Utilisation du raisonnement géométrique pour la planification en robotique d'assemblage :

Pascal Théveneau

THESE

Présentée à
L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

par

Pascal THEVENEAU

pour obtenir le grade de

DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

OOOOO

UTILISATION DU RAISONNEMENT GEOMETRIQUE POUR LA PLANIFICATION EN ROBOTIQUE D'ASSEMBLAGE : LE SYSTEME PAMELA

OOOOO

Thèse soutenue le 4 Novembre 1988 devant la commission d'examen

Président	: P. JORRAND
Rapporteurs	: P. COIFFET
	G. GIRALT
Directeur de thèse	: C. LAUGIER
	B. LACOLLE

Préparée au sein du laboratoire LIFIA

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Président : Georges LESPINARD

Année 1988

Professeurs des Universités

BARIBAUD Michel	ENSERG
BARRAUD Alain	ENSIEG
BAUDELET Bernard	ENSPG
BEAUFILS Jean-Pierre	ENSEEG
BLIMAN Samuel	ENSERG
BLOCH Daniel	ENSPG
BOIS Philippe	ENSHMG
BONNETAIN Lucien	ENSEEG
BOUVARD Maurice	ENSHMG
BRISSONNEAU Pierre	ENSIEG
BRUNET Yves	IUFA
CAILLERIE Denis	ENSHMG
CAVAIGNAC Jean-François	ENSPG
CHARTIER Germain	ENSPG
CHENEVIER Pierre	ENSERG
CHERADAME Hervé	UFR PGP
CHOVET Alain	ENSERG
COHEN Joseph	ENSERG
COUMES André	ENSERG
DARVE Félix	ENSHMG
DELLA-DORA Jean	ENSIMAG
DEPORTES Jacques	ENSPG
DOLMAZON Jean-Marc	ENSERG
DURAND Francis	ENSEEG
DURAND Jean-Louis	ENSIEG
FOGGIA Albert	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 Jean-Claude	ENSPG
JOURDAIN Geneviève	ENSIEG
LACOUME Jean-Louis	ENSIEG
LESIEUR Marcel	ENSHMG
LESPINARD Georges	ENSHMG
LONGEQUEUE Jean-Pierre	ENSPG
LOUCHET François	ENSIEG
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
PERRET René	ENSIEG
PERRET Robert	ENSIEG
PIAU Jean-Michel	ENSHMG
POUPOT Christian	ENSERG
RAMEAU Jean-Jacques	ENSEEG
RENAUD Maurice	UFR PGP
ROBERT André	UFR PGP
ROBERT François	ENSIMAG
SABONNADIERE Jean-Claude	ENSIEG
SAUCIER Gabrielle	ENSIMAG
SCHLENKER Claire	ENSPG
SCHLENKER Michel	ENSPG
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
CHOLLET Jean-Pierre
COEY John
COLINET Catherine
COMMAULT Christian
CORNUEJOLS Gérard
COULOMB Jean- Louis
DALARD Francis
DANES Florin
DEROO Daniel
DIARD Jean-Paul
DION Jean-Michel
DUGARD Luc
DURAND Madeleine
DURAND Robert
GALERIE Alain
GAUTHIER Jean-Paul
GENTIL Sylviane
GHIBAUDO Gérard
HAMAR Sylvaine
HAMAR Roger
LADET Pierre
LATOMBE Claudine
LE GORREC Bernard
MADAR Roland
MULLER Jean
NGUYEN TRONG Bernadette
PASTUREL Alain
PLA Fernand
ROUGER Jean
TCHUENTE Maurice
VINCENT Henri

Chercheurs du C.N.R.S
Directeurs de recherche 1ère Classe

CARRE René
FRUCHART Robert
HOPFINGER Emile
JORRAND Philippe
LANDAU Ioan
VACHAUD Georges
VERJUS Jean-Pierre

Directeurs de recherche 2ème Classe

ALEMANY Antoine
ALLIBERT Colette
ALLIBERT Michel
ANSARA Ibrahim
ARMAND Michel
BERNARD Claude
BINDER Gilbert
BONNET Roland
BORNARD Guy
CAILLET Marcel
CALMET Jacques
COURTOIS Bernard
DAVID René
DRIOLE Jean
ESCUDIER Pierre
EUSTATHOPOULOS Nicolas
GUELIN Pierre
JOUD Jean-Charles
KLEITZ Michel
KOFMAN Walter
KAMARINOS Georges
LEJEUNE Gérard
LE PROVOST Christian
MADAR Roland
MERMET Jean
MICHEL Jean-Marie
MUNIER Jacques
PIAU Monique
SENATEUR Jean-Pierre
SIFAKIS Joseph
SIMON Jean-Paul
SUERY Michel
TEODOSIU Christian
VAUCLIN Michel
WACK Bernard

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

E.N.S.E.E.G

CHATILLON Christian
HAMMOU Abdelkader
MARTIN GARIN Régina
SARRAZIN Pierre
SIMON Jean-Paul

E.N.S.E.R.G

BOREL Joseph

E.N.S.I.E.G

DESCHIZEAUX Pierre
GLANGEAUD François
PERARD Jacques
REINISCH Raymond

E.N.S.H.G

ROWE Alain

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

COURTIN Jacques

E.F.P.

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

DEVINE Rodericq
GERBER Roland
MERCKEL Gérard
PAULEAU Yves

Abstract

Because of the complexity of the actual world, using a robot to perform an assembly is a difficult task. The purpose of automatic programming is to generate robot control programs from a symbolic description of the task, and therefore to simplify assembly automation. For many related subproblems, this necessitates to deal with both geometry and physical phenomena. This report focuses on **fine-motions planning**; in other words it deals with the automatic generation of *small* motions necessary in order to succed in mating two parts in spite of geometric uncertainty.

After having pointed out the restrictions of numerical approaches, we present a method based on geometric reasoning and a well-suited model. Then we describe our system which automatically generates fine-motions strategies and its implementation. Finally, we present some experiments with the robot.

KEYWORDS: Robotics, Artificial Intelligence, Geometric reasoning, Automatic robot programming, Motion planning, Fine motions.

Remerciements

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

- Monsieur Philippe JORRAND, Directeur de Recherche au CNRS, Directeur du Laboratoire d'Informatique Fondamentale et d'Intelligence Artificielle.
- Monsieur Philippe COIFFET, Directeur de Recherche au CNRS.
- Monsieur Georges GIRALT, Directeur de Recherche au CNRS.
- Monsieur Bernard LACOLLE, Maître de Conférence à l'Université Joseph Fourrier de Grenoble.
- Monsieur Christian LAUGIER, Chargé de Recherche à l'INRIA, Responsable de l'équipe robotique du LIFIA.

Je tiens aussi à remercier :

- Les membres de l'équipe robotique: P. Di Giacomo, C. Gandon, C. Laugier, M. Pasquier, J. Pertin-Troccaz, P. Puget, qui ont contribué par leurs travaux à la réalisation du projet SHARP.
- Les membres du LIFIA qui ont sus par leur amitié créer une ambiance propice à la recherche.
- Les responsables du projet ARA qui ont permis le financement de ces travaux.

Sommaire

Chapitre 1

Introduction

1.1 Les robots

Les robots nous envahissent. On les trouve aussi bien dans l'espace pour mener à bien des missions scientifiques, sous la mer pour intervenir sur un site dangereux que dans les usines pour remplacer l'homme dans les tâches pénibles ou répétitives. Aujourd'hui et plus encore dans l'avenir, ils sont appelés à jouer un rôle de plus en plus important dans notre vie. Leurs facultés de perception et de raisonnement progressent chaque jour, leur permettant de se substituer de plus en plus fréquemment à l'homme dans ses tâches quotidiennes.

L'**assemblage** de pièces mécaniques reste cependant pour eux un domaine difficile et leur présence sur des sites d'assemblage reste encore rare. Cette carence est dûe à la nature même des robots et plus particulièrement aux limites des systèmes de programmation actuels.

1.2 Les robots et l'assemblage

Des machines capables d'assembler automatiquement un ensemble de pièces existaient bien avant l'apparition des premiers robots. Elles offrent des performances élevées et actuellement aucun robot ne peut les concurrencer du point de vue du rendement.

Une particularité essentielle de ces machines spécialisées est de ne pouvoir travailler que dans un environnement parfaitement défini et invariant. Les pièces à assembler

doivent être positionnées avec précision et les incertitudes pouvant subsister sont bien inférieures aux tolérances de montage. De plus, ces machines spécialisées sont complètement figées dans une tâche particulière et elles ne peuvent se concevoir que pour de grandes séries.

L'emploi des robots pour des tâches d'assemblage a pour but de lever ces contraintes en fournissant un outil **polyvalent, évolutif**, pouvant interagir avec un univers plus incertain. Grâce à leurs capteurs, les robots vont pouvoir adapter leurs actions à une configuration des pièces à assembler. Les performances d'un robot dépendent alors essentiellement de la qualité du programme de commande.

1.3 La programmation des robots

La description du travail que doit effectuer un robot est une phase préliminaire indispensable à sa mise en œuvre. Cette phase appelée **programmation** s'effectue de différentes manières suivant le but que l'on s'est fixé.

1.3.1 Programmation par l'exemple

C'est le mode de programmation utilisé sur les premiers robots. Un opérateur manipule le robot ou une réplique de celui-ci pour lui faire décrire les trajectoires que l'on souhaite ensuite lui voir réaliser. Pendant cette phase de description par l'exemple, un calculateur enregistre les différentes positions qui seront ensuite utilisées pour commander les mouvements du bras. Cette programmation conceptuellement simple est encore très largement utilisée pour des applications de peinture et de soudure par points dans l'industrie automobile. Cette technique peut être utilisée pour toutes les applications peu sensibles aux petites variations de l'environnement.

1.3.2 Programmation à l'aide d'un langage

Si la programmation par l'exemple est simple et souvent rapide, elle comporte une lacune qui lui interdit la réalisation de tâches complexes. Ce qui sépare un robot d'une machine ordinaire, c'est la présence de capteurs extérioceptifs qui ne peuvent être pris en compte dans une description par l'exemple. Pour pallier ce problème, les chercheurs puis les industriels ont développé des langages algorithmiques (LM [Maz81], VAL [SGS84]) permettant de décrire des trajectoires, de mesurer des valeurs de capteurs (forces, visions toucher, ...) et de décrire les différents comportements

à adopter en fonction des tests. Ces langages de la robotique sont formés par un langage algorithmique (fortran, lisp) qui sert de support, et par un certain nombre de primitives de commande pour les actionneurs des axes du robot. Si cette méthode s'est révélée efficace aux débuts de la robotique, on se heurte maintenant à des difficultés théoriques liées au mélange des niveaux d'expression (Cf §7). Tous ces langages permettent d'exprimer des contraintes de mouvements en utilisant des concepts abstraits, mais les valeurs des capteurs ne peuvent être appréhendées que grâce à des fonctions dédiées fournissant un résultat numérique. Cette lacune est dûe à la pauvreté des modèles qu'ils manipulent.

1.3.3 La programmation automatique

La commande d'un robot par l'intermédiaire d'un langage de commande permet une grande polyvalence mais au détriment de la facilité d'utilisation. L'écriture d'un programme et sa mise au point restent un travail de spécialiste. De plus, seul l'usage permet de garantir la robustesse du programme.

L'automatisation du raisonnement humain nécessaire à la mise en œuvre d'un robot d'assemblage constitue une étape supplémentaire indispensable. Grâce à un logiciel, la programmation ne se fait plus de manière explicite mais seulement en décrivant les tâches à accomplir. L'utilisation du robot relève alors de la programmation automatique.

Le but d'un système de programmation automatique est de traduire la description d'une tâche en un programme réellement exécutable. Cette description se fait en faisant abstraction des moyens à employer pour la réaliser et en utilisant un langage évolué. Un tel système accepte en entrée deux informations :

1. une description des composants à assembler par l'intermédiaire d'un système de type CAO.
2. une définition de la tâche à effectuer en utilisant un langage de description ne comportant que des ordres de niveau tâche.

1.4 Le problème traité

La programmation automatique de robots pour des tâches d'assemblage soulève de nombreux problèmes. Il faut être capable de calculer des prises stable pour la manipulation, de déterminer des trajectoires permettant de déplacer des objets (et

le robot) dans un univers encombré et de planifier les derniers petits mouvements réalisant le montage.

Ces petits mouvements de montage sont dûs aux différences qui existent entre l'univers théorique de la planification et l'univers réel où évolue le robot. Le programme de montage ne peut alors se contenter d'indiquer des positions à atteindre mais doit mettre en œuvre des capteurs extérioceptifs pour corriger les mouvements en fonction de l'univers réel.

L'objet de nos recherches consiste à développer un système de planification de mouvements fins de montage pouvant s'insérer dans un projet plus vaste de programmation automatique.

1.5 L'approche

Les différents travaux déjà effectués sur le sujet s'appuient tous sur le **dessassemblage**. Cette approche consiste à faire l'hypothèse que l'inverse d'une opération de démontage peut constituer un plan de montage. Nous utiliserons aussi cette démarche pour planifier des mouvements fins de montage.

Cette hypothèse introduit un certain nombre de restrictions mais comme nous le verrons dans le paragraphe 6.1, elle est moins contraignante qu'il ne pourrait sembler.

1.6 Notre contribution

La modélisation géométrique

Le problème que nous nous proposons de résoudre nécessite de combiner des calculs numériques et des calculs symboliques. Nous avons développé dans cette perspective un modèle adapté à cette bi-représentation (Cf §4). Il fournit à l'utilisateur une base de données symboliques construites automatiquement à partir du modèle numérique. Un mécanisme de liaison inter-modèle (Cf 4.3) propage les actions d'un modèle sur un autre de manière transparente pour l'utilisateur.

Le raisonnement géométrique

Sur ce modèle, nous avons construit un système de planification de mouvements fins d'assemblage appelé PAMELA (Cf §6). Ce système est capable de produire un pro-

gramme constitué d'instructions gardées et de mouvements compliants conduisant l'objet manipulé jusqu'à sa position finale d'assemblage en présence d'incertitude.

L'intégration de ce modèle et du planificateur PAMELA au système de programmation automatique développé au LIFIA : SHARP [LP85], nous a permis de valider nos recherches sur des exemples en vraie grandeur.

1.7 Plan du mémoire

Nous développerons au cours des différents chapitres de ce mémoire plusieurs points qui montreront à la fois la richesse du problème et sa complexité. Nous commencerons par étudier de manière générale l'architecture d'un tel système (§2) en présentant les différents modules nécessaires ainsi que les solutions généralement retenues. Nous aborderons ensuite le problème de la modélisation (§4) en proposant un concept propre au raisonnement géométrique. Puis nous traiterons le problème particulier des mouvements fins d'assemblage (§6)où nous présenterons les techniques que nous avons retenues pour pouvoir traiter des mouvements de montages variés incluant des translations des rotations et certaines opérations telles que le vissage. Un chapitre (§7) sera consacré aux langages cibles de la programmation automatique. En conclusion, nous présenterons une réflexion sur les directions futures de nos travaux.

Tous les travaux décrits dans ce mémoire s'insèrent dans le cadre du projet SHARP *:*

H*igh level*
S*ystem for*
A*utomatic*
R*obot*
P*rogramming*

Ce projet développé au L.I.F.I.A. a pour but, non pas de produire un système industriel, mais de définir un environnement de programmation où la confrontation simultanée des différents problèmes permet de mieux mesurer la complexité de la tâche.

Chapitre 2

Architecture d'un système de programmation automatique

2.1 Problèmes à résoudre et principes de base

La programmation automatique d'un robot ne peut dans l'état actuel de nos connaissances opérer à partir d'une description "floue" de la tâche à effectuer. Essayons donc de définir les limites de ce système. Pour cela, imaginons une tâche fictive représentée sur la figure 2.1.

Pour un être humain, le simple fait de voir le dessin des différentes pièces fournit suffisamment d'informations pour savoir (ou plutôt comprendre) ce qu'il faut faire. Pour une machine, ce même problème est très compliqué. Etudions la démarche à suivre pour réaliser l'assemblage de manière automatique.

La première étape est le choix du matériel nécessaire à l'assemblage. Il faut choisir le type de robot à mettre en œuvre ainsi que les capteurs qui devront être utilisés. Dans la pratique ce choix est souvent vite fait car on ne dispose que d'un robot. Dans le cas où on imaginerait un atelier complet devant être géré de manière automatique, ce choix du robot peut devenir un problème complexe. Il doit s'effectuer en fonction de la disponibilité mais aussi et surtout en fonction de ces caractéristiques : nombre de degrés de liberté, type de capteurs dont le manipulateur est muni, etc. Une fois le robot choisi, il faut s'intéresser à l'assemblage proprement dit, en déterminant en premier lieu l'ordre de montage des différents composants.

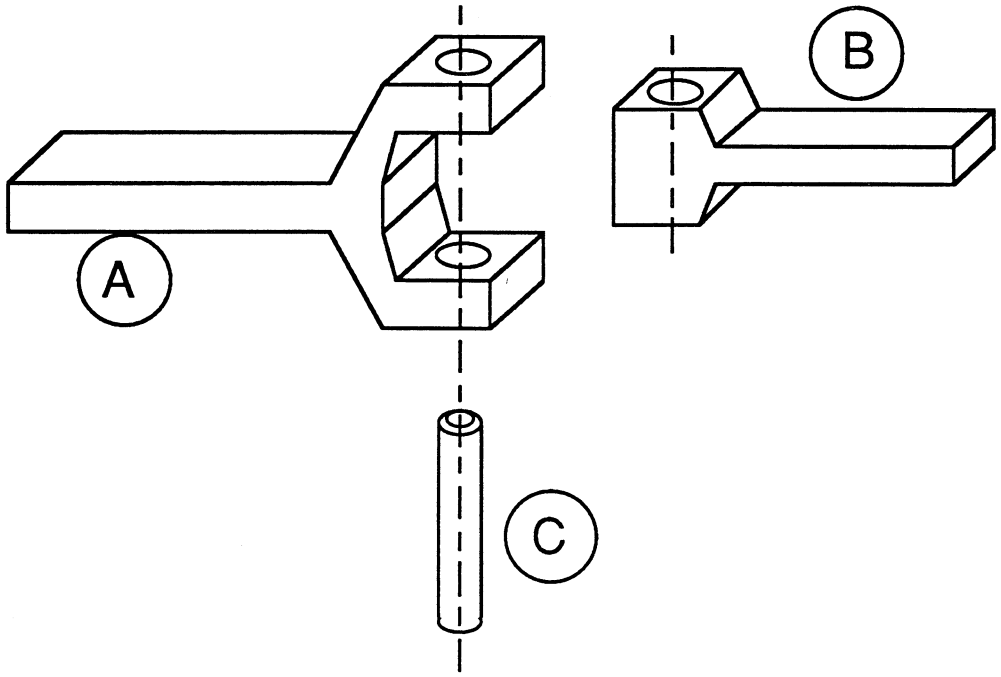


Figure 2.1: Exemple d'assemblage

Dans le cas de la figure 2.1, il faut :

1. Insérer partiellement l'axe d'articulation C dans la pièce A.
2. Placer la pièce B en alignant les axes des trous des pièces B et A.
3. Finir l'insertion de l'axe d'articulation C.

