jean-Michel
DUGARD Luc
DURAND Robert
GALERIE Alain
GAUTHIER Jean-Paul
GENTIL Sylviane
PLA Fernand
GHIBAUDO Gérard
HAMAR Sylvaine
LADET Pierre
LATOMBE Claudine
LE GORREC Bernard
MADAR Roland
MULLER Jean
NGUYEN TRONG Bernadette
TCHUENTE Maurice
VINCENT Henri

Chercheurs du C.N.R.S

Directeurs de recherche 1ère Classe

CAILLET Marcel
CARRE René
FRUCHART Robert
JORRAND Philippe
LANDAU Ioan
MARTIN

Directeurs de recherche 2ème Classe

ALEMANY Antoine
ALLIBERT Colette
ALLIBERT Michel
ANSARA Ibrahim
ARMAND Michel
BINDER Gilbert
BONNET Roland
BORNARD Guy
CALMET Jacques
DAVID René
DRIOLE Jean
ESCUDIER Pierre
EUSTATHOPOULOS Nicolas
JOUD Jean-Charles
KAMARINOS Georges
KLEITZ Michel
KOFMAN Walter
LEJEUNE Gérard
MERMET Jean
MUNIER Jacques
SENATEUR Jean-Pierre
SUERY Michel
TEDOSIU
WACK Bernard

Personnalités agrées à titre permanent à diriger des travaux de recherche (décision du conseil scientifique)

E.N.S.E.E.G

BERNARD Claude
CHATILLON Catherine
CHATILLON Christian
COULON Michel
DIARD Jean-Paul
FOSTER Panayotis
HAMMOU Abdelkader
MALMEJAC Yves
MARTIN GARIN Régina
SAINTFORT Paul
SARRAZIN Pierre
SIMON Jean-Paul
TOUZAIN Philippe
URBAIN Georges

E.N.S.E.R.G

BOREL Joseph
CHOVET Alain
DOLMAZON Jean-Marc
HERAULT Jeanny

E.N.S.I.E.G

DESCHIZEAUX Pierre
GLANGEAUD François
PERARD Jacques
REINISCH Raymond

E.N.S.H.G

BOIS Daniel
DARVE Félix
MICHEL Jean-Marie
ROWE Alain
VAUCLIN Michel

E.N.S.I.M.A.G

BERT Didier
COURTIN Jacques
COURTOIS Bernard
DELLA DORA Jean
FONLUPT Jean
SIFAKIS Joseph

E.F.P.G

CHARUEL Robert

C.E.N.G

CADET Jean
COEURE Philippe
DELHAYE Jean-Marc
DUPUY Michel
JOUVE Hubert
NICOLAU Yvan
NIFENECKER Hervé
PERROUD Paul
PEUZIN Jean-Claude
TAIB Maurice
VINCENDON Marc

Laboratoires extérieurs

C.N.E.T

DEMOULIN Eric
DEVINE
GERBER Roland
MERCKEL Gérard
PAULEAU Yves

ECOLE NATIONALE SUPERIEURE DES MINES DE SAINT-ETIENNE

Directeur : Monsieur M.MERMET
Directeur des Etudes et de la formation: Monsieur J. LEVASSEUR
Directeur des recherches : Monsieur J. LEVY
Secrétaire Général : Mademoiselle M. CLERGUE

PROFESSEURS DE 1ère CATEGORIE

COINDE Alexandre	Gestion
GOUX Claude	Métallurgie
LEVY Jacques	Métallurgie
LOWYS Jean-Pierre	Physique
MATHON Albert	Gestion
RIEU Jean	Mécanique-Résistance des matériaux
SOUSTELLE Michel	Chimie
FORMERY Philippe	Mathématiques Appliquées

PROFESSEURS DE 2ème CATEGORIE

HABIB Michel	Informatique
PERRIN Michel	Géologie
VERCHERY Georges	Matériaux
TOUCHARD Bernard	Physique Industrielle

DIRECTEUR DE RECHERCHE

LESBATS Pierre	Métallurgie

MAITRE DE RECHERCHE

BISCONDI Michel	Métallurgie
DAVOINE Philippe	Géologie
FOURDEUX Angeline	Métallurgie
KOBYLANSKI André	Métallurgie
LALAUZE René	Chimie
LANCELOT Francis	Chimie
LE COZE Jean	Métallurgie
THEVENOT François	Chimie
TRAN MINH Canh	Chimie

Personalités habilitées à diriger des travaux de recherche

DRIVER Julian	Métallurgie
GUILHOT Bernard	Chimie
THOMAS Gérard	Chimie

Professeurs à l'UER de Sciences de Saint-Etienne

VERGNAUD Jean-Maurice	Chimie des Matériaux et Chimie Industrielle

RESUME Un système de planification pour les opérations de transfert de pièces par robot manipulateur est présenté. Le système prend en charge la planification de toutes les opérations nécessaires à la réalisation d'un assemblage tels que la reconnaissance, la saisie et le transport de la pièce à monter. Une architecture basée sur l'utilisation d'une hiérarchie de filtres est utilisée comme structure de contrôle. Les algorithmes de raisonnement géométrique nécessaires à la manipulation sont présentés en utilisant un formalisme unique. Les résultats de l'expérimation ainsi que les développements futurs du système sont décrits.

ABSTRACT A system designed to plan a pick-and-place operation with a robot manipulator is described. The system automatically plans the necessary operations to perform the assembly: part recognition, grasping, path planning and re-grasping. The control structure is based on a hierarchy of tests. The algorithms used by the planner are presented with a single geometric formalism. The experimentation and the future developments of the system are discussed

MOTS CLEFS Programmation des robots - Raisonnement géométrique - planficateur de trajectoire - planificateur de saisie - vision tri-dimentionnelle - modélisation géométrique - assemblage automatique.

REMERCIEMENTS

Je remercie les membres du jury pour avoir accepter de juger ce travail:

Monsieur Gérard Veillon, Professeur à l'Institut National Polytechnique de Grenoble, Directeur de l'Ecole National Supérieure d'Informatique et de Mathématiques Appliquées de Grenoble,

Monsieur Georges Giralt, Directeur de Recherche au CNRS, responsable du projet National "Automatique et Robotique Avancées",

Monsieur Philippe Jorrand, Directeur de recherche au CNRS, Directeur du Laboratoire Informatique Fondamentale et de d'Intelligence Artificielle,

Monsieur Jean-Claude Latombe, Professeur à l'Institut National Polytechnique de Grenoble et à l'Université de Stanford,

Monsieur Tomàs Lozano-Pérez, Professeur au Massachusetts Institute of Technology.

Pour sa plus grande part, ce travail à été réalisé durant mon séjour au Laboratoire d'Intelligence Artificielle du MIT. Je voudrais exprimer ma profonde gratitude envers le Professeur Tomàs Lozano-Pérez pour son encadrement scientifique et remercier les membres de ce laboratoire pour leur extrême bienveillance à mon égard.

Un système comme Handey est avant tout le résultat d'un travail d'équipe, je souhaite remercier les personnes qui ont participé à ce projet et dont le travail et la compétence ont permi la réalisation de ce système:

- Messieurs Eric Grimson et Tomàs Lozano-Pérez pour la conception et la réalisation du système de localisation tri-dimensionelle,
- Monsieur Joe Jones pour la conception du planificateur de saisie et l'implantation du système de localisation fine,
- Monsieur Alain Lanusse pour la conception et la réalisation du système de modélisation géométrique YASM,

- Monsieur Patrick O'Donnell pour son support logisitique sur le materiel robotique et sur le logiciel,
- Monsieur Tomàs Lozano-Pérez pour la conception et la réalisation du planificateur de trajectoire,
- Monsieur Pierre Tournassoud pour la conception du système de resaisie.

Je voudrais aussi remercier John Canny, Bruce Donald, Robert Hall, Mike Erdmann, Oussama Khatib, Mat Mason, Jean Claude Latombe, Christian Laugier, Richard Paul et Pierre Puget pour les remarques et suggestions concernant ce travail, ainsi que Jocelyne Pertin, Chantal Olivier, Marie-Geneviève Picard, Vincent Hayward, Vania Joloboff, Véronique et Stéphan Ebalard pour la relecture du manuscrit et Ester Masip pour la réalisation des figures.

Je tiens à exprimer ma reconnaisance à mon frère Jean-Emile et à mère Marie-Jeanne pour leur aide, leur compréhension et leur soutien moral.

Enfin je souhaite apporter mon suffrage à ma femme Brigitte pour que lui soit décerné La Palme d'Or de l'Aide au Thésard toutes Catégories. Après avoir essuyé une première thèse, changé de travail pour me permettre de poursuivre mes recherches au MIT, supporté les états d'âmes du chercheur moyen, assuré sa nourriture, maintenu son équilibre psychologique, il lui a fallu participer à la traduction du manuscrit anglais et apprendre presque par cœur le texte français à force de relecture. Qu'une mention spéciale lui soit délivrée.

[0]Ce travail à été financé par le Centre National de la Recherche Scientifique, " The Office of Naval Research" (contrat N00014-86-K-685) et "The National Science Foundation" (Presidential Young Investigator Award (Lozano-Pérez))

Sommaire

Chapitre 1

Introduction

1.1 Présentation du problème

Dans ce rapport nous nous intéressons à la structure des systèmes de programmation de robots et, plus particulièrement, aux opérations de manipulation élémentaires tels que la prise et le montage d'un objet sur un sous-assemblage.

A l'heure actuelle, la plupart des systèmes de programmation de robots sont construits en faisant l'hypothèse suivante:

"Il est existe un ensemble de primitives de base qui permettent de construire incrémentalement des primitives de complexité croissante".

Nous maintenons qu'un tel point de vue n'est pas valable pour les opérations de manipulation. Par exemple, il est très difficile d'écrire des sous-programmes généraux de manipulation, même en utilisant des outils de programmation spécialement conçus pour la robotique. On ne peut tout simplement pas écrire un programme générique visant à prendre et à assembler une pièce sur un sous assemblage. En effet, de légères modifications de l'environnement peuvent changer radicalement la stratégie de montage. Par exemple, la présence d'un nouvel objet près du sous assemblage peut provoquer une collision au moment du montage et obliger un changement de strategie au niveau de la saisie l'objet. Il faudrait demander à l'utilisateur de plus en plus de paramètres pour spécifier les variations possibles de l'environnement.

A la limite, il faudrait lui demander de spécifier tous les mouvements correspondant à tous les possibilités rencontrées durant la manipulation.

Ceci étant valable pour la plupart des opérations de manipulation (insertion, glissement...), la programmation des robots est rendue très difficile car le programmeur ne peut pas utiliser facilement des sous-programmes. On peut utiliser le même argument afin d'expliquer pourquoi les utilisateurs de primitives évoluées sont quand même amenés à accéder à un niveau de contrôle élémentaire du robot. Quelles que soient les primitives de base, il y a toujours un cas particulier dans l'environnement qui rend ces primitives inapplicables, alors qu'une combinaison de commandes de plus bas niveau peut résoudre le problème.

Une meilleure compréhension de la structure des systèmes de programmation de robots aidera à la construction de robots plus sophistiqués. Toutefois, nous ne répondrons pas à la question:

Pourquoi est-il important de construire des robots sophistiqués ?

1.2 L'approche

Afin d'acquérir une meilleure compréhension de cette structure, nous avons construit un planificateur (Handey) qui peut planifier une simple opération de manipulation: *Prendre-et-Poser*. Ce planificateur explore l'espace de recherche associé à une description de l'environnement donnée, et crée une séquence de mouvements appropriés à la tâche à effectuer. Autrement dit, il construit une primitive de commande spécifique pour chaque combinaison de la spécification de la tâche et de la description de l'environnement. Un tel planificateur est appelé *interpréteur de niveau tâche* car il ne nécessite qu'une description de la tâche et de l'environnement, et non pas la spécification des mouvements du robots nécessaires à la manipulation.

L'étude de la structure de ce planificateur nous a permis de formuler des théories sur la structure générale d'un système de planification. Nous pensons que ces théories peuvent servir à la réalisation de planificateurs similaires pour d'autres opérations de manipulations élémentaires.

1.3 La contribution

Nous proposons deux hypotèses sur la structure d'un système de programmation de robot:

Décomposition des mouvements: *Il est possible de décomposer le planificateur associé à une opération de manipulation en plusieurs planificateurs de mouvements*

Simplification de l'espace de recherche: *L'espace de recherche associé à la planification d'une opération de manipulation est tel que l'on peut le simplifier sans pour cela supprimer toutes les solutions de l'espace de recherche initial*

L'expérimentation de Handey montre que ces hypothèses ont une base solide. Nous pensons qu'elles sont justes car elles nous ont permis de construire un planificateur sans équivalent à l'heure actuelle, et qu'elles sont générales car elles peuvent être utilisées pour construire d'autres planificateurs correspondant à d'autres opérations simples de manipulation.

1.3.1 La décomposition de mouvements

La planification d'une manipulation, telle que la prise et l'assemblage d'une pièce, nécessite la planification de plusieurs sous-opérations: la localisation de la pièce, son approche, la saisie, le transport de la pièce, son montage, etc. Ces opérations paraîssent, à première vue, séparables et indépendantes. Elles sont en fait fortement couplées et ne peuvent être traitées séparément. Par exemple, les opérations de saisie et de positionnement final sont clairement couplées. Le choix du mouvement final d'assemblage doit tenir compte de la saisie et doit éviter les collisions possibles de la pince à la fois à l'emplacement de la saisie et de l'assemblage. De même, le choix de la configuration du robot à une phase particulière de l'assemblage ne doit pas rendre les étapes suivantes impossibles à réaliser.

L'étroite interdépendance entre toutes les opérations semble nécessiter la planification de toutes les étapes en même temps [LB85]. Il est en principe possible de représenter toutes les interdépendances explicitement et d'obtenir des plans qui vont les satisfaire sans avoir recours à des méthodes par essais-erreurs. D'un point de vue théorique, il est en général très facile d'exprimer un problème de manipulation comme la recherche d'une trajectoire dans un espace donné. Par exemple, Bruce Donald [Don86] a défini un espace des configurations généralisé comme étant le produit de deux sous espaces: la position de la pièce et la déviation par rapport à la position commandée. Dans cet espace, il est possible, de représenter une opération consistant à pousser un objet sur la table comme une trajectoire de glissement sur la surface d'un obstacle de l'espace des configurations ainsi défini. En théorie, il est possible de calculer ce type de trajectoire dans des espaces de configuration complexes [Can87], mais ces méthodes générales ne sont actuellement appliquées qu'à des problèmes très simples [Bro82].

Au lieu d'essayer de construire un planificateur unique qui considérerait tous les aspects de la planification d'une opération de manipulation, une solution plus simple et plus réaliste est de concevoir plusieurs planificateurs et d'essayer de les combiner. Ce découplage nécessite toutefois de procéder par essais-erreurs car les solutions engendrées par les sous-planificateurs ne sont forcément compatibles entre elles.

1.3.2 Simplification de l'espace de recherche

Notre but étant d'obtenir un système opérationnel, nous ne pouvions nous permettre d'être bloqués par un problème trop difficile à résoudre. Afin de sortir de cette situation frustrante, nous avons fait la remarque suivante: il est plus facile de tester une propriété donnée plutôt que de produire directement un élément ayant cette propriété. Par exemple, il n'existe pas de formule permettant de calculer le n^{ieme} nombre premier, mais il est toujours possible de vérifier si un nombre est premier ou non. Ceci nous a conduit à utiliser la méthode dite de *génération-vérification* à chaque fois que nous n'avions pas de solution analytique à un problème donné.

L'écueil évident de cette approche est la nature combinatoire des problèmes difficiles. Nous nous sommes aperçus, que dans le cas de la manipulation, il était possible d'éviter cet écueil partiellement. La difficulté d'utilisation de la méthode génération-vérification provient de la grandeur du domaine de recherche et du fait que la fonction de vérification peut être longue à calculer (voir l'exemple des nombres premiers). En utilisant de simples remarques sur les propriétés de la solution cherchée, il est possible de réduire considérablement le domaine et d'appliquer la fonction de vérification à un espace de recherche limité.

Cette technique réussit bien dans le domaine de l'assemblage car il y a beaucoup de possibilités d'exécuter une opération de manipulation donnée. Les expérimentations faites avec notre système tendent à révéler la nature de l'espace de recherche associé aux opérations de manipulation: même si l'on supprime un grand nombre de solutions en supprimant de larges portions de l'espace de recherche, on ne compromet pas les chances d'en obtenir d'autres dans le reste de l'espace de recherche. Trouver une solution par une recherche aveugle est virtuellement impossible, mais l'utilisation d'un simple calcul pour guider la recherche peut s'avérer efficace. Au cours de notre expérimentation nous avons constaté que Handey trouvait toujours une solution au problème d'assemblage que nous lui soumettions. Considérant la simplicité relative des calculs géométriques et les importantes simplifications faites sur la représentation du problème, ce résultat peut paraître surprenant. Pour cette raison, nous pensons que si l'espace de recherche associé à une opération *Prendre-et-Poser* est vaste, le nombre de solutions associées est lui aussi important et que toute parcelle de connaissance sur le domaine peut être utilisée afin de réduire considérablement l'espace de recherche.

1.4 Justification

La justification de l'architecture proposée est une implémentation opérationnelle du système. Une propriété importante de cet interpréteur est sa généralité par rapport à ses paramètres. Autrement dit, quelque soit la valeur des paramètres, l'interpréteur effectuera généralement la tâche correctement. Notamment, si A et B sont deux modèles de pièces et W, le modèle de l'environnement, l'interpréteur produira une séquence de mouvements corrects afin de prendre la pièce A, de la ré-orienter le cas échéant, de l'empiler sur la pièce B et d'éviter tous les obstacles

décrits dans le modèle de l'environnement W, et ceci pour tout A, B et W. La fiabilité peut être expliquée par une structure de contrôle qui permet d'explorer efficacement l'espace de recherche associé à cette opération de manipulation et par les principes géométriques bien établis utilisés comme blocs élémentaires de construction.

1.5 Description fonctionnelle de Handey

Handey peut manipuler et assembler des polyèdres de forme quelconque, il est écrit en LISP et peut être lancé en évaluant la forme Lisp:

(prendre-et-poser pièce A pièce B environnement)

Une photographie du matériel utilisé pour l'expérimentation de notre système est présentée figure 1.1. La figure 1.4 représente un modèle CAO de la même scène.

Les lignes en pointillées apparaissant sur la table (figure 1.4) indiquent les limites du champ du système de vision tridimensionnel utilisé pour localiser les pièces. Cette aire sera appelé la V-zône par la suite. On suppose que la pièce A est positionnée dans la V-zône. On ne suppose rien au sujet de cette zône; la pièce A peut être à tout endroit dans cette zône et peut être partiellement cachée du système de vision tri-dimensionnel par d'autres objets. Ces objets ne sont pas nécessairement modélisés dans le système CAO et peuvent avoir une forme quelconque. La figure 1.2 représente une scène typique de la V-zône. Il faut remarquer que les pièces se trouvant dans la V-zône ne sont pas présentes dans la représentation CAO de la scène au début de l'exécution.

Dans un premier temps, le système calcule des manières possibles d'assembler les deux pièces. Les modèles des pièces A et B (figure 1.3) ainsi que différentes configurations d'assemblage apparaissent à l'écran (figure 1.5), puis l'utilisateur sélectionne une configuration à l'aide de la souris de la machine lisp. Cette étape est nécessaire car le système n'a aucun moyen de connaître la forme de l'assemblage final. Une solution possible serait de raisonner sur la fonctionalité de ces pièces dans l'assemblage final ([Ing87, WBKL83]) pour déterminer la position relative finale. Puis le système propose une position intermédiaire située près de l'assemblage final. Cette position remplacera la position d'assemblage car rien n'a été prévu à l'heure actuelle pour planifier les mouvements fins nécessaires à l'assemblage (voir section 6.2). Une fois la position finale des deux objets spécifiée, la planification et l'exécution de l'assemblage ne nécessite plus d'intervention de la part de l'utilisateur.

Le système de vision tri-dimensionnelle est activé. La figure 1.6 est une représentation de la carte tri-dimensionnelle construite par le système laser. Dans cette image, les niveaux de gris représentent la hauteur de chaque point de la scène. Le module de localisation est activé et la pièce A est placée à sa position correcte dans le modèle CAO.

Après la reconnaissance, l'interpréteur planifie une trajectoire sans collision allant de la position initiale du robot jusqu'à un point situé au-dessus de la V-zône. La figure 1.7 représente une simulation de cette trajectoire.

Le système planifie ensuite le guidage de la pince parmi les obstacles de la V-zône pour aller saisir l'objet (figure 1.8).

Une fois que la pièce a été saisie, un léger mouvement vertical est opéré et une trajectoire sans collision est planifiée en direction d'une position située près d'un autre système de vision actif (figure 1.9). Ce capteur de vision tri-dimensionnelle est utilisé pour améliorer la précision sur la position de la pièce A. Cette opération de localisation précise de l'objet est utile car l'objet peut légèrement se déplacer dans les mors de la pince au moment de la saisie.

Après cette phase de localisation, une dernière trajectoire sans collision est planifiée en direction d'un emplacement situé près de l'assemblage final.

Il n'est quelquefois pas possible de saisir et assembler directement une pièce. Handey peut détecter cette incompatibilité et planifier une séquence de mouvements de re-saisie. Le but de cette séquence est de ré-orienter l'objet dans les mors de la pince de sorte que l'assemblage devienne possible. Ceci est fait en plaçant successivement la pièce sur la table et en la saisissant de manière différente. Les étapes successives d'une re-saisie sont présentées dans la figure 1.10. Une trajectoire sans collision est planifiée entre chacune de ces étapes.
Afin de planifier une opération de saisie et de montage, Handey utilise les modules suivants:

Un système de modélisation géométrique qui représente le robot et la scène.

Un module de reconnaissance de forme qui localise et reconnaît les objets grâce à leurs modèles CAO.

Un planificateur de saisie qui calcule une prise stable et une trajectoire sûre pour atteindre cette prise.

Un système précis de localisation qui permet de trouver la position exacte de la pièce dans la pince après la saisie.

Un planificateur de re-saisie qui planifie les ré-orientations nécessaires afin que le robot tienne correctement la pièce pour effectuer l'assemblage.

Un planificateur de trajectoire pour planifier les mouvements entre les différentes étapes du plan général.

1.6 Description graphique du système

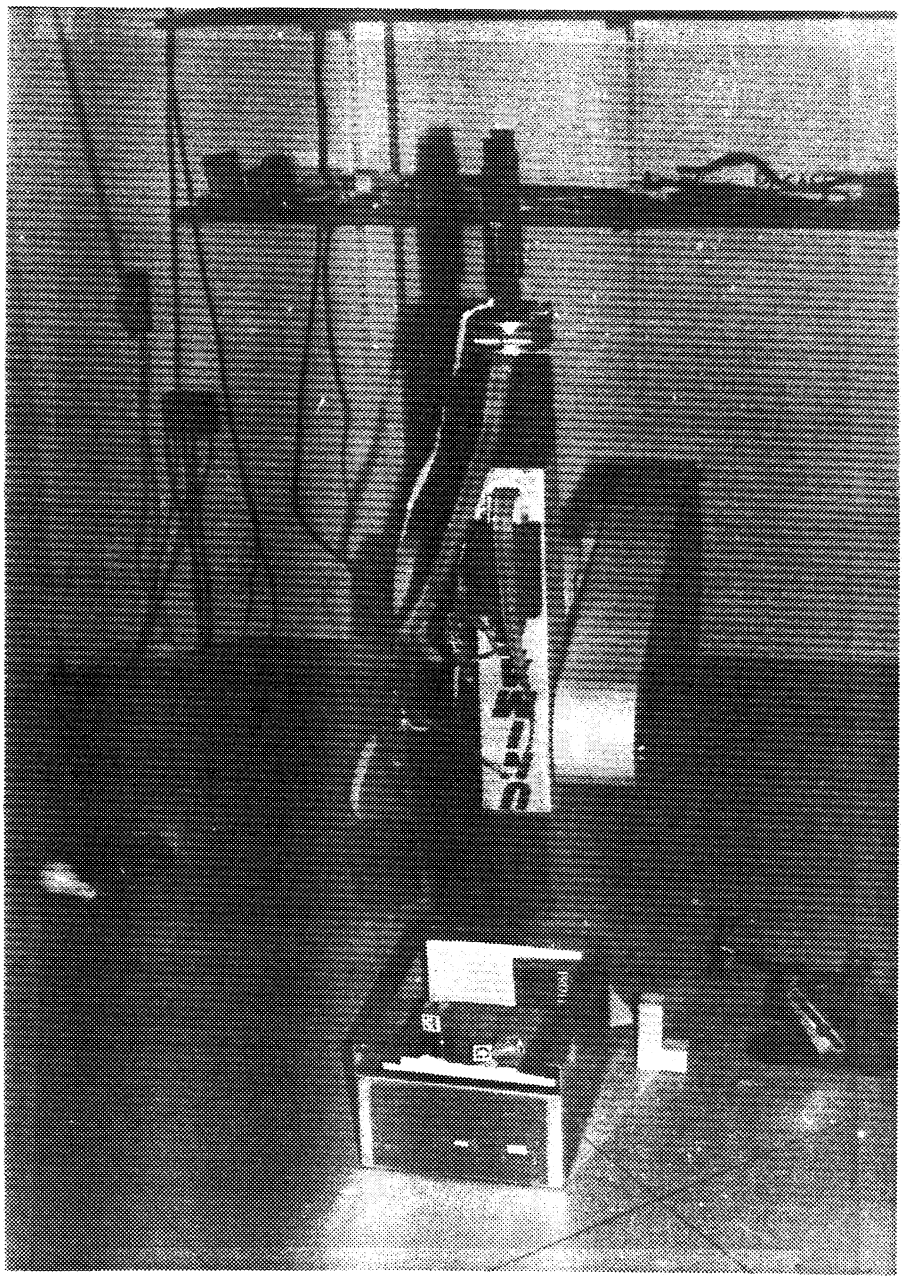


Figure 1.1: Système matériel utilisé pour l'expérimentation

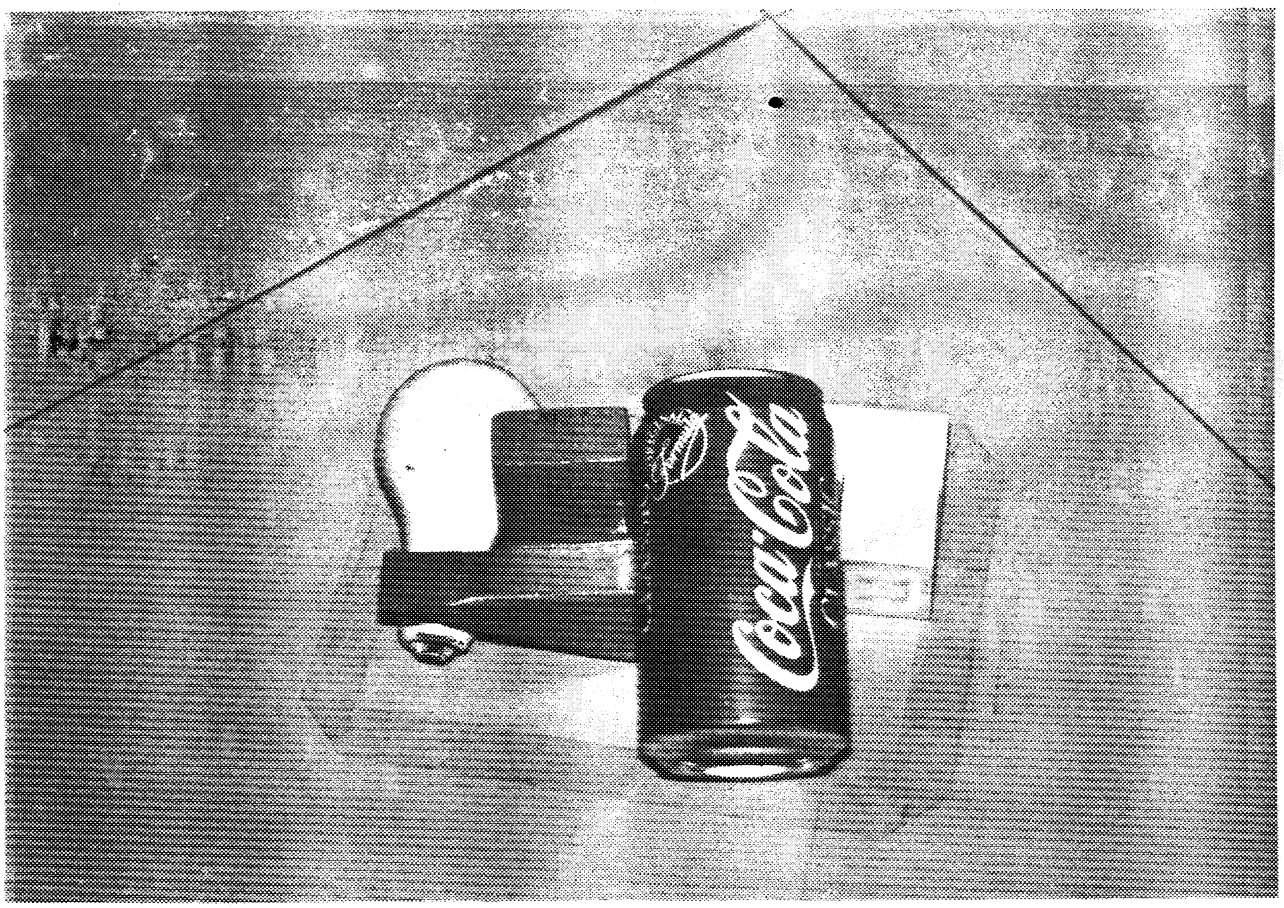


Figure 1.2: Disposition caractéristique des obstacles à la position de saisie (V-zone)

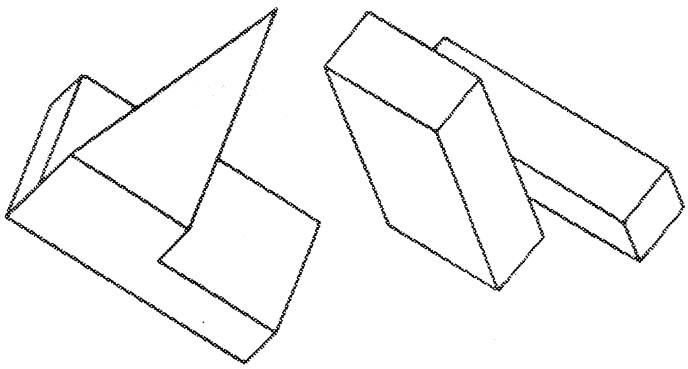


Figure 1.3: Représentation CAO des pièces A et B

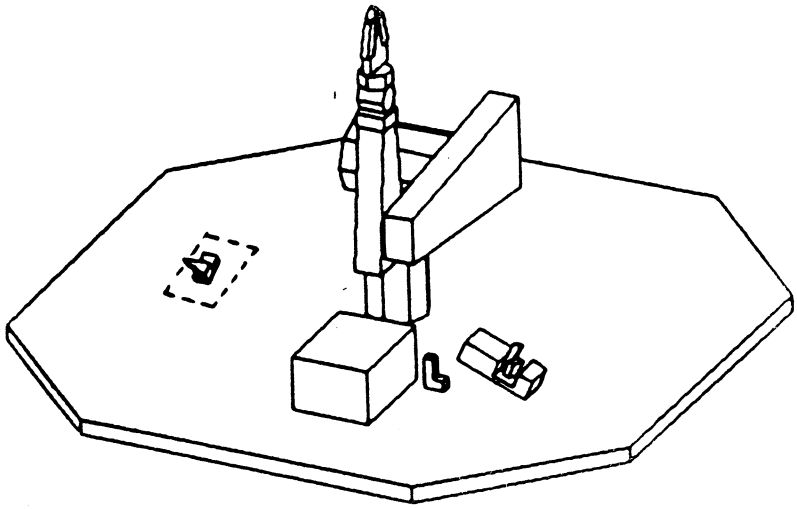


Figure 1.4: Représentation CAO du matériel

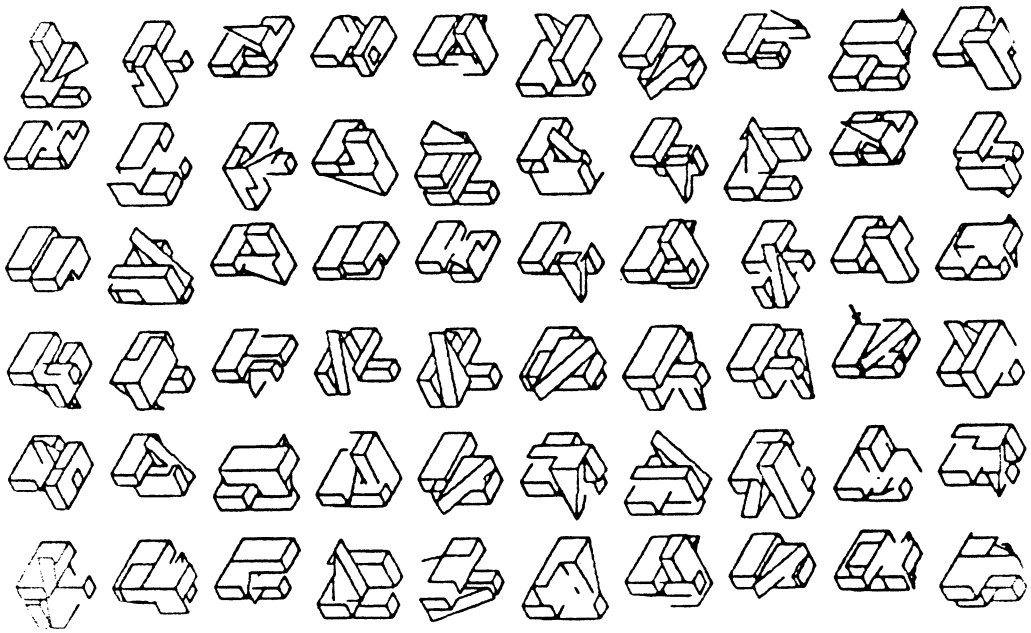


Figure 1.5: Quelques assemblages proposés par le système

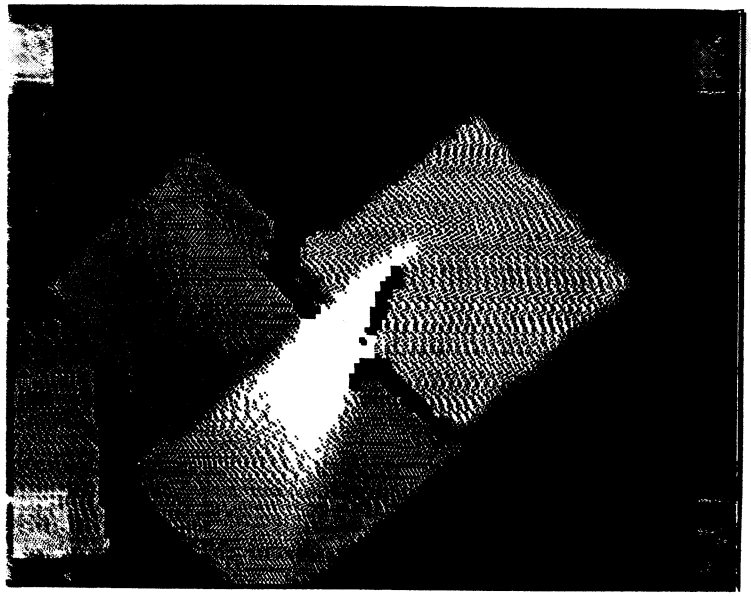


Figure 1.6: Représentation de la hauteur d'une carte tri-dimensionnelle par des niveaux de gris

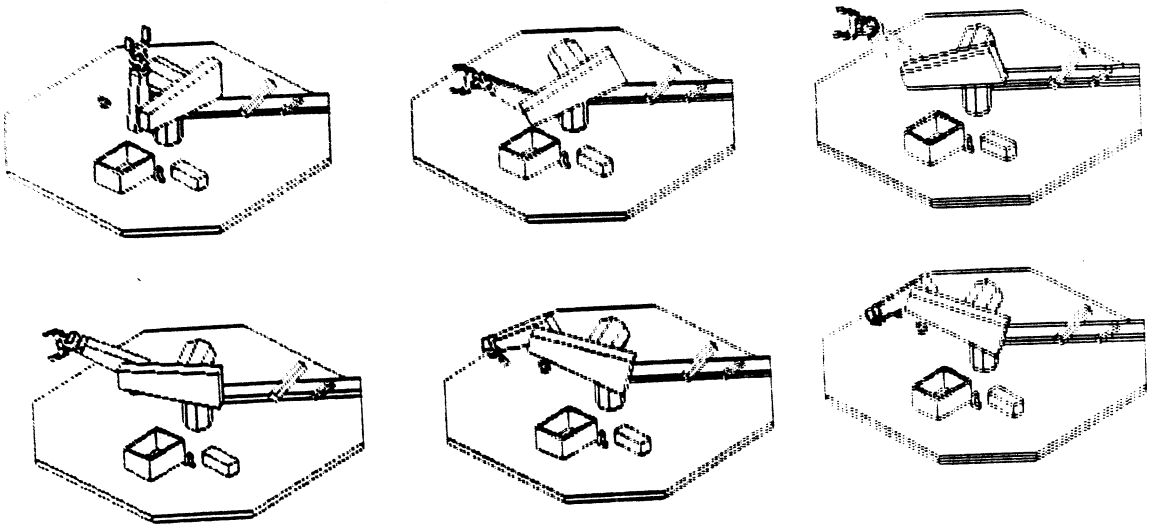


Figure 1.7: Simulation CAO de la trajectoire initiale

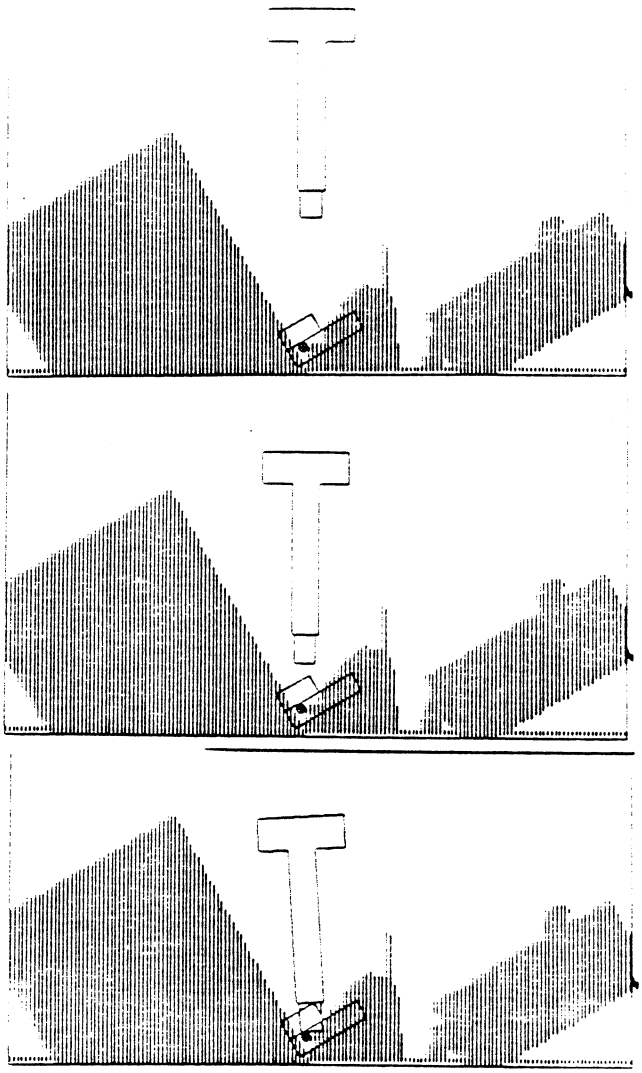


Figure 1.8: Trajectoire de la pince parmi des obstacles de la V-zône

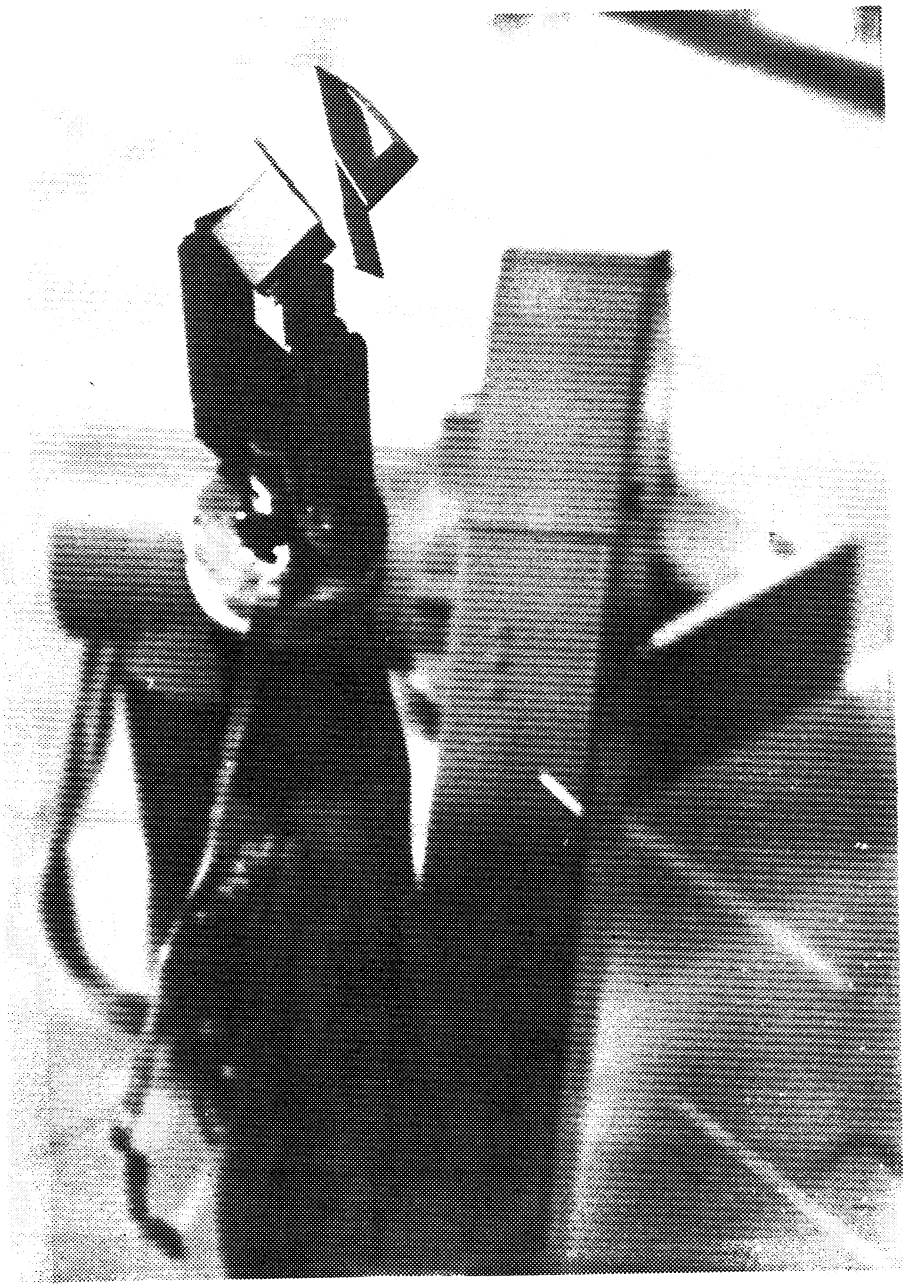


Figure 1.9: Localisation précise de la pièce avec un système de vision tri-dimensionnelle