Cette simple décomposition ne pose aucune difficulté pour un être humain, mais demande beaucoup de connaissances de la part d'un système de programmation automatique. De plus, la décomposition est fortement dépendante de la configuration de robot utilisée. Le système doit donc posséder un modèle symbolique des aptitudes du robot. Cette tâche nommée "conception de la gamme d'assemblage" peut être confiée à un module indépendant sous réserve de tenir compte des problèmes d'interdépendance.

Le système doit ensuite choisir les régions de l'espace de travail du robot les mieux adaptées à l'éxécution de sa tâche. Il doit aussi déterminer quelles vont être les pièces fixes et les pièces mobiles pendant l'assemblage. Ce problème présenté brièvement

est extrêmement complexe car il faut être capable d'estimer la qualité d'une position de travail. Pour cela de nombreux facteurs peuvent intervenir. La zone de travail où le robot dispose effectivement de 6 degrés de liberté, par exemple, est beaucoup plus petite que l'espace atteignable. Si des contraintes imposent que l'on doive travailler en dehors de cette zone, il faudra choisir une position compatible avec les mouvements à effectuer.

Cela fait, il faut être capable de manipuler les pièces dans l'ordre déterminé par le système de conception de la gamme d'assemblage. Manipuler une pièce consiste à réaliser des opérations de saisie, de transport et d'assemblage. L'ordre des opérations n'est pas figé. Il se peut que l'on doive saisir un objet, le transporter un peu plus loin et le reposer afin de pouvoir le saisir autrement. Ces problèmes de reconfiguration en cours de manipulation s'avèrent délicats car il n'y a aucune information dans le modèle pour guider le choix des opérations à effectuer. Tout relève de l'expérience du programmeur et nécessite donc un transfert de savoir entre l'expert et la machine. Le savoir à transmettre à la machine est le sens commun que possède chacun et qui lui permet de manipuler les objets de la vie courante. Ce savoir est d'autant plus difficile à transmettre que nous l'utilisons de manière réflexe. Afin de simplifier le travail d'un système de planification, une opération de manipulation se décompose très souvent en trois étapes :

- SAISIR / LACHER
- TRANSPORTER
- ASSEMBLER

Les trois opérations de base de la décomposition mettent en jeu des connaissances et des techniques de raisonnement différentes. Elles sont de ce fait traitées par des modules spécialisés. Une difficulté supplémentaire provient de ce que ces opérations sont dépendantes les unes des autres. Il est donc nécessaire de faire communiquer les différents modules. Quels problèmes doivent-ils résoudre?

1. Le module de saisie [Per86] doit être capable de choisir des prises stables dans le but de pouvoir manipuler la pièce en question pendant son montage mais aussi pendant son transport. Le choix des prises tiendra compte de la géométrie de la pièce, de la nature de l'outil de préhension et aussi des contraintes physiques rencontrées pendant les autres opérations. Considérons l'exemple de la figure 2.2.

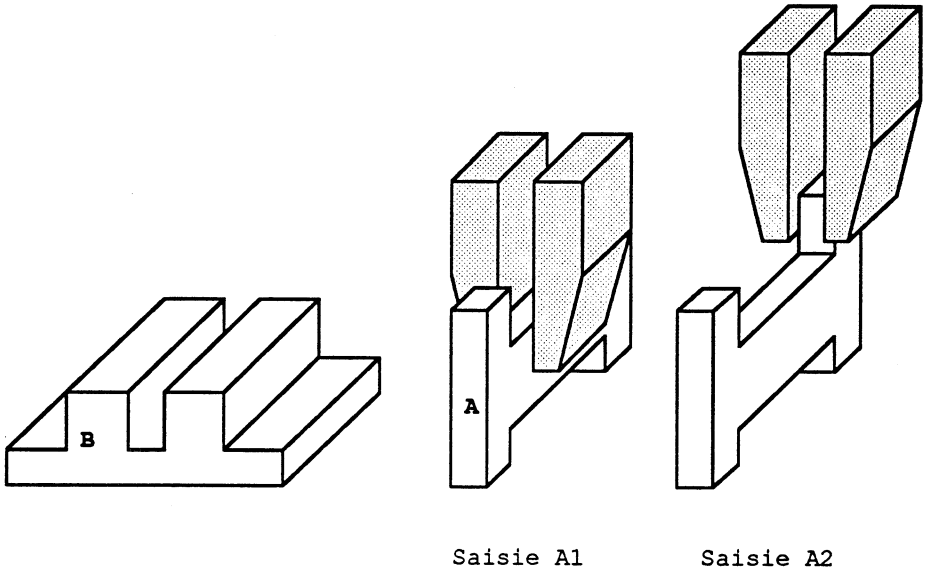


Figure 2.2: Contraintes physiques de la saisie

Le manipulateur doit engager la pièce A dans la rainure B. La saisie réalisée en A_1 est stable mais incompatible avec avec le mouvement de montage. Il faudra donc choisir une autre prise, par exemple celle présentée en A_2 qui présente une moins bonne surface de préhension mais qui permet de mener à bien l'opération de guidage.

2. Le module calculant une trajectoire sans collision pour le transport des objets manipulés doit choisir une suite de positions intermédiaires garantissant d'éviter les obstacles pendant le déplacement. La modification de la structure du robot au cours des mouvements constitue une difficulté supplémentaire : il faut modéliser dynamiquement le bras manipulateur pour s'assurer qu'il ne heurte aucun des objets environnants. La complexité de ce problème est directement liée au nombre de degrés de liberté du robot. Dans la cas de l'assemblage, un robot doit posséder 6 degrés de liberté pour ne pas être limité dans certains mouvements. Le problème est alors très complexe (Cf §2.3.5) et ne peut pour l'instant être résolu directement. Enfin, les objets manipulés possèdent une forme et un volume dont il faut tenir compte.

   Sur la figure 2.3, on voit les différentes positions que doit occuper la pièce en forme de L durant son transport. Si on considère un problème similaire mais

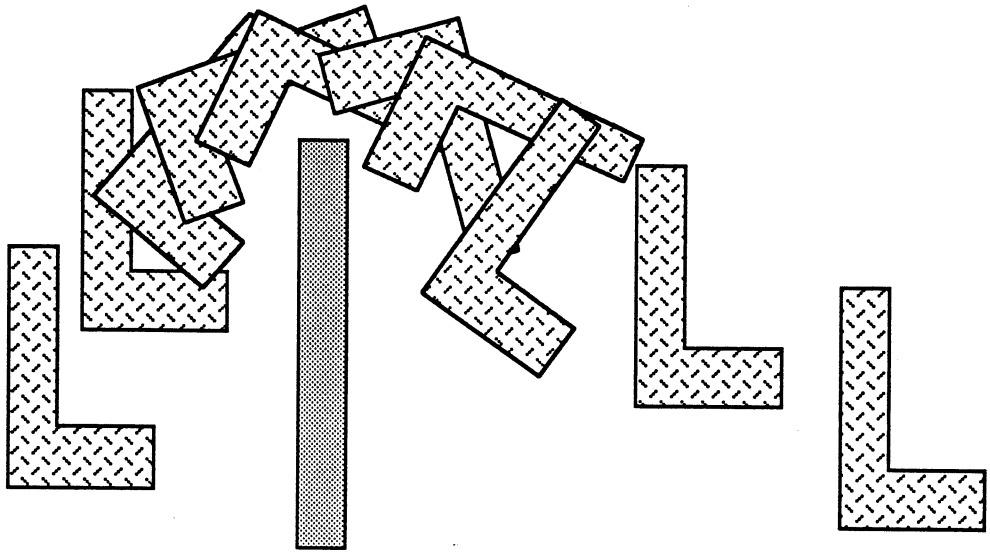


Figure 2.3: Déplacement d'un objet en 2 dimensions

avec une pièce en trois dimensions, manipulée par un robot à six degrés de liberté, la complexité de la planification d'une trajectoire devient alors telle que la résolution ne peut se faire que pour un nombre de degrés de liberté plus réduit.

3. Enfin le dernier module capable de planifier les différents petits mouvements à réaliser pour effectuer l'assemblage doit être à même de raisonner dans un espace incertain. En effet, en raison des incertitudes tant géométriques que dues à la manipulation, la seule donnée de positions à atteindre ne suffit pas pour réussir une opération de montage. L'emploi de mouvements gardés et de mouvements compliants constitue alors un excellent moyen de résoudre ces problèmes. Le rôle du planificateur de mouvement fins est donc de générer automatiquement de telles instructions.

   Il existe peu de méthodes utilisables pour cela. Nous présenterons au §3.3 quelques méthodes ainsi que leurs avantages et inconvénients. Notre approche est décrite dans le chapitre 6.

Cette décomposition hiérachique est réalisée par tous les projets ayant essayé de résoudre le problème de la programmation automatique. C'est une approche classique lorsque le problème est trop complexe pour être abordé dans son ensemble. No-

tons cependant que les différents modules sont interdépendants et qu'il faudrait pouvoir utiliser une résolution simultanée. Dans la pratique, on traite séquentiellement les problèmes, quitte à remettre en question les choix effectués lorsqu'on rencontre plus tard une contradiction ou une impossibilité. Cette technique, classique en intelligence artificielle, augmente considérablement la combinatoire et diminue par conséquent les performances. L'utilisation d'un mécanisme contrôlant simultanément les différents modules et repoussant le plus tard possible les choix propres à un seul module permet de limiter (du moins partiellement) l'explosion combinatoire.

Dans le paragraphe suivant, nous présentons brièvement trois projets du MIT qui, soit par les problèmes soulevés, soit par les solutions apportées, ont contribué à approfondir le problème.

2.2 De LAMA à HANDEY

2.2.1 LAMA

LAMA [Loz76] a été le premier système de programmation automatique abordant tous les problèmes. Il n'y a pas eu d'implantation des concepts développés par T.Lozano-Perez. Il fait en revanche le tour des différentes questions et définit une architecture possible pour un système réel.

L'ambition du système était de convertir une description symbolique d'une tâche d'assemblage réalisée grâce à des instructions du type "INSERER, PLACER, POUSSER, ...", en une séquence d'instructions LLAMA exécutables par le robot. Le langage LLAMA est un langage de manipulation défini pour pouvoir effectuer cette transformation de manière efficace.

Afin de produire du code LLAMA, le système possède un modèle des différents composants de l'assemblage. L'auteur préconise pour cela l'emploi d'un système de conception assistée par ordinateur possédant deux fonctionnalités :

1. Le système permet de décrire de manière interactive un ensemble d'objets.

2. Les positions relatives des différents composants doivent pouvoir être spécifiées par l'intermédiaire d'une description géométrique.

La description de la tâche est réalisée au moyen d'un langage symbolique de type déclaratif exprimant les actions du manipulateur, sans en préciser les détails

d'exécution. Un exemple d'une telle description est donné sur la figure 2.5. Cet exemple correspond au montage de l'ensemble bielle-piston représenté dans la figure 2.4.

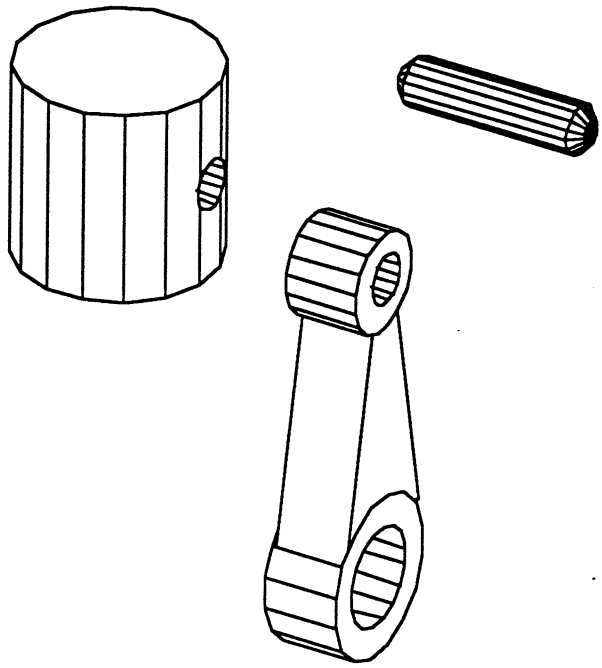


Figure 2.4: Ensemble bielle pistion axe

La première opération exécutée par le système consiste à convertir la description précédente en une suite d'actions de plus bas niveau non entièrement définie (les paramètres des mouvements ne sont pas encore spécifiés). La figure 2.6 donne la traduction du code de la figure 2.5 en actions élémentaires.

Cette traduction comporte des instructions de saisie (GRASP), de transport (PLACE) et des mouvements réalisant les petits mouvements correctifs d'assemblage (INSERT). Ce choix d'instructions élémentaires correspond à la décomposition hiérarchique du problème mentionné précédemment. L'univers du robot étant bien entendu entaché d'erreurs, le contrôle de l'exécution du programme est confié à un module intégrant les informations fournies par des capteurs (force, toucher, vision).

Les solutions retenues pour résoudre chacun des sous-problèmes sont des méthodes théoriques qui n'ont pas connu d'implantation réelle. Seule la saisie a connu une implantation partielle.

```
(insert      obj1: [piston-pin]
             obj2: [pistion pin-hole]
             such-that: (partly (fits-in obj1 obj2)))

(insert      obj1: [piston-pin]
             obj2: [rod small-end-hole])

(push-into   obj1: [piston-pin]
             obj2: (and [piston pin-hole][rod small-end])))
```

Figure 2.5: Description de l'assemblage d'un piston

En conclusion, il faut voir LAMA comme une première étude appronfondie du problème de la programmation automatique. Il met en évidence les problèmes géométriques liés aux opérations de saisie, de transport et de montage. L'influence des incertitudes est aussi clairement montrée par l'utilisation des squelettes de programme (Cf §3.3.1) et par le choix difficile des paramètres de ces procédures.

2.2.2 TWAIN

TWAIN [LB85] est une proposition de système de programmation automatique plus récent qui date de quelques années. S'il a fallu si longtemps pour passer du système LAMA au système TWAIN, c'est que les problèmes rencontrés lors de la définition des premiers systèmes de programmation automatique ont conduit à des développements théoriques et pratiques séparés. TWAIN a pour objet de rassembler ces différents résultats afin de constituer un système intégré. Il s'intéresse en particulier à tenter de résoudre le problème des interdépendances. Nous allons présenter rapidement les solutions retenues afin d'en dégager les idées fortes et ainsi mieux saisir l'architecture du système.

Approche

La méthode de séparation des problèmes proposées en 2.1 ne peut être appliquée directement car pour effectuer le transport d'un composant il faut savoir comment

```
(grasp                obj:          [piston-pin])

(place-in-vise        obj:          [piston-pin])

(ungrasp              obj:          [piston-pin])

(grasp                obj:          [piston]
                      such-that:    (facing+ ([piston] top) down))

(insert               obj1:         [piston-pin]
                      obj2:         [piston pin-hole]
                      such-that:    (partly (fits-in obj1 obj2) 0.25))

(ungrasp              obj:          [piston])

(grasp                obj:          [rod]
                      such-that:    (facing+ ([rod-bar] top) up))

(insert               obj1:         [piston-pin]
                      obj2:         [rod small-end-hole])

(ungrasp              obj:          [rod])

(grasp                obj:          [piston])

(remove-from-vise     obj:          [piston])

(push-into            obj:          [piston-pin]
                      such-that:    (and (fits-in [piston-pin] [piston pin-hole])
                                         (fits-in [piston-pin] [rod small-end])))

(ungrasp              obj:          [piston])
```

Figure 2.6: Liste des opérations d'assemblage

il est saisi, pour le saisir il faut savoir comment il va être monté et pour le monter, il faut connaitre la position d'approche choisie par le module de transport. L'interdépendance des différents problèmes a conduit les auteurs à proposer les idées directrices suivantes :

- Utiliser différents modules de planification pour la saisie, le transport sans collision et les mouvements fins afin de fournir non pas une solution mais un éventail de solutions.
- Utiliser des squelettes de programmes pour décrire les opérations du manipulateur.
- Employer un système de propagation de contraintes pour choisir les paramètres, du programme exécutable, qui influencent le plus d'opérations.
- Raisonner sur l'espace des configurations pour déterminer quels sont les mouvements possibles pour le robot.

Les deux premiers choix répondent au problème de l'interdépendance des différents modules. Grâce à cette méthode on repousse le plus loin possible les choix qui sont trop dépendants d'un sous-problème donné. Le deuxième argument consiste à essayer de fixer le plus tard possible dans la résolution les paramètres qui contraignent simultanément le plus de sous problèmes. En procédant de la sorte, on effectue une sorte de résolution parallèle où les différents modules se synchroniseraient sur les difficultés communes.

Le troisième point concerne l'utilisation des squelettes de programme. Un squelette de programme correspond à la notion informatique de procédure paramétrée. Le travail de synthèse consiste à déterminer quelle procédure appliquer, et quelles sont les valeurs des paramètres d'appel.

TWAIN est donc construit autour d'un mécanisme de propagation de contraintes et d'une bibliothèques de squelettes de programmes. Trois modules opèrent sur cette base : planification des mouvements fins, saisie et transport sans collision.