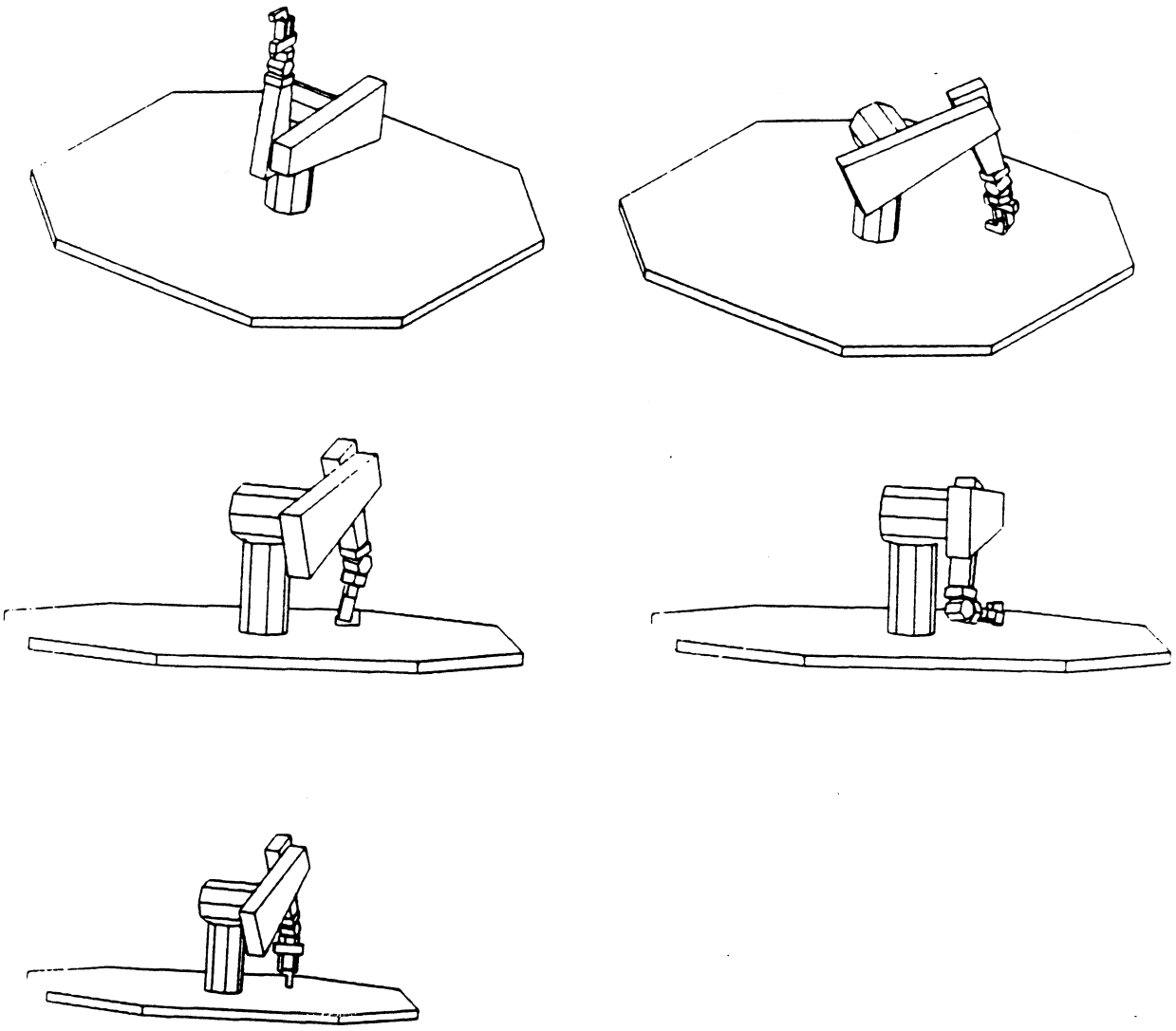


Figure 1.10: Une opération de re-saisie en deux temps

1.7 Travaux antérieurs

1.7.1 Les systèmes de programmation de robots existants

AL[Fin76], AML[TSM82], VAL II[SGS84], RAIL[FV82], LM[MM84], ML[WG75], PAL[TPB79], TEACH[Ruo79], SIGLA[Sal78], KAREL[Rob85], MCL[Dou80], T3[Cun79], LPR[Sko81], SRL[BJ82], HELP[Cor82], LUCIFER[Gir83], LAMA-S[FP80], MAL[GGGG79], RCL[SGM82], RCCL[HP86], FUNKY[Gro77], ANORAD[Cor79], EMILY[EGG76], MAPLE[DB75], LMAC[Rou83] MINI[Sil73], MHI[Ern61], JARS[Cra80], RPS[Par81], WAVE[Pau77], FPROLOG[HK86], AR-BASIC, ROL, ROBEX, sont des noms de systèmes partiellement ou totalement developpés pour la programmation des robots. Une étude les concernant sera trouvée dans [Loz82, Lat83, GSCT84]. Peu de ces de systèmes, VAL mis à part, sont utilisés par un grand nombre d'utilisateurs; cela pour deux raisons principales:

1. La difficulté de mise en œuvre d'un système de programmation de robots est toujours très largement sous-estimée. Des développements importants sont nécessaires avant qu'un système soit un outil efficace et qu'il puisse être utilisé par un groupe d'utilisateurs novices.

2. Les langages de programmation de robots sont généralement difficiles à utiliser (voir chapitre 2) et, pour la plupart des applications, ne sont pas considérés par les utilisateurs comme plus efficaces que des systèmes de programmation par l'exemple.

AML, VAL II et LM figurent parmi les quelques systèmes développés complètement. Par exemple, LM [Maz81, Mir84, LLJL84] a été complètement implanté sur une variété de robots et d'ordinateurs, montrant ainsi l'intérêt d'un système de programmation de robots indépendant de la structure géométrique des manipulateurs. Il peut être utilisé conjointement avec plusieurs utilitaires de programmation tels qu'un générateur de trajectoire complexe ou un simulateur graphique. Le système est toujours en développement bien que les premières spécifications de LM aient été écrites il y a 8 ans [ITM87b].

1.7.2 Les systèmes Prendre-et-Poser et Main-Oeil

La réalisation d'un système Prendre-et-Poser est un but qui a existé de longue date en robotique. C'est un objectif qui semble facile à atteindre et a succité de nombreuses tentatives. De la même manière, l'ajout de capteurs visuels a toujours été tentant et beaucoup de ces systèmes utilisent des capteurs visuels pour la localisation des pièces.

Les premières recherches à l'Université d'Edinbourg

Un système Prendre-et-Poser complet ainsi qu'un système Main-Oeil sont décrits dans [ABB*75] come suit:

"Un système d'assemblage universel utilisant des caméras de télévision et un bras contrôlé par ordinateur est décrit. Il exécute des assemblages simples tel qu'assembler des anneaux sur des tiges ou assembler des modèles de voitures en bois. Il sépare des pièces d' un tas, les reconnait grâce à une caméra et les assemble en utilisant ses capteurs de forces..."

Le système fonctionne en deux temps: "répartition" et "assemblage". Pendant la phase de répartition, le système tente de reconnaître des pièces parmi un tas d'autres pièces. Lorsqu'il a trouvé une pièce, il la pose à une position connue sur le plan de travail. L'opération est répétée jusqu'à ce que toutes les pièces nécessaires à l'assemblage soient trouvées. Si une ou plusieurs pièces manquent, alors le robot est capable de briser le tas dans l'espoir de rendre des pièces cachées visibles. Lorsque toutes les pièces sont positionnées à des endroits connus, la phase "d'assemblage" est activée. Etant donné que toutes les pièces sont à des positions connues, l'assemblage consiste à l'utiliser un programme classique de commande de robot.

Le système "Move-Instance"

Une description d'une version du système "Move-Instance" peut être trouvée dans [Pau72]. Le matériel utilisé comprenait un bras contrôlé par ordinateur et un système de vision en couleur. La démonstration consistait à empiler des cubes de couleur en respectant certaines conditions sur les couleurs de faces adjacentes. Le système était capable de planifier toutes les opérations nécessaires à l'empilage des cubes, y compris les opérations de manipulation nécessaires à la ré-orientation des cubes. Les principales limitations du système "Move-Instance" étaient l'absence de planificateur de trajectoire et le fait qu'il ne puisse manipuler que des parallèlogrammes. Handey peut être considéré comme une généralisation de ce système car il inclut un planificateur de trajectoire et qu'il ne souffre d'aucune restriction concernant la forme de polyèdres manipulés.

Prise dans un vrac en utilisant un système de vision stéréo

Le système présenté dans [INH*86] à été conçu afin de prendre une pièce dans un vrac et de la poser à une position connue. Le système procède en quatre temps:

1. L'identification de l'emplacement de la pièce dans une scène complexe.
2. La mesure de l'orientation de la pièce.
3. La mesure de la hauteur de la pièce.
4. Le calcul d'une configuration de prise sans collision.

Les systèmes de vision sont utilisés pour localiser l'objet à saisir et pour construire une représentation tri-dimensionnelle de la scène. De la même façon que dans Handey, la nappe tri-dimensionnelle est utilisée pour planifier l'approche et la saisie de l'objet.

1.7.3 Les systèmes de programmation niveau tâche

RAPT

RAPT[PAB80] peut être vu comme une tentative de simplification de la phase "d'assemblage" décrite en 1.7.2. L'idée principale de RAPT est de décrire un assemblage en termes de relations géométriques (figure 1.11). RAPT permet l'utilisation d'une grande variété de relation géométriques et offre la possibilité de décrire des mécanismes tels que les articulations rotoïdes ou prismatiques d'un robot. Cette dernière caractéristique fait de RAPT un outil très général étant donné que la description de la tâche comprend aussi le robot. Des "macros" peuvent être utilisées pour construire des commandes robotiques de "haut niveau" tels que l' opération Prendre-et-Poser, mais en réalité toutes les opérations complexes telles que la planification de prise ou la planification de trajectoire doivent être explicitement programmées par l'utilisateur.

L'interpréteur de RAPT est basé sur un système de ré-écriture. Ce système tente de réduire les équations déduites des relations géométriques sous une forme qui puisse être résolue au moyen de méthodes algébriques standard. Afin de rendre le système plus efficace, la description du robot a été supprimée de la spécification de la tâche, RAPT se contentant de produire du code de niveau effecteur (VAL II). Cette technique évite au système de ré-écriture le calcul de la transformation de coordonnées inverses du bras qui, dans ce cas, est pris en charge plus efficacement par l'interpréteur de VAL II. Une seconde optimisation utile est la réduction de l'ensemble des relations géométriques à un ensemble plus simple [Pop79] avant de passer au système de ré-écriture.

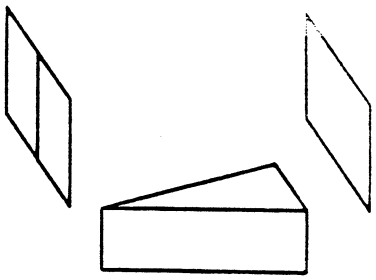
LM-Géo

LM-Géo [Maz83] est une extension du langage de programmation de robot LM [Mir84]. La conception de LM-Géo est fortemement inspirée de RAPT et LM-Géo est basé sur le concept principal de RAPT: la définition de position d'après des relations géometriques symboliques. Dans un premier temps, l'utilisateur décrit des *situations géométriques* qui seront utilisées pour définir des positions clés dans l'espace de travail (voir figure 1.11). Puis, il peut écrire un programme détaillé de *niveau manipulation*. Dans ce programme, la configuration initiale des pièces, les positions intermédiaires et finales du robot sont spécifiées grâce à des *situations géométriques*. Par exemple, les situations décrites dans la figure 1.11 peuvent être utilisées dans le programme suivant:

program demo
begin
    current situation of prism-1 is start-prism-1
    current situation of prism-2 is start-prism-2
    move gripper to achieve approach-prism-1
    open tool to grasp-prism-1+0.5
    move gripper to achieve grasp-prism-1

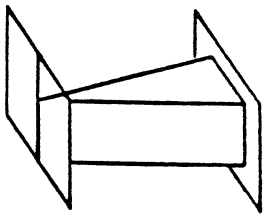
Approach_Prism_1

Face_a of Prism_1 is_against (5) Top of Gripper
Line_c1 of Prism_1 is_aligned_with Line_1 of Gripper
Face_c of Prism_1 is_against (0) finger_2 of Gripper



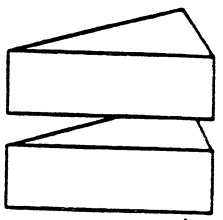
Grasp_prism_1

Face_a of Prism_1 is_against (3) Top of Gripper
Line_c1 of Prism_1 is_aligned_with Line_1 of Gripper
Face_c of Prism_1 is_against (0) Finger_2 of Gripper



Approach_Prism_2

Face_b of Prism_1 is_against (5) Face_a of Prism_2
Line_b1 of Prism_1 is_parallel_to Line_a1 of Prism_2
Line_c1 of Prism_1 is_aligned_with Line_c1 of Prism_2



Prism_1_On_Prism_2

Face_b of Prism_1 is_against (0) Face_a of Prism_2
Line_b1 of Prism_1 is_aligned_with Line_a1 of Prism_2
Line_c1 of Prism_1 is_aligned_with Line_c1 of Prism_2

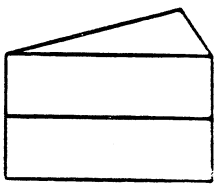


Figure 1.11: Spécification d'un assemblage à l'aide de relations géométriques

open tool to grasp-prism-1-0.5 *until touch*
attach prism-1 to gripper
move prism-1 to achieve approach-prism-2
move prism-1 to achieve prism-1-on prism-2
end

Un système de ré-écriture peut alors utiliser ce programme ainsi que les définitions des situations géométriques pour produire un programme LM exécutable. Par exemple, le programme précédent serait ré-ecrit comme le programe LM suivant:

program demo
;; variable declaration
frame array (2) object
transform array (4) situation
real array (1) opening
begin
;; variable initialization
situation(1):=translation$(t_1)\star$*rotation*(v_1, θ_1)
situation(2):=translation$(t_2)\star$*rotation*(v_2, θ_2)
situation(3):=translation$(t_3)\star$*rotation*(v_3, θ_3)
situation(4):=translation$(t_4)\star$*rotation*(v_4, θ_4)
situation(5):=translation$(t_5)\star$*rotation*(v_5, θ_5)
situation(6):=translation$(t_6)\star$*rotation*(v_6, θ_6)
opening(1):=o_1
specification of the initial situation
*object(1)=station*situation(1)*
*object(2)=station*situation(2)*
;; execution
*move to object(1)*situation(3)*
open to opening(1)$+0.5$
*move to object(1)*situation(4)*
open to opening(1)-0.5 *until touch*
affix object(1)
*move to object(2)*situation(5)*
*move to object(2)*situation(6)*
end

Les symboles $t_1, t_2 ... t_6, v_1, v_2, ..., v_6, \theta_1, \theta_2 ... \theta_6, o_1$ remplacent les valeurs réelles calculées par le système de ré-écriture d'après les relations symboliques. Une version de LM-Géo a été implantée. L'expérimentation du système a été assez difficile pour les raisons suivantes:

1. La description des situations géométriques est longue car elles doivent être décrites de manière à ce qu'elles soient totalement spécifiées. Il serait quelquefois suffisant de décrire une situation géométrique comme "étant au-dessus" sans avoir à spécifier aucune dimension.

2. Il n'est pas aussi facile de spécifier les relations symboliques entre entités géométriques qu'il pourrait paraître car il est assez difficile de donner des noms symboliques à des entités géométriques produites par les systèmes CAO.

3. Le programme final est un programme de niveau effecteur dans l'espace cartésien. Il ne prend pas en compte les problèmes reliés à l'accessibilité, à la configuration du robot ou aux collisions. Toutefois, il est probable que l'un de ces problèmes apparaisse, même dans une application très simple.

LAMA

Une *description d'assemblage* est en principe tout ce qui est nécessaire à LAMA pour planifier entièrement un assemblage. Cette description contient un nombre limité de situations géométriques intermédiaires et peu d'indications quant aux moyens de procéder à l'assemblage. L'exemple suivant peut être considéré comme la *description d'assemblage* d'un piston:

1. Insérer partiellement le goujon du piston dans le piston,

2. placer l'extrémité de la bielle dans le piston,

3. pousser le goujon dans la bielle et le piston.

De nombreux paramètres, ainsi que de nombreuses opérations détaillées permettant de réaliser l' assemblage peuvent être absents de la *description d'assemblage.*
Trois étapes sont nécesssaires afin de traduire cette description en un programme de niveau effecteur détaillé.

1. La première étape consiste à traduire la *description d'assemblage* en un *plan d'assemblage.* Un *plan d'assemblage* contient une description précise et détaillée de l'assemblage, comblant toutes les lacunes de la *description d'assemblage.* Par exemple, quelle pince doit être utilisée pour saisir quelle pièce, ou à quelle profondeur le goujon doit être inséré dans le piston.

2. Le système va décomposer le *plan d'assemblage* calculé en des opérations Prendre-et-Poser successives, et prendre en compte toutes les contraintes géométriques nécessaires à la réalisation d'une opération Prendre-et-Poser en supposant un environnement idéal sans incertitude. Ceci inclut en particulier la planification de trajectoire sans collision ainsi que la planification de prises stables.

3. L'hypothèse d'un environnement idéal étant trop forte, des stratégies d'utilisation des capteurs sont ajoutées au programme précédent afin de former le programme final.

Le travail de R. Taylor

Dans sa dissertation, Taylor [Tay76] décrit des techniques permettant d'engendrer des programmes de contrôle de manipulateurs (AL). Il développe en particulier un mécanisme permettant la prédiction de la propagation des erreurs ainsi que l'ajustement des paramètres des programmes. Un planificateur général est décrit pour l'insertion d'un goujon dans un trou. Ce planificateur engendre automatiquement la manière de saisir le goujon et la stratégie à utiliser afin de réaliser l'insertion. Il incorpore un modèle d'incertitude de la tâche, par exemple: la position de la prise va affecter la précision de la position relative entre le goujon et le trou et, par conséquent, affectera la stratégie visant à effectuer l'assemblage. Plusieurs choix de prises sont évalués afin d'optimiser la phase d'insertion.

AUTOPASS

L'envergure potentielle d'AUTOPASS [LW77] est beaucoup plus grande que de simples opérations Prendre-et-Poser car il inclut la spécification d'opérations telles que le vissage ou l'insertion. Lors de la définition d'AUTOPASS, un important travail a porté sur la définition de l'aspect syntaxique d'un système de programmation niveau tâche. Un analyseur lexicographique ainsi qu'un système de modélisation géométrique [WLL*80] ont été implantés. Ce système comprenait un planificateur Prendre-et-Poser pour un robot cartésien. Le projet AUTOPASS a été considéré comme un cadre d'étude pour la programmation automatique des robots mais le système n'a jamais été complètement implanté.

SHARP

SHARP [LP85] est le système actuellement en développement au LIFIA. Un langage cible a été défini pour SHARP (que nous appelerons TL par la suite). Ce langage est principalement de niveau effecteur mais toutes ses instructions ont des *pré* et *post* conditions très précises. Par exemple, pour l'instruction:

fermer la pince sur les faces f_1 et f_2 de la piece_a

Nous avons les pré et post conditions suivantes:

pré-condition L'angle entre la normale de la face de la pince et la normale de la face*f_1* ne doit pas être supérieur à une certaine valeur.

post-condition la *face f_1* est parallèle à la face de la pince.

Cette notation exprime le fait que la fermeture de la pince amènera la face *f_1* parallèlement aux mâchoires de celle-ci.
Deux étapes sont utilisées successivement afin de produire un programme exécutable: la phase de génération et la phase de correction.

Génération Cette étape permet de produire une première version d'un programme TL. Elle utilise trois planificateurs:

1. Un planificateur de saisie [Per87]
2. Un planificateur de mouvements fins [LT86]
3. Un planificateur de grands mouvements [Ger85]

A l'heure actuelle, ces planificateurs existent indépendament les uns des autres mais sont en passe d'être assemblés. Les planificateurs de saisie ou de mouvements fins peuvent produire plusieurs instructions TL. Le planificateur de grands mouvements ne peut en produire qu'une: *déplacer la pièce-a à la position-b.* Toutefois, un pointeur sur une description complète de la trajectoire est maintenu.

Correction La phase de correction peut être appliquée soit sur des programmes TL produits automatiquement, soit sur des programmes TL écrits par un utilisateur. Elle procède en deux temps: une phase de "vérification" et une phase "d'amendement".

Verification La vérification est accomplie en utilisant le chaînage avant et arrière pour propager les contraintes. [Lat87]. Les post-conditions les plus fortes sont utilisées pour la propagation en chaînage avant et les pré-conditions les plus faibles pour la propagation en chaînage arrière. Une erreur est détectée soit lorsqu'une pré-condition ne correspond pas à une post-condition propagée par chaînage avant, soit lorsqu'une post-condition ne correspond pas à une pré-condition propagée par chaînage arrière.

Amendement de plan Le programme est considéré comme exécutable si aucune erreur n'a été détectée lors de la phase de vérification. Dans le cas contraire, le programme doit être modifié. Afin d'améliorer le programme TL, le système recherche où il est possible de "contrôler" les variables responsables de l'erreur dans le programme. A cet endroit du programme TL, il peut ajouter des appels aux capteurs, ou une nouvelle procédure afin de réduire l'incertitude sur ces variables. La procédure de "vérification et d'amendement" est exécutée jusqu'à ce qu'un programme correct soit obtenu.

TWAIN

TWAIN[LB85] est une architecture proposée pour un système de programmation de robot niveau tâche. Dans cette architecture, l'accent a été mis sur l'étude des techniques de propagation de contraintes [Bro82] comme un moyen de résoudre les problèmes d'interdépendance entre les planificateurs. TWAIN permet la planification de l'espace de travail. Il peut, en principe, trouver la position la plus adaptée d'un objet pour accomplir une tâche donnée. Il permet aussi de planifier le choix des outils nécessaires pour effectuer l'assemblage, comme par exemple déterminer

la longueur d'un tournevis [Tay76]. Plusieurs squelettes de programmes sont utilisables pour la réalisation d'une opération donnée (prise, assemblage de pièces), la description pouvant être effectuée à plusieurs niveaux de détails. Le plan général est le suivant: déplacer le bras, réaliser la saisie, déplacer de nouveau le bras, effectuer l'assemblage éloigner le bras de la positon d'assemblage.

La structure de contrôle proposée est la suivante:

1. Mise en correspondance d'un squelette de programme pour trouver quelles stratégies de saisie et de mouvements fins il faut choisir.
2. Analyse des dépendances afin de trouver un ordre pour assembler les pièces (dans le cas d'assemblages multiples).
3. Propagation des pré-conditions (chaînage arrière). Si une erreur est détectée le système retourne à la première étape pour changer de squelette de programme.
4. Planification grossière de l'espace de travail permettant de trouver une position approximative pour les composants de l'assemblage.
5. Planification des positions de saisie amenant le système à retourner à la phase 4. en cas d'échec.
6. Affectation des variables libres concernant l'espace de travail, les planificateurs de prise et les planificateurs de mouvements fins.
7. Planification des grands mouvements.

TWAIN, lui aussi a servi comme cadre d'étude pour la programmation automatique de robots mais ne fut jamais implanté.

1.8 Discussion

La plupart des systèmes que nous venons de présenter ont (pour certains) donné lieu à une implantation visant à démontrer les principes de base du système. Selon nous, la mise en œuvre de Handey dépasse la simple démonstration de concepts. En effet, nous pouvons à tout moment: mettre en route le système expérimental, changer l'environnement du robot et lancer la planification d'une opération Prendre-et-Poser. Handey étant un système expérimental, l'issue de la planification peut être soumise à certains aléas, mais est en général conforme à notre attente.

Handey et Autopass sont les seuls systèmes intégrant un planificateur de trajectoire fonctionel. Toutefois, le planificateur de trajectoire utilisé dans Handey est plus général du fait qu'il peut manipuler des articulations rotoïdes.

Handey et le Système "Move Instance" ont tous les deux des possibilités de re-saisie. Le planificateur de re-saisie utilisé dans Handey peut manipuler tout polyèdre

alors que le planificateur utilisé dans le système "Move Instance" est limité à des parallélépipèdes.

Le planificateur de saisie utilisé par Handey est similaire au planificateur proposé dans Sharp. Il peut utiliser des données tri-dimensionnelles pour guider l'approche de la pince parmi les obstacles. Dans Sharp, les mouvements de la pince sont limités à deux degrés de liberté en translation alors que la méthode utilisée par Handey ajoute un degré de liberté en rotation, permettant ainsi de trouver plus de solutions au problème de la saisie.

La plupart des systèmes associent l'utilisation de la vision à la planification de la saisie, curieusement aucun d'eux n'envisage l'utilisation de ce capteur pour vérifier la position de l'objet après la saisie. Cette opération est pourtant indispensable dans un cadre expérimental concret. En effet, la poursuite de l'assemblage n'est possible que si l'on connait précisément la position de l'objet que l'on vient de saisir. Handey inclut un planificateur permettant de localiser précisement un objet situé dans la pince.

Chaque module de Handey peut être considéré comme au moins aussi général que les modules équivalents d'autres systèmes; de plus Handey est le seul système implantant une structure de contrôle donnée. Cependant, la planification de mouvements fins reste la partie la plus difficile, elle n'a été abordée de façon pratique dans aucun système à ce jour.

Chapitre 2

POURQUOI LA PROGRAMMATION DES ROBOTS EST-ELLE DIFFICILE ?

Un robot exécutant une opération Prendre-et-Poser ne fait certainement pas démonstration d'une technologie impressionnante. Saisir une pièce et la poser sur une autre semble être une opération simple et facile à programmer. De la même manière que dans le domaine de la vision par ordinateur, la difficulté de programmer des tâches élémentaires n'est pas évidente à première vue. L'impression de facilité est renforcée par certaines démonstrations impressionnantes faites par des robots dans le passé.

Il y a quelques années, Hitachi a présenté un système robotique capable d'assembler un aspirateur à partir de composants posés au hasard sur une table. La figure 2.1 représente un système robotique utilisé pour l'assemblage de cet aspirateur. Ce système est décrit dans [Kas77, Tak77]:

"Les caractéristiques les plus remarquables de ce robot sont: (a) un système de vision multiple, (b) un système à bras et pinces multiples et (c) une coopération visuelle et tactile que nous appelons tacto-visual feed-back control system".

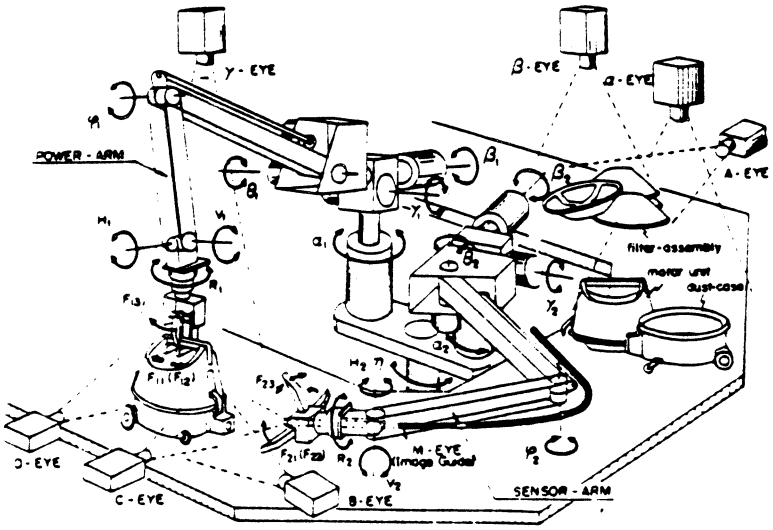


Figure 2.1: L'expérience d'Hitachi (tiré de Tak77)

Le système de robot était capable d'accomplir les tâches suivantes:

Prise dans un vrac. En utilisant ses capteurs visuels, il était capable de choisir la manière la plus aisée de saisir une pièce parmi un vrac.

Reprise. Une fois la pièce tenue par un bras, le second pouvait être utilisé afin de re-saisir la pièce pour exécuter un assemblage.

Insertion flexible Cette opération consistait à monter des pièces flexibles (en caoutchouc souple) sur des sous-assemblages pouvant éventuellement se déplacer lors du montage.

Insertion en force Lors de cette opération, les deux bras réalisaient une insertion en force.

Le système intégrait plusieurs capteurs, y compris un capteur visuel monté sur la pince d'un des bras et possédait des caratéristiques uniques telles que la re-saisie et l'insertion en force. Bien que ce système soit probablement l'un des systèmes les plus ambitieux jamais décrit dans la littérature, les conditions sous lesquelles il opérait étaient probablement très étroites. Ceci montre que l'intérêt principal d' un système est sa généralité plutôt que sa capacité à accomplir des tâches complexes. La leçon principale que l'on puisse tirer de ce genre de démonstration est que le matériel de robotique disponible actuellement offre beaucoup plus de possibilités que celles que les utilisateurs peuvent utiliser du fait d'un manque d'outils de programmation adaptés.

La complexité de la programmation d'un assemblage avec les outils de programmations actuels apparaît lorsque l'on s'attaque à quelques vrais problèmes. Par exemple, il a fallut plusieurs semaines [CSS84] pour programmer l'assemblage partiel d'un pare-choc. Un opérateur ne prendrait pas plus de deux secondes pour

Figure 2.2: Un scène et un robot élémentaires

planifier la même tâche et trente secondes pour l'exécuter. Afin d'arriver à ce que le robot exécute cette tâche, les programmeurs ont dû:

- Concevoir des outils et des attaches mécaniques spéciales afin de maintenir les pièces dans la position appropriée,
- Reconsidérer plusieurs fois la stratégie d'assemblage afin de prendre en compte les butées mécaniques ainsi que la précision du robot à la position d'assemblage,
- Concevoir des capteurs spéciaux afin de compenser l'incertitude du système.

2.1 Une étude de cas très simple: Prendre et Poser

Afin de bien montrer la difficulté de la programmation des robots, nous décrivons la programmation d'une tâche élémentaire consistant à empiler deux blocs, A et B. Nous supposons que le robot est capable de se déplacer dans les directions X et Y et qu'il peut ouvrir et fermer sa pince (figure 2.2).

2.1.1 La programmation par l'exemple

Une solution très simple afin de programmer un assemblage consiste à utiliser un boîtier de commandes manuelles et à enregistrer la trajectoire ainsi que les mouvements de la pince obtenus par télé-opération du robot. Il suffit de re-exécuter la trajectoire et les mouvements de la pince précédemment enregistrés pour faire exécuter de nouveau l'assemblage. Dans ce cas, une seule ligne de programme suffit à la programmation de cette application:

exécuter la trajectoire T.

Cette méthode est très simple et facile à comprendre. C'est la raison pour laquelle elle reste la manière la plus populaire de programmer les robots. Quelquefois, on utilise des simulations CAO du robot et de l'environnement afin d'enseigner et de

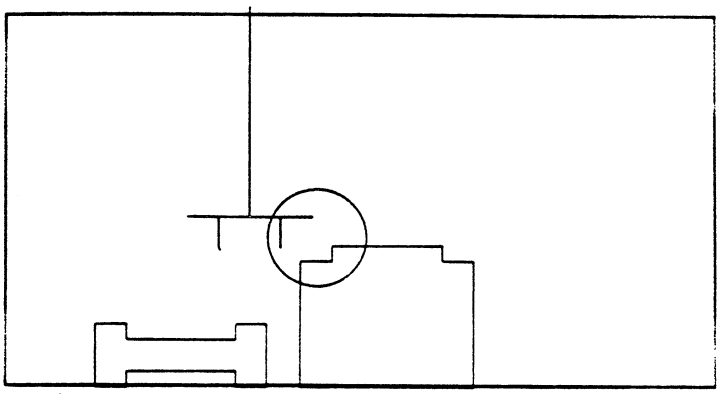


Figure 2.3: Saisie menant à une collision

tester la trajectoire, mais l'essentiel de la méthode reste identique. Une fois que la trajectoire a été enregistrée, il est très difficile de la modifier. En particulier, la prise en compte des données capteurs permettant d'ajuster la tâche est difficile.

2.1.2 La programmation textuelle

Il est évident que le programme ne pourra pas s'exécuter si les blocs A et B ne sont pas à la même place que lors de la phase "d'enregistrement" et ceci peut limiter considérablement l'utilisation du robot. Supposons maintenant que la pièce A puisse être localisée par un capteur. Il est alors possible d'utiliser un langage similaire à un langage de programmation d'ordinateur pour décrire l'assemblage.

debut
    localiser le bloc A
    aller a la prise droite du bloc B
    fermer la pince
    aller a la position d'assemblage
Fin

Si le bloc A est trop près du bloc B (figure 2.3), alors une collision est possible lors de l'approche du robot et une prise alternative doit être choisie.

debut
    localiser le block A
    si le bloc A est a droite du bloc B
      alors aller a la prise droite du bloc A
      sinon aller a la prise gauche du bloc B
    fermer la pince
    aller a la position d'assemblage
fin

Cet exemple souligne bien l'avantage principal de la programmation "textuelle" par rapport à la programmation par l'exemple. En effet, il est possible de prendre en

compte les données capteurs et de prendre des décisions en fonction de leurs valeurs. Le formalisme utilisé pour l'écriture des deux petits programmes précédents est loin d'être celui d'un langage de programmation de robot existant. Le programme suivant se rapproche plus d'un programme réel, excepté qu'il ne comprend aucune déclaration ou initialisation de variables.

```
debut
        lire position_finale dans fichier foo
        affecter position_bloc_A avec valeur_capteur
        si valeur_X(position_bloc_A)>valeur_X(position_bloc_B)
          alors affecter decalage_prise a 1.0
          sinon affecter decalage_prise a -1.0
        aller a (valeur_X(position_bloc_A)+decalage_prise, valeur_Y(position_bloc_B))
        fermer pince
        si valeur_X(position_bloc_A)>valeur_X(position_bloc_B)
          alors aller a position_finale
          sinon aller a (valeur_X(position_finale)-2.0, valeur_Y(position_finale))
fin
```

La majorité des programmes écrits à l'heure actuelle ressemblent à ce programme: en utilisant le boîtier de commandes manuelles, le robot est d'abord déplacé à des positions clés de l'assemblage (position_finale). Ces positions sont sauvegardées (dans le fichier foo) et ensuite utilisées dans le programme d'application. En fait, la programmation "textuelle" et la programmation par l'exemple sont plus ou moins séparables. La plupart des systèmes de programmation textuelle ont leurs utilitaires de programmation par l'exemple [GG78, Ban83] et de nombreux boîtiers de commandes manuelles incorporent des boutons "macros" pouvant engendrer des instructions ressemblant à des instructions de langages textuels. Il en est de même pour les systèmes CAO qui peuvent engendrer des commandes de robots à partir de simulation d'assemblage. Néanmoins, le programme précédant ne tient pas compte de deux problèmes fondamentaux en programmation de robots: la planification de trajectoire et la gestion des incertitudes.

2.1.3 La planification de trajectoire

La planification de trajectoire consiste à déplacer le robot sans qu'il entre en collision avec les obstacles environnants. Si l'on suppose que notre simple robot se déplace en ligne droite, alors le programme suivant a de grandes chances de fonctionner:

```
debut
        lire position_finale, position_approche dans fichier foo
        affecter position_bloc_A avec valeur_capteur
        si valeur_X(position_bloc_A)>valeur_X(position_bloc_B)
          alors affecter decalage_prise a 1.0
          sinon affecter decalage_prise a -1.0
        aller a (valeur_X(position_bloc_A)+decalage_prise,
```

```
            valeur_Y(position_bloc_B) + 4.0)
      aller a (valeur_X(position_bloc_A)+decalage_prise,
            valeur_Y(position_bloc_B))
      fermer pince
      aller a (valeur_X(position_bloc_A)+decalage_prise,
            valeur_Y(position_bloc_B) + 4.0)
      si valeur_X(position_bloc_A)>valeur_X(position_bloc_B)
        alors aller a position_approche
              aller a position_finale
        sinon aller a (valeur_X(position_finale)-2.0,
                       valeur_Y(position_finale))
              aller a (valeur_X(position_approche)-2.0,
                       valeur_Y(position_approche))
fin
```

La stratégie de planification de trajectoire construite dans ce programme est assez simple: la prise est choisie suivant les positions relatives des blocs A et B et le robot est commandé de façon à se déplacer "au-dessus" des obstacles, mais cette stratégie n'est pas du tout générale. Même sans ajouter de nouveaux obstacles, il est possible de trouver une position initiale pour le robot, conduisant à une collision lors du premier mouvement (figure 2.4), ou bien de trouver une position pour le bloc A qui rend le mouvement d'approche impossible du fait de la limitation des butées mécaniques, la tâche restant par ailleurs tout à fait faisable (figure 2.5).

2.1.4 La gestion des incertitudes

Si nous pouvions connaître parfaitement la position et les dimensions des pièces A et B, et si nous pouvions contrôler exactement les déplacements du robot, alors nous pourrions garantir que le programme précédent effectuerait correctement l'assemblage (en supposant que nous ayons résolu les problèmes de collision). En pratique ces conditions ne sont jamais réalisées: la plupart du temps les robots ne se déplacent pas à la position commandée, de plus et la position et la forme des pièces ne sont pas modèlisées avec une précision infinie.
Pour cette raison les programmes construits avec l'hypothèse d'un modèle exact de l'environnement sont en général peu robustes. Les erreurs résultants de l'absence de prise en compte de l'incertitude ne sont pas forcément du même ordre de grandeur que l'incertitude originale. Par exemple, une légère variation de la position de l'objet B en X peut provoquer un réelle collision en Y (figure 2.6).
La gestion des incertitudes consiste à inclure dans le programme initial des stratégies permettant de prendre en compte l'incertitude sur le modèle de l'environnement.

Ces stratégies peuvent ou non inclure l'utilisation de capteurs, mais afin de compléter notre exemple, nous supposerons que le robot est équipé d'un capteur de toucher capable de commander l'arrêt du robot sur un contact. Un appel au sous-programme suivant pourrait remplacer l' instruction qui réalise l'assemblage

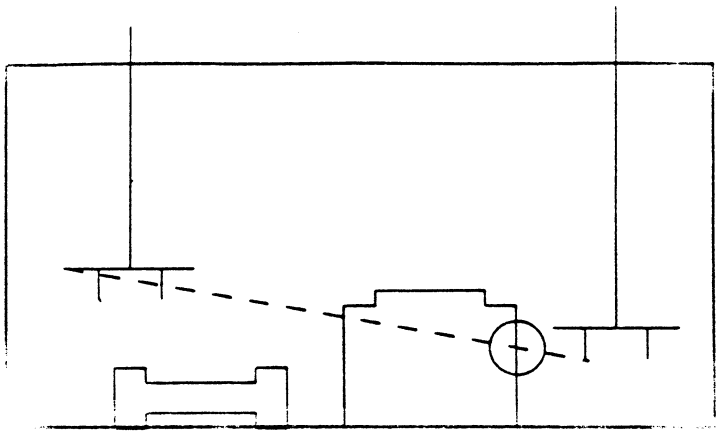


Figure 2.4: Collision avec des obstacles

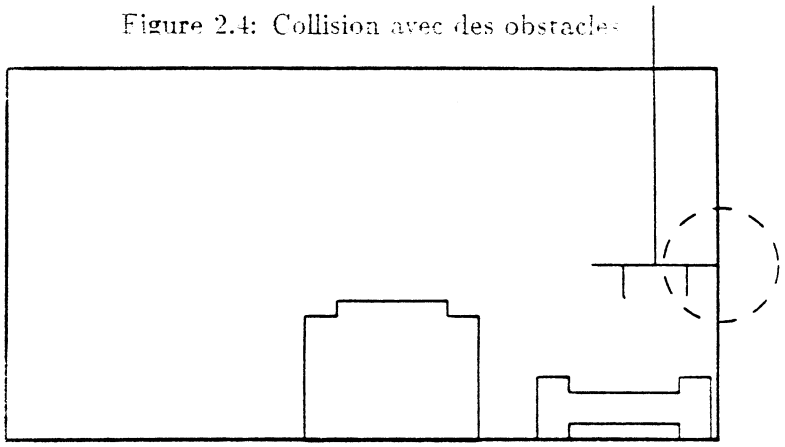


Figure 2.5: Limitation du mouvement par les butées mécaniques

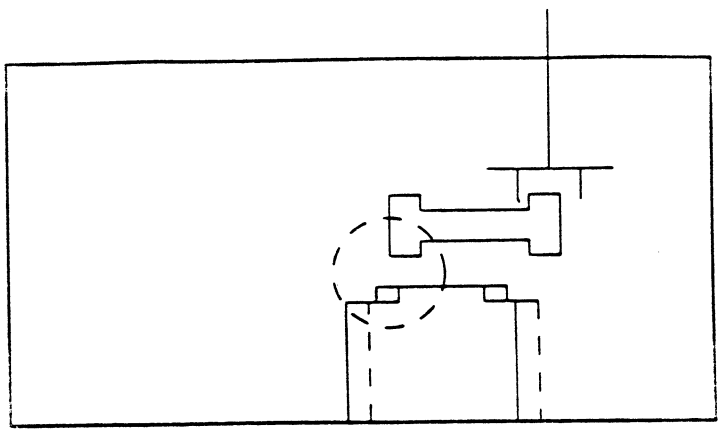


Figure 2.6: Collision dûe aux incertitudes

dans le programme précédent . Ce sous programme effectue une recherche exhaustive de droite à gauche jusqu'à ce qu'il puisse assembler les deux blocs.

```
subroutine doite_gauche(resolution,erreur_maximum)
debut
        sens=1
        registre_1=0
        registre_2=0
        tantque registre_2*resolution<erreur_maximum
                debut deplacement_relatif_Y -3.0 jusqu'a contact
                        si position_Y < 2.0
                          alors succes
                          sinon debut
                                        deplacement_relatif_Y 1.0
                                        registre_1=registre_1+1
                                        registre_2=registre_2+sens*registre_1
                                        deplacement_relatif_X registre_2*resolution
                                        sens=-1*sens
                                fin
                fin
        echec
fin
```

2.2 Une étude de quelques problèmes

L'exemple précédent montre que dans un environnement simple parfaitement connu, la programmation d'une opération de manipulation n'est pas aussi évidente qu'il parait au premier abord. Alors que nous avons mis l'accent sur les problèmes reliés à la géométrie, Taylor, dans sa dissertation [Tay76] a montré, par un simple exemple, que l'on rencontre des problèmes similaires lorsque l'on travaille sur les incertitudes.

2.2.1 Planifier et programmer de manière détaillée

L'écriture de programmes de manipulation prend généralement beaucoup de temps. Une grande partie du code consiste en déclarations et instantiations de variables. Le programme doit être écrit de manière détaillée et, la plupart du temps, de fortes suppositions "cachées" sont utilisées par le programmeur. Par exemple, la position initiale du robot est supposée être "au-dessus" du bloc A afin d'éviter une collision au cours du premier mouvement. C'est la raison pour laquelle la correction d'erreurs en ligne est nécessaire à l'écriture de programmes fiables. La correction consiste à découvrir par essais-erreurs, toutes les suppositions cachées et à adapter le programme afin d'en tenir compte.

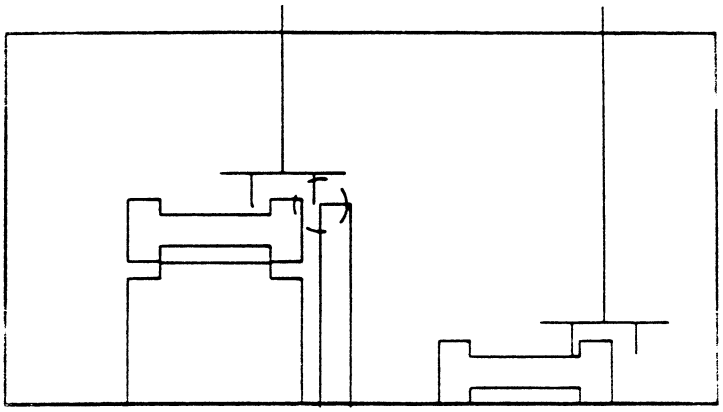


Figure 2.7: Saisie non compatible avec l'assemblage

2.2.2 Détection d'erreurs, récupération d'erreurs et complétude

Notre programme n'est pas prévu pour détecter une situation anormale. La présence du bloc B n'est pas vérifiée dans le sous programme droite-gauche et on suppose que le bloc A sera saisi correctement. La vérification des erreurs peut rendre le programme substantiellement plus long. Essayer de faire un traitement d'erreur pour relancer l'exécution est aussi assez difficile car les causes d'une même erreur peuvent être multiples. L'incertitude fait qu' il est en général très difficile d'écrire des programmes pouvant identifier de façon certaine la réussite d'une manipulation donnée. Ce problème fondamental est discuté dans [Erd84].

2.2.3 Planification globale

L'un des principaux problèmes de la programmation de robot est la difficulté de prendre des décisions basées sur des informations locales. Par exemple, la stratégie utilisée dans le programme principal pour éviter une collision lors de la saisie fonctionne bien si l'on ne considère que la saisie, mais pourrait conduire à un échec lors d'une opération d'assemblage (voir figure 2.7). De même la figure 2.5 montre que cette stratégie conduit le robot à une butée mécanique alors que l'on aurait pu, dans ce cas, choisir l'autre prise. L'aspect global de la planification d'opérations de manipulation est l'obstacle majeur à l'utilisation de sous-programme.

2.2.4 Généralisation

L'écriture d'un programme Prendre-et-Poser dans le cadre définit pour notre simple exemple n'est certainement pas un travail insurmontable. Le programme définitif serait probablement beaucoup plus long. Toutefois il serait très difficile de le généraliser et de le représenter comme une procédure Prendre-et-Poser universelle. Le nombre de paramètres va augmenter très vite si l'on change la dimension des pièces ou si l'on ajoute des obstacles. Dans ce cas, invoquer le programme peut s'avérer aussi complexe que de l'écrire. Modifier, même légèrement, la tâche va détruire complètement la structure du programme, à tel point qu'il ne sera plus possible de l'utiliser.

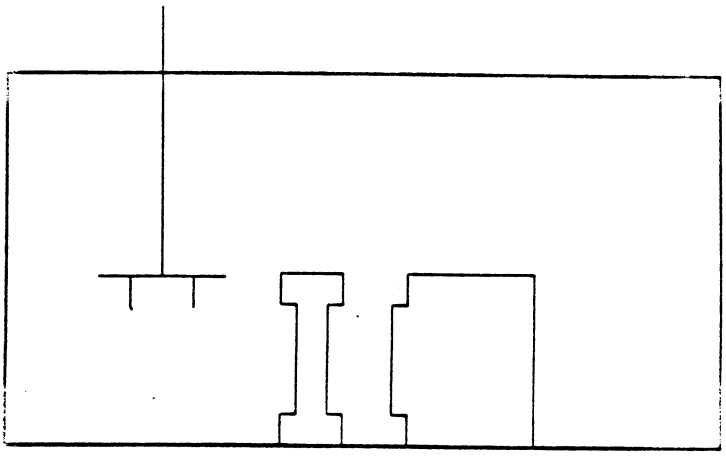


Figure 2.8: Le même assemblage peut conduire à une stratégie de montage complètement différente.

Par exemple, procéder à l'assemblage horizontalement au lieu de verticalement (figure 2.8) nécessite un programme totalement différent, alors que la tâche est approximativement la même. L'assemblage final est identique, mais la nature des opérations nécessaires change radicalement:

- Une seule stratégie de saisie est nécessaire.
- Une stratégie visant à pousser le bloc est la seule manière d'effectuer la tâche.

De plus, il est très difficile d'établir une relation entre des modifications qualitatives d'une opération de manipulation et les critères dimensionnels de cette opération. Des modifications minimes de la position de la pièce peuvent provoquer un changement complet de la stratégie, alors que la tâche resterait qualitativement identique avec des modifications dimensionnelles plus importantes.

2.2.5 Raisonnement géométrique

Les robots cartésiens sont relativement simples à visualiser et prédire leur comportement n'est pas très compliqué. Cela devient beaucoup plus complexe lorsque l'on utilise des articulations rotoïdes ou lorsque l'utilisateur doit programmer une application utilisant des rotations dans l'espace. La programmation cartésienne ne peut résoudre qu'une partie des problèmes du fait qu'elle n'est pas aisément compatible avec les configurations du robots ou les butée mécaniques. Un bon exemple pour illustrer ce problème est de considérer le programme suivant:

```
tant que non EOF
        debut
                lire position_ cartesienne dans fichier foo
                deplacer robot a position_cartesienne*decalage_cartesien
        fin
```

L'utilisation de ce programme permet théoriquement de programmer une tâche à une position donnée et de la faire exécuter à toute autre position choisie dans l'espace

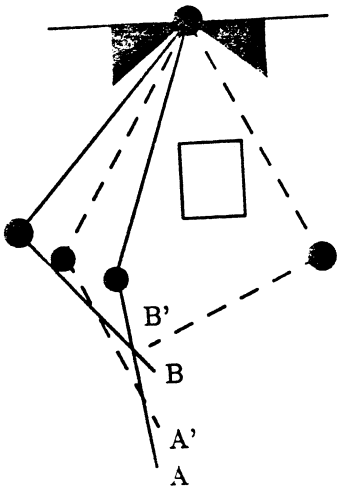


Figure 2.9: Une petite marge cartésienne provoque une reconfiguration du robot

de travail, en changeant simplement la valeur de *décalage_cartésien.* En fait, ce genre de programme n'est valide que pour des décalages limités. Tout décalage plus importante provoquera soit une re-configuration du robot soit un arrêt sur butée mécanique. Par exemple, d'un point de vue des mouvements dans l'espace cartésien, il y a très peu de différence entre aller de A à B ou aller de A' à B' (voir figure 2.9). Dans les deux cas, tous les points appartiennent à l'espace de travail du robot et sont des positions valides. Mais, même un petit décalage oblige le système à opérer une reconfiguration pour atteindre B', ce qui, dans ce cas, provoquera une collision. Ce type d'erreur est très répandu dans les systèmes de programmation de robots actuellement disponibles.

Chapitre 3

LA STRUCTURE DE CONTROLE

3.1 Le problème

Nous cherchons à résoudre le problème suivant: Trouver une trajectoire générale gp (une trajectoire générale incluant les mouvements de la pince) tel que le prédicat Prendre-et-Poser(gp) soit vérifié. Ce prédicat peut être considéré comme une généralisation du prédicat Trajectoire-sûre(p) qui est vrai lorsque la trajectoire p est sans collision. Quelques systèmes de modélisation géométrique possèdent ce type de prédicat, et il est possible de tester si une trajectoire est sûre sur ce type de système. Toutefois la difficulté ne réside pas dans le fait de tester une trajectoire mais dans le fait d'en engendrer une qui ait cette propriété. La même chose est vraie pour le prédicat Prendre-et-Poser. A notre connaissance il n'existe pas de système de CAO avec un prédicat Prendre-et-Poser, mais, en implanter un ne serait pas très difficile en utilisant le prédicat Trajectoire-sûre:

1. Tester si la trajectoire allant de la position initiale à la position de saisie est sûre.

2. Tester si la pince se ferme sur l'objet à saisir, et si la prise est stable en considérant par exemple des critères de surface, de contact ou d'excentricité.

3. En utilisant le dernier point de la trajectoire à tester, vérifier que la position de saisie de l'objet est cohérent avec sa position finale.

Une méthode grossière pour calculer une trajectoire générale serait *d'engendrer* toutes les trajectoires possibles et de les *vérifier* à l'aide du prédicat Prendre-et-Poser. Cette forme générale de structure de contrôle est connue sous le nom de génération-vérification. Notre objectif est de transformer ce planificateur grossier en un planificateur plus efficace. Pour cela nous proposons un shéma de raisonnement systématique conduisant à une décomposition descendante du problème et permettant de tenir à jour les hypothèses simplificatrices faites dans cette décomposition.

3.2 Génération-vérification "intelligente"

Pour illustrer les techniques de bases que nous avons utilisées dans la réalisation de Handey, considérons un problème géométrique simple. Supposons que le problème consiste à trouver un espace libre pour un polyèdre A sur une table encombrée d'obstacles B_i. Ignorons, les rotations possibles de A pour conserver au problème sa simplicité. L'approche génération-vérification à ce problème consisterait à explorer systématiquement toutes les positions de l'objet A sur la table et à vérifier pour chacune de ces positions si A entre en collision avec l'un des B_i. Si l'on trouve l'un de ces points, alors nous avons une solution. Bien sûr, si la table est très encombrée, la méthode n'est pas très efficace.

Il est possible d'améliorer le générateur de solutions potentielles en faisant un calcul simple qui permette d'augmenter les chances de tester une bonne position. Ceci évite de nombreux tests coûteux en calculs. Par exemple, il est évident que la position de A est sûre si tous les points de A sont sûrs, et, en particulier un point a de A. Vérifier d'abord que ce point est en dehors des obstacles pour une position donnée est beaucoup plus rapide que de calculer l'intersection entre deux polyèdres. Donc, avant d'engendrer une position possible pour a, nous testons si cette position est à l'extérieur des obstacles posés sur la table. De cette manière on peut construire une fonction génération-vérification plus efficace. Nous utilisons une condition nécessaire pour que la position de A soit sûre: la position d'un point de référence de A, a est sûre. On remarque, sur cet exemple, que si la condition nécessaire est facile à tester, elle permet de réduire rapidement l'espace de recherche.

Nous pouvons décider que le test visant à vérifier qu'un point appartient à un polyèdre est par lui-même trop long. Pour chaque obstacle nous construisons un parallélépipède englobant, et, au lieu de vérifier que le point de référence appartient au polyèdre, nous vérifions simplement si il appartient au parallélépipède englobant. Ceci est bien plus rapide mais bien sûr moins général. Dans ce cas, nous utilisons une condition suffisante pour tester si a appartient à un polyèdre. Cette technique nous permet aussi de réduire rapidement l'espace de recherche, au prix d'une perte de généralité.

Finalement l'idéal est de posséder une méthode pour calculer les obstacles dans l'espace des configurations de l'objet A. Il suffit alors de choisir un point libre de cet espace pour avoir une solution. Pour cela on peut utiliser la méthode suivante.

Par exemple, nous pouvons exploiter le fait qu'une position dans l'espace des configurations pour A est composée d'un couple de variables $< x, y >$. Nous pouvons alors définir la structure de la fonction génératrice en décidant d'engendrer des valeurs plausibles pour x en premier, puis des valeurs possibles de y (qui dépendront de la valeur de x précédement choisie). Dans ce problème, nous pouvons considérer une décomposition cylindrique des obstacles de l'espace des configurations, il est clair qu'une valeur sûre pour y ne peut changer que pour les valeurs de x correspondant à un sommet ou à l'intersection de deux arêtes de l'obstacle. Pour trouver une solution, il nous suffit d'engendrer un nombre fini de valeurs en x et de faire une recherche exhaustive en y. Cette autre technique visant à réduire la dimension de l'espace de recherche consiste à fragmenter les variables initiales pour n'avoir à explorer que des sous-ensembles de l'espace de recherche.

Nous venons d'illustrer la collection de techniques de base que nous utilisons pour rendre une fonction de génération-vérification plus efficace. Avant de rentrer plus en détail dans la méthode, nous introduisons quelques notations.

3.3 Notation et Définition

Nous pouvons représenter un ensemble S soit par un prédicat $T(s)$ qui est vrai si et seulement si: $s \in S$, ou par une fonction qui engendre les éléments de l'ensemble. Pour éviter d'introduire des notations supplémentaires, nous utiliserons le même nom pour la fonction génératrice et l'ensemble. Une fonction génératrice est définie de la façon suivante:

$S(Nil) \rightarrow s_1$
$S(s_n) \rightarrow s_{n+1}$
et si S est un ensemble fini de N elements: $S(s_N) \rightarrow Nil$

La fonction génératrice est appelée la première fois avec NIL pour obtenir le premier élément, puis avec l'élément précédent pour les suivants. Dans la suite, nous omettrons l'argument passé à la fonction génératrice en supposant simplement qu'elle tient à jour un registre contenant la dernière valeur engendrée. Dans le cas d'un ensemble de cardinalité finie, la fonction génératrice indique que l'ensemble est épuisé avant de se remettre, de nouveau, à engendrer les éléments de l'ensemble.
Si S est le produit cartésien de deux ensembles S_1 et S_2:

$$S = S_1 \otimes S_2 = \{< s_1, s_2 > | s_1 \in S_1 \text{ et } s_2 \in S_2\}$$

Alors on peut définir une fonction génératrice en "profondeur d'abord" simplement par:

fonction generatrice $S(x,y)$
$S_2(y) \rightarrow y'$
si y' *alors retourner* (x,y')
sinon $S_1(x) \rightarrow x'$
si x' *alors retourner* $S(x', Nil)$
sinon retourner (Nil,Nil)

Dans la suite nous noterons plus simplement cette fonction par:

$$S_1 \rightarrow x, S_2 \rightarrow y$$

Ceci indique simplement que nous produisons d'abord un élément x de l'ensemble S_1 puis un élément y de l'ensemble S_2 et ceci en "profondeur d'abord" (tous les éléments de S_2 sont engendrés pour un élément donné de S_1). La plupart du temps la fonction génératrice de S_2 va dépendre de la valeur choisie pour l'ensemble S_1. Dans ce cas, on donne x comme paramètre à la fonction génératrice de S_2 et l'on écrit:

$$S_1 \rightarrow x, S_2(x) \rightarrow y$$

qui est équivalent à:

fonction generatrice $S(x,y)$
$S_2(x,y) \rightarrow y'$
si y' *alors retourner* (x,y')
sinon $S_1(x) \rightarrow x'$
si x' *alors retourner* $S(x', Nil)$
sinon retourner (Nil,Nil)

Une autre méthode pour spécifier un ensemble est l'utilisation d'un prédicat $T(s)$. Nous définissons le *noyau* de $T(s)$ comme étant le sous-ensemble du domaine de $T(s)$ tel que $T(s)$ soit vrai pour tous les éléments de cet ensemble. Nous notons cet ensemble:

$$K(s) : T(s) \equiv \{s|T(s)\}$$

Si nous avons une fonction génératrice pour le domaine du prédicat D, et si nous supposons que cette fonction est réservée (qu'elle ne peut pas être appelée par une autre fonction), alors nous pouvons définir une fonction génératrice pour le noyau d'un prédicat de la manière suivant:

fonction generatrice $[K(s) : T(s)](x)$
$D(x) \rightarrow x$
tant que $\neg T(x)$ *faire* $D(x) \rightarrow x$
retourner x

Ce type de fonction est souvent connu sous le nom de fonction de génération-vérification. Dans la suite nous représenterons ce type de fonction avec la notation suivante:

$$D \rightarrow d \parallel T(d)$$

L'expression à gauche des barres parallèles correspond à la partie génération et l'expression à droite, à la partie vérification. Dans le cas d'un prédicat à plusieurs variables $T(a,b)$, $K(< a,b >) : T(a,b)$, peut être écrit comme:

$$A \rightarrow a, B \rightarrow b \parallel T(a,b)$$

L'expression à gauche des barres parallèles est une fonction génératrice produisant des valeurs pour le couple $< a,b >$ et la partie droite correspond au test.

Si le domaine D est grand et si le prédicat T est long à calculer, alors engendrer $K(s) : T(s)$ par ce biais peut être irréaliste. Dans la suite, nous discutons d'une méthode générale pour améliorer la fonction génération-vérification initiale.

3.4 Optimisation

Si K' et K'' sont deux fonctions génératrices d'un ensemble K, nous considérons que K'' est plus performant que K', si K'' engendre K dans un temps plus court que K'. Nous utilisons trois types d'heuristiques pour obtenir une meilleure fonction génération-vérification. Toutes ces heuristiques sont basées sur l'hypothèse qu'à un certain niveau de décomposition on peut trouver des fonctions génératrices efficaces.

Utilisation des conditions nécessaires

Utiliser des conditions nécessaires est un bon moyen de "filtrer" le domaine d'un prédicat, ceci permet de ne passer que des éléments potentiellement "bons" à la fonction finale de test.
Supposons que T' soit une condition nécessaire de T:

$$\forall d \in D, T(d) \Rightarrow T'(d)$$

alors,

$$K(d) : T(d) \subseteq K(d) : T'(d) \subseteq D$$

si $K(d) : T'(d)$ est substantiellement plus petit que D, la fonction:

$$K(d) : T'(d) \rightarrow d \parallel T(d)$$

est une meilleure fonction génératrice que

$$D \rightarrow d \parallel T(d)$$

s'il existe une fonction génératrice efficace pour $K(d) : T'(d)$. Par exemple, si $T'(d)$ peut être calculée très rapidement et si elle est très discréminante, alors la fonction:

$$D \rightarrow d \parallel T'(d)$$

peut être considérée comme une fonction génératrice efficace pour $K(d) : T'(d)$. Il est important de noter que ce type d'amélioration se fait sans perte de généralité.

Utilisation des conditions suffisantes

Dans la plupart des cas, nous sommes prêts à sacrifier la généralité à l'efficacité. Et au lieu de chercher à engendrer tous les éléments d'un ensemble, nous ne cherchons à engendrer qu'un sous-ensemble non vide de l'ensemble des solutions. Bien sûr cela se fait au risque d'éliminer toutes les solutions valides. Une méthode pour engendrer un sous-ensemble des solutions valides est d'utiliser un prédicat T' plus "fort" que T, autrement dit un prédicat qui élimine plus d'éléments que T sur D.
Supposons que T' soit une condition suffisante de T:

$$\forall d \in D, T'(d) \Rightarrow T(d)$$

alors,

$$K(d) : T'(d) \subseteq K(d) : T(d) \subseteq D$$

encore une fois,

$$K(d) : T'(d) \rightarrow d$$

sera une meilleure fonction génératrice que:

$$D \rightarrow d \parallel T(d)$$

s'il existe une fonction génératrice efficace pour $K(d) : T'(d)$. Par exemple:

$$D \rightarrow d \parallel T'(d)$$

sera considérée comme meilleure génératrice si T'(d) est rapidement calculable.

3.4.1 Projection des variables

Projeter une variable $d \in D$ sur deux sous espaces D_1, D_2 peut réduire la recherche sur D. Soit T(d), un prédicat qui caractérise l'ensemble des solutions cherchées. Supposons que $d =< d_1, d_2 >$ et que le prédicat $T'(d_1, d_2)$ caractérise le même ensemble que $T(d)$. Ainsi au lieu de construire la fonction génératrice pour $K(d) : T(d)$, nous pouvons construire celle pour $K(< d_1, d_2 >) : T'(d_1, d_2)$:

$$D_1 \rightarrow d_1, D_2 \rightarrow d_2 \parallel T'(d_1, d_2)$$

Il y a deux raisons pour préférer cette décomposition à la fonction génération-vérification initiale.

1. Il existe un prédicat $T'_1(d_1)$ rapide à calculer et suffisament discriminant tel que:

$$T'(d_1, d_2) \Rightarrow T'_1(d1)$$

sans perte de généralité, on peut écrire que:

$$K(d_1) : T'_1(d_1) \rightarrow d_1, D_2 \rightarrow d_2 \parallel T'(d_1, d_2)$$

est plus efficace que:

$$D_1 \rightarrow d_1, D_2 \rightarrow d_2 \parallel T'(d_1, d_2)$$

2. Si il existe une fonction génératrice efficace pour l'ensemble $K(d2) : T'(d_1, d_2)$ paramétrée par d_1 , on peut utiliser la fonction génératrice simplement donnée par:

$$D_1 \rightarrow d_1, K(d_2) : T'(d_1, d_2) \rightarrow d_2$$

Dans certain cas on peut appliquer en même temps les deux méthodes et l'on obtient:

$$K(d_1) : T'_1(d_1) \rightarrow d_1, K(d_2) : T'(d_1, d_2) \rightarrow d_2$$

qui peut être considérée comme une "bonne" fonction génératrice de $K(d) : T(d)$.

3.5 La méthode

Construire un planificateur pour l'opération de manipulation Prendre-et-Poser revient à construire une fonction génératrice pour un sous-ensemble non vide du noyau du prédicat Prendre-et-Poser(gp, W). Où gp est une trajectoire générale et où W est une constante qui représente les caractéristiques de la tâche et le modèle de l'environnement. Il n'est pas nécessaire que cette fonction génératrice engendre toutes les trajectoires possibles conduisant à une opération Prendre-et-Poser donnée, par contre elle doit pouvoir au moins produire une trajectoire pour toutes les descriptions de l'environnement et les spécifications des caractéristiques de la tâche.

Il nous faut distinguer deux problèmes:

1. Définir la fonction calculable que nous associons au prédicat, ceci inclut en particulier la définition des structures de données pour représenter l'environnement et la méthode de calcul pour décider si une trajectoire gp est ou non une trajectoire réalisant l'opération de manipulation désirée.

2. Trouver une fonction génératrice pour un sous ensemble non vide du noyau de ce prédicat qui puisse dans un temps raisonnable exhiber un élément de ce sous-ensemble.

Dans le reste du chapitre nous nous concentrerons sur le second problème en partant de la définition non formelle que nous avons donnée pour le prédicat Prendre-et-Poser. Bien que cette définition soit non formelle, il est clair que nous pouvons effectivement écrire une fonction calculable qui, étant donnés un modèle polyèdrique de l'environnement et une trajectoire générale, testerait si: la trajectoire d'approche est sans collision, la prise est stable et la trajectoire finale est sans collision. La fonction génératrice que nous souhaitons écrire doit idéalement avoir trois propriétés.

1. Elle doit être complète: pour toute description W elle doit pouvoir exhiber une trajectoire gp.

2. Elle doit être efficace: le temps de calcul d'une solution doit être suffisament court.

3. Elle doit être correcte: elle ne doit pas produire de trajectoire qui ne vérifierait pas le predicat Prendre-et-poser.

Nous verrons que nous ne garantissons "mathématiquement" aucune de ces trois propriétés pour la fonction génératrice que nous avons construite. Par exemple:

1. Il existe des environnements W ou il est possible de trouver une trajectoire sans collision alors notre fonction génératrice n'en produit aucune.

2. Nous ne pouvons pas réellement limiter le temps de calul nécessaire à l'obtention d'une solution.

3. Nous faisons des approximations qui peuvent introduire des erreurs dans des cas limites.

Cependant notre expérimentation montre que notre fonction génératrice possède ces propriétés pour les configurations couramment rencontrées de W.

Il nous faut maintenant nous interroger sur le premier problème:

Définir le prédicat calculable associé à une opération de manipulation.

Cette définition est directement reliée à la sémantique que l'on associe à une opération de manipulation. Ecrire un tel prédicat revient en fait à faire une *théorie* sur la manipulation en question. On cherche une représentation adéquate de l'environnement et l'on définit des procédures de calculs sur cette représentation. Lorsque l'on bâtit une théorie sur une partie du monde réel (ici le robot et son environnement) on se pose en général deux questions:

1. cette théorie est elle juste ?

2. cette théorie est elle générale ?

Pour répondre à ces questions il faut s'interroger pour savoir si le mode de représentation et les calculs sont appropriés à l'opération réelle que nous voulons modéliser. Il faut aussi savoir si nous avons compris le *sens* que l'on attribue à une opération de manipulation donnée. Par exemple le prédicat que nous choisissons dans ce chapitre pour représenter une opération Prendre-et-poser est assez pauvre car il n'inclut pas la notion de ré-orientation de l'objet transporté. Cette opération est pourtant très importante car elle est nécéssaire dans presque toute opération Prendre-et-poser.

Nous voyons que le premier problème est relié à la compréhension de ce qu'est une opération de manipulation, et à son mode de représentation alors que la deuxième question est reliée à l'implantation de cette opération.

3.6 La structure de Handey

3.6.1 Niveau d'optimisation zéro

Nous choisissons de projeter gp et W de la manière suivante:

$$gp =< gc, ip, fp >$$

- gc: est la position relative cartésienne de l'objet et de la pince au moment de la fermeture de la pince, cette position peut être transformée en une position pour le robot (ensemble de coordonnées articulaires)
- ip: la trajectoire (séquence de coordonnées articulaires) du robot de la position initiale à la position de saisie.
- fp: la trajectoire du robot de la position de saisie à la position finale.

$$W =< I, IC, FC, W_1 >$$

- I: est la position initiale du robot.
- IC: est la position initiale cartésienne de l'objet à saisir.
- FC: est la position finale cartésienne de l'objet.
- W_1: est le reste de la description de l'environnement.

Nous décomposons le prédicat Prendre-et-Poser de la façon suivante:

$$\text{Prendre-et-Poser}(< gc, ip, fp >, < I, IC, FC, W_1 >)$$
$$\Longleftrightarrow$$
$$\text{Trajectoire-sûre}(I, loc(gc, IC), ip, W_1) \wedge \text{ Bonne-prise}(gc, IC, FC, W_1) \wedge$$
$$\text{Trajectoire-sûre}(loc(gc, IC), loc(gc, FC), fp, W_1)$$

- Trajectoire-sûre(i, f, p, w) est un prédicat permettant de tester si une trajectoire (p) entre deux positions articulaires (i, f) est sans collision dans l'environnement W.

- Bonne-prise(gc, IC, FC, w) est un prédicat permettant de tester si une position de saisie (gc) est libre de toute collision à la position initiale (ic) et à la position finale (fc) dans l'environnement w, et, si la prise est stable.

- $loc(a, b)$ représente une fonction qui calcule essentiellement la transformation de coordonnée inverse du robot. Étant donnée la position de la pince relative à la pièce à saisir et la position de cette pièce, elle calcule l'ensemble des coordonnées articulaires du robot correspondant à cette position de la pince. Dans le cas général, il y a plusieurs solutions à ce problème pour un couple de données (a, b). En principe cette fonction devrait aussi être considérée comme une fonction génératrice de l'ensemble des solutions au problème de la transformation de coordonnées inverse. Bien que cette approche ait été prise dans Handey, nous l'ignorerons dans la suite de ce chapitre.

La fonction génératrice la plus simple pour le prédicat Prendre-et-Poser peut s'écrire:

$$C \rightarrow gc, P \rightarrow ip, P \rightarrow fp \,||\, \text{Prendre-et-Poser}(< gc, ip, fp >, < I, IC, FC, W_1 >)$$

Où C est l'ensemble des positions cartésiennes et P l'ensemble des trajectoires. Bien évidemment, il n'est pas question d'implanter une telle fonction et nous pouvons utiliser la décomposition précédente pour la remplacer par la fonction génératrice suivante:

$$\begin{aligned} K(gc) &: \text{Bonne-prise}(gc, IC, FC, W_1) \rightarrow gc, \\ K(ip) &: \text{Trajectoire-sûre}(I, loc(gc, IC), ip, W_1) \rightarrow ip, \\ K(fp) &: \text{Trajectoire-sûre}(loc(gc, IC), loc(gc, FC), fp, W_1) \rightarrow fp \end{aligned}$$

S'il existe des fonctions génératrices efficaces pour les noyaux des prédicats Trajectoire-sûre et Bonne-prise, alors nous utilisons les arguments suivants pour montrer que la fonction génératrice en profondeur d'abord que nous venons de décrire est efficace.

- En engendrant la prise de l'objet en premier, nous résolvons ce qui est considéré comme étant la partie la plus difficile de l'assemblage en premier.

- En choisissant cette décomposition de variables, nous contraignons les fonctions génératrices de trajectoires à engendrer des trajectoires d'une position initiale donnée à une position finale donnée. Ce faisant, nous supposons qu'il est en général possible d'aller d'un point à un autre.

Toutefois il est possible de trouver une "bonne" prise qui peut rendre l'un des mouvements d'approche ou d'éloignement totalement impossible (voir figure 3.1).

On peut remarquer que l'appel aux deux fonctions génératrices de trajectoire ne dépend que de gc et qu'il est donc possible d'effectuer l'appel à ces deux fonctions en parallèle une fois gc engendré par Bonne-prise.

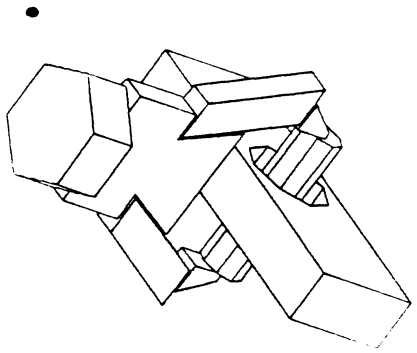


Figure 3.1: Dans cet exemple, la prise est stable et sans collision mais il est impossible de s'écarter de cette position

3.6.2 Optimisation niveau 1

Dans cette section nous décrivons comment rendre plus efficaces les fonctions génération-vérification pour les prédicats Bonne-prise et Trajectoire-sûre.

3.6.3 Optimisation de la fonction génératrice Bonne-prise

Nous donnons la définition suivante au prédicat Bonne-prise

$$\text{Bonne-prise}(gc, IC, FC, W_1)$$
$$\Longleftrightarrow$$
$$\text{Prise-stable}(gc, W_1) \ \wedge \ \text{Position-sûre}(loc(gc, IC), W_1) \ \wedge \ \text{Position-sûre}(loc(gc, FC), W_1)$$

Nous pouvons définir une fonction génératrice pour le noyau du prédicat Bonne-prise comme étant:

$$K(gc)\text{:Prise-stable}(gc, W_1) \rightarrow gc \ \| \ \text{Position-sûre}(loc(gc, IC), W_1) \ \wedge \ \text{Position-sûre}(loc(gc, FC), W_1)$$

Cette fonction génération-vérification est acceptable si l'on possède une fonction génératrice efficace pour le noyau de Prise-stable. En effet, les predicats Position-sûre peuvent être rapidement calculés.
Pour obtenir une fonction génératrice efficace pour le noyau de Prise-stable, nous définissons le prédicat Prise-parallèle-stable$(gc, face_1, face_2, W_1)$. Ce prédicat indique qu'il est possible de saisir la pièce par deux faces parallèles $(face_1, face_2)$ et que cette prise est stable. Ce prédicat est une condition suffisante pour le prédicat Prise-stable. Si la fonction génératrice

$$K(gc)\text{:Prise-parallèle-stable}(gc, face_1, face_2, W_1)$$

est efficace et permet de générer un ensemble non vide de solutions gc, alors elle est certainement meilleure (bien que moins générale) que la fonction:

$$C \rightarrow gc \parallel \text{Prise-stable}(gc, W_1)$$

Il est important de noter qu'en prenant une condition suffisante pour optimiser la fonction génératrice du noyau de Prise-stable, nous le faisons au risque d'éliminer des solutions potentiellement bonnes, essentiellement toutes les prises ne mettant pas en jeu des faces parallèles. Ce choix peut se justifier si l'on suppose que l'ensemble des prises s'appuyant sur des faces parallèles est suffisament "riche" en éléments pour fournir des solutions alternatives lors de la recherche d'une solution globale. Nous remarquons aussi que les solutions potentielles laissées de côté sont identifiables (dans ce cas, les prises autres que les prises s'appuyant sur des faces parallèles).

Nous voulons maintenant trouver une fonction génératrice efficace pour Prise-parallèle-stable. Nous utilisons une condition nécessaire de ce prédicat Faces-parallèles($face_1, face_2$) qui indique que deux faces du modèle géométrique de l'objet à saisir sont parallèles.
Ceci nous permet d'écrire la fonction génératrice suivante:

$$K(face_1, face_2)\text{:Faces-parallèles}(face_1, face_2) \rightarrow (face_1, face_2),$$
$$[C \rightarrow gc \parallel \text{Prise-parallèle-stable}(gc, face_1, face_2, W_1)] \rightarrow gc$$

Il est facile d'écrire une fonction génératrice efficace pour:

$$K(face_1, face_2)\text{:Faces-parallèles}(face_1, face_2).$$

Finalement il nous reste à écrire une fonction génératrice pour:

$$[C \rightarrow gc \parallel \text{Prise-parallèle-stable}(gc, face_1, face_2, W_1)]$$

Une fois encore nous utilisons une condition suffisante au prix de la généralité. Nous calculons avec $face_1, face_2$ un ensemble fini de points de prise stable sur les faces $face_1, face_2$ qui ont la proprieté Prise-parallèle-stable($gc, face_1, face_2, W_1$). Nous appelons Ensemble-fini-de-prise-stable($face_1, face_2$) la fonction génératrice de cet ensemble.
Finalement nous obtenons une fonction génératrice du noyau du prédicat Bonne-prise:

$$K(face_1, face_2)\text{:Faces-parallèles}(face_1, face_2) \rightarrow (face_1, face_2),$$
$$\text{Ensemble-fini-de-prise-stable}(face_1, face_2) \rightarrow gc$$
$$\parallel \text{Position-sûre}(loc(gc, IC), W_1) \ \wedge \ \text{Position-sûre}(loc(gc, FC), W_1)$$

Bien évidemment cette fonction génératrice n'engendre pas tous les éléments du noyau de bonne prise, en fait elle n'engendre qu'un ensemble discret de prises stables sur des faces parallèles.

Optimisation de la fonction génératrice Trajectoire-sûre

Considérons la projection suivante pour une trajectoire p et pour l'environnement W_1

$$p =< ipt, ip, fp >$$

- ipt: est une position intermédiaire
- ip: est une trajectoire jusqu'à ipt
- fp: est une trajectoire depuis fp

$$W_1 =< DM, WM, W_2 >$$

- DM: est la carte tri-dimensionnelle de la V-zone donnée par le système de vision tri-dimensionnelle.
- WM: est le modèle de l'environnement n'incluant pas la V-zone
- W_2: est le reste de la description de l'environnement.

Nous utilisons les conditions nécessaires suivantes pour le Trajectoire-sûre:

- Trajectoire-sûre(i, f, p, W_1) $\Rightarrow$ Position-sûre$(i, W_1) \wedge$Position-sûre(f, W_1)
- Trajectoire-sûre$(i, f, p, < DM, WM, W_2 >)$ $\Rightarrow$ Trajectoire-sûre-pince(i, f, p, DM)
- Trajectoire-sûre$(i, f, p, < DM, WM, W_2 >)$ $\Rightarrow$Trajectoire-sûre(i, f, p, WM)

Le prédicat Trajectoire-sûre-pince teste la trajectoire de la pince en ignorant le reste du robot.
Nous pouvons utiliser la condition nécessaire et suffisante:

$$\text{Trajectoire-sûre}(i, f, < ipt, ip, fp >, W_1) \Leftrightarrow \text{Trajectoire-sûre}(i, ipt, ip, W_1) \wedge \text{Trajectoire-sûre}(ipt, f, fp, W_1)$$

pour construire la fonction génératrice suivante pour Trajectoire-sûre:

$$K(< ipt, fp >) : \text{Trajectoire-sûre}(ipt, F, fp, W_1) \rightarrow < ipt, fp >$$
$$K(ip) : \text{Trajectoire-sûre}(I, ipt, ip, W_1) \rightarrow ip,$$

Nous voulons maintenant optimiser ce générateur. La première étape est d'utiliser la condition nécessaire relative à la trajectoire de la pince. Une fois encore nous essayons de résoudre les problèmes jugés difficiles en premier. Nous remplacons $K(< ipt, fp >)$:Trajectoire-sûre$(ipt, F, fp, W_1) \rightarrow < ipt, fp >$ par:

$$K(< ipt, fp >) : \text{Trajectoire-sûre-pince}(ipt, F, fp, DM) \rightarrow < ipt, fp > \| \text{Trajectoire-sûre}(ipt, F, fp, W_1)$$

En utilisant Trajectoire-sûre$(i, f, p, W_1) \Rightarrow$Position-sûre$(i, W_1)$ nous obtenons:

$$\begin{array}{c}(K(< ipt, fp >) : \text{Trajectoire-sûre-pince}(ipt, F, fp, DM) \rightarrow < ipt, fp > \| \\ \text{Position-sûre}(i, W_1)) \\ \| \text{ Trajectoire-sûre}(ipt, F, fp, W_1)\end{array}$$

Dans notre implantation, nous avons supprimé le test final, en faisant la supposition qu'un petit mouvement sûr pour la pince l'était pour l'ensemble du robot.
Étant donné un point intermédiaire, on peut écrire la seconde fonction génératrice comme:

$$K(ip) : \text{Trajectoire-sûre}(I, ipt, ip, W_1) \rightarrow ip$$

qui dans le même esprit peut être optimisée par:

$$K(ip) : \text{Trajectoire-sûre}(I, ipt, ip, WM) \rightarrow ip \| \text{ Trajectoire-sûre}(I, ipt, ip, W_1)$$

Cette fonction génératrice ignore dans un premier temps les obstacles de la V-zone et dans un deuxième temps teste la trajectoire obtenue dans cette zone. En fait nous ne faisons pas, dans notre implantation, ce dernier test en estimant que le *ipt* obtenu précédement est sûr relativement aux obstacles de la V-zone.