Mouvements fins

L'idée de base consiste à rechercher une suite de mouvements compliants permettant de réaliser la tâche d'assemblage. Les auteurs proposent d'utiliser une technique de raisonnement géométrique appelées pré-images.

De manière intuitive, il s'agit de construire un "chemin d'assemblage", en démontant le composant considéré par des mouvements élémentaires. A chaque étape, le système détermine l'ensemble des positions intermédiaires pouvant conduire de manière sûre vers le sous-but que l'on s'est fixé. Cette méthode est décrite de manière plus détaillée au paragraphe 3.3.2.

Transport

L'idée de base consiste à construire une représentation explicite de "l'espace libre" pour le manipulateur et sa charge, puis à rechercher un chemin reliant la position de départ et la position d'arrivée dans cet espace [Loz81].

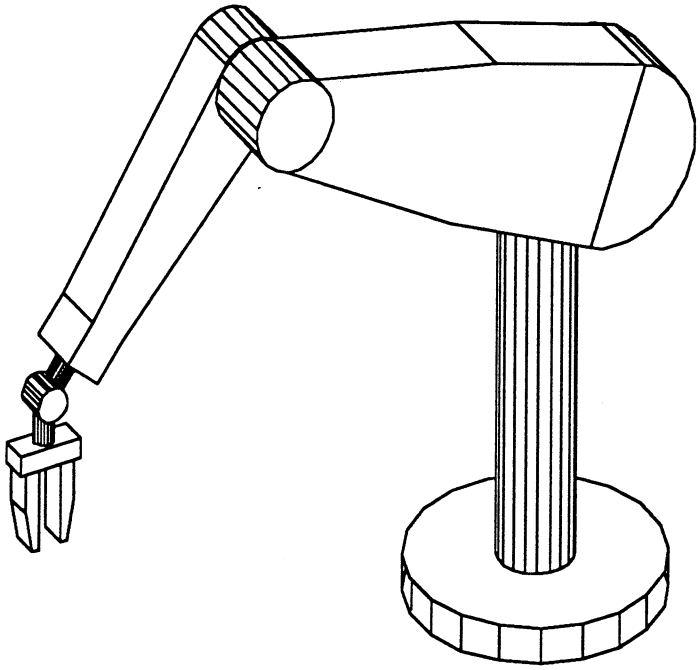


Figure 2.7: Robot Puma

Cette représentation est réalisée dans l'espace des configurations. Dans celui-ci, une configuration du robot, c'est à dire le n-uplet des valeurs articulaires, sera représenté par un point. La représentation de cet espace est réalisé de manière discrète à partir d'un découpage en cellules élémentaires du types :

$$[\theta_{1_{min}} < \theta_1 < \theta_{1_{max}}] \times [\theta_{2_{min}} < \theta_2 < \theta_{2_{max}}] \times \cdots \times [\theta_{n_{min}} < \theta_n < \theta_{n_{max}}].$$

Les configurations impossibles dues aux butées mécaniques du robot et aux obstacles de son environnement sont notées comme des zones interdites. Ce calcul est un problème complexe dans le cas général, qui est souvent résolu au moyen d'approximations (volumes englobants simples en particulier). La détermination d'une trajectoire de transfert se ramène alors à la recherche d'un chemin dans le graphe des cellules libres. On utilise pour cela des algorithmes de type A* qui garantissent une recherche optimale.

Signalons que la résolution du problème du transport sans collision est généralement faite pour 3 ou 4 degrés de liberté à cause de la complexité algorithmique introduite. D'après les auteurs, cette restriction ne contraint pas trop la classe de problèmes susceptibles d'être traités. Cela est sans doute vrai si l'on accepte de choisir des positions privilégiées de l'espace de travail du manipulateur. Ce n'est cependant pas acceptable dans le cas général.

Saisie

TWAIN recherche des prises à réaliser à l'aïde d'une pince à deux mors parallèles. Dans le but de restreindre la combinatoire, il n'analyse que des prises comportant un contact de type plan entre un des mors de la pince et une face plane de l'objet à saisir. L'autre mors de la pince peut alors s'appuyer sur une face, une arête ou même un sommet de l'objet.

Le système attribue tout d'abord à chaque face plane de l'objet à saisir une probabilité de saisie. Les faces sont ensuite étudiées dans l'ordre des priorités décroissantes. Ce traitement est réalisé par un filtre géométrique permettant de réduire le coût du calcul.

Soit P_i une face plane de l'objet à saisir, P_j la face d'appui (ou l'arête d'appui) de l'autre mors, F_1 et F_2 les deux faces de saisie de la pince et considérons que P_i et F_1 sont coplanaires. Les positions pouvant être occupées par la pince se trouvent limitées à un plan parallèle à P_i. On appelle "ensemble de prises" l'ensemble de ces positions et on le désigne par G.

Toutes les positions de G ne sont pas des positions de prises possibles car pour certaines il n'y a pas de contact entre la face P_i et le mors de la pince ou encore parce qu'il se produirait une collision entre la pince et le reste de l'objet. On contraint l'ensemble G en imposant que la face F_1 recouvre la face P_i et que la face F_2 recouvre P_j. Cet ensemble peut être explicitement calculé dans l'espace des configurations [Loz83]. On calcule de la même manière, les positions de la pince où

il y a interférence entre celle-ci et les autres objets. Les positions de saisie, sont fournies par l'intersection entre le complémentaire de cet ensemble et G.

2.2.3 HANDEY

Le système Handey [LJM*87] est le premier système de "pick and place" ayant été réellement implanté. L'objectif visé est alors beaucoup plus modeste que les ambitions initiales de LAMA et TWAIN.

Description générale

Le système Handey est capable de localiser des objets dans son espace de travail, de choisir une prise, de planifier une trajectoire pour atteindre les objets considérés et ensuite planifier une trajectoire de montage pour amener l'objet à la position requise. Cette succession d'actions s'effectue selon la démarche suivante. Tout d'abord, on fournit au système une suite de commandes d'assemblage sous la forme :

"*MOVE* `object` *TO* `destination`"

ainsi qu'un modèle des pièces à manipuler[1], et ensuite Handey effectue les opérations suivantes ;

- Le système construit une *table des distances* grâce à un capteur laser. En utilisant celle-ci, il est capable de reconnaître les objets de la scène en faisant une mise en correspondance entre les arêtes du modèle des objets et les arêtes réelles extraites de l'image [Can86]. Le résultat de cette première étape fournit la position de l'objet considéré dans le système de coordonnées du robot.

- Il choisit après une prise stable en prenant en compte les obstacles de l'environnement et la configuration d'assemblage souhaitée. Si aucune prise n'est possible, il reporte son choix sur une prise stable incompatible avec l'assemblage et fera une nouvelle saisie après avoir transporté l'objet dans une zone libre réservée à cet usage.

- Il planifie ensuite une trajectoire pour atteindre sans collision la position de saisie. Ce calcul peut être effectué plusieurs fois s'il est nécessaire de se placer auparavant dans la zone de changement de prise.

[1] Handey ne travaille qu'avec des composants à faces planes

- L'avant-dernière étape consiste à calculer un chemin qui amène l'objet à assembler près de sa position finale.
- Enfin, il exécute une commande gardée en force terminant l'assemblage du composant[2].

Les performances d'Handey sont tout à fait intéressantes mais appellent cependant un commentaire. Le dernier mouvement (la commande en force) est beaucoup trop rudimentaire pour pouvoir faire de l'assemblage dans des conditions réelles. Une telle commande ne peut assembler que des pièces qui s'emboitent bien et dont l'assemblage est peu sensible aux incertitudes géométriques et aux incertitudes de manipulation.

Architecture informatique

Handey est composé des principaux modules suivants :

- Un module de modélisation d'objets polyédriques.
- Un module capable de dresser la table des distances entre objets et capteur.
- Un module de localisation d'objets dans une image obtenue à partir du système précédent.
- Un module de calcul de trajectoire sans collision pour un manipulateur à six degrés de liberté.
- Un module de saisie capable d'engendrer des prises stables et de planifier d'éventuelles étapes intermédiaires pour effectuer une saisie selon une autre prise.
- Un module de contrôle du robot pour commander les mouvements du robot[3].

Cette architecture logicielle reflète la décomposition hiérarchique du problème de la programmation automatique d'un robot. Il convient cependant de souligner que la décomposition en problèmes fortement dépendants introduit une difficulté supplémentaire liée à la communication entre les modules et surtout au contrôle de l'appel des différents modules. Ce point sera détaillé au §2.3.3.

[2]Le système ne sait pas générer de mouvements compliants pour effectuer l'assemblage. La classe de problèmes qu'il est suceptible de résoudre est forcément restreinte.

[3]Il faut remarquer que par contrôle, on entend ici commande du robot et non pas contrôle de l'exécution du plan d'action.

2.3 SHARP

High level
System for
Automatic
Robot
Programming

2.3.1 Présentation générale

Le système SHARP [LP85,Lau87] est un système de programmation automatique en cours de développement au LIFIA. Des résultats significatifs ayant déjà été obtenus sur la résolution des problèmes de saisie, de transport sans collision et des mouvements fins, nous avons decidé de passer à l'étape suivante qui consiste à intégrer les techniques développées au sein d'un même système.

Afin de rester réalistes dans nos ambitions, nous nous sommes limités à un système capable de transformer une séquence d'instructions de type "Monter A, Placer B, ..." en un programme exécutable par un robot à six degrés de liberté. Le principe du traitement est illustré sur la figure 2.8.

Il est à noter que ce traitement conduit à considérer une action de haut niveau comme une séquence du type :

$$\text{SAISIR} \longrightarrow \text{TRANSPORTER} \longrightarrow \text{ASSEMBLER} \longrightarrow \text{LACHER}$$

Cette approche ne permet pas d'implanter des cycles de manipulation plus complexes. En particulier, elle ne permet pas de changer de prise en cours de manipulation. L'intégration d'une phase de *"regrasping"* comme dans HANDEY serait cependant envisageable car il n'y a pas d'obstacle théorique.

Cette décomposition effectuée, il s'agit alors pour trois modules (classiquement : saisie, transport, mouvements fins) de planifier les actions d'assemblage.

2.3.2 Interface homme-machine de modélisation

Deux informations doivent être données au système pour qu'il soit en mesure de produire automatiquement un programme d'assemblage : une description de l'univers du robot et une description de l'assemblage à réaliser en termes de relations géométriques. L'utilisation du système ne se faisant que par l'intermédiaire de ces deux types d'informations, il est important de fournir à l'utilisateur une interface

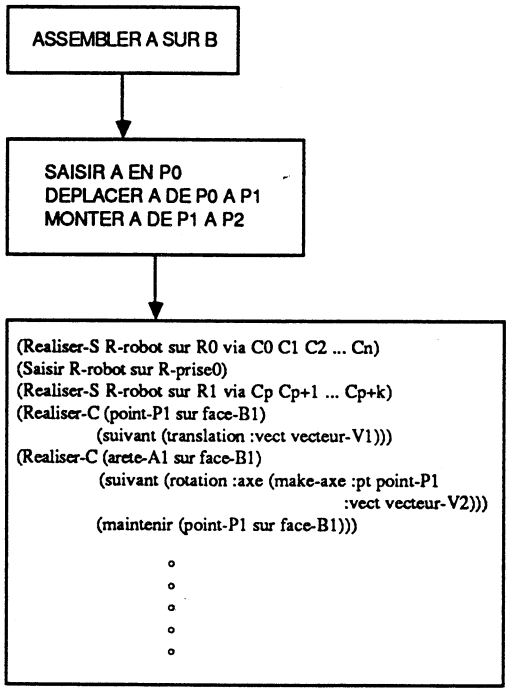


Figure 2.8: Expansion d'une instruction SHARP

la plus conviviale possible. La description de l'univers correspond à une phase de modélisation classique; Celle de l'assemblage est réalisée au moyen d'un langage symbolique manipulant des constructions géométriques [Lau87].

2.3.3 Structure de contrôle

Comme nous l'avons vu précédemment, la dépendance des sous-problèmes interdit une résolution totalement séparée. Cette situation conduit à poser deux questions auxquelles il faudra apporter une réponse :

- Quel problème résoudre en premier?
- Quelle solution doit-on remettre en cause en cas d'échec?

Résoudre ce problème de contrôle consiste à ordonner les différents modules, et à définir un mécanisme de retour arrière. Il n'y a cependant pas de solution générale. Deux attitudes sont possibles : développer une structure de contrôle simple ("backtracking" systématique) mais peu efficace, ou mettre en œuvre un mécanisme d'analyse d'échec sophistiqué.

La seconde méthode est sans doute la mieux adaptée, mais l'état de nos connaissances ne permet pas sa réalisation. Nous avons donc choisi d'implanter une structure de contrôle élémentaire qui opère comme suit :

1. Recherche d'une prise.
2. Recherche d'une stratégie de mouvements fins.
3. Recherche d'une trajectoire sans collision.

La résolution commence donc par le problème (1). Les résultats et les contraintes associées sont ensuite transmises au module (2), qui fait de même pour le module (3). Si un des modules rencontre une situation d'échec, il demande au module précédent de lui proposer une autre solution ainsi que d'autres paramètres d'entrée. Le premier module demandera à l'utilisateur d'autres conditions initiales.

Après planification des actions, le système s'assure de la validité globale du plan engendré vis-à-vis de l'incertitude. Ce problème de la preuve de programme de robotique a déjà été abordé [Pug85], mais les résultats sont encore difficilement utilisables. Nous envisageons cependant d'intégrer dans l'avenir ce type de contrôle à notre système.

2.3.4 Saisie

Nous considérons l'action de saisir comme la réalisation d'un ensemble de contacts simultanés entre l'outil de préhension et le composant à saisir.

La différence majeure avec une opération d'assemblage, est due au fait que les positions de contact à réaliser ne sont pas connues initialement et qu'elles doivent vérifier certaines contraintes de stabilité. Les méthodes utilisées dans SHARP sont issues des travaux de [LP83] et [Per86]. Elles conduisent à répondre aux quatre questions suivantes :

- Comment prendre un objet donné au moyen d'un outil donné?
- Selon quelle trajectoire aller saisir l'objet?
- Quelle doit être la configuration de l'outil au moment de la saisie?
- Comment s'éloigner une fois la saisie réalisée afin de pouvoir donner le contrôle au calcul de la trajectoire de transfert?

Les restrictions apportées par la méthode implantée portent sur deux points : les directions d'approche et de retrait sont rectilignes, l'étude de collision éventuelle ne portera que sur un voisinage de la pièce à saisir. L'implantation n'est faite que pour le cas d'une pince à deux mors parallèles.

La méthode retenue pour choisir des prises opère en deux phases. La première phase, appellée A^2LS [4] prend en compte les problèmes d'accessibilité locale et de stabilité. Cette étape a pour but de déterminer un ensemble de prises potentielles stables. La seconde phase appellée A^2G [5] prend en compte les problèmes d'accessibilité globale. Il faut dans cette phase déterminer si une prise choisie par la phase A^2LS est compatible avec l'environnement du manipulateur.

Phase A^2LS

Pendant cette phase de calcul, le système détermine un ensemble de prises π et les paramètres de préconfiguration de l'outil (écartement des mors par exemple). L'algorithme commence par recenser les différentes entités géométriques de la pièce susceptible d'être saisie (faces, arête, point). A chacune d'elles il applique des critères d'accessibilité locale et rejette ainsi toutes celles qui ne pourront pas contribuer à réaliser une saisie. Ces critères sont basés sur une étude de la topologie de la pièce à saisir. Par exemple, l'arête A1 de la figure 2.9 est concave et de ce fait ne pourra intervenir dans la prise. Plusieurs propriétés topologiques sont ainsi évaluées [LP83]. Elles conduisent à filtrer les éléments géométriques susceptible de contribuer à des prises. Le même type de traitement est appliqué sur les couples.

Formellement, le système devrait essayer tous les couples de parties de l'ensemble des entités. Dans la pratique, il n'envisage que des singletons car la combinatoire devient rapidement trop importante en regard des résultats obtenus. En cas d'échec, le système choisit une prise possible mais peu stable qu'il complétera avec d'autres entités permettant de la stabiliser. Plusieurs critères permettent de choisir ces couples d'entités susceptibles de réaliser la prise :

- Les entités se prêtent à la topologie et aux dimensions de l'outil de saisie.
- L'ensemble objet, outil de saisie est en équilibre.
- L'équilibre n'est pas précaire.

[4] A^2LS = analyse d'accessibilité locale et de stabilité

[5] A^2G = analyse d'accessibilité globale

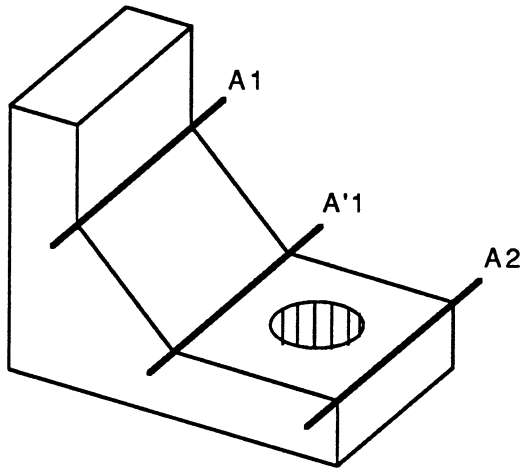


Figure 2.9: Arêtes concaves/convexes

La première contrainte est stricte et permet de rejeter toutes les prises ne correspondant pas à la morphologie de l'outil de saisie. Il faut par exemple que les entités du couple soient chacune parallèle (prise par une pince à mors parallèles), que la distance les séparant soit inférieure à l'écartement maximum des mors, etc.

Le deuxième critère concerne les rotations susceptibles d'être engendrées par le couple créé par le serrage des mors sur l'objet. Plutôt que de faire une étude mécanique des forces, on considère le critère de visibilité : deux faces sont mutuellement visibles si leur projection sur un plan parallèle commun se recouvre (Cf figure 2.10).

Le dernier critère traduit la sensibilité de la prise aux incertitudes de positionnement. Le but est de rejeter les prises qui sont théoriquement correctes mais qui risquent de provoquer une rotation de l'objet lors du serrage des mors.