La fonction génératrice obtenue pour le prédicat Trajectoire-sûre se justifie par le fait que nous disposons d'une fonction génératrice efficace pour le prédicat Trajectoire-sûre-pince. Cette fonction est utilisée pour trouver les points intermédiaires éloignés des obstacles. Ceci, à son tour, facilite la planification de trajectoire pour l'ensemble du robot.

3.7 Conclusion

Dans notre implantation, nous poursuivons le processus d'optimisation des fonctions génératrices de chaque noyau. En fait, les seules fonctions qui ne peuvent pas être obtenues par une "optimisation" récursive sont les fonctions calculant les contacts entre polyèdres et intervalles de débattement pour les degrés de liberté. La fonction génératrice que nous obtenons pour Prendre-et-Poser est bien évidemment loin d'être générale mais elle est justifiable par les résultats expérimentaux que nous avons obtenus.

Chapitre 4

LE SYSTÈME HANDEY

4.1 L'environnement expérimental

Un bon environnement expérimental est un facteur essentiel de réussite dans la réalisation d'un système tel que Handey. Pour cette raison beaucoup de groupes de recherche ont cherché à se doter d'un tel environement [LLJL84, Ala83].

4.1.1 L'architecture robotique

Le robot utilisé pour notre expérimentation est un PUMA 650 équipé d'une pince asservie en position. Le contrôleur de ce robot à été modifié [JO85] de façon à pouvoir être intégré à l'architecture présentée figure 4.1. L'avantage principal de cette architecture est de permettre un contrôle direct du robot depuis une Machine Lisp. Ceci évite en particulier le développement d'un nouveau langage pour le contrôle du robot et permet sa commande "en ligne". Il faut noter que la possibilité de commander le robot "en ligne" simplifie considérablement la planification de commande niveau tâche. En effet planifier "hors ligne" est beaucoup plus difficile car il faut prévoir à l'avance l'ensemble des états pouvant résulter d'une action, alors que la planification "en ligne" permet de ne considérer qu' un seul état à la fois.
Handey utilise un nombre restreint de primitives pour contrôler le robot, nommément:

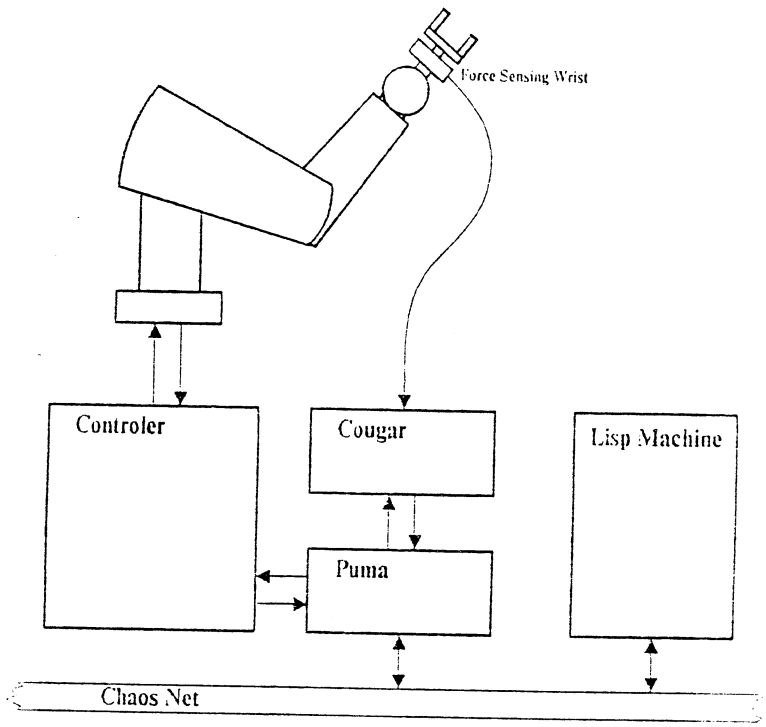


Figure 4.1: Architecture robotique utilisée pour expérimenter Handey (d'après JO85)

- Une primitive de calibrage du robot après mise sous tension,
- Une primitive de déplacement dans l'espace articulaire:

$$aller\ en\ (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$$

  Plusieurs points peuvent être envoyés successivement au contrôleur de trajectoire, qui se charge automatiquement de l'interpolation entre ces points.

- Une primitive contrôlant l'ouverture de la pince

Cet ensemble réduit de primitives fait de Handey un système facilement portable.

4.1.2 Le système laser

Le système laser est utilisé pour acquérir des images tri-dimensionnelles et former une carte 3D d'une portion de l'espace de travail. Cette carte est utilisée pour localiser des objets, et sert comme représentation des obstacles dans la V-zone. Une présentation des systèmes de vision tri-dimensionnelle peut être trouvée dans [Bor84]. L'architecture matérielle du système laser est présentée figure 4.2. La même Machine Lisp est utilisée pour contrôler ce système et le robot. Deux primitives de base sont utilisées par Handey.

1. Une primitive pour calibrer les miroirs tournants et calculer la distorsion dûe à la perspective.
2. une primitive pour créer une image tri-dimensionnelle.

Un nouveau processeur spécialisé a été ajouté pour rendre la prise d'image plus rapide [Zah87], il permet en particulier de localiser le trait laser dans l'image en une trame du signal vidéo [ITM87a].
Le succès de l'expérimentation dépend largement du bon calibrage entre le repère de référence du système laser et le repère de référence attaché à la base robot. Une procédure originale de calibrage a été developpée pour atteindre cet objectif (voir annexe B).

4.1.3 Le système précis de localisation

Ce système est utilisé pour obtenir une localisation précise d'un objet tenu dans la pince du robot [Gor86]. Il utilise le même procédé de triangularisation que le système laser mais est beaucoup plus précis car l'ensemble caméra-laser occupe une position relative fixe. Ce système est commandé directement depuis la même Machine Lisp.

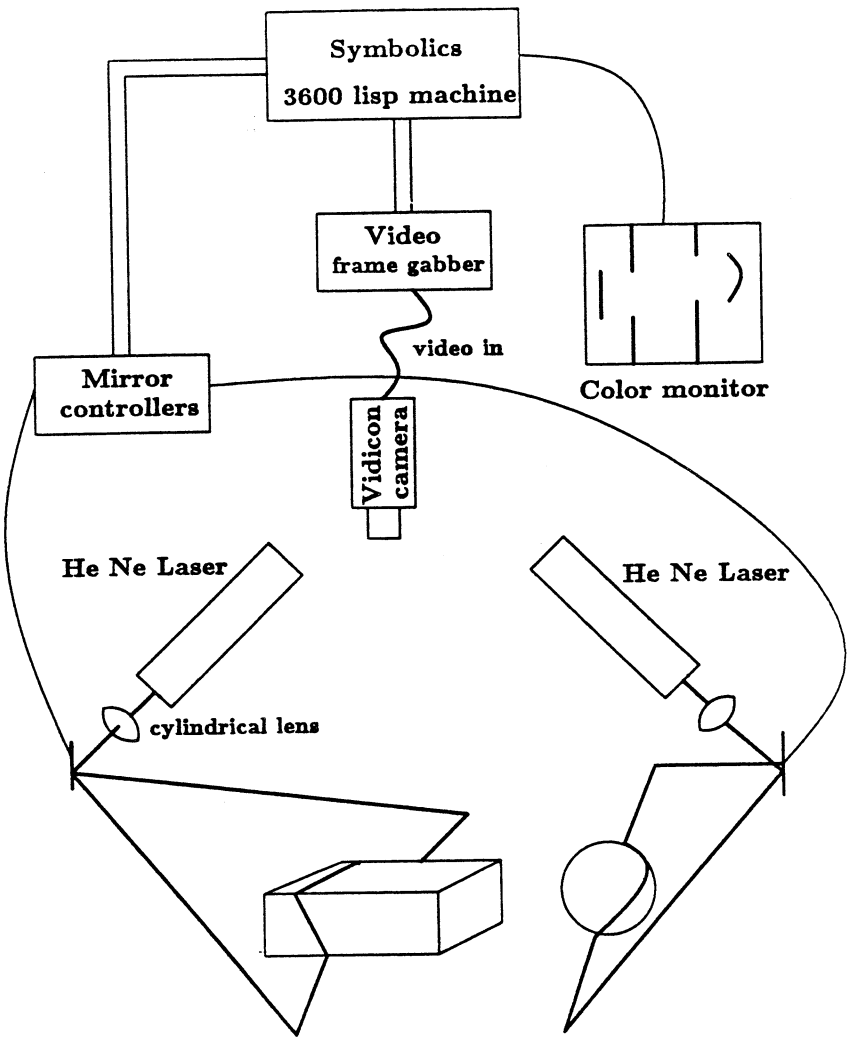


Figure 4.2: Architecture du système laser (d'après Zah87)

4.2 Le système de modélisation

Par système de modélisation, nous entendons l'ensemble des structures de données et l'ensemble des fonctions utilisées par Handey pour modéliser le robot et son environnement. Dans un système comme Handey la plupart des modules utilisent un système unique de modélisation, il s'en suit que les hypothèses et les choix faits pour construire ce système se propagent naturellement dans tous les autres modules. Le système de modélisation a deux fonctions: (a) modéliser les objets (y compris le robot), (b) tenir à jour le modèle de la scène. Les systèmes de modélisation exsistants sont en général assez rudimentaires, par exemple si un objet est relaché de la pince au dessus de la table, il restera dans cette position pour le système de modélisation, alors qu'il devrait naturellement "tomber" sur la table.
Une présentation de la représentation des solides peut être trouvée dans [Rec80]. SMGR [Per86] est l'un des seuls (GDP[WLL*80]) systèmes de modélisation spécialement développé pour la robotique. SMGR combine plusieurs fonctionnalités:

- Il permet de définir plusieurs représentations pour un même objet. Par exemple un même solide pourra être représenté par son enveloppe polyèdrique, par un ensemble de cônes généralisés ou par un ensemble de sphères englobantes. Cette représentation multiple a l'avantage de donner le choix de la représentation la plus souhaitable pour un problème donné. Par exemple la représentation par des sphères englobantes est intéressante pour la planification de trajectoire sans collision.

- Il permet de modéliser les incertitudes et de les propager durant la simulation du montage.

- Il donne la possibilité de conduire des raisonnements de physique naïve.

SMGR peut être considéré comme l'un des systèmes de modélisation les plus complets développés pour la robotique à ce jour. Il comprend un système de simulation hors ligne qui permet la représentation en images réalistes des mouvements du robot [GET87]. Dans la section suivante nous voudrions montrer que le développement d'un système sophistiqué de CAO n'est pas nécessaire pour réaliser un planificateur tel que Handey.

4.2.1 Le système de modélisation géométrique de Handey

Le système de modélisation de Handey est beaucoup plus simple que SMGR. La modélisation des objets utilise un sous ensemble d'un système implémenté par Alain Lanusse: YASM. YASM représente les objets par leurs enveloppes polyèdriques. Cette représentation "polyédrique" se retrouve dans tous les modules de Handey et constitue une de ses limitations majeures. Par exemple le module de reconnaissance et le module de changement de prise reposent largement sur cette représentation. Il faut noter que ceci est moins vrai pour le module de planification de trajectoire, qui peut continuer à s'appliquer sans perte de généralité.

Modélisation des objets et du robot

Pour représenter les objets, YASM utilise la structure de données suivante:

objet $\downarrow$
pile de position, modèle $\downarrow$
liste de faces f_i $\downarrow$
liste d'arêtes orientées $(\pm 1, e_j \downarrow)$
sommet origine, sommet extrémité $(v_r, v_l \downarrow)$
coordonnées originales des sommets

La pile de position est utilisée pour stocker des matrices 4×4 représentant la position de l'objet. Si (x, y, z) sont les coordonnées originales d'un sommet et M une matrice 4×4 alors $M * (x, y, z, 1)^T$ représente les coordonnées de ce sommet dans le repère de référence correspondant à M.

Cette pile est pratique pour représenter les coordonnées de l'objet dans différents systèmes de référence (monde, poignet, vision...). La structure de données réelle possède en fait d'autres emplacements, par exemple l'emplacement "objet" a aussi un pointeur sur la représentation de l'enveloppe convexe de l'objet. Toutes les informations stockées dans ces emplacements supplémentaires pourraient toutefois être calculées à partir de la représentation que nous avons donnée.

Dans Handey, les robots sont modélisés par des chaînes cinématiques ouvertes. A une chaîne cinématique, nous associons la structure de données suivante:

Chaîne Cinématique $\downarrow$
liste d'articulations j_i $\downarrow$
matrice "A" A_i, *butées* $(\theta_i^{max}, \theta_i^{min})$, *liste de solides.*

- Une matrice "A" est une façon standard de représenter la position d'une articulation par rapport à la précédente [DH55].
- Les butées sont utilisées pour calculer si une position est atteignable.
- La liste des solides est utilisée pour construire le modèle de l'articulation.

Étant donné un vecteur de coordonnées articulaires, il est possible de calculer les coordonnées d'un sommet $V(x, y, z)$ d'un solide appartenant à une articulation J_i avec:

$$A_0(\theta_1) * A_1(\theta_2) ... A_{i-1}(\theta_i) * (x, y, z, 1)^T$$

Les matrices "A" sont aussi utilisées pour effectuer le changement de coordonnées direct. Le changement de coordonnées inverse se fait en utilisant les formules données dans [Pau81b, Pau81a].

La plupart des modèles de solides utilisés dans Handey (notamment ceux utilisés pour le modèle du robot) ont été construit par balayage d'une face plane le long d'une direction perpendiculaire à cette face. Cependant, pour certains objets complexes la structure de données a été définie explicitement.

Globalement nous pouvons dire que le temps nécessaire pour construire une représentation du robot et de son environnement en utilisant ces simples outils a été considérablement court comparé au temps nécessaire à la construction d'un "vrai" système de CAO.

Modélisation des actions d'assemblage

Nous avons utilisé quatre fonctions pour modéliser la scène et les actions de montage.

1. Une fonction pour allouer une position au robot.
2. Une fonction pour allouer une position à un modèle dans la scène.
3. une fonction pour "lier" un objet à la dernière articulation du robot lorsque celui-ci réalise une opération de saisie.
4. une fonction pour "délier" un objet de la dernière articulation du robot lorsque celui-ci réalise une opération de lâcher.

A part ces fonctions de base, nous n'avons utilisé couramment que deux autres fonctions pour développer Handey:

- Une fonction d'affichage de la scène avec ou sans élimination des lignes cachées.
- Une fonction pour tester la collision entre deux polyèdres

Ce système de modélisation a été intensivement utilisé pendant le développement de Handey, et malgré sa simplicité, s'est révélé suffisant pour atteindre nos objectifs.

4.3 L'interface géométrique

Dans cette section nous introduisons un formalisme destiné à simplifier la présentation et l'implantation des modules de Handey. Ce formalisme permet de décrire et de résoudre la plupart des problèmes géométriques que l'on trouve dans Handey. Ces problèmes consistent toujours à trouver la réponse au même type de question:

Quelle est la position relative entre deux modèles données leurs relations géométriques mutuelles

Par exemple le module de reconnaissance construit ses hypothèses en faisant une mise en correspondance entre le modèle donné par le capteur tri-dimensionnel et le modèle géométrique de l'objet. La mise en correspondance consiste à trouver une transformation telle que:

- *l'arête A du modèle de l'image est alignée avec l'arête X du modèle géométrique.*
- *l'arête B du modèle de l'image est alignée avec l'arête Y du modèle géométrique.*

Dans le cas de la planification de trajectoire, on s'intéresse aux types de problèmes suivants:

- Existe-t-il une translation pure le long d'un axe donné amenant aux relations géométriques suivantes?:
  - *le point P de l'articulation J appartient au plan Y de l'obstacle B.*
  - *l'arête A de l'articulation J intersecte l'arête X de l'obstacle B.*
- Existe-t-il une rotation pure autour d'un axe donné amenant aux relations géométriques suivantes?:
  - *le point P de l'articulation J appartient au plan Y de l'obstacle B.*
  - *l'arête A de l'articulation J intersecte l'arête X de l'obstacle B.*

4.3.1 Equations associées aux relations géométriques

Dans Handey, nous n'utilisons que des modèles polyèdriques, et, pour cette raison nous ne nous intéresserons qu'aux entités géométriques suivantes: plan, droite et point. Les relations entre ces entités géométriques ne sont pas équivalentes aux relations obtenues avec les entités des modèles, toutefois elles peuvent servir de conditions nécessaires. Par exemple, on a:

l'arête A' intersecte l'arête B'

$\Longrightarrow$

la droite A intersecte la droite B

si les droites A et B supportent les arêtes A' et B'. Toutefois de simples calculs géométriques permettent de tester si une solution obtenue avec des droites et des plans est aussi valide pour des arêtes et des faces. Dans l'exemple précédent, le test serait de calculer les coordonnées du point d'intersection entre les deux droites A et B et de vérifier que ce point appartient aux arêtes A' et B'.

4.3.2 Notations

- $v.v'$
  représente le produit scalaire entre deux vecteurs v et v'.

- $v \times v'$
  représente le produit vectoriel entre deux vecteurs v et v'.

- $r \star v$
  représente le vecteur déduit de v par une rotation r appliquée a v.

- σv
  représente le produit d'un vecteur v par un scalaire σ.

Représentation des entités géométriques

Nous représentons les plans, droites, et points à l'aide de leurs coordonnées de Plücker [Bra53].

- Un point est représenté par un vecteur (a_0).

- Une droite est représentée par un couple de vecteurs (a_0, a) où a est un vecteur unitaire. Un point m appartient à (a_0, a) si et seulement si: $m \times a = a_0$.

- Un plan est représenté par une doublet (σ, a) où σ est un scalaire et a un vecteur unitaire. Un point m appartient à (σ, a) si et seulement si: $m \cdot a = \sigma$.

Changement de repère de référence

Translation Soient F_A et $F_{A'}$ deux repères de références tels que $F_{A'}$ soit déduit de F_A par une translation T. Si t représente le vecteur translation dans F_A alors nous pouvons écrire les relations suivantes:

- point: $[(a_0)]F_A \Longleftrightarrow [(a_0 - t)]F_{A'}$

- droite: $[(a_0, a)]F_A \Longleftrightarrow [(a_0 - t \times a, a)]F_{A'}$

- plan: $[(\sigma, a)]F_A \Longleftrightarrow [(\sigma - t \cdot a, a)]F_{A'}$

Rotation: Si F_B est déduit de $F_{A'}$ par une rotation R et si r est l'opérateur de rotation qui transforme un vecteur $[v]F_B$ en $[r \star v]F_{A'}$, alors en supposant que r^{-1} est l'opérateur de rotation inverse de r, on peut écrire:

- point: $[(a_0)]F_{A'} \Longleftrightarrow [(r^{-1} \star a_0)]F_B$
- droite: $[(a_0, a)]F_{A'} \Longleftrightarrow [(r^{-1} \star a_0, r^{-1} \star a)]F_B$
- plan: $[(\sigma, a)]F_{A'} \Longleftrightarrow [(\sigma, r^{-1} \star a)]F_B$

Relations géométriques de base et équations associées

Soient F_A et F_B les repères de références de deux modèles géométriques A et B. La position relative de B par rapport à A est connue s'il est possible de calculer une translation T, une rotation R telles que:

$$F_A \xrightarrow{T} F_{A'} \xrightarrow{R} F_B$$

Pour chaque relation géométrique, il est possible d'écrire une ou deux équations vectorielles formant un système équivalent à la relation géométrique. Dans la suite nous présentons les équations utilisées dans différents modules de Handey. Ces équations sont obtenues en écrivant les entités concernées dans le repère $F_{A'}$ et en appliquant les conditions nécessaires et suffisantes d'appartenance à une entité donnée.

- *le point* $[(a_0)]F_A$ *coïncide avec le point* $[(b_0)]F_B$

  La représentation du point a_0 dans $F_{A'}$ est $[(a_0 - t)]F_{A'}$, la représentation du point (b_0) dans $F_{A'}$ est $[r \star b_0]F_{A'}$. Si l'on écrit que les deux points sont confondus, on obtient l'équation suivante:

$$a0 - t = r \star b$$

- *la droite* $[(a_0, a)]F_A$ *est parallèle à la droite* $[(b_0, b)]F_b$

  Cette relation peut être utilisée pour spécifier que deux arêtes d'un assemblage sont parallèles. Pour représenter cette relation, il suffit d'écrire que les deux vecteurs unitaires a, b sont parallèles dans $F_{A'}$. On obtient ainsi l'équation suivante:

$$a \times (r \star b) = 0$$

- *la droite* $[(a_0, a)]F_A$ *est alignée avec la droite* $[(b_0, b)]F_B$

  Cette relation peut être utilisée pour décrire un assemblage mettant en jeu l'insertion d'un goujon dans un trou. On dira alors que l'axe du goujon est

aligné avec l'axe du trou. L'équation précèdente doit être vérifiée et un point de $[(b_0, b)]F_B$ doit appartenir à la droite $[(a_0, a)]F_A$. Le point $b \times b_0$ appartient à la droite $[(b_0, b)]F_B$ car $(b \times b_0) \times b = b_0$ (en effet $b \cdot b_0 = 0$ et $||b|| = 1$), sa représentation dans $F_{A'}$ est $[r \star (b \times b_0)]F_{A'}$ et il appartient à la droite $[(a_0 - t \times a, a)]F_A$ si et seulement si $a_0 - t \times a = (r \star (b \times b_0)) \times a$. Nous obtenons donc les équations suivantes:

$$\begin{aligned} a \times (r \star b) &= 0 \\ a_0 - t \times a &= (r \star (b \times b_0)) \times a \end{aligned}$$

- *le plan* $[(\sigma_A, a)]F_A$ *est parallèle au plan* $[(\sigma_B, b)]F_B$

Cette relation est utile pour exprimer que deux surfaces sont parallèles sans donner de précision sur leur distance relative.

$$a \times (r \star b) = 0$$

Si la normale à la surface indique la position de la matière par rapport à la surface on peut utiliser l'équation précédente pour décrire si les deux faces sont parallèles ou se font face.

$$\begin{aligned} a &= (r \star b) \\ a\times &= -(r \star b) \end{aligned}$$

- *le plan* $[(\sigma_A, a)]F_A$ *est contre le plan* $[(\sigma_B, b)]F_B$

Cette relation est utile pour décrire que deux faces d'objets sont en contact. L'équation précédente doit être vérifiée et le point $[\sigma_b b]F_B$ du plan $[(\sigma_B, b)]F_B$ doit appartenir au plan $[(\sigma_A, a)]F_A$. Nous obtenons donc deux équations:

$$\begin{aligned} a &= -(r \star b) \\ \sigma_A - t \cdot a &= (r \star (\sigma_B b)) \cdot a \end{aligned}$$

- *le plan* (σ, a) *contient la droite* $[(b_0, b)]F_B$

La droite $[(r \star b_0, r \star b)]F_{A'}$ doit être parallèle au plan $[(\sigma - t \cdot a, a)]F_{A'}$ et le point $[(r \star (b \times b_0))]F_{A'}$ doit appartenir au plan $[(\sigma - t \cdot a, a)]F_{A'}$.

$$\begin{aligned} a \cdot (r \star b) &= 0 \\ \sigma - t \cdot a &= (r \star (b \times b_0)) \cdot a \end{aligned}$$

- *le plan* $[(\sigma, a)]F_A$ *contient le point* $[b_0]F_B$

$$\sigma - t \cdot a = (r \star b_0) \cdot a$$

- *la droite* $[(a_0, a)]F_A$ *intersecte la droite* $[(b_0, b)]F_B$

Deux droites (c_0, c) et (d_0, d) s'intersectent si et seulement si (voir appendice A.2):

$$c \times d \neq 0 \; and \; (c_0 \cdot d) + (d_0 \cdot c) = 0$$

Deux équations dérivent de cette relation, seule la première est utilisée pour obtenir un résultat. La seconde ne sert qu'à vérifier le résultat obtenu.

$$(r \star b_0) \cdot a + (a_0 - t \times a) \cdot (r \star b) = 0$$
$$a \times (r \star b) \neq 0$$

Résolution du système d'équations

Si plusieurs relations géométriques définissent la position des deux objets alors un système de plusieurs équations en r et t est à résoudre. Le système est équivalent à un système d'équations algébriques. Le degré des polynômes varie selon la représentation choisie pour l'opérateur de rotation r. Si l'on utilise des quaternions alors ces polynômes sont quadratiques pour les termes provenant de la rotation et linéaires pour ceux provenant de la translation. Il est théoriquement possible de résoudre ce type de système avec une précision arbitaire. Des résultats récents [Can87] montrent que la complexité de l'algorithme est $O(r^3 d^{5r} log^2 dw)$ où r est le nombre de variables, d le degré et w le nombre de bits maximun utilisé pour représenter les coéfficients des polynômes.
Toutefois la mise en œuvre de Handey ne nécessite pas l'utilisation d'outils généraux et la méthode présentée en appendice (A.1) permet de résoudre tous les systèmes d'équations que l'on peut rencontrer.

4.4 Le système de vision

Le système de vision de Handey est utilisé pour reconnaître et localiser l'objet à saisir. Il utilise les données tri-dimensionnelles du système laser et se base sur le modèle géométrique d'un objet pour le localiser même si celui-ci n'est que partiellement visible.

Une présentation des différentes technologies utilisées pour saisir une image tri-dimensionnelle est faite dans [BJ85]. Une étude sur la vision par ordinateur et la représentation géométrique des objets peut être trouvée dans [BJ85]. Le système utilisé dans Handey est basé sur de précédents travaux décrits dans [GL85]. Comme beacoup d'autres systèmes [Lux85, Sou83, FH86, BH86], l'algorithme utilisé procède en deux phases: génération et vérification d'hypothèses. Durant la phase de génération, le système met en correspondance des entités géométriques du modèle avec des entités géométriques de l'image. Une correspondance donne jour à une hypothèse valide si elle permet de calculer la position de l'objet dans la scène. Durant la phase de vérification d'hypothèse, l'algorithme utilise la position supposée de l'objet pour prévoir la position d'autres entités géométriques dans la scène. La position qui permet de prévoir le plus de positions exactes pour les autres entités géométriques est retenue.

4.4.1 Calcul des indices visuels

Les objets manipulés par Handey étant des polyèdres, nous nous intéressons qu'aux entités géométriques suivantes: plan, droite, et point. Dans une première implémentation du système les plans étaient les seules entités géométriques utilisées. Ceci présentait l'inconvénient de nécessiter la présence de trois faces de l'objet dans l'image pour pouvoir engendrer une hypothèse valide. Il est très facile d'obtenir des scènes où cette condition n'est pas vérifiée. Par exemple un cube situé immédiatement en dessous de la caméra ne présente qu'une seule face au système de vision. Dans la version actuelle, le système se base sur les arêtes du modèle et de l'image pour engendrer les hypothèses. Cette approche ne nécessite que la présence de deux arêtes non parallèles de l'objet dans l'image. Cette condition est beaucoup plus facile à obtenir et une scène choisie aléatoirement a généralement cette propriété. Pour obtenir la liste des segments de droite le système procède en quatre étapes:

Lissage. La carte tri-dimensionnelle est tout d'abord lissée en utilisant un filtre binômial sur la hauteur de chaque point. Le lissage se fait par ligne. Si $Z(i,j)$ représente la hauteur d'un point i,j de la carte tri-dimensionnelle, alors la formule suivante est employée pour calculer la valeur lissée $SZ(i,j)$ en ce point.

$$SZ(i,j) = (Z(i,j-2) + 2*Z(i,j-1) + 3*Z(i,j) + 2*Z(i,j+1) + Z(i,j+2))/16$$

Recherche des points de rupture de pente Dans une première version du système, l'opérateur de Canny [Can83] était utilisé en remplaçant simplement les

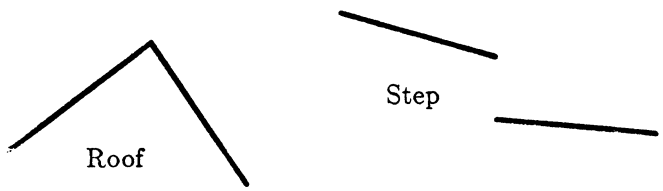


Figure 4.3: "Arêtes en forme de toît ou de marche"

valeurs d'intensité lumineuse par la hauteur des points. Les résutats donnés par cette méthode étaient relativement satisfaisants. Cependant cette méthode présente deux inconvénients majeurs:

- elle ne permet pas de retrouver dans la carte tri-dimensionnelle les arêtes en forme de "toît" (figure 4.3),
- elle a tendance à rajouter des arêtes dans les régions ayant une pente constante.

Ces problèmes sont liés à l'utilisation du passage par zéro de la dérivée seconde de la hauteur pour repérer les cassures de pente (ou lignes de contraste pour des images d'intensité lumineuse).

Le méthode utilisée actuellement est en fait plus simple, elle consiste à appliquer sucessivement deux opérateurs pour trouver les lignes de contraste. Le premier utilise le passage par zéro de la dérivée première de la hauteur pour trouver les cassures de pente en forme de "toît", le deuxième utilise le passage par zéro de la dérivée seconde pour trouver les ruptures de pente en forme de marche. Chaque opérateur fait ses calculs successivement en ligne et en colonne. L'ensemble des points de contraste est obtenu en faisant l'union de tous les points obtenus avec les deux méthodes.

Suivi de ligne de contraste. Une ligne de contraste dans une image tri-dimensionnelle représente en fait une ligne de rupture de pente. Deux étapes sont utilisées pour effectuer un suivi de ligne:

1. trouver un premier point de contraste qui n'appartienne pas déjà à une autre ligne de contraste,
2. construire une chaîne de points de contraste à partir de ce point.

Cette dernière opération peut se faire à l'aide d'une simple fonction récursive:

```
(defun chainage (ip)
        (cond ((null ip) nil)
              (t
                (marquer ip)
                (cons ip
                      (chainage (prochain-point-de-contraste ip) ip)))))
```

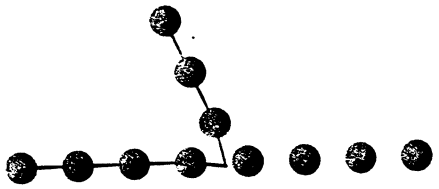


Figure 4.4: Une ligne de contraste peut être arbitrairement coupée par l'algorithme de suivi de ligne

Où *ip* est un point de contraste, *prochain-point-de-contraste* une fonction retournant un point de constraste (non marqué) adjacent à *ip*. Cet algorithme est très facile à implémenter mais présente toutefois l'inconvénient d'introduire des cassures artificielles dans les lignes de contraste (voir figure 4.4).

Segmentation des lignes de contraste. Les arêtes provenant de l'image sont construites en segmentant chaque ligne de contraste en plusieurs segments de droite. Pour obtenir ces segments de droite, on débute en prenant l'ensemble des points de la ligne de contraste. On calcule la droite des moindres carrés correspondant à cet ensemble de points. Si la distance maximum des points de la ligne de contraste à cette droite est inférieure à une valeur donnée, alors la ligne de contraste est prise comme un segment de doite, sinon la ligne de contraste est divisée en deux lignes de contraste de même longueur et le même procédé est appliqué récursivement aux deux lignes de contraste obtenues. Une fois l'ensemble des segments de droites calculé on effectue un groupement en considérant les segments de droites adjacents et parallèles.

4.4.2 Formation des hypothèses

Une hypothèse peut se représenter sous la forme d'une paire de paires d'arêtes.

$$((e_i^m, e_p^d), (e_j^m, e_q^d))$$

Où e_i^m, e_j^m sont les arêtes du modèle et e_p^d, e_q^d les arêtes construites à partir de la carte tri-dimensionnelle.

Une hypothèse est valide s'il est possible de trouver une transformation géométrique qui amène le modèle de l'objet dans une configuration où les relations géométriques suivantes soient vérifiées:

- e_i^m est alignée avec e_p^d
- e_j^m est alignée avec e_q^d

Etant donnée une hypothèse, il est possible de vérifier sa validité uniquement en faisant appel à l'interface géométrique avec les relations géométriques précédentes.

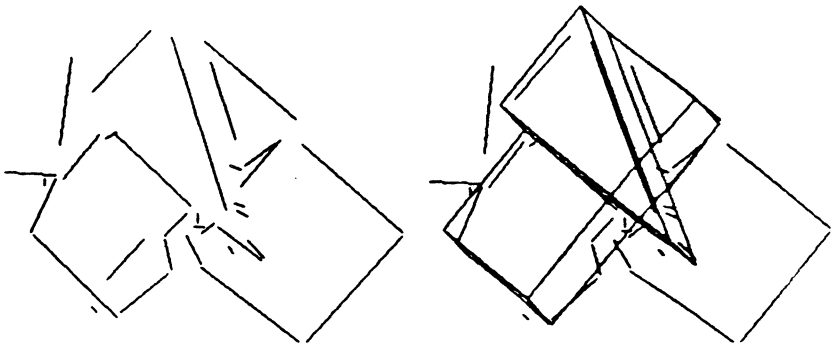


Figure 4.5: Une hypothèse valide est mise en correspondance avec les arêtes de l'image

On peut remarquer que l'algorithme utilisé dans l'interface géométrique engendre automatiquement deux ensembles d'équations pour ce type de relations géométriques et, par conséquent, l'orientation des arêtes n'est pas à prendre en compte. De plus il est possible que l'interface géométrique retourne deux solutions (une pour chaque ensemble) pour cet ensemble de relations géométriques. Dans ce cas, les deux transformations sont considérées comme hypothèses valides.

La génération d'hypothèses peut se faire simplement en engendrant toutes les combinaisons possibles de paires de paires d'arêtes. Toutefois on peut éviter un appel systématique à l'interface géométrique simplement en considérant la longueur des arêtes provenant de l'image et du modèle. En effet, une condition nécessaire pour qu'une paire d'arêtes puisse engendrer une hypothèse est que:

$$Longueur(e_i^m) > Longueur(e_p^d)$$

Cette comparaison peut se faire sans perte de généralité. Dans notre implémentation nous allons toutefois un peu plus loin en ne considérant que les paires d'arêtes vérifiant la condition:

$$Longueur(e_p^d) > k \times Longueur(e_i^m) \wedge Longueur(e_i^m) > Longueur(e_p^d)$$

avec $k < 1$. Cette condition permet de diminuer considérablement la combinatoire engendrée par l'énumération des paires de paires, par contre elle rend l'algorithme moins général (à mesure que k tend vers 1) car une hypothèse valide peut très bien se retrouver rejetée par ce test. La figure 4.5 représente une hypothèse valide (la bonne !) engendrée par le système.

Vérification des hypothèses

Etant donnée une hypothèse valide, il est possible d'amener le modèle de la pièce à l'endroit qu'il occupe dans la scène. Pour chaque arête du modèle on construit un parallélépipède entourant cette arête; une arête de l'image tri-dimensionnelle entièrement contenue dans ce parallélépipède tend à accréditer l'hypothèse considérée et un compteur est incrementé. Ce compteur représente une mesure de la

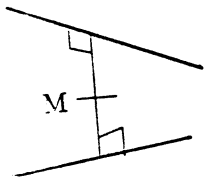


Figure 4.6: Pseudo intersection entre deux droites

validité de l'hypothèse courante. L'hypothèse ayant la validité la plus grande est retenue et la transformation correspondante est retenue comme étant la position de l'objet.

4.4.3 Ajustement de la position

Nous avons constaté que la réussite d'une expérimentation reposait largement sur une localisation précise de l'objet à saisir. Pour améliorer la précision, nous considérons deux ensembles de points I et M. I est calculé à partir de l'ensemble des arêtes de l'image tri-dimensionnelle ayant été retenu pour créer et vérifier une hypothèse. Pour chaque paire d'arêtes non parallèles de cet ensemble on calcule la pseudo-intersection entre ces arêtes (voir figure 4.6 et annexe A.3.2). Un ensemble de points M équivalent à I est créé avec le modèle en considérant l'ensemble des arêtes correspondantes. Une formule analytique permet de déduire directement la transformation qui permet de minimiser la somme des distances entre chaque paire de points [Hor87], toutefois dans notre implémentation nous utilisons une méthode basée sur l'utilisation du filtre de Kalman.

4.4.4 Optimisation de l'algorithme

La méthode proposée est éminament combinatoire, et il est important d'essayer de limiter le plus possible le nombre d'hypothèses et le nombre de vérifications. L'algorithme suivant n'est pas meilleur qu'un algorithme simpliste le plus mauvais cas mais se comporte beaucoup mieux la plupart du temps.

validite-max=0
boulce pour i de 1 *a* $n-1$ *; n nombre d'aretes dans la scene*
  boucle pour j de $i+1$ *a n*
    si $n-j <$ *validite-max alors retourner la meilleure hypothese ;voir (2)*
    boucle pour p de 1 a m ;nombre d'aretes dans le modele
      boucle pour q de 1 a m
        si bonne-hypothese$((e_i^m, e_p^d),(e_j^m, e_q^d))$
        ;;; calcul de la validite
        debut
        validite=0
        boucle pour j_1 *de* $j+1$ *a n ; voir(1)*
          boucle pour k de 1 *a m*
            si valide$(e_k^m, e_{j_1} p^d)$

validite=validit e+1
validite-max=max(validit e-max,validite)

Deux remarques permettent de comprendre cet algorithme:

1. Il n'est pas nécessaire de vérifier la validité des arêtes de l'image dont l'indice est inférieur à $j+1$. En effet, supposons qu'une arête e_k^d avec $k < j+1$ soit valide pour l'hypothèse $((e_i^m, e_p^d), (e_j^m, e_q^d))$, l'hypothèse suivante $((e_i^m, e_p^d), e_k^m, e_q^d))$ est équivalente et sa validité a déjà été calculée

2. En tenant compte de la première remarque la valeur maximum de la validité est $n - j + 1$. Donc il n'est pas nécessaire de poursuivre plus loin car on a déjà atteint le maximum.

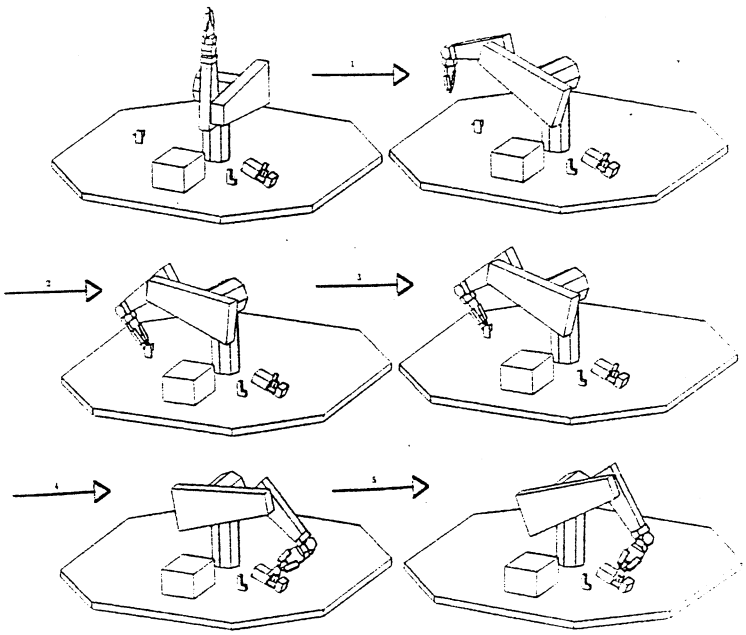


Figure 4.7: Positions initiale et finale pour les différents planificateurs de trajectoire d'une opération Prendre-et-Poser: (1) grand mouvement, (2) approche et saisie, (3) éloignement, (4) grand mouvement et (5) mouvements fins

4.5 Planification des grands mouvements

Dans une opération Prendre-et-Poser sans re-saisie, deux mouvements sont clairement identifiables: le mouvement nécessaire pour aller prendre l'objet et le mouvement nécessaire pour faire l'assemblage. Pour un observateur extérieur il serait naturel de penser que deux appels ont étés faits au même planificateur de trajectoire pour conduire cette opération Prendre-et-Poser. Dans un système tel que Handey nous utilisons en fait quatre planificateurs différents pour venir à bien de cette opération. La figure 4.7 représente l'ordre d'appel de ces planificateurs.

Planificateur de grands mouvements. Ce planificateur est conçu pour calculer des trajectoires dans un environnement peu encombré d'obstacles, en particulier il ne peut pas être utilisé lorsque les pièces de l'assemblage rentrent en contact.

Planificateur de saisie. Ce planificateur a pour rôle de planifier le déplacement de la pince parmi les obstacles de la V-zone de façon à atteindre une prise stable pour l'objet.

Planificateur d'éloignement Ce planificateur serait utilisé pour dégager la pince et l'objet transporté de la zone de préhension.

Planificateur de mouvements fins. Ce planificateur serait nécessaire pour planifier les mouvements compliants et/ou gardés visant à assembler la pièce transportée sur le sous assemblage.

La version actuelle de Handey ne comprend pas de planificateur d'eloignememt ni de planificateur de mouvements fins. Actuellement le dégagement de la pièce se fait simplement "par le haut" en commandant un mouvement vertical de la pince après la saisie. Il faut noter que ce mouvement donne lieu à quelques erreurs soit parce qu'il cause une collision de l'objet transporté soit parce qu'il cause une reconfiguration du robot pendant le mouvement de dégagement. Le planificateur de mouvement fins fait partie des extensions prochaines de Handey (voir Section 6.2).

Dans cette section nous décrivons l'algorithme utilisé pour la planification des grands mouvements. L'algorithme utilisé pour le planificateur de saisie est présenté dans la section 4.6. Le planificateur de grands mouvements est un des modules clés de Handey. Dans un montage avec n poses intermédiaires (cas de la re-saisie) $2n+1$ appels à ce planificateur sont nécessaires et un algorithme inefficace pourrait rendre le système trop lent pour être expérimenté.

Travaux similaires

Une présentation des algorithmes utilisés pour le calcul de trajectoires sans collision peut être trouvée dans [Lau87, Lat87], et, des résultats théoriques sur la complexité de ce problèmes dans [Can87, SHS87].

Les méthodes utilisées en planification de trajectoire peuvent être classées en deux catégories: les méthodes locales et les méthodes globales. Les méthodes globales représentent explicitement toutes les contraintes de l'environnement. On cherche la trajectoire d'un point représentant la position du robot dans le nouvel espace définissant le problème. Les méthodes locales, par contre, ont tendance à éviter ce type de représentation et les décisions se font sur la base d'une vue limitée de l'environnement. Les méthodes de génération et test [Fav86], du potentiel [Kat86], et de limitation de la vitesse relative aux obstacles [FT87] peuvent être considérées comme des méthodes locales. Les méthodes du "free-way" [Bro83], la représentation globale de l'espace des configurations [Don84] et la représentation de l'espace des configurations par coupes [Loz86, Ger85] peuvent être considérées comme des méthodes globales.

Toutefois il est assez difficile d'obtenir une classification très nette entre toute ces méthodes. Par exemple la méthode décrite dans [Don84] peut être considérée comme une sorte de méthode des potentiels dans l'espace des configurations: le point représentant la position du robot dans cet espace reste à distance fixe des C-obstacles. Réciproquement, construire une représentation en coupe de l'espace des configurations revient à tester explicitement les collisions à chaque point de l'espace sur un sous-ensemble de l'espace complet.

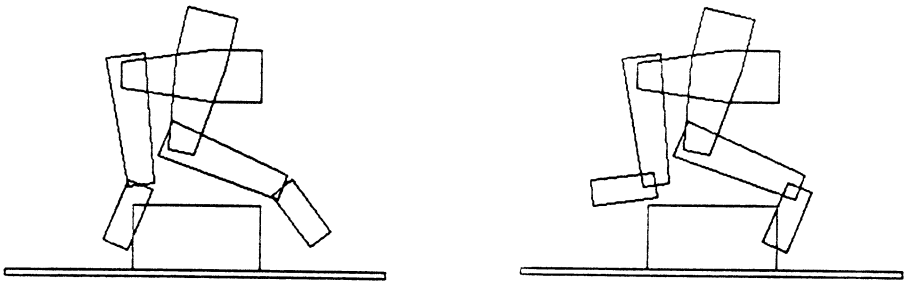


Figure 4.8: "Position initiale et finale d'un robot planaire"

4.5.1 Présentation générale

Le planificateur de trajectoire utilisé dans Handey est basé sur la méthode décrite dans [Loz86]. Les coordonneés articulaires sont utilisées pour définir l'espace des configurations. Pour trouver une trajectoire les système procède en deux phases:

1. Dans la première phase, on construit une représentation des obstacles dans l'espace des configurations en utilisant des coupes de dimensions infèrieures.

2. Dans la seconde phase, on cherche une trajectoire sans collision dans la nouvelle représentation du problème.

Construire une représentation explicite d'un espace à 6 dimensions n'est pas possible pour des raisons pratiques de temps calcul et d'espace mémoire. L'approche prise dans Handey ne requiert que la construction d'un espace des configurations à 3 dimensions bien qu'elle permette de construire des trajectoires à 6 dimensions. La méthode n'est pas générale et, il est possible de construire des contre-exemples où, bien qu'il soit possible de touver une trajectoire sans collision, la méthode proposée n'en trouve aucune. Toutefois notre expérimentation a montré que cette méthode est satisfaisante dans un espace relativement peu encombré .

4.5.2 Réduction de la dimensionnalité de l'espace des configurations

La méthode utilisée pour réduire la dimensionnalité de l'espace des configurations peut être illustrée avec un robot planaire à trois degrés de liberté. Soient $(\theta_1^i, \theta_2^i, \theta_3^i)$ et $(\theta_1^f, \theta_2^f, \theta_3^f)$ les positions initiale et finale du robot. La figure 4.8 représente:

1. Le robot à la position $(\theta_1^i, \theta_2^i, \theta_3^i)$ et à la position $(\theta_1^f, \theta_2^f, \theta_3^i)$.

2. Le robot à la position $(\theta_1^f, \theta_2^f, \theta_3^f)$ et à la position $(\theta_1^i, \theta_2^i, \theta_3^f)$.

étape 1 La dernière articulation du robot est remplacée par une boîte suffisament large pour contenir un mouvement de l'articulation 3, de la position initiale θ_3^i à la position finale θ_3^f. La figure 4.9 représente une telle boîte.

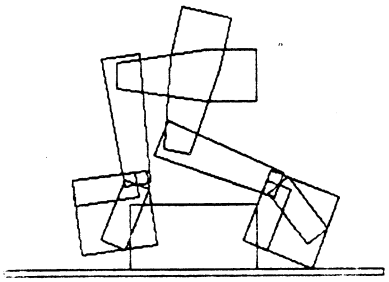


Figure 4.9: Boite englobant le mouvement du dernier degré de liberté

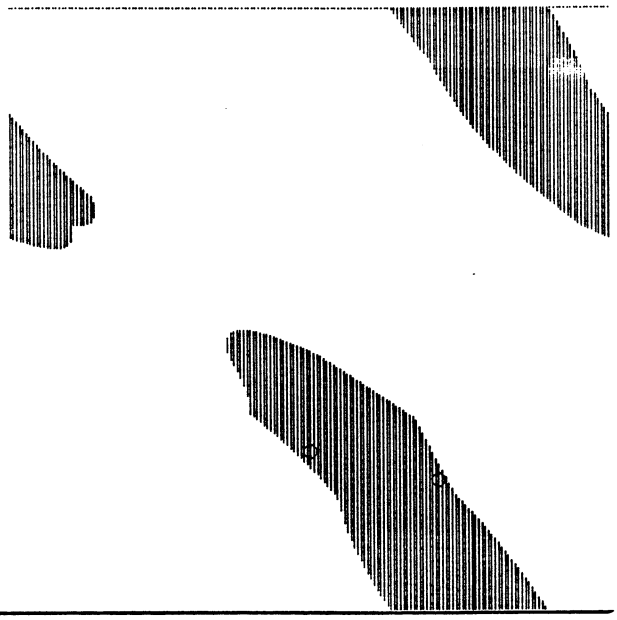


Figure 4.10: "Un espace des configurations bi-dimensionel"

étape 2 Une représentation des obstacles dans l'espace des configurations θ_1, θ_2 est construite en considérant que la boîte est solidaire de la deuxième articulation (figure 4.10). Notons que dans notre exemple la position finale et la position initiale en θ_1, θ_2 sont interdites.

étape 3 Les points dans l'espace libre les plus près des positions initiale et finale sont calculés, on les note (θ_1^1, θ_2^1),(θ_1^2, θ_2^2) respectivement. L'algorithme cherche une trajectoire sans collision entre (θ_1^1, θ_2^1) et (θ_1^2, θ_2^2) (voir figure 4.11). La boîte ayant été conçue pour inclure un mouvement de l'articulation 3 de θ_3^i à θ_3^f, il est possible d'effectuer ce mouvement en tout point de la trajectoire précédente sans provoquer une collision. Il est même possible d'effectuer continuellement ce mouvement le long de la trajectoire pour obtenir un mouvement global plus lissé.

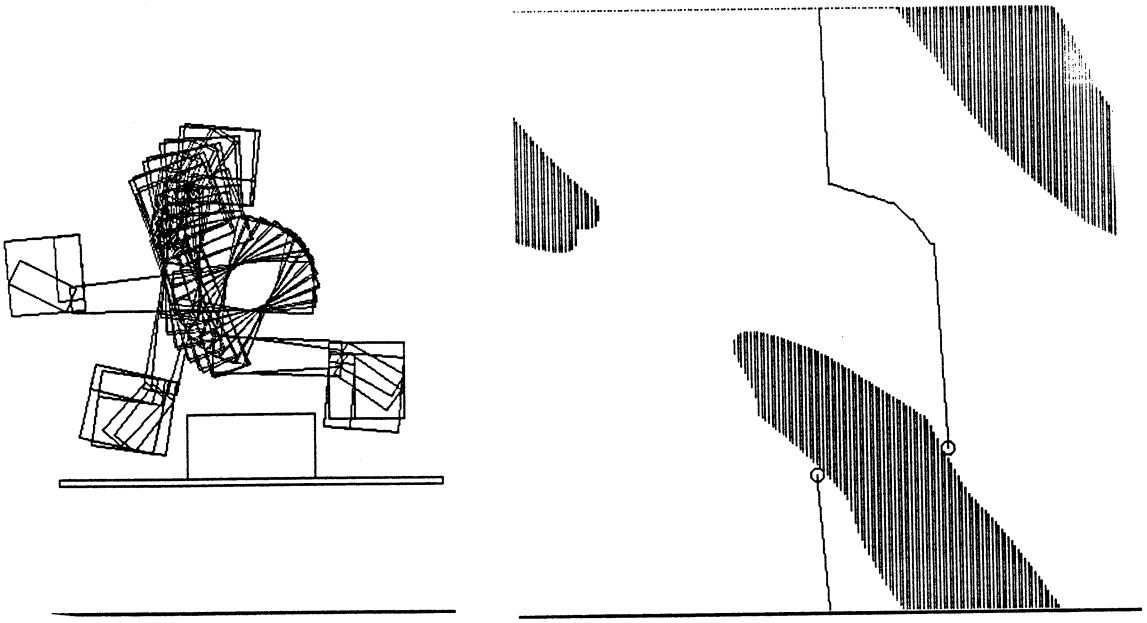


Figure 4.11: Calcul d'une trajectoire sans collision avec boîte englobante

étape 4 Le reste de l'algorithme consiste à trouver une trajectoire entre (θ_1^i, θ_2^i) et (θ_1^1, θ_2^1), et une trajectoire entre (θ_1^2, θ_2^2) et (θ_1^f, θ_2^f). La même méthode est appliquée en supprimant la boîte et en donnant à θ_3 une valeur fixe (respectivement θ_3^i et θ_3^f). La figure 4.12 représente l'espace des configurations dans ces deux cas. Notons qu'il n'est, en général, pas nécessaire de calculer complètement l'espace des configurations mais qu'il est simplement nécessaire de le calculer dans de petits intervalles autour de θ_1^f et θ_1^i.

étape 5 Une fois les deux trajectoires précédentes calculées, il ne reste plus qu'à juxtaposer les trois segments de trajectoire pour obtenir la trajectoire complète. Cette trajectoire est une trajectoire dans un espace à trois dimensions alors qu'elle n'a nécessité que la construction d' espaces des configurations de dimension 2.

Dans le cas d'un robot à six degrés de liberté la méthode reste essentiellement la même: Le système ne construit l'espace des configurations que pour les trois premiers degrés de liberté, les trois derniers restant fixes durant la première et la troisième partie de la trajectoire. La partie intermédiaire de la trajectoire est calculée en utilisant une boîte englobant les trois derniers degrés de liberté et l'objet transporté dans toutes les positions intermédiaires entre la position initiale et finale. Deux raisons peuvent faire échouer cette approche, là où un calcul complet de l'espace des configurations à six dimensions permettrait de trouver un solution.

1. L'ajout de l'enveloppe englobant le déplacement des trois derniers degrés de liberté détruit la connexité de l'espace des configurations (figure 4.13). Les

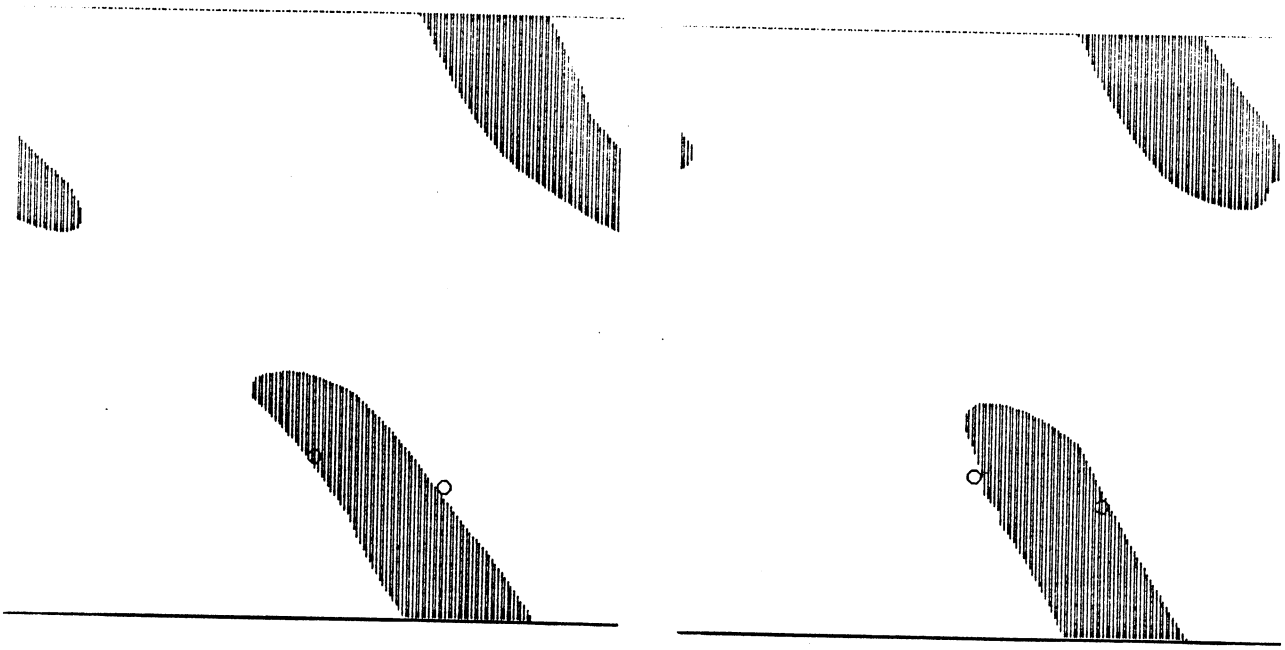


Figure 4.12: Espaces des configurations ... inal

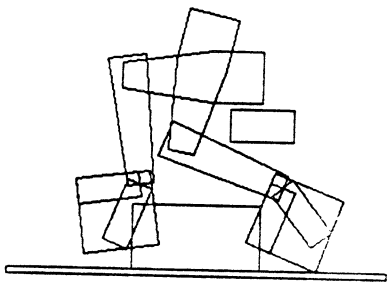


Figure 4.13: Destruction de la connexité de l'espace des configurations par ajout d'une boite autour du dernier degré de liberté

deux positions proches de la position initiale et finale sont dans deux composantes connexes de l'espace des configurations et il n'est pas possible de trouver un chemin de l'une à l'autre.

2. Le mouvement a lieu dans un environnement très contraint, et, le déplacement simultané de tous les degrés de liberté est nécessaire pour éviter les obstacles (figure 4.14).

Dans Handey nous supposons qu'il n'est pas possible de déconnecter l'espace des configurations en ajoutant une enveloppe autour des trois derniers degrés de liberté. Autrement dit, nous supposons qu'il est toujours possible d'effectuer une ré-orientation des trois derniers degrés de liberté en un point de l'espace et que ce point est atteignable depuis la configuration initiale et la configuration finale du robot. Ceci est en général vrai pour un PUMA travaillant au-dessus d'une table. En effet, la position représentée figure 4.15 est en général libre et permet une

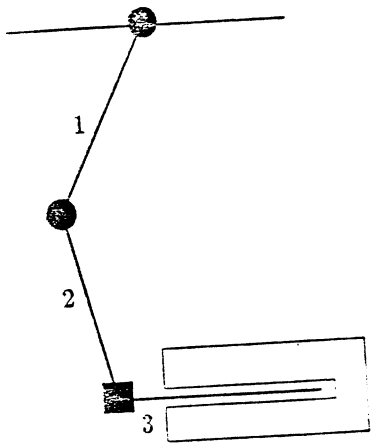


Figure 4.14: Les trois articulations doivent se mouvoir en même temps pour dégager l'articulation 3

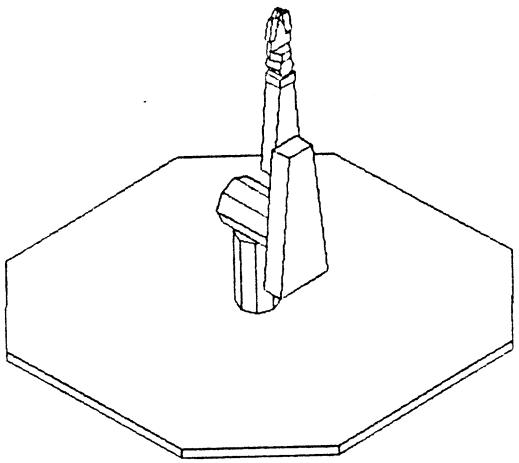


Figure 4.15: La plupart du temps, cette position verticale de robot peut être atteinte de tous les points de l'espace et permet n'importe quelle ré-orientation du poignet

ré-orientation du poignet.
Par construction, le planificateur de mouvement n'est jamais appelé à planifier une trajectoire dans un voisinage trop contraint, et donc la deuxième cause d'échec est peu probable.

4.5.3 Calcul de l'espace des configurations

Il est nécessaire de calculer une représentation de l'espace des configurations pour pouvoir planifier une trajectoire. Les premières versions de Handey utilisaient l'algorithme proposé dans [Loz86]. Cependant il est possible d'améliorer considérablement ce type d'algorithme en tenant compte de la géométrie particulière d'un robot donné. Par exemple pour le PUMA, il a été possible de réduire le temps nécessaire au calcul de l'espace des configurations de plusieurs minutes à quelques dizaines de secondes, et en conséquence, le planificateur de trajectoire est souvent

l'un des modules les plus rapides du système.

Technique de Base

Le planificateur n'utilisant que trois degrés de liberté pour planifier une trajectoire, nous ne présentons les algorithmes que dans le cadre d'un espace à 3 dimensions, alors que la méthode peut être applicable à des espaces de dimensions supérieures [Loz86].

Soit C^3 un obstacle de l'espace des configurations à trois dimensions. Nous approximons C^3 par une union de coupes dans l'espace à deux dimensions. Une coupe à 2 dimensions dans l'espace des configurations correspond à un C-obstacle dans l'espace des configurations défini par θ_2, θ_3 quand θ_1 a une valeur donnée. Chaque coupe bi-dimensionnelle peut à son tour être décomposée par une union de coupes mono-dimensionnelles ou "intervalles".
Ces intervalles correspondent à un C-obstacle de l'espace des configurations à une dimension définie par θ_3 quand θ_1 et θ_2 sont fixés. En admettant la possibilité de calculer un intervalle légal pour une seule articulation, il est possible d'appliquer l'algorithme suivant:

étape 1 Ignorer les articulations autres que la première. Trouver les intervalles pour θ_1 dans lesquels la première articulation peut se mouvoir librement.

étape 2 Echantillonner les intervalles de positions valides pour θ_1 à un pas donné. Faire les étapes de 3 à 5 pour chaque valeur échantillonnée de θ_1.

étape 3 Ignorer les articulations autres que la deuxième. Trouver les intervalles de valeurs légales pour θ_2 pour la valeur courante de θ_1.

étape 4 Echantillonner les intervalles de positions valides pour θ_2 à un pas donné. Faire l' étape 5 pour chaque valeur echantillonnée de θ_2.

étape 5 Trouver l'ensemble des intervalles valides pour θ_3 pour les valeurs courantes θ_1 et θ_2.

En prenant cette approche, le calcul des C-obstacles dans un espace des configurations à trois dimensions est ramené à un calcul d'intervalles valides pour un seul degré de liberté. Pour celà, il suffit de calculer le débattement d'une articulation par rapport aux obstacles. Si nous utilisons une modèlisation polyèdrique, alors nous cherchons le débattement d'un polyèdre pour qu'il rentre en contact avec un autre. On dénombre trois types de contacts entre deux polyèdres: A,B,C:

Type A sommet-plan,

Type B plan-sommet,

type C arête-arête.

Il est possible de décrire ces contacts comme des relations géométriques entre polyèdres. Par exemple un contact de type A peut s'écrire avec la relation géométrique suivante:

Type A: le plan $[(\sigma, a)](obstacle)$ contient le sommet $[b_0](articulation)$

La technique utilisée pour résoudre cette équation consiste principalement à exprimer les coordonnées du plan et du sommet dans le bon repère de référence.

- Pour une articulation rotoïde, ce repère de référence a son axe z aligné avec l'axe de rotation.
- Pour un degré de liberté prismatique, ce repère de référence à son axe z aligné avec l'axe de translation.

Le repère de référence est immédiatement calculable à partir des matrices "A". Une fois le changement de repère effectué, une méthode générale de résolution (voir Annexe A.1) peut être appliquée pour trouver le débattement relatif à cette relation géométrique.

La même relation peut être utilisée pour les contacts de type B:

Type B: le plan $[(\sigma, a)](articulation)$ contient le sommet $[b_0](obstacle)$

En gardant la même équation que pour les contacts de "type A", on facilite la résolution de l'équation associée. En utilisant cette notation, c'est l' obstacle qui se déplace relativement à l'articulation et non pas l'inverse. Il faut donc changer le signe du résultat obtenu.

Le débattement pour un contact de type C est calculé avec la relation géométrique suivante:

Type C: la droite $[(a_0, a)](articulation)$ intersecte la droite $[(b_0, b)](obstacle)$

La méthode utilisée pour résoudre ces équations est donnée dans l'annexe A.1. Il faut cependant noter que calculer les débattements pour chaque type de contraintes n'est pas suffisant pour obtenir le débattement du degré de liberté et qu'il faut aussi calculer:

- Si le résultat obtenu s'applique aux entités géométriques finies considérées. Par exemple, si l'on connaît la position relative des deux polyèdres tel qu'un sommet d'un polyèdre appartienne au plan supportant une face de l'autre polyèdre, il faut aussi vérifier que le point se trouve effectivement à l'intérieur de la face.
- Les contraintes d'accessibilité d'un contact par rapport à un autre.

Ces calculs peuvent se faire en utilisant la méthode proposée dans [Loz86].

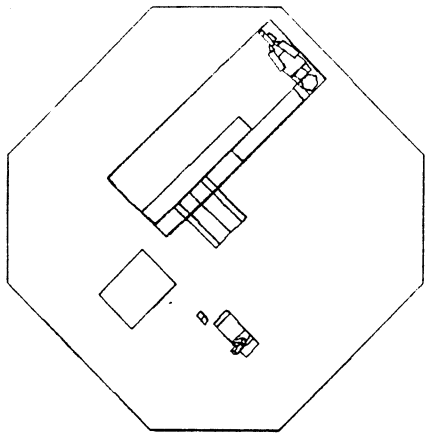


Figure 4.16: Une décomposition de l'espace des configurations par approximation du modèle

Optimisation des calculs

La méthode employée pour calculer la représentation de l'espace des configurations est générale et pourrait être utilisée pour n'importe quel type de robot. Cependant il est possible de simplifier considérablement les calculs en profitant de la structure géométrique d'un robot donné. Par exemple, dans le cas du PUMA il est possible de déterminer facilement des intervalles de θ_1 pour lequels les coupes bi-dimensionnelles en θ_2 et θ_3 sont les mêmes.
Considérons le parallélépipède représenté figure 4.16. Ce parallélépipède contient les mouvements possibles de θ_2 et θ_3 en supposant les articulations 4,5,6 immobiles. En rendant ce parallélépipède solidaire de la première articulation, il est possible, de calculer les intervalles valides pour le premier degré de liberté avec ce parallélépipède. Le parallélépipède incluant tous les mouvements des degrés de liberté 2 et 3, la même coupe en (θ_2, θ_3) est valide pour toutes les valeurs de θ_1 appartenant à ces intervalles. Ceci permet de calculer cette coupe pour une seule valeur de θ_1 et de l'avoir disponible pour toutes les valeurs de θ_1 appartenant à ces intervalles. L'échantillonnage de θ_1 est donc limité, en fait, aux complémentaires de ces intervalles. On peut remarquer que la même technique s'applique récursivement aux degrés de liberté suivants. A θ_1 constant, on peut définir un parallélépipède contenant tous les mouvements possibles de θ_3 (avec les articulations 4,5,6 fixes) et calculer les intervalles légaux pour θ_2 lorsque l'articulation 2 bouge avec ce parallélépipède. Des techniques similaires sont décrites dans [Gou84, Fav86].

Recherche d'une trajectoire dans l'espace des configurations

Chaque coupe est divisée en régions. Une région est un ensemble d'intervalles se recouvrant. On définit le *noyau* d'une région comme étant la partie de recouvrement commune à tous les intervalles. Il est toujours possible d'aller d'un point d'une région au noyau, et, de ce fait, il est toujours possible d'aller d'un point d'une région à une autre. Un graphe des régions est construit: les nœuds du graphe sont les régions et les arêtes indiquent que deux régions ont une frontière commune. On créé aussi des arêtes entre les régions de deux coupes adjacentes si leurs noyaux se

recouvrent. Trouver un chemin d'un point à un autre dans l'espace des configurations est alors équivalent à trouver un chemin d'un nœud du graphe des régions à un autre.

4.6 Planification de la saisie et de l'approche

L'objet de ce module est de planifier une prise stable et le mouvement d'approche conduisant à cette prise. Ceci est acompli en trois étapes.

1. Le problème est tout d'abord ramené d'un problème de planification dans l'espace à un problème de planification dans le plan.
2. Un point de saisie stable est ensuite déterminé ainsi qu'un point de départ pour la trajectoire d'approche.
3. Une méthode utilisant des pseudo-potentiels est ensuite appliquée pour trouver une trajectoire.

4.6.1 Conversion du problème dans un espace plan

Le modèle de la pince utilisé par le planificateur de trajectoire est formé de deux mors opposés pouvant se refermer sur un objet à saisir en glissant sur un pièce transversale (voir figure 4.17). Nous ne considèrons que les prises utilisant des faces parallèles de l'objet. Une prise légale sera telle que chaque mors repose sur une des faces et que la surface de contact soit supérieure à une valeur donnée. Nous ne considérons pas l'excentricité de la prise comme un facteur de stabilité estimant que la pince peut exercer une pression suffisament forte comparée aux poids des objets transportés. Le problème de la planification du mouvement d'approche est simplifié considérablement si l'on se ramène à un mouvement plan. Pour cette raison nous nous sommes limités à planifier des mouvements dans le plan médian aux deux faces parallèles définissant la prise, dans la suite nous appèlerons ce plan: le plan de préhension.

Lorsque la pince se déplace en translation ou en rotation dans le plan de préhension ses éléments constitutifs balayent une portion de l'espace que nous appèlerons le volume de préhension. Tout objet faisant partie du volume de préhension peut provoquer une collision avec le modèle de la pince. Considérons uniquement le volume balayé par un seul mors de la pince (voir figure 4.18). Pour plus de clarté, les volumes balayés par l'autre mors et par la partie transversale de la pince ont été omis de la figure. Le modèle du mors de la pince sur le plan de préhension peut être construit comme étant l'ombre projetée du mors sur le plan de préhension. De la même façon il est possible de construire le modèle d'un obstacle du plan de préhension simplement en considérant l'ombre projetée d'un obstacle du volume de préhension sur le plan de préhension . Dans les deux cas on suppose que l'on "éclaire" par derrière le volume de préhension et selon une direction perpendiculaire au plan de préhension. Une condition nécessaire et suffisante pour qu'une partie du modèle de la pince rentre en collision avec un obstacle est que son "ombre" recouvre l' ombre de l'obstacle. Autrement dit à toute trajectoire sûre de la projection de la pince parmi les projections d'obstacles sur le plan de préhension correspond une trajectoire sans collision dans l'espace réel. Il faut cependant noter qu'il existe des positions sûres pour la pince qui correspondent à des situations de collision dans le

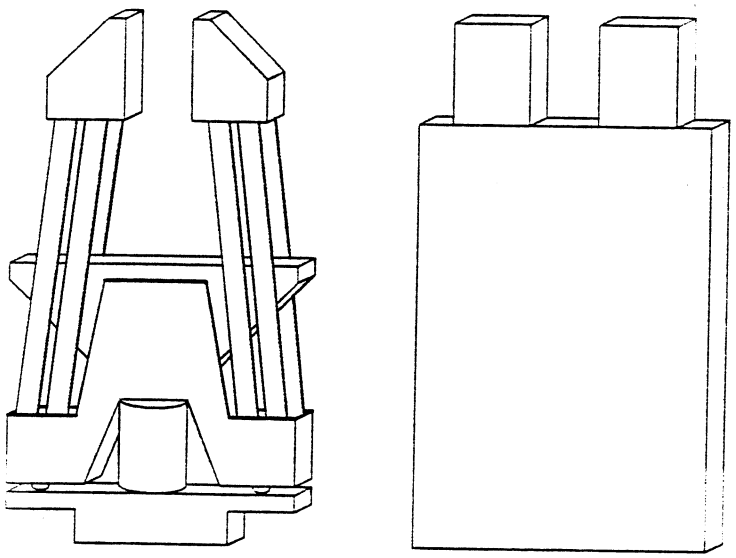


Figure 4.17: Représentation schématique de la pince et du modèle utilisé par la planificateur

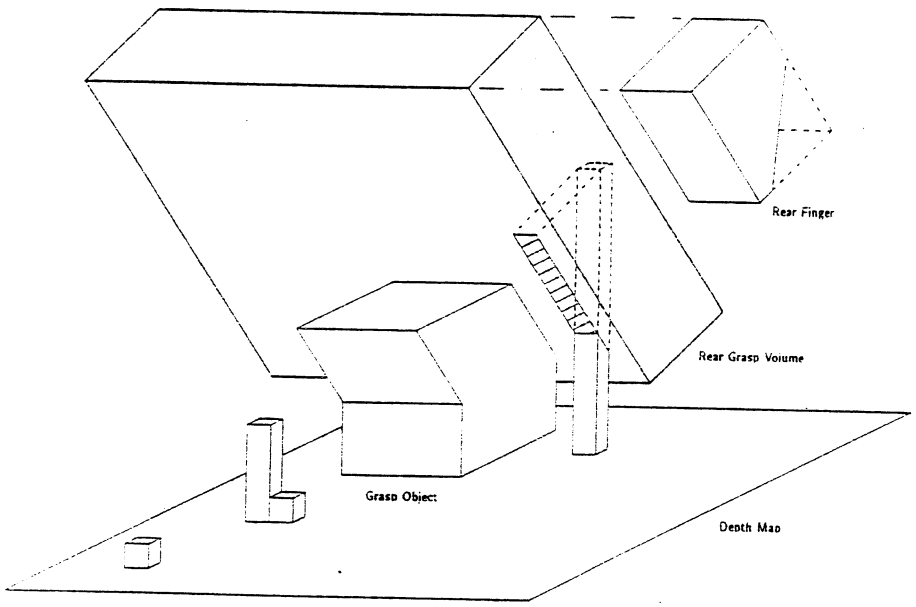


Figure 4.18: Comment les obstacles situés dans le volume de préhension sont projetés sur le plan de préhension.

plan de préhension. Ceci résulte du fait que le modèle utilisé pour la pince englobe le volume occupé son volume réel.

La projection des obstacles sur le plan de préhension est faite de la manière suivante.

Pour chaque obstacle O
Pour chaque point P de O
faire
    si P appartient au volume de prehension
      alors projeter P sur le plan de prehension

En fait nous ne disposons pas dans la V-zone d'une représentation complète des obstacles mais d'une carte tri-dimensionnelle de la V-zone. Par sécurité il est donc préférable de considérer que tous les points situés en dessous la nappe tri-dimensionnelle sont des obstacles. Il nous faut donc legèrement modifier l'algorithme précédent:

Pour chaque point P de la nappe tri-dimensionnelle
faire
    projeter le point P verticalement sur la table en P'
    si une portion segment PP' intersecte le volume de prehension
      alors le projeter sur le plan de prehension.

L'objet à saisir ne se distingue pas des obstacles dans la carte tri-dimensionnelle et, en utilisant l'algorithme précédent, il peut se faire que l'objet lui-même rende inaccessible une partie du plan de préhension qui est, en fait, libre.

Considérons par exemple la figure 4.19. Si l'on applique l'algorithme précédent, tous les points de la partie noire donnent lieu à une projection (partie hachurée) sur le plan de préhension, rendant ainsi inaccessible une face de l'objet qui pourrait servir d'appui à un prise stable. Pour cette raison nous nous servons de la position de l'objet à saisir pour marquer les points de la nappe tri-dimensionnelle qui appartiennent à cet objet et nous ne les prenons pas en compte dans la projection des obstacles sur le plan de préhension.
L'objet lui même peut intersecter le volume de préhension (figure 4.20) et provoquer une collision durant la saisie. Pour éliminer ce cas nous appliquons notre premier algorithme de projection à l'objet que nous désirons saisir.

Il reste qu'un obstacle peut être situé en dessous de la partie surplombante de l'objet. Ce cas n'est pas détéctable avec notre capteur et dans notre implémentation nous n'en tenons pas compte. Il serait cependant souhaitable que les déplacements effectués "sous" l'objet à saisir le soient sous le contrôle d'un moniteur ayant accès à des capteurs tel qu'un capteur de proximité ou de force.

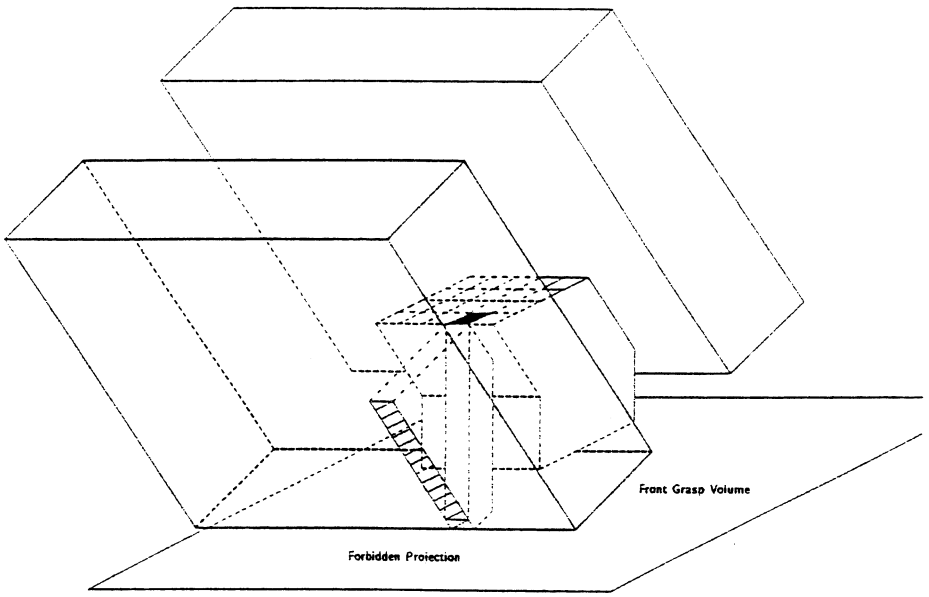


Figure 4.19: Pourquoi le modèle doit être retiré de la nappe tri-dimensionnelle

4.6.2 Choix de la paire de faces

Le plan de préhension est implémenté sous la forme d'un tableau. Ceci permet de changer arbitrairement la résolution de la projection. Avant de faire une analyse détaillée du plan de préhension correspondant à une paire de faces données. Une première passe à basse résolution est effectuée pour sélectionner un couple de faces. Le planificateur de saisie applique l'agorithme de projection à très basse résolution à tous les couples de faces parallèles. En comptant le nombre de cellules libres de chaque plan de préhension on se dote d'une mesure sur la qualité de la paire de faces correspondante. Les paires sont rangées par ordre croissant selon cette mesure et testées à haute résolution par ordre décroissant.

Dans l'idéal le planificateur de saisie doit choisir une prise pour l'objet qui puisse être utilisée à la fois au moment de la saisie et au moment de la dépose de l'objet. La position finale de l'objet étant connue à l'avance, il est possible de considérer l'influence des objets situés au voisinage de la position finale, au moment de la prise. Pour cela il suffit de considérer les positions relatives de l'objet à saisir avec les autres objets de la scène dans la position finale. Puis, d'utiliser ces positions pour placer artificiellement les objets avec la même position relative mais, cette fois, en considérant la position initiale de l'objet à saisir. Ainsi si I et F sont les matrices en coordonnées homogènes correpondant à la position initiale et finale de l'objet à saisir, et si 0_i représente la position d'un objet i de la scène, on donnera

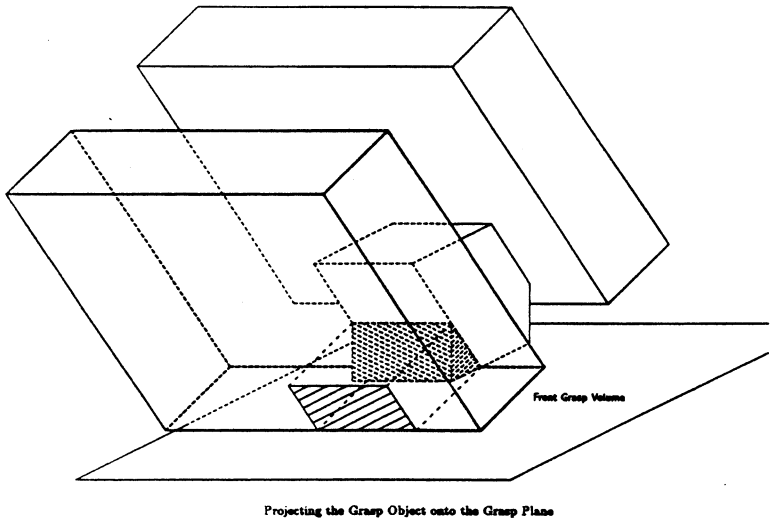


Figure 4.20: L'objet à saisir peut intersecter le volume de préhension et doit être projeté sur le plan de préhension

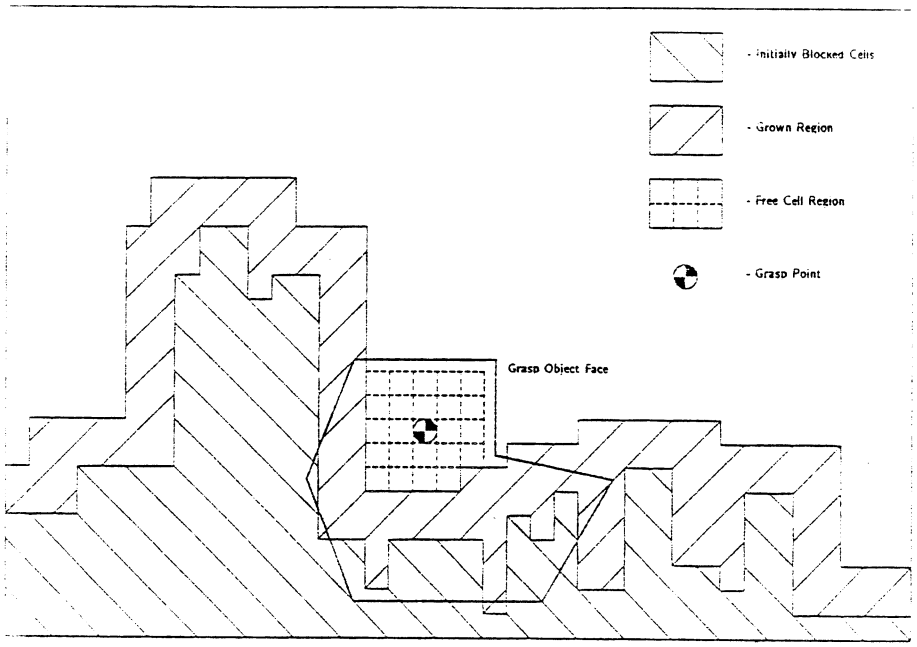


Figure 4.21: Représentation du plan de préhension

la position $O_i' = F^{-1}IO_i$ au modèle de l'objet i dans le système de modèlisation géométrique. On utilise ensuite ce modèle "rétro-projeté" pour l'intersecter avec le volume de préhension et en déduire sa projection comme obstacle dans le plan de préhension. Il est clair que si nous pouvons trouver une trajectoire et une prise dans un plan de préhension contenant à la fois les obstacles de la V-zone et les obstacles "rétro-projetés", alors la prise trouvée sera valide à la fois à la position initiale et à la position finale de l'objet. Cette technique permet, en fait, de propager des contraintes géométriques prenant source à différents endroits de la scène.

Le planificateur de saisie réalise une première passe à basse résolution avec les objets "rétro-projetés"; si aucune paire ne totalise un nombre suffisant de cellules libres alors une opération de re-saisie est nécessaire pour effectuer le montage. Dans ce cas, une nouvelle passe à basse résolution est lancée mais, cette fois, sans la contrainte provenant de la position finale.

Une fois une paire de faces choisie (avec ou sans contrainte finale) une représentation haute résolution du plan de préhension est construite et les contours des faces choisies sont représentées sur le plan de préhension (figure 4.21).

Choix du point de prise et du point de départ

Le plan de préhension peut être considéré comme un tableau de cellules marquées "libre" si une cellule représente une portion de l'espace libre et "occupée" si un obstacle se projette sur cette cellule. Pour utiliser notre méthode des pseudo-potentiels nous avons besoin de créer un point d'attraction dans le plan de préhension. Pour se faire nous utilisons la méthode suivante:

1. La première étape consiste à "grossir" la frontière des régions occuppées d'un nombre de cellules correspondant approximativement à une demi-largeur de mors. Cette étape a pour but de sélectionner un point cible qui ne soit "pas trop près" d'un obstacle pour les mors de la pince.