La dernière étape permet de vérifier l'accessibilité locale des prises retenues, compte tenu de la répartition de la matière au voisinage des entités participant au serrage.

Les prises obtenues à l'issue de ce traitement sont des "prises potentielles", non entièrement spécifiées. C'est le rôle de la phase suivante.

Phase A^2G

Cette phase vérifie la compatibilité des prises calculées par l'A^2LS par rapport à l'environnement de manipulation. Elle permet en même temps de fixer les paramètres de prises non encore instanciés. Cette étape du calcul d'une prise peut être vue

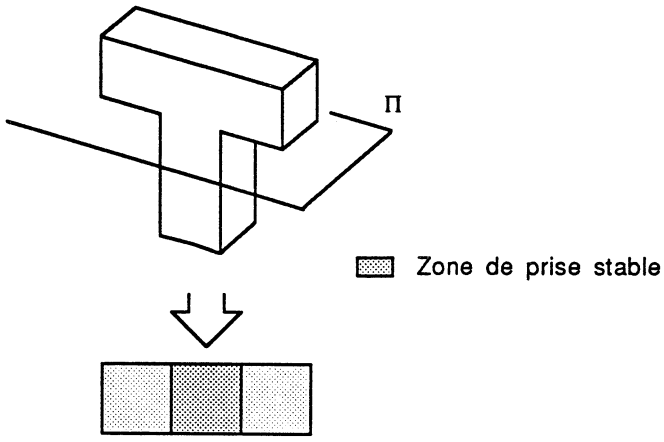


Figure 2.10: Propriété de visibilité

comme la recherche d'une trajectoire sans collision pour atteindre la position finale de saisie où l'évitement d'obstacle se fait en ne prenant en compte que la zone de manipulation.

Compte tenu des contraintes de mouvements imposées par les contacts avec les deux mors parallèles, il est possible d'étudier ce problème dans un espace de dimension 2. Afin de réduire la complexité du problème, le système prend pour hypothèse de base que la prise (et le lâcher) se font généralement à orientation constante.

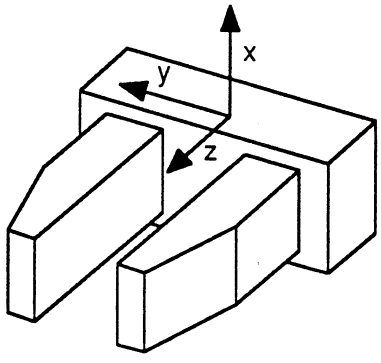


Figure 2.11: Repère associé à la pince

Garder une orientation constante jusqu'à ce que la surface des mors réalise un contact, revient à fixé Y_p constant. Le point O_p se déplace alors dans un plan : le

plan de préhension. L'orientation de Z_p est fixée heuristiquement si elle ne l'est pas déjà par les contraintes de prise. Le problème est alors réduit à un problème de mouvements en translation, pour lequel il est possible de construire une représentation de l'espace des configurations.

La partie mobile (pince + objet manipulé) est ainsi ramenée à un point, et les obstacles sont grossis de manière inverse à la forme du mobile. Ce calcul coûteux en 3D est ramené à un calcul en 2D grâce à un artifice de calcul : l'espace tridimentionnel est découpé par des plans parallèles à (O_p, X_p, Z_p).

Les calculs de grossissement sont alors exécutés pour chacun des plans et les domaines restant valides sont ensuite réunis en utilisant un mécanisme de propagation de contraintes. Lorsque les contraintes imposées par deux plans consécutifs sont trop différentes, le système introduit un nouveau plan par un mécanisme de découpage dichotomique.

Une présentation plus détaillée des algorithmes est faite dans [Per87].

2.3.5 Transport sans collision

Le transport sans collision a pour but de déterminer une trajectoire de transfert réalisable par le robot de manière la plus optimale possible. Notre méthode repose sur une étude de l'espace des configurations du robot. Elle s'appuient sur les travaux de F.Germain [Ger85] et M.Pasquier[Pas88].

Notion d'espace des configurations

Le problème initial consiste à trouver une représentation non-ambiguë pour une structure articulée à p degrés de liberté évoluant dans un espace de dimension n. La position exacte de chacun des composants de la chaîne articulée est complètement déterminée par la donnée des valeurs des p degrés de liberté. L'ensemble des p-uplets que peut vérifier la structure constitue l'espace des configurations. Une trajectoire sera définie par la donnée d'une liste de p-uplets.

Calculer une trajectoire de transfert, pour un robot à six degrés de liberté, revient alors à trouver un chemin dans un espace à six dimensions. Cet espace sera limité par les butées mécaniques du bras, par les obstacles constituant l'univers du robot et par les composants de l'assemblage devant être manipulés.

Modélisation des différents composants

Deux modèles sont nécessaires. Le premier modèle est un modèle cinématique du bras permettant de passer des coordonnées articulaires aux coordonnées de son extrémité terminale dans l'espace 3D où évolue le robot et réciproquement. Un "changeur de coordonnées" [Pau81] fournit ce modèle. Le second modèle se rapporte aux objets physiques (constituants du robot et objets de l'environnement). Pour cela le système dispose du modèle géométrique détaillé au chapitre 4. Il contient des informations qui permettent de construire facilement une modélisation de l'espace des configurations.

Il s'agit alors de modéliser un objet dans l'espace à 6 dimensions à partir d'un modèle à trois dimensions. Si cette transformation est univoque, elle n'est pas simple et ne conserve pas les proprietés topologiques et géométriques des objets. Devant la complexité du problème, nous avons choisi de découper l'espace des configurations en cellules élémentaires afin d'approximer ces objets par des hypervolumes simples. On comprend alors sans peine que le nombre de cellules augmente comme la puissance 6 de la résolution choisie.

L'explosion combinatoire qui en résulte interdit l'utilisation systématique de cette méthode pour le traitement de tous les degrés de liberté. Seuls les trois premiers seront donc traités de la sorte.

Recherche d'une trajectoire

La recherche d'une trajectoire dans un espace de cellules peut se ramener à un parcours de graphe, dont les nœuds sont constitués par les cellules de l'espace libre. Il existe un arc entre deux cellules s'il est possible de passer librement de l'une à l'autre. Le problème du parcours de graphe se résout alors facilement par un algorithme de type A*. A ce niveau, il faut remarquer l'importance du pas de discrétisation. S'il est trop important, le modèle de l'espace libre sera surcontraint et on risque de ne pas trouver de solution. S'il est trop fin, le coût en temps de calcul sera trop élevé. Dans la pratique, le pas de découpage n'est pas le même pour tous les degrés de liberté : le premier degré est découpé plus finement que le dernier.

Traitement des rotations du poignet

Les derniers degrés de liberté correspondent pour notre bras manipulateur aux degrés "d'orientation" du poignet. Leur prise en compte s'effectue en approximant le volume qu'ils peuvent balayer par une sphère. La première phase détermine une trajectoire en ne considérant que des problèmes de position, l'orientation étant traitée dans cette seconde phase grâce à des méthodes locales.

Les heuristiques employées imposent les contraintes suivantes :

- Les mouvements au voisinage de la position de départ et d'arrivée sont réalisés à orientation constante.
- Les réorientations éventuelles sont réparties le long de la trajectoire.

Ces heuristiques peuvent conduire à des situations d'échec avec des objets longs, cas où la sphère englobante est très grosse. La résolution de se problème passe par l'utilisation de méthode permettant de planifier des morceaux de trajectoire à partir d'informations locales [Tou87].

2.3.6 Mouvements d'assemblage

En raison des incertitudes liées à toutes manipulation, un assemblage ne peut être mené complètement à son terme en utilisant uniquement qu'une commande en position. Il faut que les derniers mouvements intègrent des opérations sensorielles afin de s'affranchir des erreurs.

Ils amènent l'objet manipulé d'un voisinage de la position finale jusqu'à celle-là. La position initiale est telle que l'effet des incertitudes est négligeable par rapport aux objets environnants.

La prise en compte de l'incertitude se fait à travers le choix des mouvements. Ils sont choisis de manière à "tasser" l'incertitude : quand on réalise un contact entre deux entités, l'incertitude absolue de position reste la même ou diminue peu tandis que l'incertitude relative est supprimée. Le planificateur de mouvements fins recherche une suite ordonnée de mouvements destinée à créer et/ou maintenir des contacts.

La méthode utilisée procéde par chaînage arrière en déduisant un plan de montage du plan de démontage. Au cours d'un démontage fictif, chaque position est qualifiée de manière symbolique et numérique (propriété géométrique et dimension de la surface des contacts par exemple).

L'ensemble de ces positions est regroupé dans un graphe qui traduira les différentes trajectoires de démontage possible. Nous extrairons ensuite une trajectoire grâce à un parcours de graphe.

Nous développerons plus en détail cette question et les solutions utilisées aux chapitres 5 et 6 en présentant l'approche mathématique et le système PAMELA.

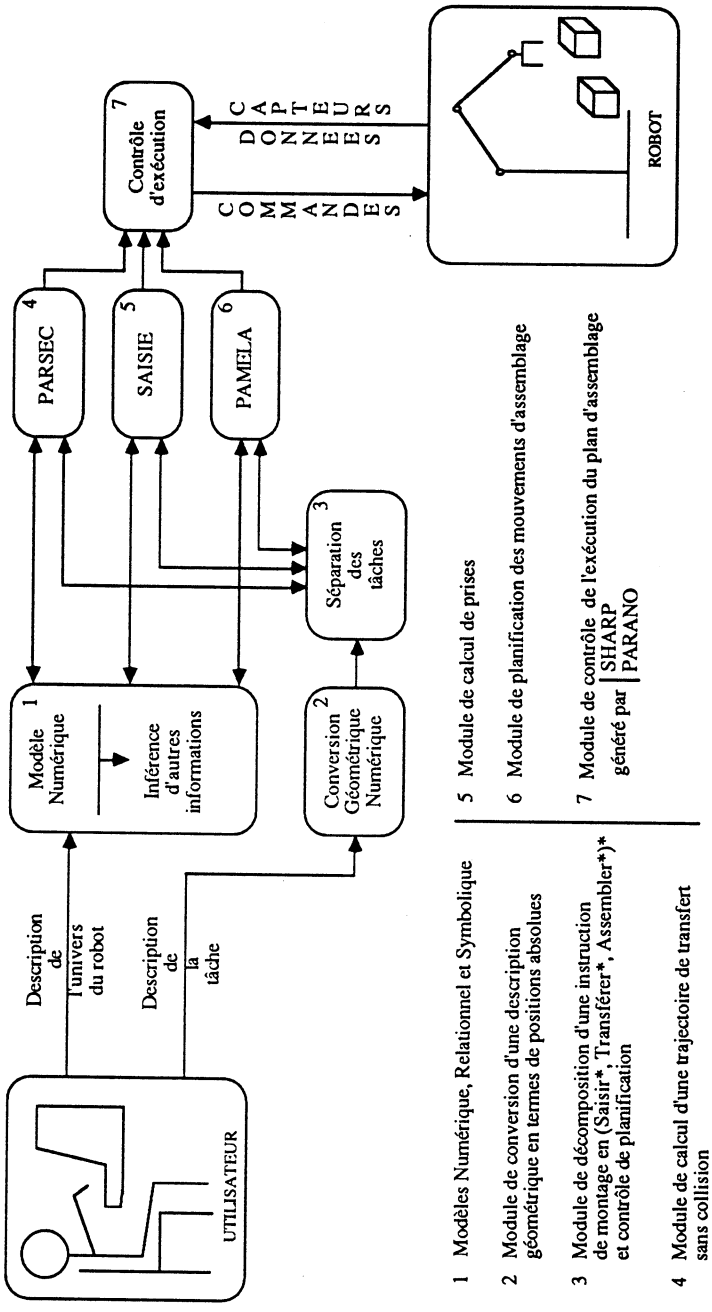


Figure 2.12: Architecture du système SHARP

Chapitre 3

Planification de mouvements de montage : problèmes et approches

3.1 Le problème abordé

A chaque instant, le modèle représente l'univers du robot. Certains paramètres comme la forme des objets ou leur dimension sont connues avec une précision satisfaisante vis-à-vis de la tâche à exécuter. En revanche, la position des objets est souvent entachée d'un terme d'erreur susceptible de faire échouer l'exécution du programme d'assemblage. Plusieurs sources d'erreurs existent. Les principales sont les suivantes :

- La précision limitée des mécanismes de localisation des objets et de l'outil terminal du robot.
- Les perturbations dues à la nature physique de l'environnement (par exemple : rotation d'objets lors de saisie).
- Les calculs ...

Un programme de manipulation constitué uniquement de position absolue à atteindre ne peut fonctionner que dans un univers parfaitement défini. En raison des

incertitudes, ce type de programme ne peut mener à bien une opération d'assemblage sans risque de collision. Pour pallier ce problème, les instructions de commandes doivent faire appel à des instructions de perception qui permettent de tenir compte de l'environnement en prenant en compte les positions réelles des objets.

L'incertitudes de position des objets se présente sous deux formes :

- L'incertitude absolue qui désigne l'imprécision de la position du repère d'un objet par rapport au repère de l'univers.
- L'incertitude relative qui désigne l'imprécision de la position d'un objet par rapport à un autre.

Cette distinction provient des méthodes devant être mises en œuvre pour évaluer puis réduire l'incertitude. Cette réduction ne peut se faire qu'en effectuant des **mesures** sur l'univers réel et en les reportant éventuellement sur le modèle. Mesurer consiste à comparer une grandeur connue à la valeur que l'on veut déterminer.

Dans le cas de l'incertitude absolue, les mesures à effectuer portent sur des repères. Ces entités immatérielles ne peuvent être explicitement manipulées. Les méthodes utilisées pour les localiser consistent à déduire leur position à partir de mesures effectuées sur des entités réelles. L'augmentation de précision apportée par ces techniques ne permet pas de réaliser des opérations d'assemblage très precises.

L'incertitude relative concerne, elle, des objets réels. Deux objets quelconques possèdent l'un par rapport à l'autre six degrés de liberté. Pour localiser un objet, il suffit de fixer les six valeurs des degrés de liberté. Le positionnement est incertain si chaque valeur ne peut être fixée exactement. La réalisation de contacts entre ces deux objets supprime certains degrés de liberté réduisant ainsi l'incertitude qui leur est liée. Ce type d'action est équivalent à une mesure où l'on compare la distance entre les entités mises en contacts. Les opérations tactiles constituent de ce fait un bon moyen pour restreindre l'incertitude relative entre objets.

Un programme d'assemblage robuste doit être constitué d'instructions de mouvements dépendantes d'opérations de perception visant à réduire l'incertitude relative par la réalisation de contact.

Définition : On appelle **mouvements fins** une suite de mouvements amenant un objet d'une position où les effets de l'incertitude sont négligeable à sa position finale d'assemblage.

La planification de mouvements fins d'assemblage a pour objet de produire automatiquement une suite d'instructions incluant ces heuristiques de réduction de l'incertitude.

3.2 Les moyens

Une technique classique pour améliorer les performances des robots consiste à utiliser des informations sensorielles. Parmi l'éventail des capteurs existants, le capteur de forces constitue une source d'informations privilégiée. Son utilisation préférentielle est due à la place prépondérante qu'occupe le sens tactile dans notre vie quotidienne.

L'utilisation conjointe de capteurs proprioceptifs et extérioceptifs permet d'aborder **la commande en force et en position.** Dans ce mode de commande, les mouvements du robot sont spécifiés par des contraintes faisant intervenir simultanément les positions de l'outil terminal et les valeurs des forces mesurées. Ce problème ouvert donne lieu à de nombreux travaux de recherche. [Gan86,Mer86].

3.2.1 Les mouvements gardés

Un mouvement gardé en force est constitué d'une instruction décrivant une trajectoire ou une direction de déplacement et d'une expression logique faisant intervenir les mesures du capteur de forces. L'évaluation de cette expression fournit un résultat vrai ou faux. Le mouvement s'arrête soit lorsque le résultat de l'expression est vrai, soit lorsque le robot a atteint une position objectif spécifiée dans la commande.

exemple : `déplacer` *robot* `selon` $\overrightarrow{V_z}$ `jusqu'à` $F_z > 40$

Ce type de commande permet de réaliser des contacts, malgré la présence d'incertitude sur la position des objets concernés. La généralisation de ce type de commande peut réclamer l'évaluation de condition d'arrêt d'expression complexe.

3.2.2 Les mouvements compliants

Le terme "compliant" et les mots qui en dérivent sont des anglicismes provenant de l'adjectif anglais "compliant". Une traduction plus précise pourrait être "en souplesse".

Un mouvement compliant est une trajectoire asservie à des données de capteurs. Parmi les six degrés de liberté que possède potentiellement l'objet manipulé, certains sont utilisés pour définir les directions des mouvements contrôlés en position par le système, d'autres sont contraints par des lois où interviennent les mesures de capteur.

exemple : **déplacer** *robot* **selon** $\overrightarrow{V_x}$
en maintenant $(20 < F_y < 40)$ **et** $(10 < F_z < 25)$

Ce type de mouvements permet de se déplacer en prenant appui sur l'univers environnant. En pratique, les mouvements gardés et compliants sont utilisés de manière complémentaire. Par exemple, on réalise un ensemble de contacts S_1 en maintenant un autre ensemble de contacts S_2 en exécutant un mouvement gardé sur S_1 et compliant sur S_2.

3.2.3 L'utilisation d'autres données sensorielles

Les recherches menées conjointement en robotique et en vision nous fournissent des outils permettant au robot de mieux percevoir son univers et ainsi d'agir en conséquence. Malheureusement, la mise en œuvre de ce type de système est coûteuse et délicate. Seules quelques applications spécifiques peuvent trouver une solution ponctuelle.

3.3 Les principales techniques de résolution

La planification automatique de mouvements d'assemblage a pour objet de produire un programme de commande de robot susceptible de s'adapter à certaines variations prévues de l'environnement.

La prise en compte des incertitudes peut alors être faites de manière explicite en raisonnant sur les domaines d'incertitude ou de manière implicite en supposant que toutes les incertitudes sont de même nature et parfaitement bornées.