2. La deuxième étape consiste à faire l'intersection des regions libres avec les deux faces de l'objet définissant le plan de préhension.

3. Après la précédente opération, l'ensemble des cellules restantes se trouvent en regard des deux faces et éloignées des obstacles. Nous determinons les composantes connexes de cet ensemble de cellules avec un algorithme classique de coloriage emprunté à la vision par ordinateur et sélectionnons la composante connexe possèdant le plus de cellules.

4. Nous rétrécissons progressivement cette composante connexe juqu'à n'obtenir qu'un point, que nous appelons point de préhension de l'objet.

Ce point sert d'attracteur à notre méthode de pseudo-potentiels. Le point de départ est choisi dans la même composante connexe que cet attracteur dans le plan de préhension originale. En fait plusieurs points sont choisis arbitrairement dans cette zone et seront pris successivement comme points de départ jusqu'à ce qu'une solution soit trouvée ou que ce plan de préhension soit abandonné.

La méthode des pseudo-potentiels

Le planificateur de saisie utilise une pseudo-méthode des potentiels pour planifier la trajectoire de la pince dans le plan de préhension. Une méthode plus rigoureuse est présentée dans [Kat86]. Elle consisterait:

1. à calculer, dans le plan de préhension, la distance entre chaque point du modèle de la pince et chaque point de l'espace occupé,

2. à en déduire la force répulsive s'appliquant au modèle de la pince en utilisant une pseudo-loi de Newton

3. à faire la somme de cette force avec une force attractive entre le point de préhension de l'objet et un point situé dans les mors de la pince (point de préhension de la pince).

4. à en déduire le mouvement du modèle de la pince dans le plan de préhension.

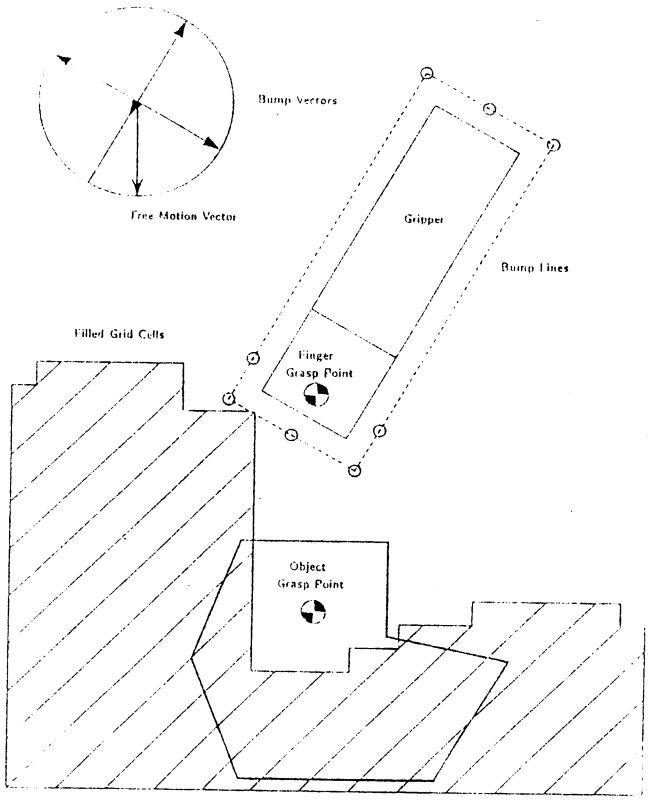


Figure 4.22: La méthode des pseudo-potentiels

Cette approche demande de nombreux calculs et peut s'avérer très coûteuse en temps calcul. Pour cette raison nous avons opté pour une méthode plus simple.

Nous définissons autour du modèle de la pince des segments de droite "pare-choc" (figure 4.22). Ces segments de droite se trouvent à une distance d du contour du modèle et peuvent se déplacer avec lui. La méthode consiste à tester si un des "pare-chocs" se trouve dans une région marquée et d'en déduire un mouvement correctif. A chaque "pare-choc" nous associons un vecteur unitaire perpendiculaire au segment

En l'absence d'intersection entre un "pare-choc" et une région marquée du plan de préhension, le mouvement de la pince est une simple translation ayant pour direction le vecteur défini par le point de préhension de la pince et le point de préhension de l'objet. Nous associons à cette direction un vecteur unitaire (appelé dans la suite le vecteur de déplacement libre).
Nous examinons chaque pare-choc pour tester s'il intersecte une cellule marquée. Si un pare-choc intersecte une cellule alors le vecteur associé est marqué 0 sinon il est marqué 1 .

Pour calculer la translation de la pince nous comparons le vecteur de déplacement libre au précédent vecteur. Si le vecteur de déplacement libre ne se projette pas sur un vecteur marqué 0 alors le déplacement en translation a lieu le long du vecteur de

déplacement libre. Sinon nous projetons le vecteur de déplacement libre sur chaque vecteur marqué 1 et fixons la direction de translation comme étant le vecteur associé à un pare-choc qui maximise la projection du vecteur de déplacement libre.

La rotation se fait autour de point de préhension de la pince. Chaque pare-choc intersectant un obstacle engendre un couple s'appliquant au point de préhension de la pince. Ce couple est proportionnel au produit vectoriel du vecteur associé au pare-choc et au vecteur de déplacement libre. La rotation finale est simplement calculée en faisant la somme de tous les couples appliqués.

Le mouvement de la pince est limité à une cellule par itération, ceci évite qu'une cellule marquée puisse toucher la pince. A chaque itération le système effectue une transformation de coordonnées inverse, il vérifie que la nouvelle position est atteignable en continuité avec la précédente (sans reconfiguration du bras). Si cela n'est pas le cas, alors les autres directions de déplacement sont utilisées.

Conditions d'arrêt

Le planificateur utilise trois conditions d'arrêt:

- Le système vérifie régulièrement la progression de la pince vers son but. Si aucune progression n'est enregistrée alors le planificateur cherche un autre point de départ pour planifier la trajectoire ou considère cette prise impossible.
- Un temps maximum est alloué pour la recherche d'une trajectoire. Si ce temps est écoulé, alors le planificateur cherche un autre point de départ pour planifier la trajectoire ou considère cette prise impossible.
- Régulierement le planificateur teste la surface de contact existant entre les mors et les faces de l'objet définissant le plan de préhension. Si cette surface est supérieure à un seuil donné alors le planificateur retourne la trajectoire trouvée.

4.7 La Re-saisie

Une re-saisie est nécessaire à chaque fois qu'une prise effectuée par le robot n'est pas compatible avec la tâche qu'il doit effectuer. Du fait d'une conjonction de contraintes à la position de saisie et à la position de montage, il se peut que la tâche ne puisse être effectuée avec une seule prise, par exemple:

- L'interaction géométrique entre l' objet, la pince du robot et l'environnement.
- La cinématique du manipulateur, et en particulier, les butées mécaniques.

Pour être plus précis, nous devons ajouter les contraintes dûes au mouvement du bras entre la position de saisie et la position de montage. Nous ne traiterons pas ce type de contraintes, et nous supposerons que la cellule robotique est disposée d'une manière telle, qu'il est toujours possible d'aller un point à un autre.
Les opérations de re-saisie peuvent être séparées en deux grandes classes. L'une inclut les mouvements durant lesquels l'objet reste en contact avec la pince. Ce type de manipulation ne peut se faire qu'à l'aide de pinces spécialement conçues pour la manipulation complexe [Fea84] car elle nécessite un controle précis des forces exercées sur l'objet manipulé. Par exemple si l'on doit tourner un stylo de 180° en utilisant l'une de ces pinces, on commencera par orienter la pince de telle façon que le point de prise se situe légèrement en dessous du centre de gravité du stylo, puis on relachera légèrement la pression des mors sur le stylo de façon à permettre au stylo de tourner. L'autre classe d'opérations de re-saisie exclut tout mouvement dynamique. Elle consiste en un ensemble de manipulations telles que la position relative de l'objet et de la pince reste constante pendant les mouvements de transport, les seuls changements intervenant lors de la pose de l'objet sur la table et de la re-saisie. Dans ce chapitre nous nous limiterons à la deuxième classe d'opération.

Le problème

Nous pouvons définir informellement le problème de la re-saisie comme: une position initiale et une position finale de prise données, puis la construction d'une séquence de prise et de dépose de l'objet permettant de passer de l'une à l'autre. Nous supposons qu'entre deux phases de saisie l'objet est placé sur la table dans une position stable. La figure 1.10 représente une opération de re-saisie d'un objet en forme de L. Cette opération se fait en deux étapes avec un robot à six degrés de liberté.
Pour les objets polyèdriques, nous conjecturons qu'en l'absence d'obstacle et de butée mécanique au voisinage de la zone de dépose, toute opération de resaisie peut se faire en une seule étape. En pratique, ces contraintes ne sont jamais réalisées et plusieurs étapes sont en général nécessaires pour effectuer complètement cette opération.
Notre approche est la suivante: nous représentons l'ensemble des prises d'un objet ainsi que l'ensemble de ses positions stables sur la table. Dans les deux cas, nous

caractérisons ces ensembles par un ensemble discret de positions combiné avec un paramètre continu. Le problème de la re-saisie est alors résolu en calculant les transitions dans un espace où nous représentons toutes les combinaisons de prise et de pose pour l'objet.
Une prise est compatible avec une pose si l'on peut utiliser cette prise pour réaliser la pose. Ceci peut être testé d'une part en considérant les collisions de la pince avec la table, et, d'autre part, en considérant les positions articulaires permettant de réaliser la pose.
Le choix d'une prise stable pour saisir un objet a fait l'objet de nombreuses études [Pau72, Loz76, Loz81, Tay76, Bro80, Win77, Lau81, HA77, LP83, AHM85, BFG85, HM86, BVD*86, Ngu86, JL86]. A notre connaissance, la seule étude systématique du problème de la re-saisie à été présentée par Paul [Pau72] dans le cadre du "Move Instance System". Paul supposait que la re-saisie pouvait se faire en une seule étape et ne tenait pas compte complètement des contraintes cinématiques. Toutefois la plupart des idées présentées dans ce chapitre sont inspirées de son travail.

4.7.1 Les prises d'un objet

Choisir des prises stables

Nous nous intéressons aux prises possibles pour saisir des objets polyèdriques pas forcément convexes. La pince est composée de deux mors parallèles ayant des plans comme surface de saisie. En ignorant la friction, il est possible d'obtenir une prise stable pour un objet convexe en réalisant les contacts suivant avec les plans des mors:

1. Deux faces parallèles de l'objet dont les projections mutuelles s'intersectent (propriété de visibilité mutuelle).

2. Une face et une arête parallèle ayant la propriété de visibilité mutuelle.

3. Une face et un sommet, si le sommet se projette dans la face.

Dans le cas d'objet non-convexe, une première classe de prises peut être obtenue avec les paires d'entités géométriques précédentes. De surcroit, il est possible d'obtenir des prises stables en utilisant les parties concaves de l'objet. Dans ce dernier cas, l'orientation de l'objet dans la pince appartient à un ensemble discret d'orientations possibles et ne constitue pas une prise souhaitable pour le problème de la re-saisie. Nous nous limitons dans la suite aux prises de type (1), toutefois la méthode présentée pourrait être étendue sans difficulté majeure aux prises de type (2) et (3).
Dans tous les cas, une paire d'entités géométriques représente une collection finie de plans $G_i, i \in \{1, ..., n\}$ sur lesquels les faces des mors s'appuiront durant la saisie. Nous définissons $\mathbf{n}_i$ le vecteur unitaire perpendiculaire à l'un de ces plans. C'est un vecteur perpendiculaire à une face de l'objet.

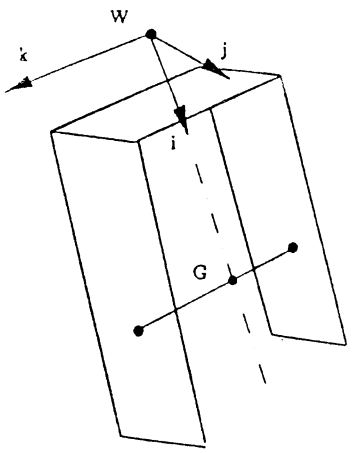


Figure 4.23: Repere de référence lié au modèle de la pince

4.7.2 Les paramètres d'une prise

Pour chaque classe de prises G_i, on définit un repère de référence standard lié à l'objet ayant les axes x-y dans le plan de préhension. Nous écrivons ce repère $(\mathbf{C}_i, \mathbf{u}_i, \mathbf{v}_i, \mathbf{w}_i)$, avec $\mathbf{w}_i = \mathbf{n}_i$.
Soit $(\mathbf{c}, \mathbf{u}, \mathbf{v}, \mathbf{w})$ le repère s de référence dans lequel l'objet est décrit, $\mathbf{C}$ étant le centre de gravité de l'objet et soient $\mathbf{M}_1$ et $\mathbf{M}_2$ deux points de contact quelconques entre l'objet et les mors de la pince, appartenant aux deux surfaces opposées de la prise. Nous appelons

$$e_i = \frac{1}{2}(\overline{\mathbf{CM}}_1 + \overline{\mathbf{CM}}_2) \cdot \mathbf{n}_i$$

l'excentricité de la prise G_i. Nous imposons la condition $\overline{\mathbf{CC}}_i \cdot \mathbf{n}_i = e_i$ sur l'origine du repère de référence associé à la prise G_i. De ce fait, le point $\mathbf{C}_i$ est équidistant des deux faces de contact de la prise. Dans la suite, $(\mathbf{C}_i, \mathbf{u}_i, \mathbf{v}_i)$ est appelé le plan de préhension.
Nous supposons un modèle standard pour une pince à mors parallèles, avec deux parallélépipèdes rectangles symétriques par rapport à l'axe de la pince. Nous supposons que le modèle de la pince est exprimé dans un repère de référence ayant pour origine le centre du poignet du robot et que l'axe des trois dernières articulations s'intersectent en ce point. Nous écrivons $(\mathbf{W}, \mathbf{i}, \mathbf{j}, \mathbf{k})$ ce repère (voir figure 4.23). Le vecteur $\mathbf{k}$ est normal à la surface de contact des mors, le vecteur $\mathbf{i}$ est aligné avec les mors et pointe à l'extérieur de la pince. Durant une saisie, le centre du poignet est contraint d'appartenir au plan de préhension et, $\mathbf{w}_i$ est aligné avec le vecteur $\mathbf{k}$.
Comme le montre la figure 4.24, une prise donnée peut être décrite par: x et y, les coordonnées du centre du poignet dans le plan de préhension, et par l'orientation des mors dans ce plan. Cette orientation est donnée par ϑ, l'angle entre les vecteurs $\mathbf{u}_i$ et $\mathbf{i}$.
L'espace de recherche défini par (x, y, ϑ) est cependant trop grand pour être exploré

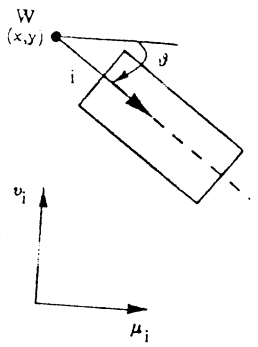


Figure 4.24: La position de la pince est complètement spécifiée par x, y, ϑ

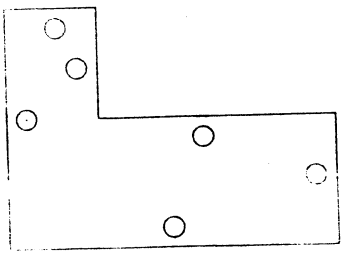


Figure 4.25: Positions de prise associeés à l'une des faces d'un objet

complètement. Nous souhaiterions nous restreindre à un seul paramètre pour décrire une classe de prise. Ceci est possible si l'on choisit une position pour le point de préhension de la pince. Nous appelons ce point **G**. Nous pourrions sélectionner la position de **G** relative de façon à maximiser l'intervalle de positions légales pour l'angle ϑ. Dans notre implémentation, nous nous limitons à un nombre fini de positions pour le point **G** dans le plan de préhension. Ceci a pour conséquence de rendre l'algorithme de re-saisie moins général et d'éclater chaque classe de prises G_i en un nombre discret de classes de prises.
Soient $\{e_1^i, e_2^i, \cdots, e_n^i\}$ et $\{e_1^j, e_2^j, \cdots, e_m^j\}$, les ensembles d'arêtes délimitant les faces F^i, F^j utilisées pour définir le plan de préhension. Pour chaque arête e_k^l de ces deux ensembles, nous associons un point p_k^l de la face F^l tel que:

1. p_k^l appartient F^l, et
2. p_k^l est à une distance donnée d du point milieu de e_k^l.

Les points p_k^l associés à la face F^l de la pièce en forme de L sont représentés figure 4.25. La projection de ces points sur le plan de préhension définit des positions possibles pour **G**. Pour être valides, ces points doivent aussi se projeter à l'intérieur de la face opposée.
Ces points peuvent être utilisés comme origine $\mathbf{C}_i$ au repère définissant une classe de prises. Dans la suite, nous appelons le point $\mathbf{C}_i$ *le point de préhension* de l'objet pour la classe G_i. Nous écrivons $\overline{\mathbf{CC}_i} = x_i\mathbf{u}_i + y_i\mathbf{v}_i + e_i\mathbf{w}_i$. L'angle ϑ définit

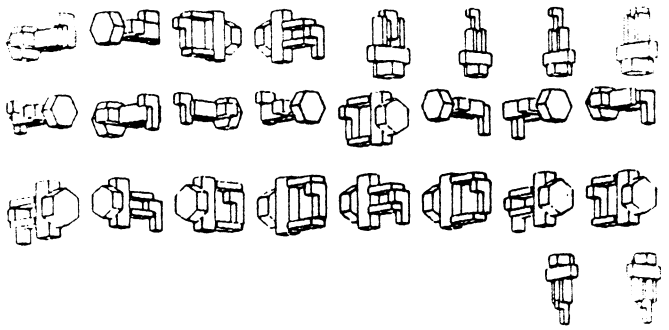


Figure 4.26: Différentes prises utilisées par le planificateur de re-saisie

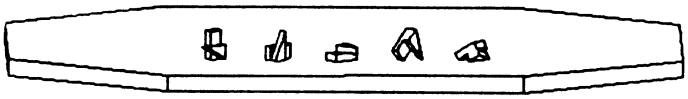


Figure 4.27: Faces d'équilibre d'un objet

alors complètement la prise. Plus précisement, nous avons: $x = -\Delta g \cos\vartheta$ et $y = -\Delta g \sin\vartheta$, avec Δg la distance entre le point de préhension de la pince et le centre du poignet.

Nous définissons l'ensemble des prises comme étant l'ensemble des couples $(G_i: \vartheta)$:

- G_i est une classe de prises (dépendant du choix d'un point de prise dans le plan de préhension).
- ϑ est l'angle formé par les mors dans un repère lié à l'objet .

La figure 4.26 représente l'ensemble des G_I pour un objet en forme de L. Dans ces exemples, la valeur de ϑ a été fixée à zéro.

4.7.3 Les poses d'un objet

Nous associons une pose d'un objet à une face de l'objet. Pour être valide, cette face doit appartenir à l'enveloppe convexe de l'objet et doit contenir la projection du centre de gravité de l'objet. Nous définissons $\mathbf{n}_j$ la normale extérieure à la face et par d_j la distance entre la face et le centre de gravité. Pour un objet donné, nous obtenons une collection de poses $P_j, j \in \{1, ..., m\}$. La figure 4.27 représente l'ensemble des poses possibles d'un objet utilisé pour notre expérimentation.
Jusqu'à présent, nous n'avons pas défini la position de l'objet par rapport à un repère de référence absolu. Nous limiterons une pose comme étant la position de l'objet à une position donnée de la table. Soit $\mathbf{H}_j$ la projection du centre de gravité sur la face $(\mathbf{n}_j, d_j)$ associée à la pose. Nous imposons que ce point $\mathbf{H}_j$ coïncide avec un point donné de la table $\mathbf{H}$, origine d'un repère de référence absolu $(\mathbf{H}, \mathbf{I}, \mathbf{J}, \mathbf{K})$.

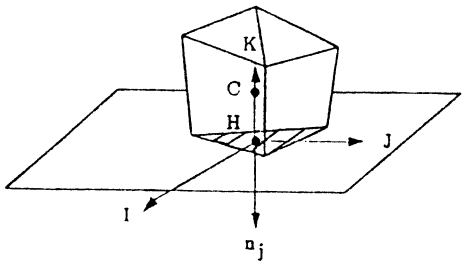


Figure 4.28: Définition de la position d'une pose

Ce repère de référence est choisi de telle sorte que $(\mathbf{H}, \mathbf{I}, \mathbf{J})$ représente la surface de la table et que $\mathbf{K}$ soit la normale extérieure à la table.
La position de l'objet pour une pose donnée est complètement spécifiée par l'angle de rotation autour du vecteur $\mathbf{K}$: φ.
Le point $\mathbf{H}$ doit être choisi de sorte que le poignet puisse librement se déplacer au dessus de cette position. Essentiellement le point $\mathbf{H}$ doit appartenir à une zone où la pince peut être positionnée avec un maximum d'orientations.

Nous définissons l'ensemble des placements comme l'ensemble des couples $(P_j: \varphi)$:

- P_j est une classe de poses définie par la normale à la face de contact entre l'objet et la projection de centre de gravité sur cette face. Ceci suppose qu'un point de la table ait été choisi.
- φ est l'angle de rotation de l'objet autour d'un axe vertical passant par le point $\mathbf{H}$

4.7.4 Construction de l'ensemble produit Prise-Pose

Représentation d'un couple Prise-Pose

Nous pouvons maintenant définir plus formellement le problème de la re-saisie. Le choix d'une prise $(G_i: \vartheta)$ fixe entièrement la position de l'objet dans la pince. De même, une pose définit complètement la position d'un objet sur la table. De ce fait, une opération de saisie ou de lâcher (se déssaisir de la pièce sur la table) est complètement specifiée par la donnée de $(G_i: \vartheta)$ et $(P_j: \varphi)$. Toute combinaison n'est cependant pas valide: il faut pouvoir exhiber au moins un ensemble de coordonnées articulaires pour le bras qui corresponde à la position du poignet définie par $(G_i: \vartheta)$ et $(P_j: \varphi)$. Nous devons, en plus, prendre en compte les contraintes d'accessibilité:

1. Les intersections entre le modèle de l'objet et la pince. Celles-ci limitent les choix du paramètre ϑ dans $(G_i: \vartheta)$.
2. Les interactions de la pince avec l'environnement au milieu duquel l'opération de re-saisie est effectuée. Celles-ci limitent le choix des paramètres à la fois dans $(G_i: \vartheta)$ et $(P_j: \varphi)$.

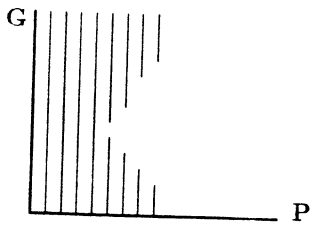


Figure 4.29: Topologie d'une table Prise-pose

3. Les butées et limitations mécaniques du bras.

A chaque couple (G_i, P_j), nous associons un sous-ensemble $\overline{\mathbf{q}_{ij}}$ de l'espace produit $[0, 2\pi] \times [0, 2\pi]$ correspondant aux valeurs du couple (ϑ, φ) pour lesquelles la position de saisie (ou de lâcher) est définie. Un exemple de la topologie d'un ces sous-ensembles est donné figure 4.29. L'ensemble $\overline{\mathbf{q}_{ij}}$ est obtenu en échantillonnant l'ensemble $[0, 2\pi] \times [0, 2\pi]$ et en appliquant les trois critères de validité définis précédement. La construction de ce sous-ensemble est reprise en détail en 4.7.5. Il faut noter que pour certains couples G_i, P_j, l'ensemble $\overline{\mathbf{q}_{ij}}$ est vide.

Recherche d'une séquence de prises

Une opération de re-saisie en un temps est une transition entre deux prises: $(G_{i_1}: \vartheta_1)$ et $(G_{i_2}: \vartheta_2)$ en passant par une pose $(P_j: \varphi)$ compatible avec les deux prises. De façon duale, on peut définir une opération de re-lâcher en un temps comme une transition entre deux poses $(P_{j_1}: \varphi_1)$ et $(P_{j_2}: \varphi_2)$ en passant par une prise $(G_i: \vartheta)$ compatible avec les deux poses. Plus formellement, nous pouvons écrire les deux propositions suivantes:

Proposition 1: Une transition entre deux prises $(G_{i_1}: \vartheta_1)$ et $(G_{i_2}: \vartheta_2)$ est légale si et seulement si il exite une pose $(P_j: \varphi)$ telle que:

$$(\vartheta_1, \varphi) \in \overline{\mathbf{q}_{i_1 j}}$$

$$(\vartheta_2, \varphi) \in \overline{\mathbf{q}_{i_2 j}}$$

Proposition 2: De facon équivalente, une transition entre deux poses $(P_{j_1}: \varphi_1)$ et $(P_{j_2}: \varphi_2)$ est légale si et seulement il existe une prise $(G_i: \vartheta)$ telle que:

$$(\vartheta, \varphi_1) \in \overline{\mathbf{q}_{i j_1}}$$

$$(\vartheta, \varphi_2) \in \overline{\mathbf{q}_{i j_2}}$$

Une version du problème de la re-saisie peut maintenant être définie comme:

Figure 4.30: Une séquence de poses et de prises conduisant à la prise désirée

Étant donné une prise initiale $(G^{ini}: \vartheta^{ini})$ et une prise prise finale $(G^{goal}: \vartheta^{goal})$, trouver une séquence de prises $(G_{i_0}: \vartheta_0), ..., (G_{i_n}: \vartheta_n)$, telle que $(G_{i_0}: \vartheta_0) = (G^{ini}: \vartheta^{ini})$, $(G_{i_n}: \vartheta_n) = (G^{goal}: \vartheta^{goal})$, et telle qu'il existe une transition légale par une pose entre deux prises. S'il existe une solution à ce problème, nous cherchons celle qui minimise la longueur de la séquence.
Une telle séquence est illustrée figure 4.30. Le méthode de calcul de cette séquence est donnée dans la section 4.7.7. En pratique les prises finale et initiale ne sont pas forcément complètement specifiées. Par exemple la classe de prise finale G^{goal} peut être spécifiée et une contrainte sur l'intervalle de variation de $[\vartheta^{goal}]$ imposée sans que la valeur de $[\vartheta^{goal}]$ ne soit pour autant fixée. Cette version plus générale du problème de la re-saisie est celle que nous avons implantée.

4.7.5 Calcul des contraintes de l'espace prise-pose

Nous présentons maintenant la méthode de calcul du sous ensemble de $[0, 2\pi] \times [0, 2\pi]$: $\overline{\mathbf{q}_{ij}}$ des valeurs légales du couple (ϑ, φ) pour que la prise $(G_i: \vartheta)$ soit compatible avec la pose de $(P_j: \varphi)$.

Interaction de la pince et de l'objet

Pour une classe de prises donnée G_I (ce qui implique le choix d'un point de préhension dans le plan de préhension), l'espace des configurations de la pince peut être paramétrée par l'angle ϑ. Dans cet espace, nous pouvons calculer les obstacles, c'est-à-dire les intervalles de ϑ pour lesquels la pince rentre en collision avec des obstacles. La méthode est exactement la même que celle présentée dans la section 4.5.3. Le complément de ces intervalles correspond aux intervalles légaux de ϑ: $\overline{\theta}_i^1$. Nous avons donc:

$$\mathbf{q}_{ij} \subset \overline{\theta}_i^1 \times [0, 2\pi]$$

Lorsque nous calculons les C-obstacles, nous ne tenons compte que de l'objet à saisir et de la table.

4.7.6 Contraintes cinématiques et butées mécaniques

Le dernier ensemble de contraintes que nous devons satisfaire se rapporte au choix d'une prise $(G_i: \vartheta)$ et d'une pose $(P_j: \varphi)$ données. Nous devons trouver au moins une solution au problème de la transformation de coordonnées de l'espace cartésien à l'espace articulaire et ceci en tenant compte des éventuelles butées mécaniques présentes sur les articulations du robot.

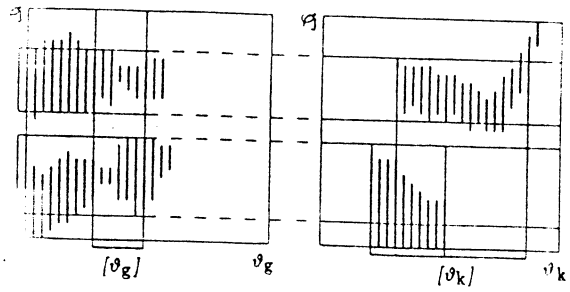


Figure 4.31: Propagation des intervalles dans le processus de chaînage arrière

Le calcul de l'ensemble $\overline{\mathbf{q}_{ij}}$ se fait de la manière suivante:

1. Calculer des intervalles légaux pour ϑ.
2. Discrétiser $\overline{\theta}_i^1$ avec un pas donné, faire les étapes 3 et 4 pour chaque valeur de ϑ ainsi obtenue.
3. Discrétiser l'intervalle $[0, 2\pi]$ avec un pas donné. Pour chaque valeur de φ ainsi obtenue, faire l'étape 4
4. Calculer les solutions dans l'espace des coordonnées articulaires correspondant à la position du poignet définie par $(G_i: \vartheta)$ et $(P_j: \varphi)$.

Pour chaque valeur de ϑ, nous présentons les valeurs légales de φ sous la forme d'intervalle.

4.7.7 Algorithme de recherche d'une séquence de prises

L'algorithme de recherche d'une séquence de prises procède par chaînage arrière. Le but est specifié par une classe de prises finales $(G_{i_g}: [\vartheta_g])$. Notons que ϑ_g peut appartenir à des intervalles de valeurs légales. Chaque étape consiste à trouver une autre classe de prise G_{i_k} telle qu'il existe un placement et un ensemble d'intervalles non nuls de ϑ:

$$[\vartheta_k] = \{\vartheta' | \exists\varphi \exists\vartheta \in [\vartheta_g](\vartheta, \varphi) \in \overline{\mathbf{q}_{i_g j}} \wedge (\vartheta', \varphi) \in \overline{\mathbf{q}_{i_k j}}\}$$

Si un tel ensemble d'intervalles existe, alors cela veut dire que P_j peut être utilisée comme pose entre certaines prises de la classe $(G_{i_g}: [\vartheta_g])$ et toute prise de la classe $(G_{i_k}: [\vartheta_k])$. La recherche s'arrête quand on atteint une classe contenant la prise initiale et que la valeur initiale de ϑ est incluse dans un des intervalles légaux.
Dans notre implantation le chaînage arrière se fait de la manière suivante: Pour chaque pose $(P_j: \varphi)$ nous calculons l'ensemble d'intervalles $[\varphi_j]$ comme étant l'union des coupes-φ de $\overline{\mathbf{q}_{i_g j}}$ quand $\vartheta \in [\vartheta_g]$. On calcule alors ϑ_k comme l'union des coupes-ϑ de $\overline{\mathbf{q}_{i_k j}}$ quand $\varphi \in [\varphi_j]$. (voir figure 4.31).

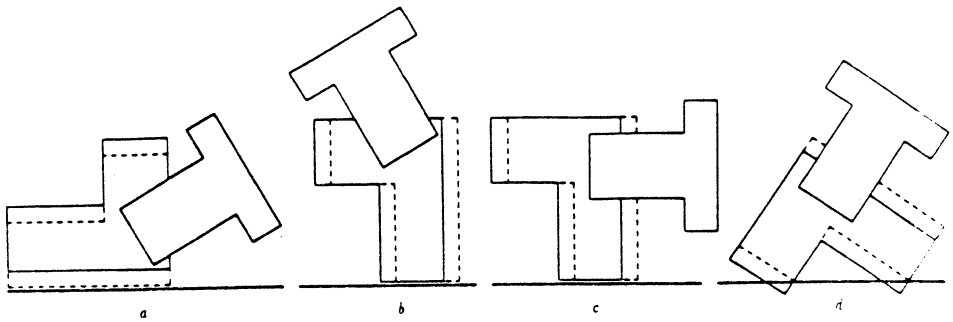


Figure 4.32: Une légère erreur de positionnement à la saisie entraîne une collision durant une séquence de re-saisie

4.8 Le système précis de localisation

4.8.1 Pourquoi la saisie est-elle imprécise

Il y a plusieurs raisons qui rendent la saisie imprécise:

- Un erreur de calibrage entre le repère de référence du robot et le repère de référence de système de vision tri-dimensionnelle. Cette erreur introduit un décallage systématique entre la position de saisie planifiée et la position réelle.
- L'incertitude résultant du processus de localisation, la localisation de la pièce étant calculée avec une précision donnée.
- La précision nominale du robot.

La plupart du temps ces erreurs sont relativement négligeables comparées aux erreurs dûes à des mouvements imprévisibles de la pièce au moment de la saisie. Dans une expérimentation réelle, la pièce peut aussi glisser légèrement durant le mouvement de dégagement. Les conséquences de ces petits mouvements ne sont pas forcément détectables immédiatement, ces erreurs peuvent en effet se propager [PP87] et donner lieu à une collision plusieurs étapes après l'opération défectueuse. Par exemple dans une opération de re-saisie une erreur de position à la saisie (figure 4.32) peut ne pas causer de collision en *b* et *c* mais en *d*. La fonction de la phase de localisation fine de l'objet dans les mors de la pince a pour but d'éviter ce genre de problème.

4.8.2 Utilisation d'un système de vision actif

La méthode est inspirée de [Gor86], elle fait appel à un système de vision actif utilisant un plan lumineux et une caméra. Après calibrage, il est possible de calculer les coordonnées tri-dimensionelles d'un point appartenant à l'intersection du plan

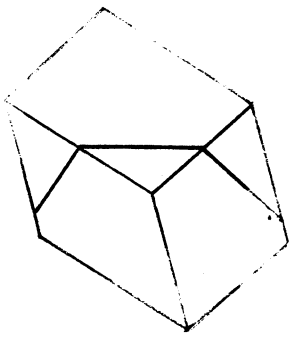


Figure 4.33: Utilisation de trois faces pour localiser l'objet dans l'espace

laser et d'un solide (si ce point est dans le champ de vision de la caméra). Si le plan laser intersecte une face d'un polyèdre, alors l'intersection est une droite et il est possible de calculer les coordonnées de cette droite dans le repère attaché au robot.

Si l'on réussit à faire en sorte que le plan laser intersecte 3 faces du polyèdre (F_1, F_2, F_3) alors on obtient l'équation de trois droites (L_1, L_2, L_3) (figure 4.33). Dans ce cas la position de la pièce peut être calculée en prenant l'inverse de la transformation géométrique correspondant à l'ensemble des relations géométriques suivantes:

- *le plan* F_1 *contient la droite* L_1
- *le plan* F_2 *contient la droite* L_2
- *le plan* F_3 *contient la droite* L_3

Il est, en général assez difficile d'obtenir une configuration de l'objet qui permette d'avoir trois faces simultanément illuminées par le plan laser. Cependant il est possible d'utiliser notre connaissance de la position appoximative de l'objet dans la pince pour réduire le nombre de faces à illuminer. Bien qu'il ne soit pas possible de connaître la position exacte de l'objet dans la pince, il est cependant possible de savoir quelle face F de l'objet est contre quel mors J de la pince (voir figure 4.34). En réduisant le nombre de degrés de liberté pour la pièce nous pouvons n'utiliser que deux intersections du plan lumineux pour la localiser. La position peut être obtenue avec les relations géométriques suivantes:

- *le plan* F *est contre le plan* J
- *le plan* F_1 *contient la droite* L_1
- *le plan* F_2 *contient la droite* L_2

Ces équations n'ont d'intérêt que si J, L_1, L_2 sont exprimés dans le même repère de référence. Ceci est obtenu par un calibrage précis des deux systèmes de référence associés au robot et au système de vision laser [Bor84].

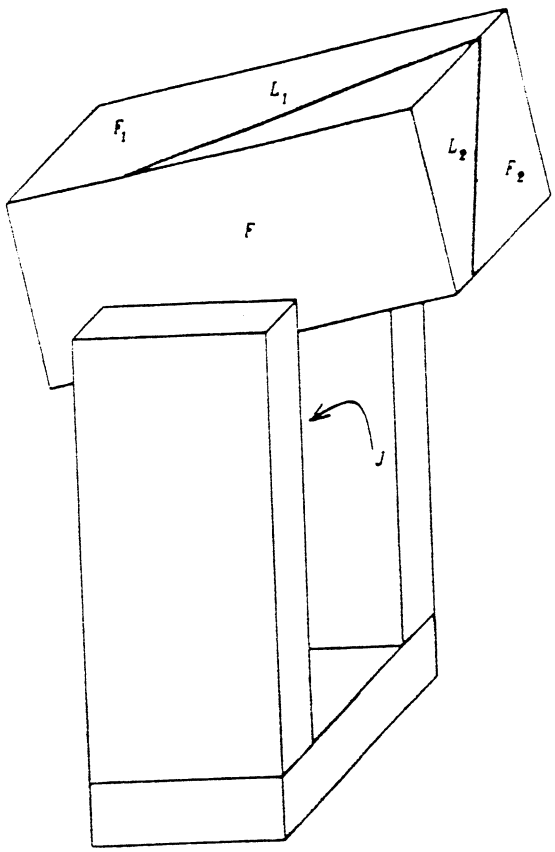


Figure 4.34: Deux faces sont utilisées pour localiser l'objet

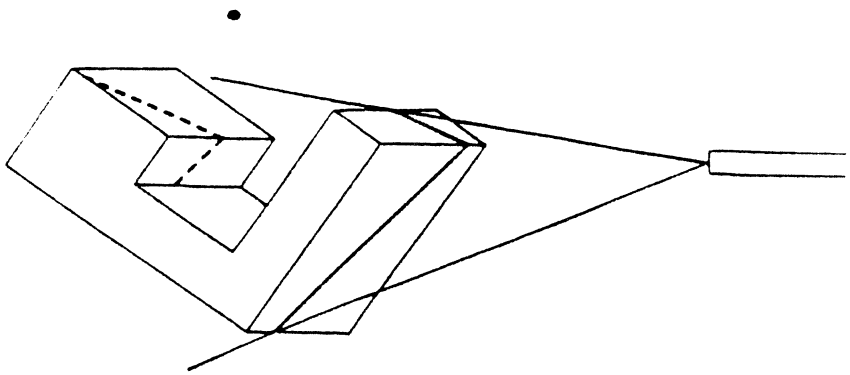


Figure 4.35: L'objet lui-même occulte le plan laser

4.8.3 Sélection des faces canditates

Une fois la pièce dans la pince, il nous faut sélectionner une paire de faces (F_1, F_2) susceptible d'être utilisée pour la localisation. La localisation n'est possible que si ce couple vérifie les quatres conditions suivantes:

1. Le vecteur parallèle à l'arête définie par F_1, F_2 ne doit pas être perpendiculaire à la normale du plan définie par F (voir figure 4.34), si cela est le cas le système est sous contraint.
2. Le plan laser doit intersecter les deux faces F_1 et F_2.
3. L'intersection avec le plan laser doit être dans le champ de la caméra.
4. La configuration qui satisfait les contraintes de 1 à 3 doit être atteignable par le robot.

Pour choisir une paire de faces appropriée nous considérons successivement toutes les arêtes du modèle. Pour chaque arête E nous utilisons les conditions précédentes comme filtres successifs pour les faces F_1, F_2 définissant l'arête E.

Si nous ne considérons que l'objet et si nous choisissons F_1 et F_2 appartenant l'enveloppe convexe de l'objet, nous obtenons une condition suffisante pour la condition 2. En utilisant cette condition on évite en particulier le type de configuration présentée figure 4.35.

Une fois la paire de faces selectionnée, nous positionnons l'objet. En utilisant (E, F_1, F_2) nous calculons: m le point milieu de l'arête E et un vecteur V bissecteur de l'angle formé par les deux normales aux faces F_1 et F_2. Nous définissons les constantes suivantes (voir figure 4.36):

- L'axe optique de la caméra: OA.
- Le point C de OA où OA intersecte le plan laser.

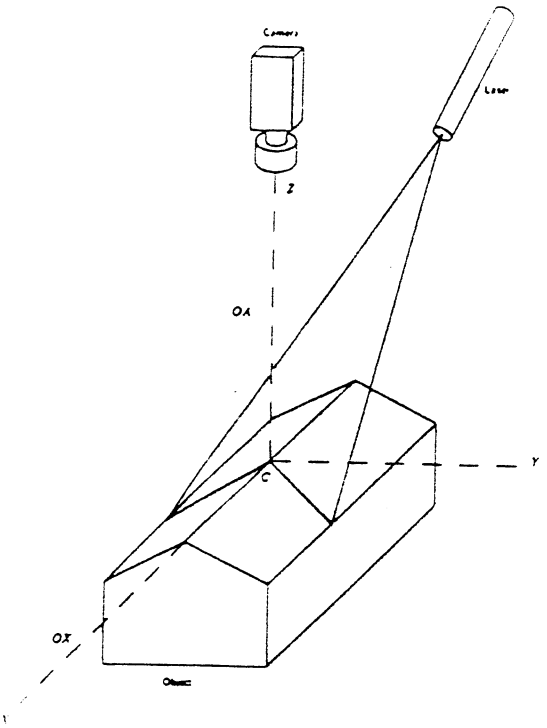


Figure 4.36: Determination de la position de l'objet sous le capteur

- Le plan XZ contenant OA et perpendiculaire au plan laser.
- La droite OX passant par C, perpendiculaire à OA et appartenant à XZ.

Ces constantes sont determinées aproximativement au moment du calibrage. En utilisant la position *approximative* de la pièce dans la pince nous utilisons l'interface géométrique pour calculer la positon de l'objet telle que:

- *le vecteur V est aligné avec l'axe OA*
- *le point m coïncide avec le point C*
- *L'arête E est alignée avec l'axe OX*

Si la position de l'objet dans la pince n'est pas trop différente de la position estimée, et s'il existe une solution à la transformation de coordonnées inverse pour cette position alors, il est possible d'amener l'objet dans une position telle que deux de ses faces intersectent le plan laser.

Il reste cependant à tester: (a) que le trait laser atteint les deux faces et (b) que l'intersection est visible depuis la caméra. Pour se faire, nous retirons l'objet du

modèle du robot et considérons deux parallélépipèdes rectangles associés respectivement à la caméra et à la pince (figure 4.37). Ces parallélépipèdes sont construits de sorte qu'un objet dans le champ de la caméra ou dans le champ du laser intersecterait l'un ou l'autre des parallélépipèdes. Il nous suffit donc de faire un test de collision entre ces deux parallélépipèdes est les autres polyèdres de la scène pour savoir si les traits laser seront visibles ou non.

Arrivé à ce point, toutes les conditions sont remplies pour présenter la pièce au système laser et obtenir une meilleure estimation de la position de l'objet dans la pince.

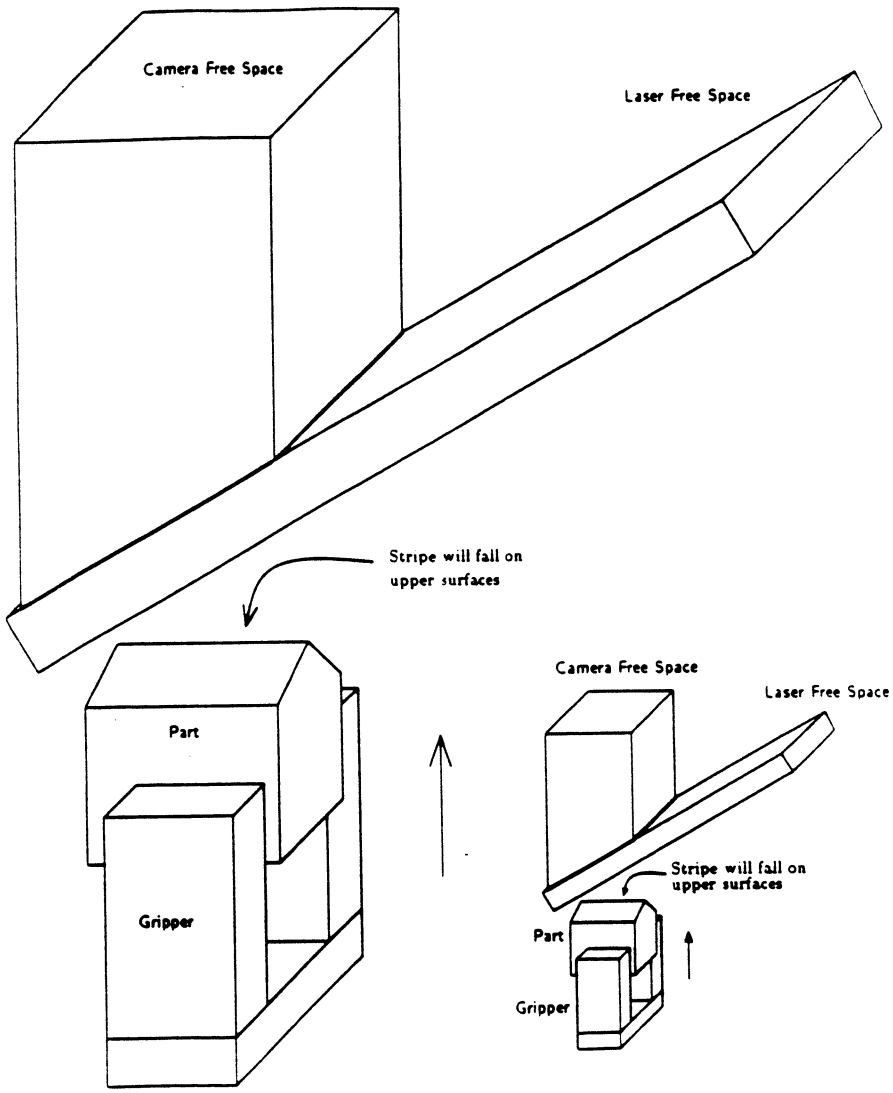


Figure 4.37: Test de la visibilité des traits laser

Chapitre 5

EXPERIMENTATION

L'expérimentation du système Handey a été une expérience très intéressante. Dans Handey, la planification des mouvements se fait dans des espaces non-intuitifs tels que l'espace des configurations ou le plan de préhension. Les mouvements résultant de cette planification peuvent être très surprenants, même pour les concepteurs. Nous pensons que cette "indépendance" du système fait son attrait et d'une certaine façon montre sa complexité. L'expérimentation du système n'a pas seulement été passionnante, elle nous a aussi aidé à obtenir une meilleure compréhension des problèmes reliés à la manipulation et nous a suggéré de nombreuses améliorations.

La plupart des expérimentations ont consisté à essayer d'empiler deux blocs ayant la forme d'un L, avec un petit nombre d'obstacles (cinq). Ces deux pièces ont été utilisées pour tester le système durant son implémentation. Toutefois, une des démonstrations la plus impressionnantes des possibilités du système a été réalisée avec les pièces données en exemple dans ce document.

Après avoir construit le modèle des pièces formant l'assemblage et décrit les obstacles de l'environnement, la pièce A était placée dans le champ de vision du système de localisation. 15 minutes plus tard un plan de manipulation était engendré et était exécuté sans erreur par le robot.

Bien qu'il faille reconnaître la part de chance dans cet exemple particulier, cela montre aussi l'utilité d'un tel planificateur: si l'on donne une nouvelle série de pièces

et un modèle particulier de l'environnement, il n'est pas nécessaire de reprogrammer le système. Le planificateur sera automatiquement capable d'exécuter les opérations nécessaires afin d'atteindre l'objectif donné. Bien que 15 minutes paraîssent être un temps assez long pour planifier une tâche aussi simple, c'est un temps probablement relativement court comparé au temps nécessaire à l'écriture d'un programme de robot de niveau effecteur effectuant la même tâche. Dans la suite de ce chapitre, nous considèrerons uniquement les expérimentations effectuées avec les deux pièces en forme de L.

5.1 Les faits

5.1.1 Le temps d'exécution

Un des points les plus importants pour avoir de bonnes conditions expérimentales est la mise en place et le fonctionnement rapide du système. Cette condition n'est pas atteinte à l'heure actuelle: la mise en route et le calibrage du système demandent plus d'une heure.

Le temps nécessaire pour planifier une opération Prendre-et-Poser dépend en grande partie du nombre de re-saisies nécessaires à la réalisation de l'assemblage. Un découpage typique de la planification d'une manipulaiton sans re-saisie peut être:

balayage laser et	
construction de la carte tri-dimensionnelle:	1mn 30s
localisation:	2mn
planification de trajectoire	1mn
planification de la saisie	2mn
planification de trajectoire	1mn
temps total de planification	7mn 30s

L'introduction de la re-saisie augmente sensiblement le temps de planification car le planificateur de saisie doit être utilisé deux fois et il faut plusieurs appels au planificateur de trajectoire. De plus ce dernier planificateur tend à prendre plus de temps pour planifier des trajectoires au voisinage immédiat des obstacles (cas de la re-saisie). Le temps caractéristique nécessaire pour une manipulation incluant une seule opération de re-saisie est de:

balayage laser et	
construction de la carte tri-dimensionnelle:	1mn 30s
localisation	2mn
planification de trajectoire	1mn
planification de saisie	2mn
planification de saisie	2mn
planification de re-saisie	2mn
planification de trajectoire	1mn
planification de trajectoire	1mn 30s

planification de trajectoire	1mn
temps total de planification	15 mn

Le temps nécessaire à l'exécution réelle de la tâche dépend de la vitesse de fonctionnement du robot. Du fait que les trajectoires ne sont pas du tout planifiées de manière optimum (aussi bien du point de vue temps que distance), le temps d'exécution sera en général légèrement supérieur à celui obtenu avec un programme de manipulation conventionnel.

5.1.2 Succès et Echecs

Les tâches demandées au sytème peuvent être décomposées en trois catégories:

1. localisation de la pièce,
2. planification du mouvement,
3. exécution du mouvement.

A partir du moment où la première pièce en forme de L a été localisée, le système ne connaît jamais d'échec lors de la planification d'un mouvement qui, en théorie permettra d'empiler une pièce sur l'autre. Autrement dit, il trouve toujours un moyen de saisir la pièce, la re-saisir si nécessaire, et de déplacer le robot entre les étapes intermédiaires de l'opération. Ce succès "logique" tend à conforter notre opinion sur l'espace de recherche associé aux opérations de manipulation des robots: il existe beaucoup de solutions, et il n'est pas nécessaire de considérer l'espace de recherche entier pour trouver une solution, donc on peut appliquer des méthodes simples afin de résoudre le problème. Toutefois, il est plus difficile d'arriver à ce que le système expérimental réalise les trois sous-tâches avec succès, notamment si l'on utilise la stratégie de re-saisie.

Si l'on utilise une simple opération Prendre-et-Poser (sans re-saisie), la plupart des échecs seront provoqués par l'inaptitude à localiser correctement la pièce dans la V-zône. Effectuer une opération Prendre-et-Poser est une opération très contraignante et la plupart du temps, il faut avoir recours à une stratégie de re-saisie. Dans ce cas, la proportion d'échecs est beaucoup plus grande, rendant le pourcentage total d'échecs très élevé. La différence de performance entre la planification avec ou sans re-saisie s'explique par la nature de l'opération visant à poser la pièce sur la table. Cette opération est planifiée de façon purement géométrique et s'agissant d'un assemblage, il n'est pas étonnant qu'elle conduise à de nombreux échecs.

Avec une opération de re-saisie, le système arrivera à effectuer complètement la tâche (étape (1), (2) et (3)) une fois tous les quatorze essais. La reconnaissance de forme est responsable de vingt pour cent des échecs, et dès lors que l'étape (2) produit toujours un plan, quatre-vingt pour cent des échecs se produisent durant l'exécution du mouvement. Puisque le planificateur est responsable de la production

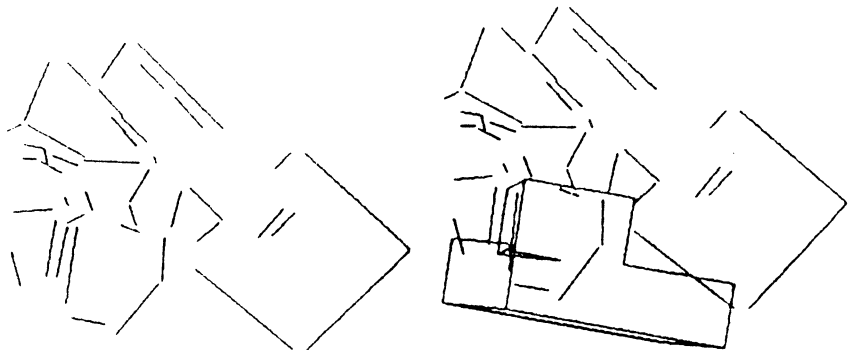


Figure 5.1: Mise en correspondance incorrecte entre les arêtes de la carte tri-dimensionnelle et du modèle

d'un plan correct, on peut considérer qu'il est responsable de quatre-vingt pour cent des échecs. Cependant, nous considérons que le planificateur réussit s'il produit un plan qui peut être correctement exécuté sur un simulateur CAO. Nous attribuons les échecs durant l'exécution du mouvement aux différences entre l'environnement réel et le modèle utilisé pour sa représentation. Par exemple:

- La pièce devant être saisie ne se trouve pas au même emplacement dans l'environnement et dans le modèle (erreur de vision),
- Le modèles du robot ou des obstacles ne représentent pas exactement la réalité ou ne sont pas aux mêmes emplacements,
- le comportement physique des pièces n'est pas modélisé, par exemple, le glissement de la pièce dans les mors de la pince.

5.2 Echecs

Dans cette section, nous présentons les échecs rencontrés durant l'expérimentation de Handey et nous essayons aussi d'apporter quelques éléments d'explication de ces échecs ainsi que quelques améliorations possibles.

5.2.1 Reconnaissance de forme

Le module de reconnaissance de forme échoue lorsqu'il effectue une mise en correspondance incorrecte entre les arêtes du modèle et les arêtes de la scène (voir figure 5.1).

La raison fondamentale de cette erreur est le manque de précision lors de la reconnaissance des arêtes dans la carte tri-dimensionnelle. Afin de compenser ce manque de précision, le programme de vérification fait de larges approximations pour définir si une arête confirme une hypothèse. En conséquence, une arête donnée peut être utilisée pour confirmer une hypothèse, même si elle ne fait pas partie du modèle.

Cette tendance est renforcée par la présence d'arêtes aléatoires qui peuvent être considérées comme faisant partie de la pièce.
Il existe deux possibilités d'amélioration de la fiabilité du module de reconnaissance de forme.

- Définir un meilleur détecteur d'arêtes pour les données tri-dimensionnelles,
- Utiliser chaque arête valide pour re-calculer la position de la pièce lors de la vérification d'hypothèse. Cela permet d'améliorer la précision sur l'hypothèse de la position de la pièce et de réduire la dimension du parallélépipède utilisé pour trouver les arêtes à confirmer.

5.2.2 La saisie

Le système n'a jamais manqué de trouver une manière de saisir la pièce, même dans un environnement très encombré d'obstacles. C'est la raison pour laquelle nous pensons que la méthode utilisée pour planifier la saisie ainsi que la trajectoire de saisie est relativement fiable et bien adaptée aux pinces à mors parallèles. Nous avons toutefois connu les incidents suivants lors de l'exécution de plans de saisie:

- *Erreurs:*

  1. La pince entre en collision avec un obstacle de la V-zone lors de l'approche rendant l'approche de la pièce impossible.
  2. La pince ne se déplace pas dans le plan de saisie réel et ne se ferme pas sur la pièce.
  3. Le robot change sa configuration au cours du mouvement d'éloignement, la dernière articulation faisant un demi tour complet, provoquant ainsi une collision avec l'environnement.
  4. La saisie trouvée par le planificateur n'est pas suffisament stable et la pièce tombe de la pince.

- *Les causes:*

  1. La calibration entre le robot et le capteur tri-dimensionnel est la seule explication valable. Nous nous sommes aperçus que recalibrer de temps à autre les systèmes de coordonnées permet d'obtenir de meilleurs résultats. Pour l'instant, les causes de ce dé-calibrage n'ont pas été déterminées.
  2. Le calibrage et/ou la reconnaissance peuvent être considérés comme les causes possibles de cette erreur.
  3. Le mouvement de dégagement est le seul mouvement "heuristique" utilisé dans Handey. Une fois que la pince est fermée, le robot se déplace verticalement de 15 cm au-dessus de sa position actuelle. Nous exécutons les opérations suivantes afin de calculer la position finale du robot:

- La position cartésienne du poignet est calculée,
- un décalage vertical est ajouté à cette position et les coordonnées articulaires correspondant à cette position sont calculées. En général, plusieurs ensembles de valeurs pour les coordonnées articulaires correspondent à cette position cartésienne.
- La solution adaptée est celle considérant un déplacement minimal (cette heuristique est utilisée pour sélectionner la même configuration que la position initiale).

Il se peut que toutes les solutions trouvées nécessitent une reconfiguration du robot, provoquant une trajectoire imprévisible des dernières articulations. Il faut noter que la reconfiguration n'est pas la seule source d'erreurs au cours de ce mouvement, un mouvement vertical parfait pouvant aussi conduire à une collision.

4. Le choix du point de préhension ne tient pas compte de la stabilité de la prise. Il est basé uniquement sur l'accessibilité et suppose que la pièce est relativement petite par rapport à la pince. On suppose que la pince serrera assez fort pour pouvoir maintenir l'objet fermement. Cette dernière supposition n'est pas entièrement valable dans notre expérimentation. Il ne sera pas très difficile d'incorporer un critère de stabilité dans la planification de saisie.

5.3 Planification de trajectoire

La planification de trajectoire réussit la plupart du temps. Les problèmes principaux arrivent lors de la phase de re-saisie, lorsque le robot doit se déplacer près des obstacles. La figure 5.2 représente une vue de haut des positions initiale et finale du robot lors de la phase de re-saisie. Il est assez difficile de faire en sorte que le robot s'échappe de la position initiale car il n'est pas possible d'utiliser uniquement les articulations 2 et 3 pour l'éloigner de cet emplacement.

L'articulation 1 doit aussi être utilisée et l'on peut rencontrer des problèmes reliés à l'échantillonnage de l'espace libre. Si l'on suppose que la pièce est située à une distance de 150 cm de la base du robot, un pas d'échantillonnage de un degré sur la première articulation provoquera un mouvement d'environ 2.6 cm à l'extrémitié de la pince. L'erreur la plus courante résultant de cet échantillonnage peut être décrite de la manière suivante:

- Lors du processus d'échantillonnage, les articulations du robot ne sont pas grossies pour tenir compte du mouvement entre deux pas d'échantillonnage successifs. Il est donc possible qu'une position intermédiaire à deux valeurs échantillonnées libres soit interdite. Le programme de recherche de trajectoire fait l'hypothèse inverse et peut donc planifier des trajectoires incorrectes. On peut cependant remarquer que ce genre d'erreurs peut être détectée à l'aide d'un testeur de collision conventionnel.

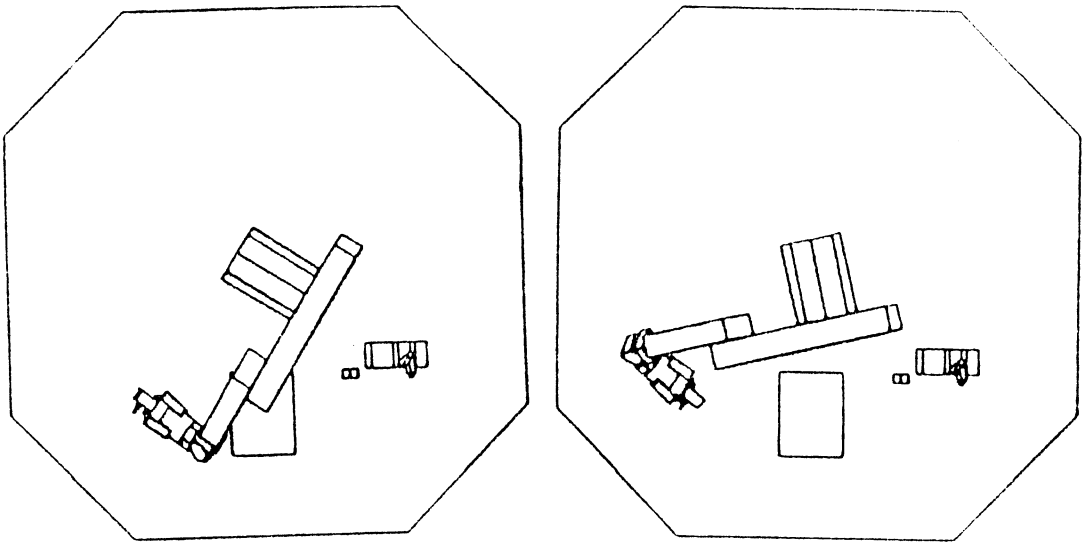


Figure 5.2: Configurations initiale et finale du robot lors d'une phase de re-saisie

5.4 Re-saisie

Le planificateur de re-saisie ne manque jamais de trouver une manipulation permettant de passer d'une prise donnée à une autre. Etant donné une prise initiale et finale, il trouve toujours une séquence de mouvements optimale en terme du nombre de déplacements intermédiaires. Il serait toutefois possible d'optimiser cette séquence en choisissant les prises initiale et finale de manière plus approprié (5.5.2).

Dans toute la durée d'exécution, la re-saisie est le seul moment au cours duquel a lieu un assemblage réel. En effet, poser une pièce sur une table peut être considéré comme un assemblage, étant donné que la pièce et la table rentrent en contact. Dans Handey, rien n'a encore été prévu au sujet de la planification de mouvements fins, et l'on peut s'attendre à ce que cet assemblage ne se fasse pas sans problèmes. En particulier, nous avons enregistré de nombreuses erreurs dues à une collision entre l'objet transporté et la table.

5.5 Les leçons

5.5.1 Mise au point du système expérimental

Utiliser un matériel d'expérimentation fiable est une condition nécessaire à une bonne recherche en robotique, mais ceci n'est pas une condition forcément facile à obtenir. Une étape importante visant à atteindre ce but est la mise au point du matériel et du logiciel. En particulier, l'obtention des valeurs exactes des constantes définissant le système est un point crucial dans l'interprétation des résultats de l'expérimentation.

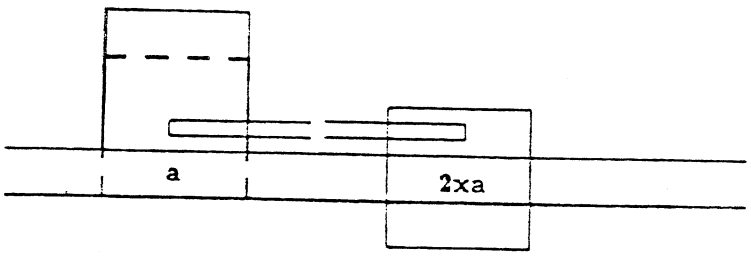


Figure 5.3: Un exemple de la propagation rapide des incertitudes

Lors des premières expérimentations faites avec Handey, nous avons supposé que la table était horizontale et à une hauteur donnée. Ceci était "approximativement" vrai mais, de nombreux échecs du système sont imputables à cette approximation. L'exemple suivante illustre ce qu'il se passe dans ce cas et démontre la vitesse à laquelle l'incertitude se propage dans les expériences de robotique.

La tâche consiste à ré-orienter un cube. La hauteur de la table a été mesurée avec une erreur a (voir figure 5.3), la position du cube est donnée par rapport à la table et on suppose que le robot se déplace sans erreur. La saisie est effectuée avec la même erreur a; la ré-orientation du cube va doubler l'erreur relative du cube par rapport à la table et de ce fait, provoquer une collision entre le cube et la table. Beaucoup d'erreurs de mesures vont se propager de la même manière faisant diverger rapidement le système de son modèle. En particulier, le succès de l'expérimentation dépend en grande partie de la précision de la reconnaisssance, qui elle même dépend d'un calibrage correct entre les systèmes de références du robot et de la vision. Nous pensons que de nombreux problèmes rencontrés lors de l'expérimentation de Handey pourraient être résolus par une étude systématique des problèmes de calibrage et par une représentation plus fidèle de l'environnement.

5.5.2 Optimisation

L'expérimentation a suggéré deux optimisations concernant l'architecture globale du système.

La planification de saisie

Dans la première version de Handey, la planification de la saisie était effectuée à l'aide de deux modules:

- Le premier module produisait des plans et des points de préhension possibles, ceci indépendemment des obstacles situés dans la V-zone.

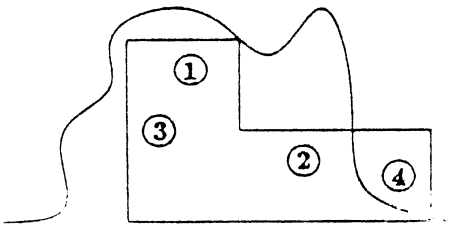


Figure 5.4: Points cibles successifs proposés par le planificateur de trajectoire

- Le deuxième module utilisait une pseudo méthode des potentiels pour guider la pince vers un point de saisie, lui même selectionné parmi l'ensemble défini précédemment.

Les choix des points et plans de saisie étant fait indépendemment des obstacles, il était nécessaire d'avoir recours à de nombreux cheminements arrières avant de pouvoir choisir une saisie adéquate. Dans l'exemple présenté en figure 5.4, le planificateur de saisie va engendrer les points de saisie 1,2,3,4 dans cet ordre. Ces points seront utilisés successivement par la méthode des potentiels comme points d'attraction et une trajectoire correcte ne sera trouvée que pour le dernier point. Dans la version actuelle du système, les obstacles de la V-zone sont utilisés afin de trouver directement un point cible possible dans une partie libre du plan de saisie. Cette stratégie limite le nombre d'appels au planificateur d'approche.

Re-saisie

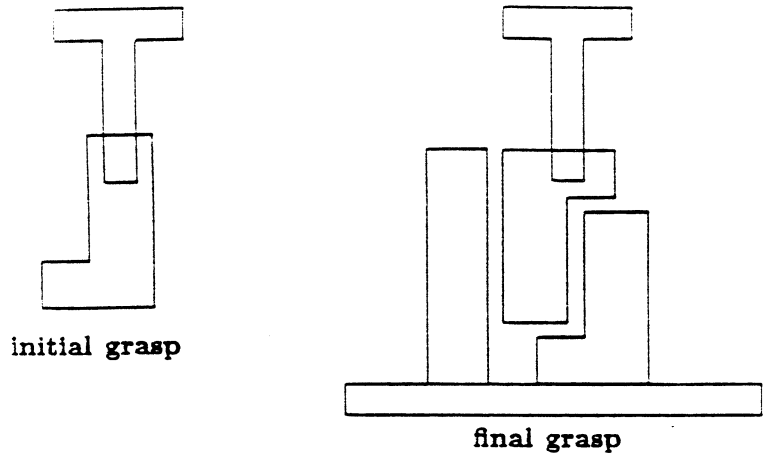
Il suffit habituellement d'une re-saisie à une seule phase pour réussir une opération Prendre-et-Poser. Dans le système actuel, on donne une prise initiale et finale au planificateur de re-saisie et il calcule le nombre optimal d'étapes afin de passer d'une prise à l'autre. Cette façon de procéder aboutit très souvent à une re-saisie en deux temps, alors qu'une stratégie différente aboutirait plus souvent une opération en une étape. Une version optimisée du système travaillerait de la manière suivante:

1. Un ensemble de prises finales possibles est donné comme paramètre planificateur de re-saisie.

2. Pour chacune de ces prises, il produit un ensemble de prises initiales possibles pouvant être atteintes avec une seule pose intermédiaire.

3. Les prises ainsi obtenues sont utilisées afin de contraindre le planificateur d'approche et de saisie, en rétro-projectant la table dans la V-zone.

Afin d'illuster ce processus, nous supposons qu'une prise est définie par une simple transformation entre la pièce et la pince. La figure 5.5 représente la stratégie actuelle: les prises initiale et finale sont données et entraînent une opération en deux temps. Considérons maintenant la figure 5.6. Elle représente toutes les prises



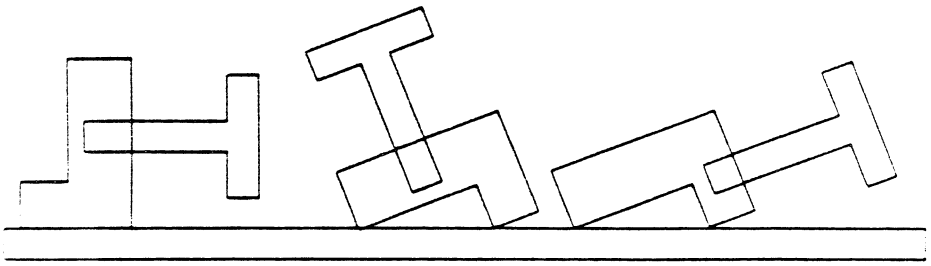


Figure 5.5: Lorsqu'on lui donne une position initiale et finale, le planificateur produit une stratégie de saise en deux étapes

finales possibles afin d'effectuer un assemblage donné. La figure 5.7 représente les prises initiales pouvant être effectuées en un temps à partir de la prise finale et la pose correspondante. La figure 5.8 montre comment cette pose est utilisée pour contraindre l'accessibilité lors de la planification de prise.

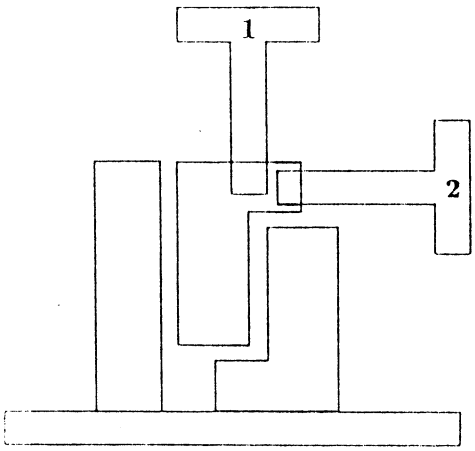
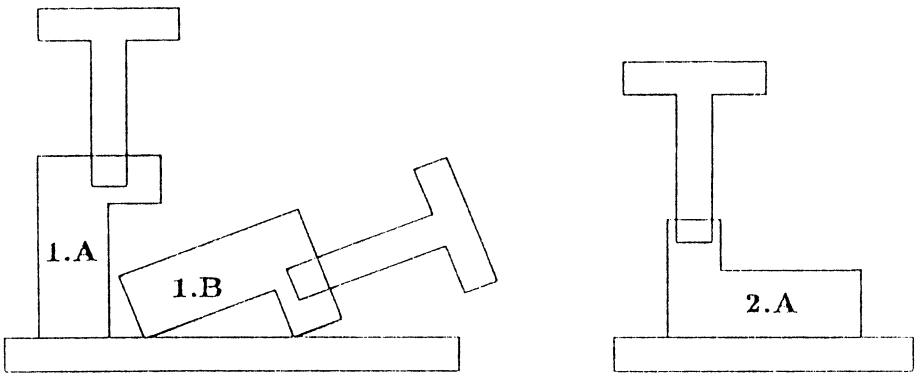


Figure 5.6: Prise finale possible, permettant d'assembler les deux pièces en forme de L



stable placements for grasps 1 and 2

Figure 5.7: Pose possible, compatible avec la prise finale

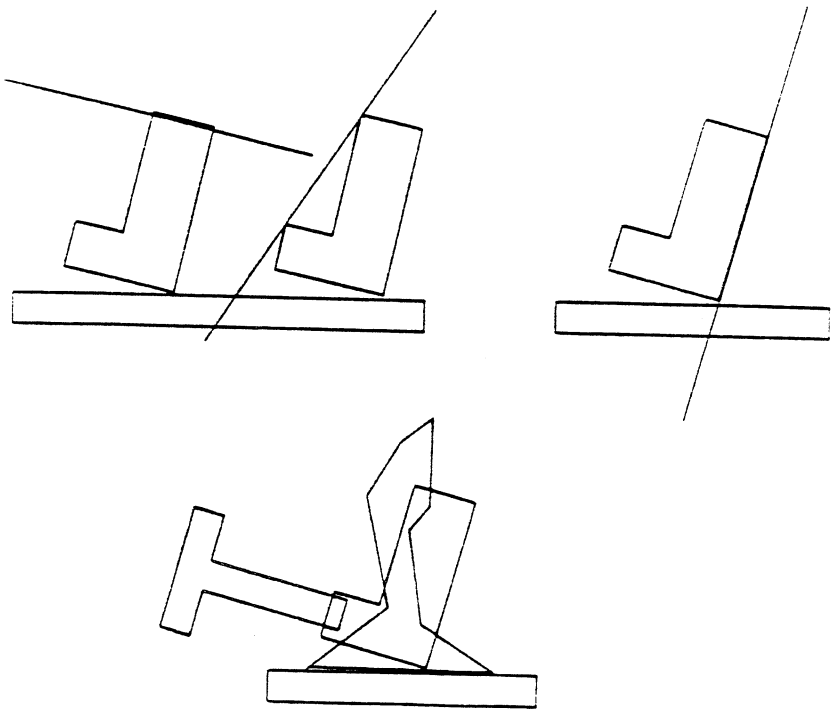


Figure 5.8: Les placements sont utilisés afin de contraindre le planificateur de trajectoire pour trouver une prise initiale telle qu'une opération de re-saisie en un temps soit nécessaire pour atteindre la prise finale

Chapitre 6

TRAVAUX FUTURS

6.1 Liens avec des planificateurs de haut niveau

Il serait difficile d'utiliser un planificateur basé sur la logique des prédicats tel que STRIPS [FN71] pour planifier le même type d'opérations que nous planifions avec Handey. En fait, les planificateurs basés sur la logique des prédicats ne conviennent pas bien à la planification de bas niveau des robots, car les actions des robots dépendent d'une manière très complexe et non discrète de leur environnement. Il n'est, en général pas possible d'utiliser des "pré" et "post" conditions pour des opérations de robots de bas niveau. Le succès d'une opération robotique donnée ne peut pas être défini par un ensemble de pré-conditions discrètes (voir chapitre 2) et il est fort possible qu'une action valide ne mène pas aux post-conditions prédites. D'autre part, il est impossible de penser qu'une tâche complexe puisse être planifiée à l'aide d'un planificateur tel que Handey. Handey est un planificateur géométrique et il est impossible qu'une tâche complexe puisse être planifiée au seul niveau de détail géométrique.

Une solution possible pour avoir des robots réalisant des tâches réellement complexes est la combinaison de planificateurs géométriques et logiques. Avant de les combiner, il est important de bien comprendre la compétence de chacun d'eux. Dans un moyen terme, notre stratégie sera d'étudier de simples interactions entre Handey et un planificateur basé sur la logique des prédicats responsable du découpage d'une opération Prendre-et-Poser de base. Nous supposerons, par exemple, que les actions

devant être effectuées par le robot peuvent être décomposées en des opérations délicates telles que l'insertion d'un nouveau module dans un satellite. La séquence d' opérations sera programmée par un opérateur. Un planificateur logique utilisant une représentation traditionnelle des conditions nécessaires à ces opérations calculera les manipulations élémentaires qu'Handey peut planifier.

6.2 Planification de mouvements fins

A l'heure actuelle, Handey n'a pas de planificateur de mouvements fins. En principe, on peut utiliser les techniques de planification de trajectoire afin de planifier tous les mouvements nécessaires à la réalisation d'un assemblage. Mais la présence d'incertitudes contraint à arrêter la planification des grands mouvements lorsque le robot et la pièce transportée sont près des obstacles.

La connexion d'un planificateur de mouvements fins au reste du système sera très similaire à la connexion d'un planificateur de saisie. Ceci n'est pas surprenant car la saisie peut être considérée comme une opération d'assemblage. Elles sont toutefois légèrement différentes car dans le cas d'un planificateur de saisie, l'interpréteur a quelque liberté en ce qui concerne la position relative de la pince par rapport à la pièce. La prise peut être stable sans que la position idéale soit atteinte. Le planificateur de mouvements fins sera considéré comme la fonction génératrice de l'ensemble des positions intermédiaires permettant de commencer la planification des mouvements d'assemblage. De la même manière que pour le planificateur de saisie, des contraintes artificielles peuvent être utilisées afin de forcer le choix de ces positions intermédiaires. Une position intermédiaire étant choisie, elle sera considérée comme la position finale par le système existant. S'il n'est pas possible de planifier une séquence de mouvements visant à l'atteindre, alors le planificateur de mouvements fins sera de nouveau activé afin de produire une nouvelle position intermédiaire.
Des études théoriques de planification de mouvements fins peuvent être trouvées dans [LMT84, Mas82, Erd84, Don87]. Ces études utilisent toutes le même cadre théorique basé sur les *pré-images* et les *cônes de friction.* Des améliorations du modèle d'incertitudes relatives aux mouvements ainsi qu'un planificateur de mouvements fins tri-dimensionnel (sans rotation) sont présentés dans [Buc87].
La difficulté majeure réside dans l'implémentation. La majorité de ces résultats a été démontrée dans un espace de faible dimension. Il peuvent en principe être généralisés à des espaces de dimensions plus importantes, mais les solutions demanderaient un temps de calcul trop important.

Il existe plusieurs alternatives à l'implémentation d'un planificateur de mouvements fins basé sur les *pré-images* et *cônes de friction.*

Planificateur basé sur l'utilisation de règles de production. On trouvera la description d'un tel planificateur dans [DL84]. L'idée est d'utiliser des situations géométriques comme parties gauche et droite pour les actions du robots. Dans ce cas, ces actions sont des mouvements gardés dans une direction. Le

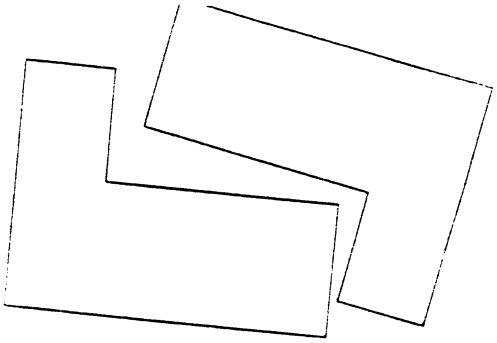


Figure 6.1: L'orientation correcte de la pièce B facilitera l'assemblage lorsque la pièce B sera lâchée

système a pour objectif d'atteindre une situation géométrique correpondant à l'assemblage final, il procède par chaînage arrière pour trouver les actions à effectuer. Apres chaque action, il tente d'évaluer la situation géométrique dans laquelle il se trouve, pour éventuellement modifier la séquence initiale. De plus, ce planificateur est couplé à un système d'apprentissage.

Planificateur de désassemblage. Un planificateur de désassemblage est présenté dans [Val85] et [LT86]. L'idée principale est d'utiliser la position relative finale des deux pièces afin de trouver un plan de démontage. Il suffit ensuite d'inverser ce plan pour obtenir un plan d'assemblage.

Planification assistée par gravitation. Le fait de lâcher une pièce au-dessus de sa position finale provoque souvent l'assemblage correct. Les chances de succès peuvent être grandement améliorées si on trouve un position correcte pour les deux pièces au moment du lâcher. Par exemple une rotation légère de la pièce B améliore les chances de réussite de la tâche (voir figure 6.1). L'objectif de ce planificateur est de calculer une position relative acceptable entre les deux pièces.

Cette dernière méthode sera utilisée pour complèter l'implémentation de Handey. Afin qu'elle soit réellement efficace, il faut disposer de deux robots afin de pouvoir positionner correctement les deux pièces.

6.3 Utilisation de deux robots et d'une main à trois doigts.

Un autre robot PUMA est actuellement ajouté au matériel d'expérimentation. Une main à trois doigts [SC82] sera installée sur ce nouveau robot. Les deux robots seront placés de manière qu'ils puissent partager un espace de travail commun et qu'ils puissent tous les deux atteindre la V-zone. L'utilisation de cette architecture rendra possible l'expérimentation de plusieurs algorithmes, notamment:

- La planification de trajectoire pour deux bras manipulateurs fonctionnant en étroite collaboration.

- La planification de l'approche et de la saisie pour la main à trois doigts.
- L'assemblage par gravité.
- La planification des mouvements fins utilisant les possibilités de mouvements compliants de la main à trois doigts.

6.4 Autres structures de contrôle.

Dans le chapitre 3, nous avons proposé une méthode générique, visant à la construction de planificateurs ayant à traiter des espaces de recherche associés à des problèmes géométriques. Nous avons illustré notre approche en construisant un planificateur spécial pour une opération de manipulation simple: Prendre-et-Poser. Les résultats de notre expérimentation montrent que ce planificateur réussit asez bien dans un environnement simple. On peut se poser deux questions intéressantes à propos de ce planificateur et de la méthodologie utilisée pour le concevoir:

1. *Handey se comportera-t-il bien dans un environnement plus complexe ?*
2. *Quel niveau de simplification est admissible dans la planification d'opérations de manipulation ?*

Il est assez difficile de répondre à la première question sans avoir testé le planificateur dans une scène complexe réelle. Nous pensons qu'Handey continuera à se comporter correctement dans un environnement substantiellement plus complexe. Nous sommes persuadés que l'ajout de nouveaux obstacles à la fois dans la V-zone et sur la table ne changera pas sa capacité à trouver des solutions et que modifier la forme des pièces n'affectera pas son comportement aussi longtemps que ces pièces sont des polyèdres. Nous basons notre confiance dans ce système sur différents arguments.

- Le planificateur de saisie a déjà démontré sa capacité à planifier des prises dans un environnement difficile,
- Le planificateur de re-saisie, bien que n'étant pas général, peut produire un plan pour la plupart des ré-orientations nécessaires à un assemblage quelconque,
- Il est possible d'imaginer des situations où le planificateur de trajectoire ne réussira pas alors qu'une trajectoire aurait pu être trouvée à l'aide de techniques plus générales. Ces scènes ne dépendent pas de la quantité d'obstacles, mais de leur répartition topologique. Par exemple, si l'assemblage doit être exécuté à travers une fenêtre, le planificateur de trajectoire de Handey ne trouvera probablement pas de solution car cet arrangement va créer deux régions non connexes dans l'espace des configurations. Nous sommes malgré tout convaincus que la plupart des assemblages faits sur une table peuvent être effectués à l'aide de notre planificateur de trajectoire.

Il nous faut néanmoins expérimenter Handey dans des situations plus complexes avant de confirmer ou d'infirmer nos hypothèses.