Nous présentons dans la suite quatre méthodes ayant fait l'objet de publications et représentant bien les différentes démarches de résolution. Parmi celles-ci seule la méthode des pré-images est susceptible de tenir compte explicitement de l'incertitude.

3.3.1 Utilisation de schémas de programmes

Le principe de cette méthode consiste à analyser les contraintes de précision imposées par la tâche afin de choisir puis de compléter des schémas types regoupés dans une bibliothèque. Les travaux de Brooks [Bro82] ont permis de perfectionner la méthode en fournissant un moyen pour propager de manière symbolique les incertitudes dans le programme. Le mode de calcul utilisé permet d'estimer en tout point les erreurs engendrées par les actions antérieures. Il permet aussi d'évaluer les valeurs initiales qu'il est possible d'affecter à certain paramètres, sans dégrader les chances de succès du programme.

Une limitation importante du procédé réside dans la difficulté qu'il y a à apprécier certaines erreurs, et dans les techniques de propagation utilisées qui conduisent à surestimer très largement les incertitudes.

3.3.2 Raisonnement sur les pré-images

L'idée directrice de la méthode des pré-images est d'essayer de calculer un ensemble de positions qui permettent d'atteindre le but que l'on s'est fixé, en un mouvement élémentaire. Si la position initiale de l'objet manipulé ne fait pas partie de cet ensemble, on réitère le procédé jusqu'à ce qu'une position convienne. Détaillons davantage cette méthode à partir d'un exemple.

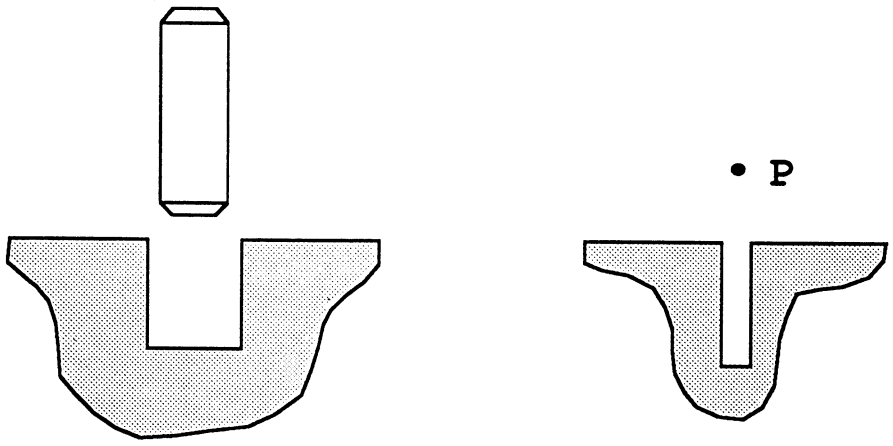


Figure 3.1: A: Insertion d'un goujon B: Problème équivalent ramené à un point

Le problème de la figure 3.1A est de réaliser l'insersion d'un goujon avec l'hypothèse que les axes du trou et du goujon sont toujours parallèles. La technique pour aborder ce problème consiste à déterminer les mouvements possibles en réduisant le goujon à la dimension d'une ligne. Le trou est réduit de la même manière. La position d'un point **p** de l'axe du goujon détermine complètement sa position à cause de l'hypothèse de parallélisme entre son axe et celui du trou. Grâce a ce grossissement des obstacles, le problème est ramené à l'insertion du point **p** dans un alésage de diamètre $D - d$ (voir figure 3.1B).

Pour ce problème reformulé, nous allons déterminer un ensemble de positions qui permettent d'atteindre le but G en un mouvement élémentaire de translation. Chacun de ces mouvements peut être représenté par un vecteur vitesse v_i. Pour chaque v_i, on calcule tous les P_i tels que si l'on applique le mouvement de vitesse v_i à un point à la position P_i, alors il atteint le but. Un tel ensemble de positions est appellé **pré-image** du but pour un vecteur vitesse donné. La figure 3.2 représente la pré-image associée au vecteur vitesse v_i.

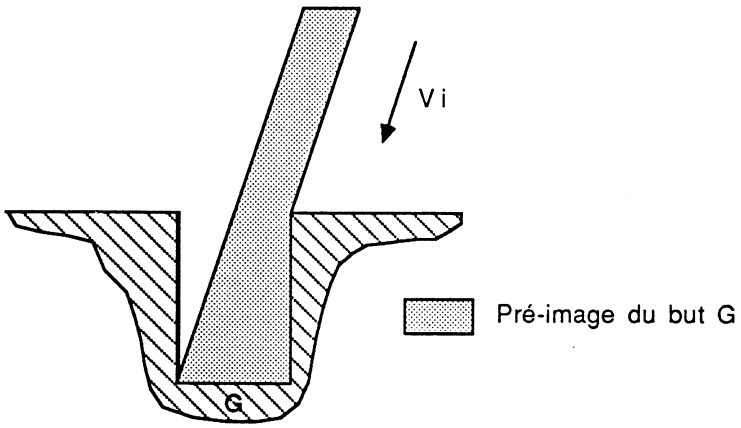


Figure 3.2: Pré-image

Dans le cas où aucune pré-image de G ne contient la position initiale du goujon, on itère le processus en considérant chaque pré-image comme un sous-but à atteindre, cela jusqu'à ce qu'on trouve une pré-image qui contienne la position initiale. A partir de la suite de pré-images (et de leurs vecteurs vitesse associés), on peut élaborer un plan d'assemblage complet. Pour cela, il faut déterminer les valeurs des capteurs permettant de caractériser les situations géométriques des positions correspondant

aux fins de mouvements. Ce calcul des conditions d'arrêt est un problème difficile dans le cas général. Le programme produit est constitué de mouvements gardés et de mouvements compliants.

Afin de prendre en compte l'incertitude sur la direction du mouvement, les auteurs ont introduit la notion de "pré-image large" et de "pré-image forte". Ces deux notions tentent de représenter respectivement l'ensemble des points à partir desquels il est **possible** d'atteindre le but suivant la direction de mouvements fixée (à l'incertitude de commande près), et ceux à partir desquels la réalisation du but est garantie. Si on note :

$P_{im}(G, \overrightarrow{V_i})$ la pré-image du but G selon la direction $\overrightarrow{V_i}$,

et $I(\overrightarrow{V})$ l'incertitude associée au vecteur $\overrightarrow{V}$ (partie ouverte et convexe de $\Re^3$).

Ces deux types de pré-image peuvent être définis comme suit :

$$large\_Pim(G, \overrightarrow{V}) = \bigcup_{\overrightarrow{V_i} \in I(\overrightarrow{V})} P_{im}(G, \overrightarrow{V_i})$$
$$forte\_Pim(G, \overrightarrow{V}) = \bigcap_{\overrightarrow{V_i} \in I(\overrightarrow{V})} P_{im}(G, \overrightarrow{V_i})$$

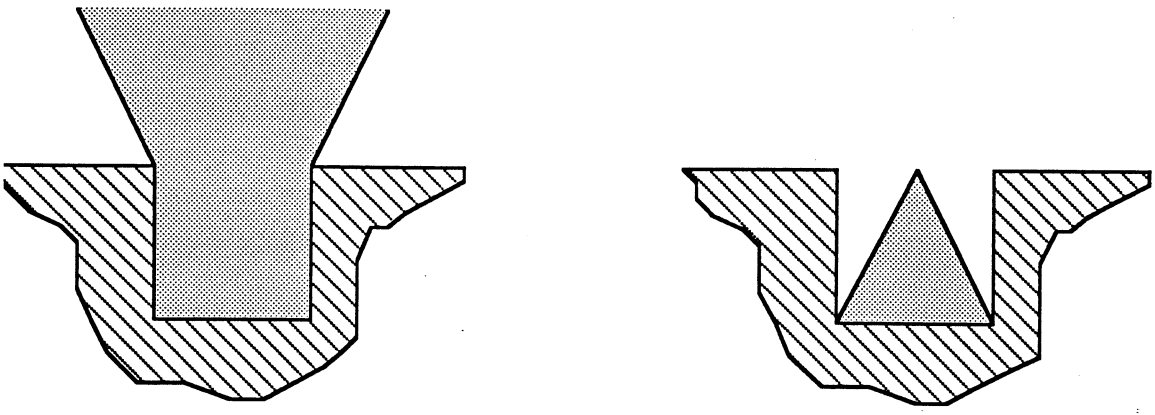


Figure 3.3: Pré-image large et pré-image forte

Cette approche permet de prendre en compte les incertitudes sur la direction des mouvements. Les incertitudes sur la géométrie des objets peuvent aussi être prises en compte grâce aux techniques de grossissement déjà mentionnées.

Les différents effets de l'incertitude sont ramenés au point P. Il se transforme alors en un petit domaine D_P contenant P. On effectue alors une majoration stricte de l'erreur en compressant D_P jusqu'à la taille d'un point comme cela a déjà été fait pour le goujon. La même opération est appliquée aux objets environnants. Le problème initial est ainsi surcontraint par l'incertitude.

Cette majoration des incertitudes ainsi que l'utilisation des pré-images fortes permet de garantir la robustesse du programme de montage produit. Il subsiste cependant le risque de ne pas trouver de solution en raison des trop fortes majorations effectuées pendant la résolution. Pour l'instant aucune implantation véritable n'a été réalisée en raison de la complexité algorithmiques de la méthode. De plus, elle est difficilement applicable aux mouvements de rotation.

3.3.3 Apprentissage à partir de traces d'exécutions

Les difficultés à raisonner explicitement sur les incertitudes ont conduit les chercheurs du LIFIA [Duf83] à développer une méthode basée sur une analyse des effets des incertitudes. Le principe de la méthode consiste à produire un plan d'assemblage, incluant des mouvements fins, en combinant les résultats de plusieurs tentatives d'exécution de la tâche. Elle procède en deux phases :

phase d'exercice : Dans cette phase, le programmeur réalise plusieurs exécutions de la tâche en connexion avec le robot. Il dispose pour cela d'un langage de commande spécialisé et d'une bibliothèque de stratégies élémentaires pour effectuer des corrections si besoin est. Chaque expérimentation, jugée significative par l'utilisateur, est enregistrée sous forme d'un graphe linéaire. Tous les choix effectués lors de cette première phase sont dus au programmeur.

phase d'induction : Les traces d'exécution enregistrées lors de la première phase sont ensuite utilisées pour synthétiser un programme d'assemblage. La méthode consiste à appliquer des transformations itératives sur le graphe obtenu par fusion des graphes linéaires de la première phase. Le graphe résultant de ces diverses transformations représente la structure de contrôle du programme et les nœuds définissent les instructions de manipulation.

Cette méthode décrite sommairement peut sembler la plus adaptée à la génération de programmes d'assemblage en raison des manipulations réelles utilisées pour la synthèse du programme finale. Elle présente cependant quelques lacunes :

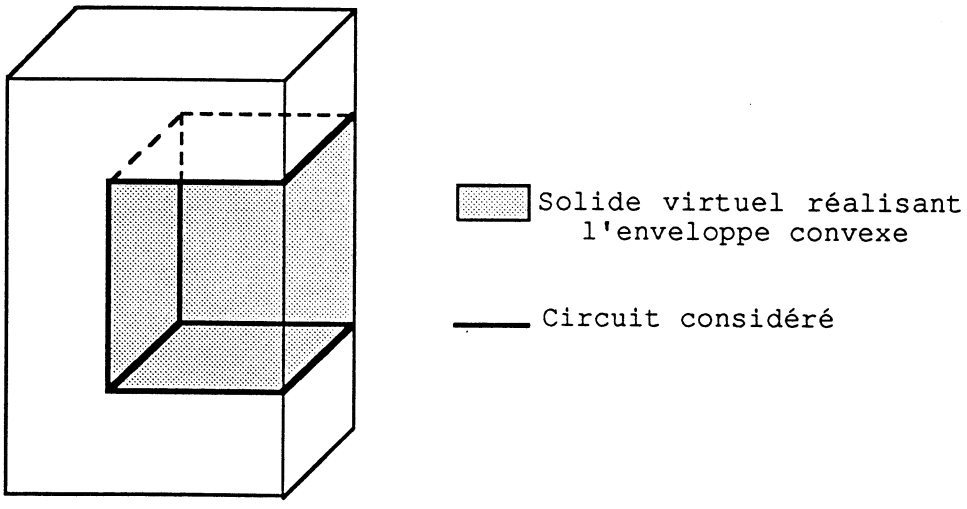
- L'hypothèse selon laquelle après N exécutions de la tâche, toutes les situations ont été rencontrées, n'est qu'un point de vue statistique.
- La reconnaissance d'une situation à partir de données de capteurs est difficile sinon impossible dans le cas général.
- L'absence de modèle classique conduit à avoir une description exhaustive des situations possibles (position, force, relation géométrique, type de mouvement).

Cette méthode d'apprentissage souffre visiblement des problèmes liés à l'utilisation d'un modèle uniquement symbolique. Le lien entre les situations symboliques et les situations réelles est difficile à établir et nécessite l'intervention humaine.

3.3.4 Raisonnement sur les enveloppes convexes

La méthode des enveloppes convexes utilise des résultats de la théorie des graphes [Val85]. Elle repose sur la constatation suivante : "S'il y a assemblage entre deux composants alors leurs enveloppes convexes s'intersectent". Il est donc possible d'isoler des zones d'assemblage et ainsi orienter la recherche d'une solution. Le système tente pour cela de démonter l'assemblage pour en déduire un plan de montage.

La méthode développée se divise en deux étapes. Tout d'abord, le système calcule l'enveloppe convexe des objets et met en évidence les *circuits* qui s'appuient sur leurs fermetures convexes. Intuitivement, cela consiste à s'intéresser aux faces communes entre l'objet et les solides virtuels réalisant l'enveloppe convexe. Le calcul des circuits s'appuie sur des techniques de la théorie des graphes.



Dans un deuxième temps, chaque circuit est utilisé pour déterminer des directions de mouvements de translation. Deux directions sont alors prises en compte : selon la direction des arêtes du circuit et selon la normale aux faces adjacentes à ces mêmes arêtes.

Cette méthode comporte deux restrictions importantes :

- Le calcul des enveloppes convexes sur des objets non-polyhédriques est difficile dans le cas général. L'étude des circuits s'appuyant sur des arêtes non rectilignes ne fournit pas suffisamment d'informations pour proposer un mouvement de démontage. Dans la pratique la méthode se restreint à des objets polyhédriques.

- L'implantation qui a été faite de la méthode, ne permet de traiter que des mouvements de translation. Cette restriction est importante car la mise en œuvre de mouvements correctifs en rotation est indissociable de l'assemblage en trois dimensions.

L'extension de la méthode aux rotations est possible selon l'auteur mais le coût de cette amélioration risque d'être très important. En revanche le traitement et la prise en compte des incertitudes, qui ne sont pas réellement abordés dans l'état actuel de la méthode, semble plus difficile à intégrer.

Les algorithmes proposés ont été utilisés avec succès pour calculer de manière théorique des trajectoires de montage de différentes pièces parmi lesquelles figurent de véritables assemblages. Une difficulté subsiste cependant : la génération d'un programme incluant la gestion de capteurs. La méthode décrite dans [Val85] ne permet pas pour l'instant la génération d'un programme réel pouvant s'exécuter sur un manipulateur.

Chapitre 4

Le système de modélisation : SIMONS

Système
Interactif de
Modelisation d'
Objets
Numériques et
Symboliques

L'étape de modélisation consiste à représenter, en utilisant un codage donné, un ensemble de données manipulable par l'intermédiaire d'un calculateur. La définition d'un système de modélisation général utilisable dans tous les domaines est sans doute illusoire. En effet chaque application a besoin d'un modèle qui lui est propre et toute tentative consistant à faire l'union des besoins de chacune se solde par une perte d'efficacité pouvant conduire à un ensemble inutilisable. Quatre questions doivent être considérées avant de spécifier les caractéristiques du système de modélisation :

- Quels sont les objets à représenter ?
- Quels type de traitement doivent-ils subir ?
- Quel type de représentation choisir ?
- Comment réaliser l'interface utilisateur ?

4.1 Les besoins de la robotique

Définir un modèle général pour la robotique est aussi sans doute trop ambitieux. Selon que l'on s'intéresse à la commande ou à la planification, les modèles choisis seront complètement différents. Notre but est de définir un système de modélisation dédié à la robotique d'assemblage, et plus spécialement à la synthèse automatique de programme de montage. Pour cela, il nous faut un modèle de l'environnement de travail sur lequel il soit possible d'automatiser le raisonnement.

Topologie des objets

Les objets appartenant à l'univers du robot sont les premiers éléments à représenter. Le modèle retenu doit permettre d'effectuer facilement des calculs géométriques sur les entités constituant les objets. Pour cela chaque entité "géométrique" (point, arête, face), est représentée par son(ses) équation(s). La structure de cet ensemble d'entités sera choisie de telle façon que l'on puisse "parcourir" de manière ordonnée un objet. Il doit exister des primitives qui à partir de l'objet permettent de trouver les composants puis les faces, les arêtes, les points. Des primitives réalisant l'opération inverse sont aussi nécessaires. Ce modèle hiérarchique peut se schématiser par un graphe $M = S \times A$.

S = { nœuds }
= { entités constituant l'objet : objet, composant, face, arete, sommets }
A = { arcs }
= { A_{ij} / Il existe un arc entre S_i et S_j si S_i "se décompose en " S_j }
∪ { A_{kl} / Il existe un arc entre S_k et Sl si S_l "se décompose en " S_k }

La relation *"S_i se décompose en S_j"* signifiant :

- S_i et S_j appartiennent au même objet.
- $D(S_i) - D(S_j) \geq 0$ où

$$\left| \begin{array}{lcl} D(objet) & = & 4 \\ D(composant) & = & 3 \\ D(face) & = & 2 \\ D(arête) & = & 1 \\ D(point) & = & 0 \end{array} \right.$$

Seules les entités telle que $D(E) \leq 2$ devront être caractérisées par la donnée des équations exactes de leur support mathématique.
Cette première couche de modélisation représente un modèle théorique parfait des objets. En pratique, les objets diffèrent de leur modèle mathématique à cause de l'incertitude. Il faut la modéliser. Deux sortes d'incertitudes peuvent être identifiées :

- L'incertitude géométrique (dimensions des objets).
- L'incertitude sur la position de l'objet complet.

La représentation de l'incertitude géométrique s'effectue en général par la donnée d'un intervalle de variation ou d'une loi de probabilité. De nombreuses représentations peuvent convenir à condition de disposer des outils permettant de manipuler les incertitudes.

Position des objets

L'incertitude en position est plus importante dans la mesure où les ordres de grandeur ne sont pas comparables (les moyens dont on dispose pour mesurer une pièce étant bien plus précis que ceux utilisés pour la localiser) et aussi parce qu'on ne sait pas réellement traiter l'incertitude de "forme".
Le positionnement des objets est fait de manière classique à l'aide de repères cartésiens. Toutes les entités géométriques sont définies par rapport à un repère absolu : le repère de l'univers. A chaque objet est associé un repère local. La donnée de la transformation entre le repère de l'univers et le repère local détermine complètement sa position. Le choix d'une représentation pour la transformation est libre (coordonnées homogènes, translation + quaternion, ...), mais on s'attachera à en choisir une le moins sensible possible aux erreurs de calculs.

Structure articulée

Parmi les objets faisant partie de l'univers du robot, les composants mêmes du robot doivent être représentés. Leur fonctionnalité conduit à une modélisation particulière. Chaque composant est lié aux autres en ne conservant que quelque degrés de liberté (en général un seul). Le $n^{ième}$ composant de la chaîne articulée qui constitue le robot est localisé par la donnée de la position du $n-1^{ième}$ composant, et des valeurs de degrés de liberté existant entre eux. Si le robot possède p degrés de liberté, une configuration du robot sera décrite de manière univoque par la donnée

de p paramètres. Le passage d'un tel p-uplet vers une position cartésienne, et réciproquement, constitue la partie cinématique du modèle.

Propriétés symboliques

Cette première modélisation quantitative du robot et de son univers n'est pas suffisante pour "raisonner". Le sens commun dont on veut doter les robots s'appuie sur un modèle qualitatif. Un certain nombre de propriétés géométriques doivent pouvoir être consultées par l'intermédiaire d'une base de données symboliques. Une représentation par l'intermédiaire des prédicats du premier ordre est bien adaptée. Les contraintes liées au modèle qualitatif sont imposées par le langage d'interrogation. Il doit fournir une réponse aux deux type de questions suivantes :

Q1: Est-ce que les entités E_1, E_2, $\cdots$, E_n, vérifient la propriété P ?

Q2: Quelles sont les entités qui vérifient la propriété P ?

De manière plus générale, le langage doit être capable de fournir une réponse à une question formée à partir de questions de type Q1 ou Q2 et d'opérateurs logiques AND, OR, NOT. Cette dernière contrainte est difficile à réaliser et demande souvent des hypothèses restrictives.

Mécanisme d'évolution dynamique

Ces trois aspects du modèle fournissent une représentation statique d'un état de l'univers du robot. La nature évolutive d'une tâche d'assemblage induit un certain nombre de contraintes supplémentaires :

1. Le fait de bouger des objets modifie simultanément la partie quantitative et qualitative du modèle.

2. Il peut exister des liaisons mécaniques rigides (temporaires ou permanentes) entre objets telles que le fait de bouger un objet implique le déplacement d'un autre.

Le premier point dû à la double représentation peut se résoudre manuellement, mais il est préférable de définir un mécanisme qui automatise le procédé. Toute modification sur une donnée du modèle se répercute ainsi dans l'ensemble du modèle. Nous proposons un tel mécanisme en 4.3.2.

Le second point provient des relations qui existent entre objets physiques. Si on considère la "liaison mécanique" comme une propriété particulière de la base symbolique, son évolution peut alors être gérée automatiquement par le même mécanisme que précédemment. Ce mécanisme de mise à jour instantanée et automatique constitue à lui seul un modèle du comportement physique de l'univers du robot.

Un modèle pour la programmation automatique comporte alors quatre composantes essentielles :

- Une représentation géométrique adaptée aux calculs géométriques.
- Une représentation cinématique du robot : changeur direct et inverse.
- Une représentation symbolique des propriétés géométriques.
- Une représentation du comportement physique, succeptible d'assurer la mise à jour automatique des différents aspects du modèle.

4.2 Les structures de données

4.2.1 Les données numériques

Deux types de représentation sont susceptibles d'être utilisés pour modéliser les entités géométriques :

La représentation de type "boundary representation" (ou BR)

Ce mode de représentation très utilisé en graphique, consiste à modéliser la surface extérieure des objets. Un objet est ainsi décrit par les surfaces qui le limitent, ces dernières étant bornées par des arêtes, elles mêmes bornées par des points. Chaque entité géométrique est caractérisée de manière univoque par une liste de paramètres numériques. Un exemple d'une telle décomposition est donné figure 4.1. Ce type de modélisation est assez bien adapté au calcul de propriétés géométriques. En revanche la décomposition immédiate d'un objet en face supprime des possibilités que permet la représentation CSG.

La représentation de type "constructive solid geometry" (ou CSG)

Avec un modèle CSG, chaque objet est défini à partir de composants élémentaires combinés entre-eux au moyen d'opérateurs ensemblistes (Cf figure 4.2).

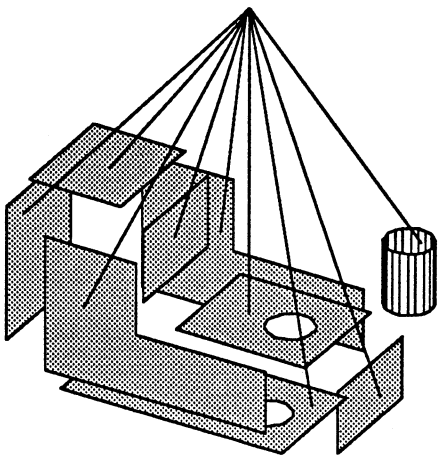


Figure 4.1: Modélisation de type BR

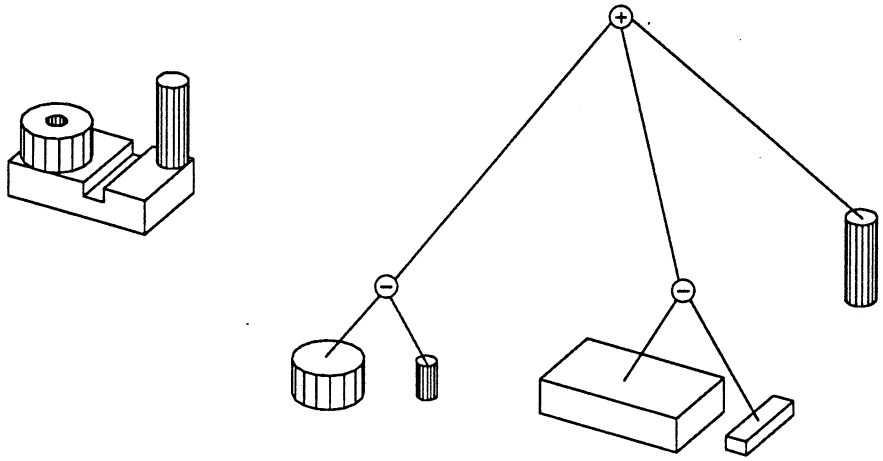


Figure 4.2: Modélisation de type CSG

Le nombre et la variété des composants élémentaires influe sur les possibilités de modélisation. La difficulté majeure liée à ce type de représentation se situe dans l'utilisation des opérateurs. S'il est facile de définir des opérateurs variés, l'exploitation des objets créés est plus délicate. L'application d'un opérateur entre deux composants peut créer, supprimer ou modifier des entités. Le calcul de propriétés géométriques à partir d'un tel modèle est difficile et celui-ci semble donc peu adapté à la robotique. Ce type de représentation présente en revanche une

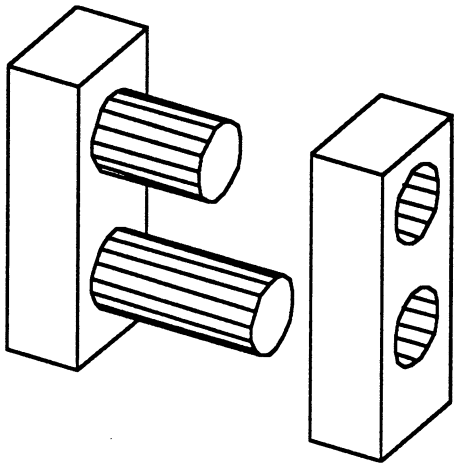


Figure 4.3: Pièce à double insertion

caractéristique intéressante : il est possible de décomposer facilement un solide en volumes élémentaires. Considerons l'exemple de la figure 4.3. Une simple décomposition de type BR permet difficilement de mettre en évidence la fonction de guidage associée aux cylindres. Avec une représentation CSG (n'utilisant que l'opérateur d'union) une telle information peut figurer dans le nœud représentant le cylindre. Qui plus est, les deux informations caractérisant le guidage peuvent "remonter" jusqu'à la racine sous forme d'une information relative à un guidage double. L'utilisation de ce genre de données pour la synthèse de trajectoire de montage permet d'affiner le choix des stratégies.

Principe de notre représentation

Afin de conserver les avantages des deux modes de représentation, nous avons adopté une solution mixte. Chaque objet est alors décomposé en composants élémentaires, décomposés eux-mêmes en face, arête, sommet. Cette approche permet à la fois de raisonner sur les propriétés "technologiques" des composants et de calculer des propriétés géométriques liées à la morphologie locale des objets. Un exemple est donné sur la figure 4.4. Des chaînages [Cam81] supplémentaires permettent de lier entre-eux les éléments géométriques voisins afin de les retrouver rapidemement pour l'évaluation des propriétés (Cf 4.5). Ces informations caractérisent la "topologie locale" (Cf 2.2.2).

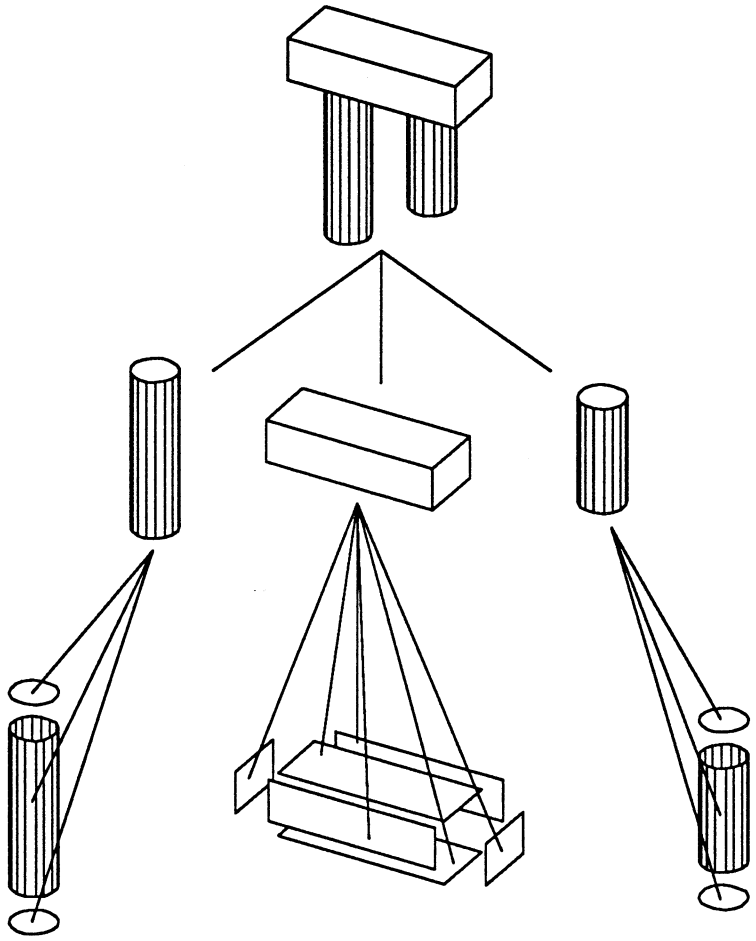


Figure 4.4: Modélisation mixte

L'exemple de la figure 4.5 illustre ce type de construction avec une arête d'un cube.

Les opérateurs Lors de la spécification d'un système de modélisation de type CSG, on est tenté de définir de nombreux opérateurs permettant les opérations les plus variées. Ainsi un opérateur comme le perçage peut sembler indispensable pour la représentation d'objets réels. Pour notre part, nous n'avons conservé que l'opérateur de collage qui permet de faire l'union de deux objets.

Les composants élémentaires La liste des composants élémentaires n'est potentiellement pas figée. Comme il n'existe pas d'opérateur de perçage, elle contient un ensemble d'objets pré-percés assemblés à la demande (Cf figure 4.6). Cet en-

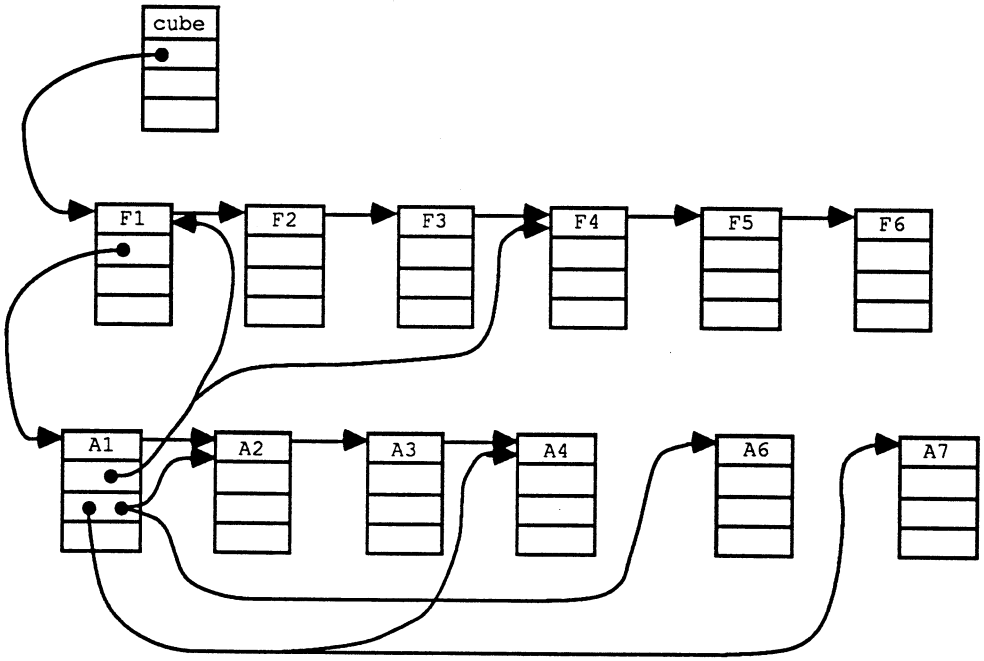


Figure 4.5: Chaînage vers les voisins

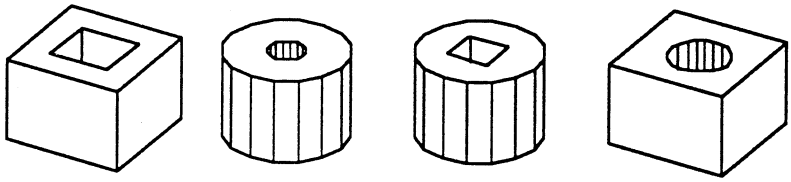


Figure 4.6: Exemple d'objets pré-percés

semble d'objets élémentaires définit une classe d'objets modélisables réduite mais suffisante pour aborder de nombreux problèmes de robotique d'assemblage.

La position des objets Toutes les positions des composants d'un objet sont données par rapport au repère propre de l'objet. La position de l'objet est définie par la transformation entre le repère de l'objet et le repère absolu de l'univers. Cette transformation est représentée à l'aide d'un vecteur de translation et d'un quaternion de rotation [PW82]. Cette représentation permet d'effectuer de longues suites

d'opérations sur les positions, en garantissant une grande stabilité des résultats vis-à-vis des erreurs d'arrondi.

4.2.2 Les données symboliques

Diverses techniques pour représenter des données symboliques ont été développées par l'intelligence artificielle. En particulier, le langage des prédicats offre un support suffisamment général pour exploiter les aspects symboliques de notre modèle.

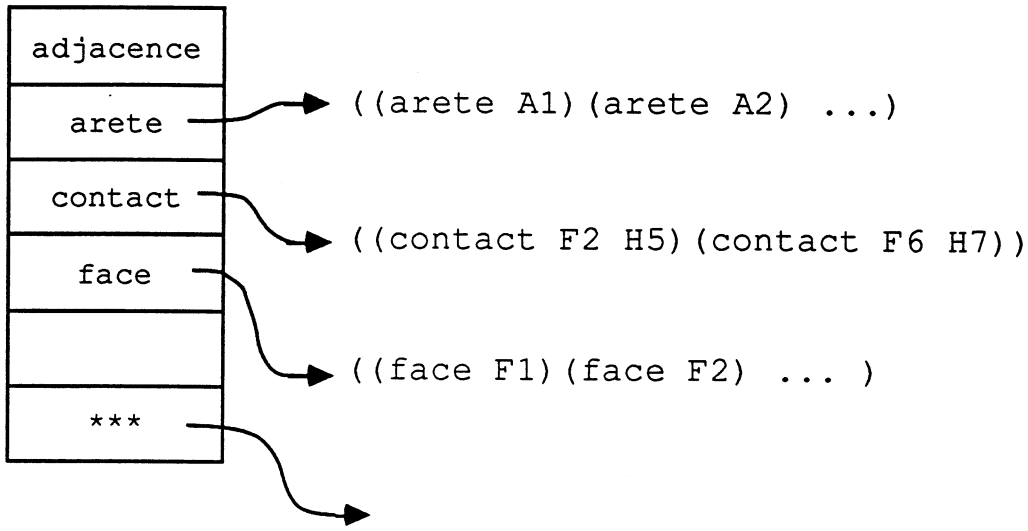
Parmi les nombreuses représentations qui peuvent convenir, on choisit celle qui permet de répondre facilement aux questions et en particulier à celle de base qui est :

$$[Pred\ ?x_1\ ?x_2 \cdots ?x_n] \equiv \text{Quel ensemble } \{x_1 x_2 \cdots x_n\} \text{ vérifie le prédicat Pred ?}$$

Naturellement, une représentation sous forme de listes s'impose à nous. A chaque fois que l'ensemble $\{x_1\ x_2 \cdots x_n\}$ vérifie un certain prédicat "Pred", il figure parmi notre ensemble de données une liste traduisant cette information :

$$Pred(x_1\ x_2 \cdots x_n) = Vrai \equiv (Pred\ x_1\ x_2 \cdots x_n)$$

Le modèle symbolique est constitué par un ensemble de listes représentant l'ensemble des prédicats qui sont vrais à un instant donné. La représentation interne de cet ensemble à peu d'importance sur le plan théorique et une simple liste de listes peut convenir. En revanche, sur le plan pratique, il est intéressant de structurer cet ensemble pour optimiser l'accès et éventuellement minimiser le nombre de comparaisons lors d'une recherche. Parmi les critères pouvant fournir un moyen de classement, une clé utilisant le nom du prédicat est sans doute le moyen le plus efficace. Toutes les listes traduisant un même prédicat seront regroupées dans une même liste et ces listes de listes seront mémorisées dans un tableau dont la clé d'accès sera le nom du prédicat.