L'éventail de complexité d'un planificateur Prendre-et-Poser peut varier d'un planificateur très évolué qui tiendrait compte de toutes les contraintes de la tâche pour produire un plan, à une simple fonction génératrice qui engendrerait des trajectoires avant de les tester à l'aide d'un prédicat Prendre-et-Poser. Handey se situe quelque part dans cet éventail et à partir de là, plusieurs directions sont possibles:

En amont dans la complexité: Le fait d'avoir un planificateur réellement intelligent est certainement un but très enviable. On peut souligner que ce n'est pas un but impossible à atteindre en ce qui concerne l'aspect géométrique de la planification. D'autres aspects, tels que la gestion des incertitudes ou l'interaction avec des données sensorielles sont des objectifs à plus long terme. Il est nécessaire d'obtenir des résultats fondamentaux dans ces domaines afin de réussir à concevoir un "planificateur complet".

En aval dans la complexité: Nos hypothèses concernant le comportement de Handey dans un environnement plus complexe semblent suggérer que Handey peut être sur-dimensionné par rapport à la complexité du type d'assemblage pour lequel il a été initialement conçu. Nous pensons en fait qu'il est possible de construire une version simplifiée de Handey moins générale mais qui continuerait à travailler dans le même type d'environnement dans lequel Handey travaille actuellement. Cette version simplifiée de Handey adopterait la même structure de contrôle, mais utiliserait une représentation de la tâche moins sophistiquée. Nous décrivons une version simplifiée de Handey dans la section suivante.

6.5 Une version simplifiée de Handey

Nous présentons ci-dessous un scénario possible pour l'implémentation d'un planificateur Prendre-et-Poser plus simple.

1. Une fois que la pièce aura été localisée, le système générera des plans de préhension potentiels, utilisant des paires de faces. Il les triera en rangeant le plan de préhension le plus vertical en premier.

2. Etant donné un plan de préhension, il utilisera la projection du centre de gravité du plan de préhension comme cible pour le point de préhension sur la pince.

3. En utilisant cette cible, il calculera un mouvement vertical d'une longueur fixe en direction de la pièce, et testera ce mouvement pour les collisions et les solutions au problème de changement de coordonnées inverses. Si cette trajectoire s'avère impossible, alors un nouveau plan de préhension est sélectionné (étape 1).

4. En placant le modèle de la pièce à sa destination, il planifiera une trajectoire et une prise utilisant les étapes (1), (2) et (3)

5. Il effectuera systématiquement une opération de re-saisie:

   - Une position unique sur la table est assignée à chaque face de l'enveloppe convexe,
   - Nous appelerons L_i et L_f l'ensemble de faces pouvant être placées sur la table avec respectivement la prise initiale et finale
   - Si $L_f \cap L_i = \Phi$, alors on change de couple de prises, sinon on choisit une face de $L_f \cap L_i$ pour effectuer la re-saisie.

6. Il planifiera tous les mouvements intermédiaires utilisant une représentation sphérique des obstacles (en faisant usage de plusieurs sphères pour chaque articulation du bras) et une sphère pour les trois dernières articulations.

Il est très difficile de prédire le comportement d'un Handey simplifié mais il se pourrait qu'il agisse de manière tout à fait satisfaisante au cours de simples assemblages faits sur une table.

Chapitre 7

CONCLUSION

7.1 Résumé

Nous avons formulé trois hypothèses sur la structure d'un système de programmation de robot.

1. Il n'est pas possible d'écrire des primitives générales pour de simples opérations de robots.

2. Il est possible de décomposer un planificateur de mouvements en plusieurs planificateurs spécifiques.

3. Il existe de nombreuses solutions potentielles au problème de la planification d'opérations de manipulation simple, et il est donc possible de construire des planificateurs simples qui, n'étant pas généraux, atteindront la plupart du temps leur objectifs.

Dans le chapitre 2, nous avons justifié la première hypothèse. Dans le chapitre 3, nous avons proposé une méthodologie au problème de décomposition de la planification de mouvements, nous avons illustré cette méthodologie en décomposant la planification d'une opération Prendre-et-Poser en plusieurs planificateurs simples: cette décomposition et son expérimentation supportent notre seconde hypothèse. Dans le chapitre 4, nous avons donné les outils algorithmiques permettant la construction de

ces planificateurs. Enfin, les expérimentations faites avec Handey (chapitre 5) tendent à confirmer notre troisième hypothèse: en dépit de nombreuses simplifications, Handey réussit la plupart du temps à planifier une opération Prendre-et-Poser.

7.2 Ce que nous avons appris

- Handey fonctionne car il utilise des raisonnements géométriques génériques bien établis. Ces raisonnements ne permettent pas de résoudre tous les problèmes géométriques reliés aux opérations de manipulation, mais les conditions de leur applicabilité sont bien définies et Handey est bâti sur ces fondements solides.

- La structure de contrôle définie pour Handey paraît être un moyen approprié de combiner des planificateurs. On peut considérer cette structure de contrôle comme efficace pour deux raisons:

  1. Chaque planificateur est en fait capable de raisonner dans le domaine d'intérêt (la géométrie). Bien qu'ils ne soient pas généraux, ils opèrent sur une description de l'environnement pour produire des solutions.
  2. Il existe un mécanisme simple qui permet aux planificateurs de s'accorder entre eux lorsque des problèmes globaux se présentent.

Nous pensons que ces leçons sont importantes pour d'autres problèmes robotiques non résolus, tels que la planification de mouvements fins. Nous croyons en particulier qu'il est important de débuter avec une théorie générale solide avant de tenter de construire un planificateur de mouvements fins. Il sera toujours possible d'apporter des simplifications à cette théorie pour faire de nouvelles intégrations, et ce faisant, faire un pas supplémentaire vers la construction de *vrais robots.*

Annexe A

CALCULS GÉOMÉTRIQUES

A.1 Résolution des équations de l'interface géométrique

Notre algorithme procède en trois étapes:

A.1.1 Recherche d'une direction commune

Dans Handey nous n'avons à considérer que des relations géométriques ayant la propriété suivante:

Si l'ensemble des relations géométriques définit complètement la position de l'objet A par rapport à l'objet B, alors il est possible de trouver deux vecteurs unitaires U_A, U_B *associés respectivement à A et B tel que* $U_A = U_B$.

Par exemple si l'ensemble des relations géométriques inclut la relation suivante:

la droite $[(a_0, a)]F_A$ *est alignée avec la droite* $[(b_0, b)]F_B$

alors $U_A = a, U_B = b$ ou $U_A = a, U_B = -b$

de même, si nous avons la relation géométrique suivante:

le plan $[(\sigma_A, a)]F_A$ *est contre le plan* $[(\sigma_B, b)]F_B$

alors $U_A = a, U_B = -b$

Cette relation est aussi vraie lorsque l'on cherche à résoudre les équations provenant des contacts de types A,B et C entre une articulation et un obstacle. Pour les articulations linéaires la direction commune est simplement l'axe de translation et pour les articulations rotoïdes l'axe de rotation.
Dans certains cas (*une droite est alignéee avec une droite*) on obtient deux valeurs possibles pour U_B: $(U_B = b, U_B = -b)$. Dans ce cas on applique le reste de l'algorithme pour les deux valeurs de U_B. L'ensemble des solutions est alors l'union des solutions obtenues avec les deux valeurs de U_B.

A.1.2 Changement de repère de référence

On choisit deux nouveaux repères de références pour A et B: F_A^1 et F_A^2. Ces deux repères sont tels que les axes "Z" des repères F_A^1 et F_B^1 sont respectivent égaux à U_A et U_B.

Si oz_A est l'axe Z du repère F_A alors le repère F_A^1 peut être déduit de F_A par une rotation r_A définie comme:

$$\begin{aligned} &si\ oz_A = U_A\ alors\ r_A = Rot(ox, 0) \\ &\qquad sinon\ si\ oz_= - U_A\ alors\ r_A = Rot(ox, \pi) \\ &\qquad\qquad sinon\ r_A = Rot(U_A + oz, \pi) \end{aligned}$$

Les entités géométriques sont exprimées dans F_A^1 en utilisant les formules données dans la section (4.3).

La même opération est conduite avec le modèle B.
Après cette étape les nouveaux repères de référence de A et B ne diffèrent que d'une translation, et d'une rotation autour de U_A ou du nouvel axe Z du repère F_A^1.

A.1.3 Résolution d'un système pseudo-linéaire

Soient x, y, z les coordonnées du vecteur translation de F_A^1 à $F_{A'}^1$ dans F_A^1, et r la rotation de $F_{A'}^1$ à F_B^1 dans $F_{A'}^1$. Il est possible de représenter r de la manière suivante:

$$r = \begin{pmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$c^2 + s^2 = 1$$

En utilisant cette représentation, il est facile de montrer que chaque équation vectorielle peut s'écrire sous la forme d'une ou trois équations linéaires en x, y, z, c, s.

Donc, pour chaque ensemble de relations géométriques, on obtient un ensemble d'équations linéaires et une équation non-linéaire: $c^2 + s^2 = 1$.

$$\left(\quad K \quad \right) \star \left| \begin{matrix} x \\ y \\ z \\ c \\ s \end{matrix} \right| + \left| \; L \; \right| = \left| \; 0 \; \right|$$

$$c^2 + s^2 = 1$$

Résolution du système d'équations linéaires

Pour résoudre ce système d'équations on utilise une méthode de Gauss classique. Cette méthode permet d'obtenir directement le rang de la matrice K, et, de tester si le système est cohérent.

Utilisation de la contrainte $c^2 + s^2 = 1$

Si le système d'équations est incohérent, alors le système de relations géométriques est aussi incohérent. Sinon il existe trois possibilités:

1. Rang(K) = 5: Il existe une solution unique en x, y, z, c, s pour le système d'équations linéaires. Si $c^2 + s^2 = 1$, alors il y a une solution sinon il n'y a pas de solution.

2. Rang(K) = 4: Après la quatrième étape de la méthode de Gauss il est possible écrire:

   $$\alpha c + \beta s = \gamma$$

   α et β ne pouvant être simultanément nuls.

   En combinant cette équation avec $c^2 + s^2 = 1$ on obtient une équation du second degré en c ou s qui peut avoir une, deux ou aucune solution.

3. Rang(K) < 3: La position relative des deux objets ne peut pas être déterminée de manière unique avec le système de relations donné.

Soit x_0, y_0, z_0, c_0, s_0 une solution du système d'équations pseudo-linéaires, la position de A par rapport à B est donnée par la transformation suivante:

$$r_A \star translation(x_0, y_0, z_0) \star rotation_z(c_0, s_0) \star r_b$$

A.1.4 Résolution des équations provenant des contacts de type A,B ou C

La méthode utilisée pour résoudre les équations provenant des contacts de type A,B ou C est similaire à la méthode précédemment décrite.

Changement de repère

Les entités géométriques des obstacles sont exprimées dans un repère ayant l'axe Z parallèle à l'axe de translation ou de rotation de l'articulation.
Pour les articulations prismatiques, les entités géométriques de l'articulation sont exprimées dans un repère ayant son axe Z parallèle à l'axe de translation. On peut simplifier légèrement les calculs nécessaires à la commande de l'articulation si l'on choisit l'origine de ce repère au zéro "mécanique" de l'articulation.
Pour les articulations rotoïdes les entités géométriques sont exprimées dans un repère ayant l'axe Z parallèle à l'axe de rotation. Il est aussi pratique de choisir l'axe X de ce repère aligné avec le zéro mécanique de l'articulation.

Résolution des équations

Articulations prismatiques Dans le cas d'une articulation prismatique, les contacts de type A,B ou C ne donnent lieu qu'à une seule équation en z.

Type A : *le plan* $[(\sigma, a)](obstacle)$ *contient le point* $[b_0](articulation)$
avec $t = (0, 0, z)$ et $r = I$ (I: rotation identité). Nous obtenons l'équation suivante:

$$a_z z = \sigma - (b_{0x}a_x + b_{0y}a_y + b_{0z}a_z)$$

Type B : *le plan* $[(\sigma, a)](articulation)$ *contient le point* $[b_0](obstacle)$
Correspond à la même équation que celle obtenue pour les contacts de type A, toutefois le signe du résultat doit être inversé pour obtenir le débattement du degré de liberté par rapport aux obstacles et non pas l'inverse.

$$a_z z = (b_{0x}a_x + b_{0y}a_y + b_{0z}a_z) - \sigma$$

type C : *la droite* $[(a_0, a)](articulation)$ *intersecte la droite* $[(b_0, b)](obstacle)$

$$(r \star b_0) \cdot a + (a_0 - t \times a) \cdot (r \star b) = 0$$

$$a \times (r \star b) \neq 0$$

avec $t = (0, 0, z)$ et $r = I$ (I: rotation identité). Nous obtenons l'équation suivante:

$$a \times b \neq 0$$

Si $a \times b = 0$ alors il n'y a pas de solution sinon la solution est donnée par:

$$(b_x a_x - a_y by)z = (a_{0x}b_x + a_{0y}b_y + a_{0z}b_z) + (b_{0x}a_x + b_{0y}a_y + b_{0z}a_z)$$

Articulation rotoïde Dans le cas d'une rotation les contacts de type A,B ou C donnent lieu à une seule équation linéaire en c et s la contrainte $c^2 + s^2 = 1$ doit être utilisée pour trouver une solution.

Type A : *le plan* $[(\sigma,a)](obstacle)$ *contient le point* $[b_0](articulation)$

$$\sigma - t \cdot a = (r \star b_0) \cdot a$$

Avec $t = (0,0,0)$ et $r = rotation_z(c,s)$. On obtient l'équation suivante:

$$(a_x b_{0x} + a_y b_{0y})c + (a_y b_{0x} - a_x - b_{0y})s = \sigma - a_z b_{0z}$$

Type B : *le plan* $[(\sigma,a)](articulation)$ *contient le point* $[b_0](obstacle)$
donne la même équation que l'équation de type A, cependant le signe de l'angle obtenu doit etre inversé, de façon à obtenir le débattement correct.

type C : *la droite* $[(a_0,a)](articulation)$ *intersecte la droite* $[(b_0,b)](obstacle)$

$$(r \star b_0) \cdot a + (a_0 - t \times a) \cdot (r \star b) = 0$$

$$a \times (r \star b) \neq 0$$

Avec $t = (0,0,0)$ et $r = rotation_z(c,s)$. On obtient l'équation suivante:

$(a_x b_{0x} + a_y b_{0y} + a_{0x} b_x + a_{0y} b_y)c +$
$(a_y b_{0x} - a_x - b_{0y} + a_{0y} b_x - a_{0x\_} b_y)s +$
$+a_{0z} b_z + a_z b_{0z} = 0$

Si il existe une solution alors elle doit aussi vérifier la condition:

$$a \times (r \star b) \neq 0$$

A.2 Intersection de deux droites

A.2.1 Conditions nécéssaires

- Soient $C = (c_0, c)$ et $D = (d_0, d)$ les équations de deux droites, si C et D s'intersectent, alors:

$$c \times d \neq 0 \qquad \text{(A.1)}$$

- Les points $c \times c_0$ et $d \times d_0$ appartiennent respectivement à C et à D. Si C intersecte D, le vecteur $(c \times c_0\text{-}d \times d_0)$ appartient au plan défini par C et D, on a donc:

$$\times c_0 - d \times d_0) \cdot (c \times d) = 0$$

$\Longleftrightarrow$

$$((c \times c_0) \cdot (c \times d) - (d \times d_0) \cdot (c \times d) = 0 \qquad \text{(A.2)}$$

$\Longleftrightarrow$

$$((c \cdot c)(c_0 \cdot d) - (c_0 \cdot c)(c \cdot d)) - ((d \cdot c)(d_0 \cdot d) - (d_0 \cdot c)(d \cdot d)) = 0 \qquad \text{(A.3)}$$

D'autre part $c \cdot c = d \cdot d = 1$ et $c_0 \cdot c = d_0 \cdot d = 0$:

A.3 $\Longleftrightarrow$

$$c_0 \cdot d + d_0 \cdot c = 0 \qquad \text{(A.4)}$$

A.2.2 Condition suffisante

Montrons que A.1 et A.4 est une condition suffisante. Si A.4 alors A.2, et comme $c \times d \neq 0$ les droites C et D appartiennent au même plan et s'intersectent.

A.3 Pseudo-intersection de deux droites

A.3.1 Distance entre deux droites

Soient (a_0, a) et (b_0, b) les coordonnées de deux droites L_a et L_b. Si $a = b$ alors $||(a \times a_0) - (b \times b_0)||$ représente la distance entre les deux droites. Autrement, le vecteur $v = (a \times b)/||(a \times b)||$ est perpendiculaire aux deux droites. Le point $a \times a_0$ appartenant à la droite L_a le plan $((a \times a_0) \cdot v, v)$ contient L_a et est parallèle à L_b, la même chose est vraie pour le plan $((b \times b_0) \cdot v, v)$ qui contient L_b et qui est parallèle à L_a. Ces deux plans étant parallèles, leur distance peut s'écrire: $|((b \times b_0) - (a \times a_0)) \cdot v|$ et correspond aussi à la distance entre les deux droites.

A.3.2 "pseudo" intersection de deux droites

Si l'on note $\sigma = ((b \times b_0) - (a \times a_0)) \cdot v$ alors la translation $\sigma \star v$ amène L_a en $L'_a(a_0 - \sigma \star v \times a, a)$ qui intersecte L_b. Le système d'équations
$a \times x = (a_0 - \sigma \star v \times a)$
$b \times x = b_0$

a une solution unique m correspondant à l'intersection entre les deux droites L'_a et L_b et le "pseudo-point" d'intersection peux être défini par:

$$m - (\sigma/2) \times v$$

Annexe B

Calibrage

La bonne marche du système dépend d'un bon calibrage des différents systèmes de références. Un de nos objectifs dans Handey est d'automatiser le plus possible ces calibrages. Bien que nous soyons encore éloigné de cet objectif nous décrivons un des calibrage clés du système.

Les deux principaux repères de référence de Handey sont: le repère de référence du capteur tri-dimensionnel et le repère de référence du système de modélisation géométrique. Nous sommes intéressés par la mise en correspondance de ces repères avec le repère attaché à la base du robot. Pour illustrer le processus de calibrage, nous décrivons dans cette section le calibrage entre le système de vision et le repère du robot.
La procédure de calibrage suppose que l'axe x du repère du capteur est parallèle à l'axe z du repère du robot et que la hauteur de la table est connue dans les deux repères. L'architecture mécanique du système permet de justifier ces deux hypothèses. La problème du calibrage est donc ramené à un problème à deux dimensions: trouver une rotation autour de l'axe Z et une translation dans le plan x, y.

Phase un: Calcul de la rotation. Un cube est placé dans la pince du robot de telle façon qu'une de ses faces soit contre un des mors de la pince, cette condition étant réalisée aucune contrainte suplémentaire n'est nécessaire. En particulier il n'est pas nécessaire de connaître la position relative du cube dans

la pince. On commande au robot d'amener le cube dans le champ de vision du capteur. Soient (xw_1, yw_1) les coordonnées du poignet du robot dans le plan $x - y$ du repère du robot et (xv_1, yv_1) les coordonnées du centre de la face du cube visible par le capteur.

On commande alors au robot de se déplacer de $(xr_1 + dx, yr_1)$ sans changer son orientation. dx est choisi de telle façon que le cube reste entièrement dans le champ de vision. Soient (xv_2, yv_2) les nouvelles coordonnées du centre de la face. Le vecteur $(xv_2 - xv_1, yv_2 - yv_1)$ est parallèle au vecteur x du repère du robot, ceci nous permet de calculer la rotation entre le repère de vision et le repère du robot:

$$R = Rot(\vec{z}, atan2(xv_2 - xv_1, yv_2 - yv_1))$$

$Rot(\vec{v}, \theta)$ est la rotation d'angle θ autour du vecteur $\vec{v}$.

Phase deux: Calcul de la translation. Rien n'étant supposé concernant la position relative du cube et de la pince, appelons (fx, fy) la projection de cette position sur le plan x-y du repère du robot. L'objectif du mouvement suivant est de calculer fx et fy. On commande au robot de tourner de π autour de l'axe z en maintenant la position du centre du poignet fixe. Durant ce mouvement, le centre de la face décrit un demi-cercle et se retrouve à l'opposé de sa position initiale par rapport au centre du poignet. Appelons (xv_3, yv_3) la nouvelle position du centre de la face donnée par le système de vision. Il est possible d'écrire

$$(fx, fy) = 0.5R^{-1}(xv_2 - xv_3, yv_2 - yv_3)$$

La transformation entre le repère du robot et le repère du système de vision est donc donnée par:

$$(o_x, o_y) = (xr_2, yr_2) + (fx, fy) + R^{-1}(xv_2, yv_2))$$

Si $h - r$ et h_v sont respectivement les hauteurs de la table dans le repère du robot et le repère du système de vision alors $o_z = h_r - h_v$ correspond à la différence de hauteur entre les deux repères.

Finalement la tranformation permettant d'écrire les coordonnées d'un point du repère de vision dans le repère du robot est donnée par:

$Trans(o_x, o_y, o_z)R^{-1}$

$Trans(x, y, z)$ correspondant à une translation selon le vecteur x, y, z.

Bibliographie

[ABB*75] A.P. Ambler, H.G. Barrow, C.M. Brown, R.M. Burstall, and R.J Popplestone. A versatile system for computer controlled assembly. *Artificial Intelligence*, 6:129–156, 1975.

[AHM85] J.M. Abel, W. Holzmann, and J.M. McCarthy. On grasping planar objects with two articulated fingers. In *IEEE International Conference on Robotics and Automation*, pages 576–581, St Louis, March 1985.

[Ala83] Rachid Alami. *Un environnement lisp pour l'intégration et la mise en œuvre de système complexes en robotique.* PhD thesis, Institut National Polytechnique de Toulouse, November 1983.

[Ban83] J.P. Bansard. *Le système LM-EXE.* LIFIA internal report, Laboratoire d'Informatique Fondamentale et d'Intelligence Artificielle, Institut National Polytechnique de Grenoble, 1983.

[BFG85] B.S. Baker, S. Fortune, and E. Grosse. Stable prehension with multi-fingered hande. In *IEEE International Conference on Robotics and Automation*, pages 570–575, St Louis, March 1985.

[BH86] Robert C. Bolles and Radu Horaud. 3dpo: a three-dimensional part orientation system. *The International Journal of Robotics Research*, 5(2):3,26, Fall 1986.

[BJ82] C. Blume and W. Jakob. *Design of a structured robot language (SRL).* Technical Report, University of Karlsruhe, 1982.

[BJ85] Paul J. Besl and Ramesh C. Jain. Three-dimensional object recognition. *Computing Surveys*, 17(1):75–145, March 1985.

[Bor84] Paul-Louis Borianne. *Contributions à la vision par ordinateur tridimentionnelle.* PhD thesis, Institut National Polytechnique de Grenoble, April 1984.

[Bra53] L. Brand. *Vector and tensor analysis.* Chapman and Hall Editor, 1953.

[Bro80] P. Brou. *Implementation of high level commands for robots.* M.S Thesis, Massachusetts Institute of Technology, december 1980.

[Bro82] Rodney A. Brooks. Symbolic error analysis and robot planning. *International Journal of Robotics Research*, 1(4):29–68, December 1982.

[Bro83] Rodney A. Brooks. Planning collision free motions for pick-and-place operations. *The International Journal of Robotics Research*, 2(4):19–44, Winter 1983.

[Buc87] Stephen John Buckley. *Planning and teaching compliant motion strategies.* PhD thesis, Massachusetts Institute of Technology, January 1987.

[BVD*86] J. Barber, R. Volz, R. Desai, R. Runinfeld, B Schipper, and J. Wolter. Automatic two-fingered grip selection. In *IEEE International Conference on Robotics and Automation*, pages 890–896, San Francisco, April 1986.

[Can83] John Francis Canny. *Finding edges and lines in Images.* Technical Report TR-720, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, June 1983.

[Can87] John Francis Canny. *The complexity of robot motion planning.* PhD thesis, Massachusetts Institute of Technology, May 1987.

[Cor79] Anorad Corporation. *Operation and programming Manual.* Technical Report, Anorad Corporation, December 1979.

[Cor82] General Electric Corporation. *Automation system A12, Assembly robot operator's manual.* Technical Report P50VE025, General Electric Corporation, 1982.

[Cra80] John Craig. *JARS: JPL Autonomous Robot System.* Technical Report, Jet Propulsion Laboratory, 1980.

[CSS84] M. Crebassa, F. Salmon, and P. Schmitt. *Montage d'un amortisseur.* LIFIA internal report, Laboratoire d'Informatique Fondamentale et d'Intelligence Artificielle, Institut National Polytechnique de Grenoble, 1984.

[Cun79] C.S Cunningham. Robot flexibility through software. In *Ninth International Symposium on Industrial Robot*, pages 297–307, Washington D.C, 1979.

[DB75] J.A Darringer and M.W Blagsen. *MAPLE: A high level language for research in mechanical assembly.* Technical Report RC 5606, IBM T.J Watson Research Center, September 1975.

[DH55] J. Denavit and R.S. Hartenber. A kinematic notation for lower-pair mechanisms based on matrices. *Journal of Applied Mechanics*, 22, June 1955.

[DL84] Bruno Dufay and Jean-Claude Latombe. An approach to automatic robot programming based on inductive learning. *The International Journal of Robotics Research*, 4(3), 1984.

[Don84] Bruce R. Donald. *Motion Planning with Six Degrees of Freedom.* Technical Report TR-791, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, May 1984.

[Don86] Bruce R. Donald. Robot motion planning with uncertainty in the geometric models of the robot and environment: a formal framework for error detection and recovery. In *IEEE International Conference on Robotics and Automation*, San Francisco 1986.

[Don87] Bruce R. Donald. *Error Detection and Recovery for Robot Motion Planning with Uncertainty.* PhD thesis, Massachusetts Institute of Technology, july 1987.

[Dou80] McDonnell Douglas. *Robotic system for Areospace Batch Manufacturing.* Technical Report, McDonnell Douglas, Inc, February 1980.

[EGG76] R.C. Evans, D.G. Gernett, and D.D Grossman. *Software system for computer controlled manipulator.* Technical Report RC 6210, IBM T.J Watson Research Center, May 1976.

[Erd84] Michael Erdmann. *On motion planning with uncertainty.* Technical Report TR-810, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 1984.

[Ern61] H.A.A Ernst. *A Computer-Controlled Mechanical Hand.* PhD thesis, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 1961.

[Fav86] Bernard Faverjon. Object level programming of industrial robots. In *IEEE International Conference on Robotics and Automation*, 1986.

[Fea84] R. Fearings. Simplified grasping and manipulation with dextrous robot hands. In *American Control Conference*, pages 32–38, 1984.

[FH86] Olivier D. Faugeras and Marc Hebert. The representation, recognition, and localization of 3-d objects. *The International Journal of Robotics Research*, 5(2):27,52, Fall 1986.

[Fin76] R.A. Finkel. *Constructing and debbuging manipulator programs.* Technical Report AIM 284, Artificial Intelligence Laboratory, Stanford University, August 1976.

[FN71] R.E. Fikes and N.J. Nilson. Strips: a new approach of the application of theorem proving to problem solving. *Artificial Intelligence Journal*, 3,4(2), 1971.

[FP80] D. Falek and M. Parent. An evolutive language for an intelligent robot. *The Industrial robot*, September 1980.

[FT87] Bernard Faverjon and Pierre Tournassoud. *A local based approach for path planning of manipulators with high number of degrees of freedom.* Technical Report, Insitut National de la Recherche en Informatique et Automatique, February 1987.

[FV82] J. W. Franklin and G.J. Vanderburg. Programming vision and robotics systems with rail. In *SME Robots 5*, pages 392–406, March 1982.

[Ger85] Florence Germain. Génération statique de trajectoires: un algorithme paramétrable par le contexte. In *5 ème Congrès AFCET Reconnaissance des Formes et Intelligence Artificielle*, Grenoble France, November 1985.

[GET87] GETRIS. *HELIOS: Le système graphique temps réel pour la représentation d'images réalistes.* Technical Report, GETRIS Image, 1987.

[GG78] G. Gini and M. Gini. Object description with a manipulator. *The industrial Robot*, March 1978.

[GGGG79] G. Gini, M. Gini, R. Gini, and D. Giuse. Introducing software systems in industrial robots. In *Ninth International Symposium on Industrial Robots*, pages 309–321, 1979.

[Gir83] A. Giraud. Generalized active compliance for part mating with assembly robots. In *First International Symposium on Robotics Research*, Bretton-Wood, 1983.

[GL85] W. Eric L. Grimson and Tomás Lozano-Pérez. *Recognition and localization of overlapping parts from sparse data.* Technical Report A.I. Memo 841, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, June 1985.

[Gor86] Steven J. Gordon. *Automated assembly using feature localization.* Technical Report TR-932, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, December 1986.

[Gou84] L. Gouzenes. Strategies for solving collision free trajectories problems for mobile and manipulator robots. *The International Journal of Robotics Research*, 4(3), Winter 1984.

[Gro77] D.D Grossman. *Programming of a computer controlled industrial manipulator by guiding through motion.* Technical Report RC 6393, IBM T.J Watson Research Center, March 1977.

[GSCT84] William A. Gruver, Barry I. Soroka, John J. Craig, and Timothy L. Turner. Industrial robot programming languages a comparative evaluation. *IEE Transaction on Systems, Man, and Cybernetics*, 14(4):565–570, July/August 1984.

[HA77] H. Hanafusa and H. Asada. Stable prehension of objects by robot hand with elastic fingers. In *Seventh International Symposium on Industrial Robots*, pages 361–368, Tokyo, October 1977.

[HK86] S.A Hutchinson and A.C Kak. Fprolog: a language to integrate logic and functional programming for automated assembly. In *IEEE International Conference on Robotics and Automation*, pages 904–909, San Francisco, April 1986.

[HM86] W. Holzmann and J.M. McCarthy. Computing the friction forces associated with a three-fingered grasp. In *IEEE International Conference on Robotics and Automation*, pages 594–600, San Francisco, April 1986.

[Hor87] Berthold. K.P. Horn. Closed form solution of absolute orientation using unit quaternions. *Journal of the Optical Society A*, 4(4):629–642, April 1987.

[HP86] V. Hayward and R.P. Paul. Robot manipulator control under UNIX RCCL: a robot control "C" library. *The International Journal of Robotics Research*, 5(4):94–111, Winter 1986.

[Ing87] François F. Ingrand. *Inférence de formes à partir de fonctions: application à la conception de montages d'usinage.* PhD thesis, Institut National Polytechnique de Grenoble, February 1987.

[INH*86] Katshushi Ikeuchi, H. Keith Nishihara, Berthol K. P. Horn, Patrick Solbalvarro, and Shigemi Nagatha. Determining grasp configuration using photometric stereo and the prism binocular stereo system. *The International Journal of Robotics Research*, 5(1):46–65, Spring 1986.

[ITM87a] ITMI. *GTR-86/3D: Le processeur temps réel pour l'extraction des contours en vision par ordinateur.* Technical Report, Industrie et Technologie de la Machine Intelligente, 1987.

[ITM87b] ITMI. *LM3-PC: Le Language de programmation des robots sur IBM/PC-AT et compatibles.* Technical Report, Industrie et Technologie de la Machine Intelligente, 1987.

[JL86] J.W. Jameson and L.J. Leifer. Quasi-static analysis: a method for predicting grasp stability. In *IEEE International Conference on Robotics and Automation*, pages 890–896, San Francisco, April 1986.

[JO85] Joe. L. Jones and Patrick A. O'Donnell. *Using the Puma System.* Technical Report Working paper 271, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, April 1985.

[Kas77] JS. Kashioka. An approach to the integrated robot with multiple sensory feedback: visual recognition techniques. In *Seventh International Symposium on Industrial Robots*, Tokyo, October 1977.

[Kat86] Oussama Kathib. Real-time obstacle avoidance for robot manipulator and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98, Spring 1986.

[Lat83] Jean-Claude Latombe. A survey of advanced software for robot manipulators. *Computers in Industry*, 4(3), October 1983.

[Lat87] Jean-Claude Latombe. *Automatic synthesis of manipulator robot programs.* to be published, 1987.

[Lau81] Christian Laugier. A program for automatic grasping of objects with robot arm. In *Eleventh International Symposium on Industrial Robots*, Tokyo, October 1981.

[Lau87] Christian Laugier. *Vers des robots intelligents: les apports respectifs des systèmes robotisés contemporains, de l'intelligence artificielle, et du raisonement géométrique.* PhD thesis, Université Scientifique et Médicale de Grenoble, Institut National Polytechnique de Grenoble, December 1987.

[LB85] Tomás Lozano-Pérez and Rodney A. Brooks. *An Approach to Automatic Robot Programming.* Technical Report A.I Memo 842, Massachusetts Institute of Technology , Artificial Intelligence Laboratory, April 1985.

[LLJL84] J.C. Latombe, C. Laugier, and J.F Miribel J.M. Lefebvre, E. Mazer. Design and implementation of the lm robot programming system. In *Second International Symposium on Robotics Research*, Kyoto, August 1984.

[LMT84] Tomás Lozano-Pérez, Matthew T. Mason, and Russel H. Taylor. Synthesis of fine-motion strategies for robots. *The International Journal of Robotics Research*, 3(1), 1984.

[Loz76] Tomás Lozano-Pérez. *The design of a mechanical assembly system.* Technical Report A.I. T.R. 397, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, December 1976.

[Loz81] Tomás Lozano-Pérez. Automatic planning of manipulator transfer movement. *IEEE Transaction on SMC*, SMC-11(10):681–698, October 1981.

[Loz82] Tomás Lozano-Pérez. *Robot programming.* Technical Report A.I Memo 698, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, December 1982.

[Loz86] Tomás Lozano-Pérez. *A simple motion planning algorithm for general robot manipulator.* Technical Report A.I Memo 896, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, June 1986.

[LP83] Christian Laugier and Jocelyne Pertin. Automatic robot programming: the grasp planner. In *International Symposium on Advanced Software in Robotics*, Liège, Belgium, 1983.

[LP85] Christian Laugier and Jocelyne Pertin. Sharp: a system for automatic robot programming of manipulation robots. In *Third Symposium of Robotics Research*, Paris, October 1985.

[LT86] Christian Laugier and Pascal Theveneau. Planning sensor-based motions for part-mating using geometric reasoning techniques. In *European Conference on Artificial Intelligence*, Brighton U.K, 1986.

[Lux85] Augustion Lux. *Algorithme et contrôle en vision par ordinateur.* PhD thesis, Université Scientifique et Médicale de Grenoble, Institut National Polytechnique de Grenoble, Septembre 1985.

[LW77] L.I. Liebermann and M.A Wesley. Autopass an automatic robot programming system for computer controlled mechanical assembly. *IBM Journal of Reasearch and Development*, 4(21):321–333, 1977.

[Mas82] Matthew T. Mason. *Manipulator Grasping and Pushing Operations.* Technical Report TR-690, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 1982.

[Maz81] Emmanuel F. Mazer. *Réalisation d'un support expérimental de recherche pour le projet de robotique Pandore: Définition et implantation du language LM.* PhD thesis, Institut National Polytechnique de Grenoble, January 1981.

[Maz83] Emmanuel Mazer. Geometric programming of assembly robots (lmgeo). In *International Symposium on Advanced Software in Robotics*, Liège, Belgium, 1983.

[Mir84] Jean-Francois Miribel. *Conception et implantation d'un système de programmation de robots.* PhD thesis, Institut National Polytechnique de Grenoble, Octobre 1984.

[MM84] Emmamnuel Mazer and Jean-Francois Miribel. *Le language LM: Manuel de référence.* CEPADUES, 1984.

[Ngu86] Van-Duc Nguyen. *The synthesis of stable force closure grasps.* Technical Report TR-905, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, July 1986.

[PAB80] R.J Popplestone, A.P. Ambler, and I.M Bellos. An interpreter for a language for describing assemblies. *Artificial Intelligence*, 14:79–107, 1980.

[Par81] William T. Park. *The SRI robot programming system (RPS).* Technical Report, Stanford Research Institut International, March 1981.

[Pau72] Richard Paul. *Modeling, trajectory calculation and servoing of a computer controlled arm.* PhD thesis, University of Pennsylvania, November 1972.

[Pau77] R.P Paul. Wave: a model based language for manipulator control. *The Industrial Robot*, March 1977.

[Pau81a] Richard P. Paul. Kinematic control equations for simple manipulators. *IEEE Transactions on Systems, Man, and Cybernetics*, June 1981.

[Pau81b] Richard P. Paul. *Robot Manipulators: Mathematics, programming, and control.* MIT Press, Cambridge, Mass, 1881.

[Per86] Jocelyne Pertin-Troccaz. *Modélisation du raisonement géométrique pour la programmation des robots.* PhD thesis, Institut National Polytechnique de Grenoble, March 1986.

[Per87] Jocelyne Pertin-Troccaz. On-line automatic robot programming: a case study in grasping. In *IEEE International Conference on Robotics and Automation*, Raleigh, 1987.

[Pop79] R.J Popplestone. *Specifying manipulation in terms of spatial relation.* Technical Report 117, Department of Artificial Intellignece University of Edinburgh, 1979.

[PP87] Jocelyne Pertin and Pierre Puget. Dealing with uncertainties in robot planning using program proving techniques. In *Fourth Symposium of Robotics Research*, Santa Cruz, August 1987.

[Rec80] A.A.G. Rechiqua. Representations for rigids solids: theories, methods and systems. *ACN Computing Surveys*, 12(4), December 1980.

[Rob85] GMF Robotics. *introduction to the Karel programming language1.* Technical Report, GMF Robotics, September 1985.

[Rou83] R. Rousseau. *Définition et implémentation d'un système modulaire de programmation pour la productique: LMAC.* PhD thesis, Université de Franche-Comté, April 1983.

[Ruo79] Karl Ruoff. Teach: a concurrent robot control language. In *IEEE COMPSAC*, pages 442–445, November 1979.

[Sal78] M. Salmon. Sigla: the olivetti sigma robot programming language. In *Eight International Symposium on Industrial Robots*, Stuttgart, West Germany, 1978.

[SC82] J. Kent Salisbury and John J. Craig. Articulated hands: force control and kinematics issues. *The International Journal of Robotics Research*, 1(1), Spring 1982.

[SGM82] K.G. Shin, P Vukkadala G, and N.D. McKay. Development of an explicit robot control language. In *IEEE COMPSAC*, Chicago, November 1982.

[SGS84] B.E. Shimano, C.C. Geshke, and C.H. Spalding. Val 2: a new robot control system for automatic manufacturing. In *IEEE International Conference on Robotics and Automation*, Atlanta, 1984.

[SHS87] J. Schwartz, J. Hopcroft, and M. Sharir. *Plannning, geometry and Complexity of robot motion planning.* Albex Publishing Corp., 1987.

[Sil73] D. Silver. *The little robot system.* Technical Report AIM 273, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, January 1973.

[Sko81] F. Skoda. *LPR: Language de programmation pour robots.* Technical Report, Régie Renault, 1981.

[Sou83] Viviane Souvignier. *PVV - Un système d'interprétation d'images par prédiction-vérification.* PhD thesis, Institut National Polytechnique de Grenoble, 1983.

[Tak77] S. Takeyasu. An approach to the integrated robot with multiple sensory feedback: construction and control functions. In *Seventh International Symposium on Industrial Robots*, Tokyo, October 1977.

[Tay76] Russel H. Taylor. *A synthesis of manipulator control programs.* PhD thesis, Stanford University, Computer Science Department, July 1976.

[TPB79] K. Takase, R.P Paul, and E.J. Berg. A structered approach to robot programming and teaching. In *IEEE COMPSAC*, Chicago Illinois, November 1979.

[TSM82] R.H. Taylor, P.D. Summers, and J.M. Meyer. Aml: a manufacturing language. *The International Journal of Robotics Research*, 1(3):19–41, Fall 1982.

[Val85] Jean-Michel Valade. *Raisonement géometrique et synthèse de trajectoire d'assemblage.* PhD thesis, Université Paul Sabatier, January 1985.

[WBKL83] Patrick H. Winston, Thomas O. Binford, Boris Katz, and Michael Lowry. *learning physical description from functional definitions, examples, and precedents.* Technical Report AI-Memo 679, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, January 1983.

[WG75] P.M. Will and D.D Grossman. An experimental system for computer controlled mechanical assembly. *IEEE Transactions on Computers*, 91–97, 1975.

[Win77] M. Wingham. *Planning how to grasp objects in a cluttered environment.* PhD thesis, University of Edinburgh, 1977.

[WLL*80] M.A. Wesley, T. Lozano-Pérez, L.I. Lieberman, M.A. Lavin, and D.D. Grossman. A geometric modeling system for automated assembly. *IBM Journal of Research and Development*, 24(1), January 1980.

[Zah87] Joseph Steven Zahavi. *A DSP based stripe finder for a laser range finder.* Technical Report, Massachusetts Institute of Technology, 1987. Bachelor of science in computer science and engineering.

RESUME Un système de planification pour les opérations de transfert de pièces par robot manipulateur est présenté. Le système prend en charge la planification de toutes les opérations nécessaires à la réalisation d'un assemblage tels que la reconnaissance, la saisie et le transport de la pièce à monter. Une architecture basée sur l'utilisation d'une hiérarchie de filtres est utilisée comme structure de contrôle. Les algorithmes de raisonnement géométrique nécessaires à la manipulation sont présentés en utilisant un formalisme unique. Les résultats de l'expérimation ainsi que les développements futurs du système sont décrits.

ABSTRACT A system designed to plan a pick-and-place operation with a robot manipulator is described. The system automatically plans the necessary operations to perform the assembly: part recognition, grasping, path planning and re-grasping. The control structure is based on a hierarchy of tests. The algorithms used by the planner are presented with a single geometric formalism. The experimentation and the future developments of the system are discussed

MOTS CLEFS Programmation des robots - Raisonnement géométrique - planficateur de trajectoire - planificateur de saisie - vision tri-dimentionnelle - modélisation géométrique - assemblage automatique.