4.3 Le mécanisme de maintien de la cohérence

4.3.1 Le principe

L'ensemble des données numériques et symboliques fournit à un instant donné un modèle figé de l'univers réel du robot. Deux types d'agents peuvent modifier l'univers réel :

- Les robots qui manipulent les objets.
- Les lois physiques (gravité, ...) qui tendent à faire évoluer les objets vers des positions stables.

Quand on modifie l'univers réel, le modèle doit aussi évoluer. Ces modifications peuvent être explicites ou implicites. Les changements explicites sont dus aux actions de l'utilisateur tandis que ceux implicites sont régis par les lois qui définissent les propriétés physiques de l'univers. S'il est possible d'effectuer une mise à jour systématique après chaque action sur le modèle, il est préférable de définir un mécanisme de maintien automatique de cohérence. Ce mécanisme constitue un modèle du comportement de l'univers du manipulateur.

Pour décrire les propriétés physiques de l'univers du robot, il suffit d'un ensemble de lois (de règles) décrivant le comportement à adopter pour une situation donnée. Ces règles sont constituées d'une condition d'activation caractérisant une situation et d'une liste d'actions à effectuer.

Les règles sont appliquées dans trois cas :

- Ajout d'une donnée dans le modèle.
- Suppression d'une donnée dans le modèle.
- Modification d'une donnée du modèle.

L'ensemble des règles est divisé en trois classes correspondant chacune à un type d'action précise. Grâce à ce classement, le nombre de règles activées par chaque action sur le modèle, est réduit. La donnée d'une règle et d'un cas d'activation constitue un "démon" [Win77].

$$\text{DEMON} \equiv \left\{ \begin{array}{c} \textit{Cas d'activation} \\ + \\ \textit{Description de situation} \\ + \\ \textit{Règles à appliquer} \end{array} \right.$$

4.3.2 Les démons

Intuitivement, un démon peut être vu comme un processus indépendant qui a une propriété à maintenir et qui s'exécute chaque fois qu'il le faut. Pour cela, il surveille en permanence l'ensemble du modèle et les conditions d'activation qui lui sont associées. Si cette image présente bien l'idée directrice du mécanisme, en revanche elle soulève de nombreux problèmes tant matériels que logiciels. Il est impossible qu'un ensemble de processus (de démons) modifie parallèlement un ensemble de données (le modèle) sans que l'unité soit compromise. Le mécanisme des démons est donc implanté de manière séquentielle avec les restrictions suivantes :

1. Les démons sont explicitement activés par le système.
2. Si plusieurs démons doivent être activés simultanément, Ils sont classés afin d'être activés séquentiellement.

D'un point de vue informatique, un démon peut être vu comme une règle attachée à un type d'opération donnée. Quand on effectue une action A sur le modèle, tous les démons qui sont attachés au type A essaient de s'activer.

Syntaxe d'écriture des démons

<demon> ::= (DEFDEMON :NAME <idf>
:WHEN <moniteur>
:CONDITION <expression d'activation>
:ACTION <liste d'actions>)

<moniteur> ::= ADD // SUP // TOUCH

<expression d'activation> ::= voir paragraphe 4.3.4

<liste d'actions> ::= (<expression lisp> { <expression lisp> })

<idf> ::= <caractère> { <caractère> }

<caractère> ::= a // b // c // ... // z //
A // B // C // ... // Z //
0 // 1 // ... // 9

Exemple

Soit P, la propriété suivante : *"S'il existe deux faces appartenant à deux objets distincts telles que ces faces soient en contact, alors ajouter l'information correspondante dans la base de connaissance"*. Le démon correspondant peut être défini comme suit à l'aide de notre langage :

```
(defdemon
    :name contact
    :when add
    :condition  (and [face ?f1]
                     [face ?f2]
                     [lier-objet-face ?o1 ?f1]
                     [lier-objet-face ?o2 ?f2])
                     (not(equal ?o1 ?o2))
    :actions    (if (contact-p ?f1 ?f2)
                    (ajouter '(contact ,?f1 ,?f2))))
```

4.3.3 Les moniteurs :when

Le moniteur est un outil logiciel permettant de s'affranchir des contraintes liées aux programmes mono-processus. Au lieu d'observer en permanence l'état du modèle, tous les démons vont "sommeiller" en attendant le signal d'un moniteur qui sélectionnera ceux qui sont concernés. Chaque opération sur le modèle devra alors communiquer au gestionaire de moniteurs les informations suivantes :

1. Quelle est la nature de l'opération?
   - Ajout d'une donnée dans la base.
   - Suppression d'une donnée de la base.
   - Modification d'une donnée de la base.

   Grâce à cette donnée, le moniteur attaché au type de l'action pourra s'activer.
2. Quelles données sont concernées par cette opération? Le moniteur sélectionné en 1 peut ainsi déclencher les démons concernés par la modification du modèle. Il les active alors de manière séquentielle.

Dans ce contexte, une propriété géométrique P n'est entièrement spécifiée qu'à travers les réponses aux trois questions suivantes :

- Quand P est-elle crée ?
- Comment P évolue-t-elle ?
- Quand P est-elle supprimée ?

La description complète du "processus de vie" de P est alors réalisée par l'écriture de trois démons correspondant aux actions suivantes :

- On ajoute des données $\longrightarrow$ moniteur ADD
- On supprime des données $\longrightarrow$ moniteur SUP
- On modifie des données $\longrightarrow$ moniteur TOUCH

4.3.4 Le mécanisme d'activation

Rappel sur la "mise en correspondance"

Dans ce qui suit, nous allons utiliser le mécanisme de la "mise en correspondance" (ou matching) pour déclencher nos démons. La mise en correspondance consiste à chercher les substitutions minimales à effectuer sur deux expressions pour qu'elles soient identiques. Les substitutions s'appliquent sur les caractères spéciaux suivants :

? qui **doit** être substitué par un symbole quelconque.

$\star$ qui **peut** être substitué par un ensemble, éventuellement vide, de symboles.

?x qui **est** substitué par un symbole quelconque fixant alors la valeur de la variable.

Le résultat d'une opération de mise en correspondance est soit FAUX (si rien ne correspond), soit la liste des couples (variable + valeur) substitués pour établir l'égalité. Un résultat à NIL signifie qu'il y a correspondance sans utilisation de variables.

On notera $[x_1\ x_2 \cdots x_n]$ une expression qui doit être mise en correspondance avec chaque terme de la base de données symboliques. Le résultat sera la liste des résultats significatifs de chaque mise en correspondance.

Exemple : `Base=((P a b)(P a c)(P a d)(Q a b)(Q s t))`

```
[P e f]    →  Faux
[P a ?x]   →  (((?x b))((?x c))((?x d)))
[Q a b]    →  nil
[Q ?x ?y]  →  (((?x a)(?y b))((?x s)(?y t)))
```

L'expression d'activation

L'expression d'activation a un double rôle :

- elle détermine si un état d'activation est vérifié ou non.
- elle founit un ensemble de {variable + valeur } qui peuvent être utilisées par les actions du démon.

Sa syntaxe est la suivante :

<expression d'activation> ::= <facteur> //
(<opérateur>
<expression d'activation>
{<expression d'activation>})

<facteur> ::= <expression lisp> //
<expression de matching> //
(NOT <expression de matching>)

<opérateur> ::= AND // OR

<expression de matching> ::= [<idf> { <terme> }]

<var> ::= ? <idf>

<terme> ::= ⋆ // ? // <var> // <idf>

En plus d'une écriture syntaxiquement correcte, il faut que l'expression vérifie quelques règles sémantiques liées au mélange d'expressions complètement connues et d'expressions dont les paramètres doivent être recherchés dans la base. Ces règles sont dues à l'hypothèse d'implantation suivante : *"Chaque expression est représentée sous forme d'un arbre où les nœuds sont les opérateurs et les feuilles les opérandes. L'évaluation des termes se fait de gauche à droite et en profondeur d'abord"*. L'expression d'activation doit alors vérifier les règles suivantes :

1. Un opérande de type <*expression de matching*> permet de fixer la valeur des variables qu'il comporte.

2. Une expression LISP comportant des variables ne peut être évaluée que si elles ont toutes pu être associées à une valeur en étudiant le même niveau d'arborescence ou une partie gauche d'arbre[1].

[1]Le système d'évaluation utilise la propriété de commutativité des opérateurs OR et AND pour réordonner les arguments de telle sorte que les expressions de mise en correspondance soient traitées en priorité.

- (OR [P ?X](FN ?X ?Y)[Q ?Y]) ⟶ peut s'évaluer
- (OR (OR [P ?X](FN ?X ?Y))[Q ?Y]) ⟶ est impossible à évaluer

Plus simplement, les valeurs des variables doivent être connues au moment du calcul d'une expression car le système est incapable d'étudier le reste de l'expression pour déterminer les valeurs manquantes.

3. L'opérateur NOT ne peut s'appliquer à une expression de matching que si toutes les variables sont instanciées.

Parmi ces remarques, seul le point 3 constitue une restriction. Les autres remarques impliquent uniquement une certaines rigueur lors de l'écriture.

Le mélange des types d'opérandes complique l'évaluation d'un résultat et demande que l'on précise exactement le mécanisme de calcul. Une présentation plus détaillée de celui-ci sera faite dans le chapitre 6 lors de la présentation du mécanisme d'évaluation des règles de planification. Nous expliquerons alors la raison pour laquelle l'opérateur NOT ne peut être appliqué qu'à une expression de mise en correspondance où toutes les valeurs des variables sont connues.

L'activation des démons

L'activation d'un démon ne peut se faire que si les deux conditions suivantes sont vérifiées :

1. La donnée traitée par le moniteur peut être mise en correspondance avec un terme de la condition d'activation.
2. L'évaluation complète de la condition en considérant toute la base de données retourne une liste de contexte d'activation. Cette évaluation se fait en tenant compte du résultat de la mise en correspondance effectuée en 1.

Si ces deux conditions sont impératives, elles ne précisent cependant pas complètement le mécanisme d'activation. Lors de la recherche d'une mise en correspondance possible entre un terme de l'expression d'activation et la donnée traitée, plusieurs stratégies peuvent être adoptées :

1. On considère uniquement la première correspondance possible.
2. On active le démon pour toutes les correspondances possibles.

3. On effectue un choix "intelligent" parmi les correspondances possibles.

La meilleure des stratégies est évidemment la dernière, mais sa mise en œuvre pose cependant des problèmes difficiles à résoudre. Cette nécessité d'un choix de stratégie intervient dans les expressions d'activation où il y a plusieurs termes qui peuvent être mis en correspondance. Montrons pourquoi les stratégies 1 et 2 ne peuvent être appliquées systématiquement.

Soient E une expression d'activation sur la $\mathcal{B}ase$ et $\mathcal{T}$(E) l'ensemble des opérandes de type "mise en correspondance" qui la composent.

$$\mathcal{T}(\mathrm{E}) = \{ \text{ opérande de mise en correspondance de E } \} = \{ t_i, i \in [1 \cdots\cdots \mathrm{n}] \}$$

On note :

- ${}^{t}/_{S}$ un opérande de type "mise en correspondance" pour lequel la substitution S à été effectuée.
- EV(t_i) le résultat de l'évaluation d'un élément $t_i \in \mathcal{T}$(E).

  EV(t_i) est un ensemble de liste de couples de variables et de leur valeur. EV(t_i) = { S_{ij} } avec :

$$S_{ij} = \left\{ \begin{array}{l} (< nom\ de\ variable >, < valeur >) \\ tel\ que\ \exists D_0 \in \mathcal{B}ase\ v\acute{e}rifiant\ {}^{t_i}/_{S_{ij}} = D_0 \end{array} \right\}$$

Considérons une expression d'activation E contenant plusieurs (au moins deux) opérandes de type "*mise en correspondance*". Soient t_i et t_j deux de ces opérandes. t_i,$t_j \in \mathcal{T}(E)$.

Méthode d'activation numéro 1 :

Si t_i est le premier terme étudié lors de l'évaluation.
et $H = EV(t_j) - (EV(ti) \cap EV(tj)) \neq \emptyset$

Alors la stratégie consistant à ne prendre en compte qu'un seul terme de mise en correspondance possible, élimine des cas d'activations. Toutes les substitutions possibles de H ne sont jamais prises en compte.

Méthode d'activation numéro 2 :
S'il existe deux substitutions S_{ik} et S_{jl} appartenant aux ensembles solutions respectifs de t_i et t_j tels que :

$$t_i/_{S_{ik}} = t_j/_{S_{jl}}$$

$$\text{et} \quad \text{EV}(t_i) \cap \text{EV}(t_j) = \{ S_{ij}^1, S_{ij}^2, \cdots, S_{ij}^n \} \neq \emptyset.$$

Alors dans le cas d'une application systématique, les substitutions S_{ij}^p , $1 \leq p \leq n$ multiplient inutilement les activations du démon. En effet le système d'activation va déclencher le démon au moins deux fois pour chacune des S_{ij}^p.

4.3.5 Les actions :actions

Le traitement des variables

Les actions associées à un démon sont formées d'instructions LISP.
Une fois que le démon est réveillé et qu'il a déterminé un ensemble de situations d'activation, il exécute pour chacune d'elles l'ensemble d'instructions associées à sa partie "actions".
L'utilisation des variables est bien entendu possible à condition que celles-ci aient été instanciées lors de l'évaluation de la condition d'activation.

Le traitement de la récursivité

Il est toujours possible d'activer un ou plusieurs autres démons par l'intermédiaire des instructions contenues dans la partie "actions" d'un démon et ainsi créer des appels récursifs. Cette possibilité offre une grande souplesse pour la description des démons. En contrepartie, il faut qu'il existe un dispositif contrôlant l'arrêt des appels récursifs. Considérons à titre d'exemple le démon suivant :

```
(defdemon
    :name       exemple
    :when       touch
    :condition  [objet ?o]
    :actions    ((touch '(* ?o * ))))
```

Ce démon a pour but de répercuter une action portant sur un objet `?o`, sur toutes les données symboliques faisant intervenir `?o`. Des démons ad hoc se chargent ensuite du traitement spécifique à chaque donnée. Il peut être défini intuitivement par la phrase : *"Si on bouge un objet* `?o`*, alors bouger tout se qui s'y rapporte "*.

Le formalisme proposé permet à l'utilisateur de traduire "naturellement" sa pensée mais il risque de déclencher des appels intempestifs.

L'accès à l'objet `cube` (`touch '(objet cube)`) activera le démon qui déclenchera l'action (`touch '(* cube *)`) susceptible d'atteindre à nouveau l'objet `cube`. Le démon serait alors pris dans une boucle infinie.

Pour pallier ce problème, les moniteurs pouvant provoquer des boucles infinies gardent une trace des appels de démons ainsi que de leurs paramètres. Ils vérifient à chaque appel qu'il ne s'agit pas d'un cas déjà traité. Ce problème concerne essentiellement les actions de type TOUCH qui contrôle les accès à la partie symbolique de la base de données, sans modification de son contenu.

4.4 Exemple de problèmes de l'activation

Montrons les lacunes des différentes méthodes d'activation à partir d'un exemple. Pour cela, nous allons développer le mécanisme de maintien des "liaisons" entre objets. Deux repères d'objet sont *"liés"* s'il existe une liaison rigide entre eux. Cette propriété présente deux caractéristiques :

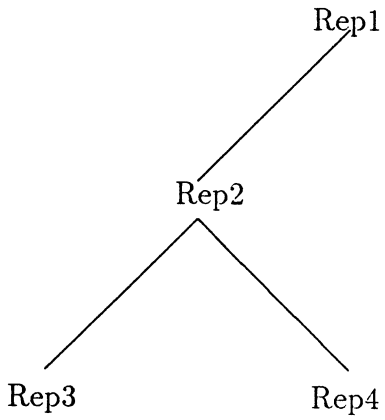
Transitivité : (A lié_à B) $\wedge$ (B lié_à C) $\Longrightarrow$ (A lié_à C) .

Complétude : (A lié_à B) $\wedge$ (A lié_à C) $\Longrightarrow$ (B lié_à C).

On se propose de modéliser ces propriétés par l'intermédiaire de deux démons: Transitivité et Complétude. L'éxécution pas-à-pas de leur activation permettra de mettre en évidence la difficulté du mécanisme de contrôle.

```
(defdemon                              (defdemon
  :name Transitivité                     :name Complétude
  :when add                              :when add
  :condition (and [lié ?x ?y]            :condition (and [lié ?x ?y]
                  [lié ?y ?z])                           [lié ?z ?y])
  :actions ((ajouter '(lié ,?x ,?z))))   :actions ((ajouter '(lié ,?x ,?z))))
```

Soit le contexte d'application suivant :



ⓐ ajouter '(lié Rep2 Rep1)

ⓑ ajouter '(lié Rep3 Rep2)

ⓒ ajouter '(lié Rep4 Rep2)

Les liaisons initiales sont modélisées grâce aux instructions a, b, c. Soit Δ le symbole désignant l'opérateur de mise en correspondance entre deux listes.

Scénario 1 : Considérons tout d'abord l'activation basée sur la première correspondance possible.

B = {}
action (a) $\longrightarrow$ B = {(lié Rep2 Rep1)}
action (b) $\longrightarrow$ B = {(lié Rep2 Rep1)(lié Rep3 Rep2)}

Le démon *"Transitivité"* se réveille alors :

[lié ?x ?y] Δ (lié Rep3 Rep2) $\longrightarrow$ C_1 = (((?x Rep3)(?y Rep2)))
L'expression (and [lié ?x ?y][lié ?y ?z]) s'évalue alors sur toute la base en prenant en compte le contexte C_1. Elle engendre comme résultat la liste (((?x Rep3)(?y Rep2)(?z Rep1))) ce qui a pour effet d'ajouter le prédicat (lié Rep3 Rep1) à la base.
Effectuons la même opération en inversant l'ordre d'application de a et b.

B = {}
action (b) $\longrightarrow$B = {(lié Rep3 Rep2)}
action (a) $\longrightarrow$B = {(lié Rep3 Rep2)(lié Rep2 Rep1)}

Le démon *"Transitivité"* se réveille alors :

[lié ?x ?y] Δ (lié Rep2 Rep1) $\longrightarrow$ C_2 = (((?x Rep2)(?y Rep1)))

L'expression (and [lié ?x ?y][lié ?y ?z]) s'évalue sur toute la base en considérant le contexte C_2. Le résultat est faux, il n'y a donc aucune activation. Si t_1 = [lié ?x ?y] et t_2 = [lié ?y ?z] :

$$H = EV(t_2) - (EV(t_1) \cap EV(t_2)) = (((?y\ Rep2)(?z\ Rep1)))$$

L'évaluation de la condition d'activation en prenant en compte les valeurs de H aurait permis une activation correcte du démon.

Scénario 2 : Une solution simple à ce problème consiste à essayer de faire une mise en correspondance entre la donnée traitée et tous les termes de l'expression d'activation. Recommençons la dernière manipulation avec cette nouvelle méthode :

Le démon *"Transitivité"* se réveille

[lié ?x ?y] Δ (lié Rep2 Rep1) $\longrightarrow$ (((?x Rep2)(?y Rep1)))

L'expression (and [lié ?x ?y][lié ?y ?z]) s'évalue a faux.

[lié ?y ?z] Δ (lié Rep2 Rep1) $\longrightarrow$ (((?y Rep2)(?z Rep1)))

L'expression (and [lié ?x ?y][lié ?y ?z]) s'évalue alors à : (((?x Rep3)(?y Rep2)(?z Rep1))) ce qui a pour effet d'ajouter le prédicat (lié Rep3 Rep1) à la base.
Le mécanisme d'activation a cette fois fonctionné correctement. A la vue de cet exemple, la seconde méthode semble être le bon mécanisme d'activation. Il est cependant facile de trouver un exemple qui le mette en échec.

Scénario 3 : Considérons maintenant le démon qui maintient à jour la propriété

de "Complétude".

```
                B = {}
action (b) ⟶ B = {(lié Rep3 Rep2)}
action (c) ⟶ B = {(lié Rep3 Rep2)(lié Rep4 Rep2)}
```

Le démon *"Complétude"* se réveille alors :

[lié ?x ?y] Δ (lié Rep4 Rep2) $\longrightarrow$ (((?x Rep4)(?y Rep2)))

L'expression (and [lié ?x ?y][lié ?z ?y]) s'évalue alors sur toute la base. Son résultat est (((?x Rep4)(?y Rep2)(?z Rep3))), ce qui ajoute le prédicat (lié Rep4 Rep3) à la base. De même

[lié ?z ?y] Δ (lié Rep4 Rep2) $\longrightarrow$ (((?y Rep2)(?z Rep4)))

L'expression (and [lié ?x ?y][lié ?z ?y]) s'évalue alors sur toute la base en produisant (((?x Rep3)(?y Rep2)(?z Rep4))) ce qui ajoute le prédicat (lié Rep3 Rep4) à la base.

En faisant un test d'activation sur tous les termes de la condition d'activation, le démon s'active deux fois. Dans ce cas particulier on pourrait concevoir soit de faire figurer deux fois l'information en raison de la commutativité du prédicat "lié", soit vérifier avant l'ajout d'une donnée dans la base qu'elle n'y figure pas déjà.

Conclusion : Cet exemple a mis en évidence les difficultés liées à l'activation des démons. Des solutions ponctuelles peuvent être trouvées mais le cas général ne pourra être résolu. Il faut alors faire un choix en connaissant les contraintes qu'il implique. Pour notre implantation, nous avons choisi d'effectuer l'activation pour toutes les mises en correspondance possibles entre les termes de l'expression considérée et la donnée manipulée.

4.5 Coût

La représentation des propriétés physiques de l'univers d'un robot par l'intermédiaire des démons, est un outil puissant et assez bien adapté à la modélisation en

vue de la résolution des problèmes de la programmation automatique des robots d'assemblages.

L'utilisation systématique de ce moyen est cependant compromise par le coût élevé des calculs annexes liés au déclenchement des démons. Pour toute donnée ajouter, supprimer, modifier, dans le modèle il faut effectuer la mise en correspondance entre elle et chaque terme constituant l'expression d'activation de chaque démon associé au type de l'opération réalisée. Une fois cette première sélection effectuée, pour chaque terme de l'expression d'activation de tout démon retenu il faut essayer de faire une mise en correspondance avec chaque donnée symbolique de la base. Le coût d'une opération élémentaire de mise en correspondance étant élevé on conçoit que le coût total est très élevé.

Si N représente le nombre de démons concernés par l'opération effectuée sur le modèle, N_{tm} le nombre moyen de termes de mise en correspondance dans une condition d'activation et N_b le nombre de données symboliques dans la base, chaque action sur le modèle déclenchera :

$$N \times (1 + N_{tm} \times N_b) \text{ opérations de mise en correspondance.}$$

Il est possible de réduire ce terme en en structurant la base de données symboliques (on diminue alors le terme N_b). Cette réduction reste cependant superficielle et ne résout pas le problème.

Chapitre 5

Résolution analytique du problème : méthodes de calcul et limitations algorithmiques

Dans ce chapitre, nous allons formaliser le problèmes des mouvements fins en établissant les différentes équations permettant de calculer des trajectoires de montage.

5.1 Les problèmes abordés

5.1.1 Formalisation des mouvements contraints

La nature des systèmes d'équations modélisant les mouvements possibles est liée au type de mouvements retenus.
Les incertitudes diverses conduisent à employer des mouvements compliants et des mouvements gardés pour effectuer les opérations de montage. Les mouvements gardés impliquent des conditions de fins de déplacements et les mouvements compliants influent sur la forme de la trajectoire.
Il est donc nécessaire de pouvoir spécifier des contraintes qui portent sur la géométrie de ces trajectoires. Dans le cadre des mouvements fins d'assemblage, ces contraintes sont exprimées par des contacts à maintenir. Une approche mécanique "classique" de ce problème consiste à exprimer le fait que le vecteur vitesse du

mobile appartient au plan local tangent du contact.

Cette mise en équation conduit à écrire un système d'équations différentielles où interviennent les dérivées partielles du mouvement et les équations des entités en contact. Il faut alors résoudre ces systèmes pour obtenir le mouvement. Cette résolution est complexe et ne peut souvent se faire qu'en faisant appel à des techniques d'analyse numérique.

5.1.2 Calcul d'amplitude de mouvements

Le calcul de l'amplitude possible d'un mouvement est une étape indispensable à la planification des mouvements fins de montage. La complexité du calcul dépend alors de notre connaissance éventuelle de la nature du mouvement.

Quand la nature du mouvement est connue, calculer le débattement possible consiste à "introduire" les équations de l'entité géométrique déplacée. Le problème réside alors dans la recherche de l'amplitude minimum du déplacement telle que le système constitué des équations des entités fixes et mobiles admette une solution.

Si la nature du mouvement n'est pas connue explicitement (cas de la résolution numérique d'un système différentiel), alors le calcul de l'amplitude doit se faire simultanément à la résolution. A chaque pas de convergence, il faut vérifier si la situation géométrique a changé ou non. Le coût du calcul est alors très élevé.

Nous étudierons au paragraphe 5.5 le principe de calcul dans le cas où les équations sont connues.

5.2 Formulation analytique : Notations et définitions

5.2.1 Le modèle des objets

Définissons de manière plus formelle ce que sont les entités géométriques d'un objet. Soit S une partie de l'ensemble des surfaces géométriques de l'espace à trois dimensions. Les éléments de S sont des surfaces que l'on peut trouver dans la constitution d'un objet de type polyédre généralisé : plan, cylindre, cône. Seuls des morceaux de surfaces font partie de l'objet considéré car la pluspart de ces surfaces mathématiques sont infinies. Soit $L = S \times S$, l'ensemble constitué des courbes définies par l'intersection de deux surfaces. On définit les segments de L, SL, comme la donnée de deux points et d'un élément de L qui les relient. $SL = P \times P \times L$ où P est l'ensemble $\Re^3$.

- Un sommet est un point qui appartient à au moins trois faces de l'objet. L'ensemble des sommets de l'objet O est noté : $\mathcal{P}(O)$ par analogie avec les points.
- Une arête d'objet est définie par la donnée d'un élément de L caractérisant sa nature. Si l'élément $l \in L$ est infini (droite par exemple), la définition de l'arête se fait par la donnée d'un élément de SL. L'ensemble des arêtes est noté : $\mathcal{A}(O)$.
- Une face d'objet est définie par la donnée d'un élément de S précisant la nature de la surface. De même que pour les arêtes, si la surface n'est pas bornée (cas de la sphère), la définition est complétée par la donnée d'un ensemble d'éléments de $\mathcal{A}(O)$ définissant les bords de la surface considérée. L'ensemble des faces de O est noté $\mathcal{F}(O)$.

5.2.2 Le mouvement

Un mouvement d'objets peut être représenté par une fonction de $\Re^4 \longrightarrow \Re^3$, définie comme suit :

$$M: \begin{cases} \Re^4 & \longrightarrow & \Re^3 \\ (x,y,z,t) & \longmapsto & (x,y,z) \end{cases}$$

Où (x,y,z) représente un point de l'espace cartésien 3D où sont définis les solides que l'on considère, et t est un paramètre représentant l'amplitude du mouvement. Nous définisons alors un mouvement par la donnée de 3 fonctions permettant de calculer chacune des coordonnées.

$$M: \begin{cases} \Re^4 & & & \longrightarrow & \Re^3 \\ (x,y,z,t): \begin{cases} \Re^4 & \longrightarrow & \Re \\ (x,y,z,t) & \longmapsto & \varphi_x(x,y,z,t) \\ \Re^4 & \longrightarrow & \Re \\ (x,y,z,t) & \longmapsto & \varphi_y(x,y,z,t) \\ \Re^4 & \longrightarrow & \Re \\ (x,y,z,t) & \longmapsto & \varphi_z(x,y,z,t) \end{cases} & \longmapsto & \left| \begin{array}{l} \varphi_x(x,y,z,t) \\ \varphi_y(x,y,z,t) \\ \varphi_z(x,y,z,t)) \end{array} \right. \end{cases}$$

On note M(t,p), l'image du point p par un déplacement M d'amplitude t.

Propriété : $\forall p_1, p_2 \in \Re^3, \forall t \in \Re \parallel \overline{p_1 p_2} \parallel = \parallel \overline{\mathcal{M}(t,p_1), \mathcal{M}(t,p_2)} \parallel$

Définition 1 : Un mouvement m est qualifié de nul si et seulement si :

$$\forall t_1, t_2 \in \Re, \forall p \in \Re^3, \qquad m(t_1, p) = m(t_2, p).$$

Définition 2 : Deux mouvements m_1 et m_2 sont dits équivalents si et seulement si :

- m_1 et m_2 ne sont pas des mouvements nuls.
- $(\Im(m_1) \subset \Im(m_2)) \vee (\Im(m_2) \subset \Im(m_1))$ où
  $\Im(m) = \{p \in \mathcal{R}^3, tel\ que\ \exists\ q \in \mathcal{R}^3, \exists\ t \in \mathcal{R}\ et\ p = m(t, q)\}$

Exemple de mouvement hélicoïdal

En robotique, on utilise habituellement des transformations pour exprimer des déplacements mais celles-ci ne fournissent hélas pas d'information sur la trajectoire des éléments déplacés. Développons comme exemple, le mouvement en forme d'hélice autour de l'axe Oz.

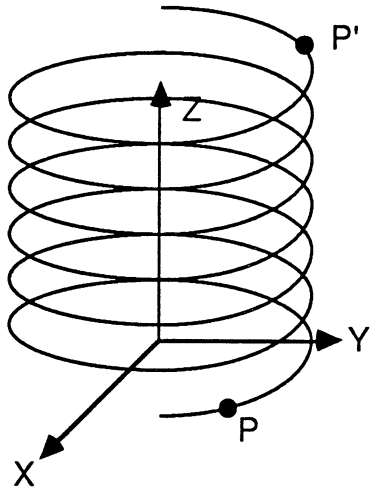


Figure 5.1: Mouvement hélicoïdal

L'équation classique d'une hélice de rayon R et de pas h est :

$$\begin{cases} x &= R \times \cos t \\ y &= R \times \sin t \\ z &= h \times t \end{cases}$$

Ce système ne définit qu'une courbe et pas le mouvement de vissage. Le diamètre de l'hélice doit dépendre de la distance entre l'axe Oz et le point considéré. Les équations s'obtiennent facilement en faisant dépendre le rayon R des coordonnées du point à déplacer.

Soit $P_0(x_0, y_0, z_0)$ le point auquel on veut appliquer un vissage d'axe O_z et de pas h. P_0 se trouve à une distance $R = \sqrt{x_0^2 + y_0^2}$ de l'axe O_z et vérifie :

$$\begin{cases} x_0 &= R \times \cos\theta_0 \\ y_0 &= R \times \sin\theta_0 \\ z_0 &= \frac{2\pi}{h} \times \theta_0 \end{cases}$$

Le point $P(x, y, z)$ obtenu en appliquant à P_0 un mouvement de vissage d'amplitude t vérifie le système :

$$\begin{cases} x &= R \times \cos(\theta_0 + t) \\ y &= R \times \sin(\theta_0 + t) \\ z &= \frac{2\pi}{h} \times (\theta_0 + t) \end{cases}$$

D'où en développant :

$$\begin{cases} x &= R \times \cos\theta_0 \times \cos t - R \times \sin\theta_0 \times \sin t \\ y &= R \times \sin\theta_0 \times \cos t + R \times \cos\theta_0 \times \sin t \\ z &= \frac{2\pi}{h} \times \theta_0 + \frac{2\pi}{h} \times t \end{cases}$$

Ce qui donne après simplification et changement de variable :

$$M: \begin{cases} \varphi_x(x, y, z, t) &= x \times \cos t - y \times \sin t \\ \varphi_y(x, y, z, t) &= x \times \sin t + y \times \cos t \\ \varphi_z(x, y, z, t) &= z + \frac{h}{2\pi} \times t \end{cases}$$

On définit ainsi une famille d'hélices représentant un mouvement de vissage autour de Oz.

Si on s'intéresse au même mouvement mais autour d'un axe quelconque passant par le point ω et de vecteur directeur $\overrightarrow{R}$, les équations deviennent plus complexes car il faut alors combiner des tranformations à ces équations de mouvements. Soit $\mathbf{T}$ la transformation amenant le point ω en O et le vecteur $\overrightarrow{R}$ sur l'axe Oz.

$$\mathbf{T}: \begin{cases} \Re^3 & \longrightarrow & \Re^3 \\ (x, y, z) & \longmapsto & Rot_{\omega, \overrightarrow{R}}(x, y, z) \circ Transl_{\omega \to O}(x, y, z) \end{cases}$$

Les équations représentant le mouvement peuvent s'obtenir en appliquant $\mathbf{T}$ puis $\mathbf{M}$ et enfin $\mathbf{T^{-1}}$. Pour conserver la cohérence des dimensions des espaces manipulés, on définit :

$$\mathbf{T^{\star}}: \begin{cases} \Re^4 & \longrightarrow & \Re^4 \\ (x,y,z,t) & \longmapsto & \mathbf{T}^{\star}(x,y,z,t) = (\mathbf{T}(x,y,z),t) \end{cases}$$

On peut ainsi définir le mouvement généralisé correspondant à un mouvement de vissage autour d'un axe quelconque par :

$$Mg: \begin{cases} \Re^4 \longrightarrow \Re^3 \\ \mathbf{T^{-1}} \circ \mathbf{M_0} \circ \mathbf{T}^{\star} \end{cases}$$

Où M_0 est un élément distingué de la famille des mouvements considérés pour lequel l'ensemble des équations est facile à établir.

5.2.3 Les contacts

- Deux entités géométriques e_1 et e_2 sont en **contact** si et seulement si :
  - $e_1 \cap e_2 \neq \emptyset$.
  - $\exists \partial t \in \Re$ et au moins deux mouvements $m_1(t)$, $m_2(t)$ non équivalents tels que : Si e_j^i représente l'entité e_j déplacée par le mouvement $m_i(\partial t)$,

$$\forall i \in [1,2], \qquad e_1^i \cap e_2^i \neq \emptyset.$$

    Cette formulation permet d'éliminer les entités géométriques qui se touchent mais dont la stabilité de la zone de contact n'est pas garantie vis-à-vis de l'incertitude.

- On note $C(e_i, e_j)$, un contact entre l'entité fixe e_i et l'entité mobile e_j.

- On désigne par $EN(C)$ l'ensemble des normales au contact $C(e_1, e_2)$. Dans le cas des entités géométriques que nous utilisons, $EN(C)$ est caractérisé par le tableau suivant :

$e_1 \rightarrow$ e_2 $\downarrow$	*point*	*arête* vecteur directeur $= \overrightarrow{A_1}$	*face*
point	$\emptyset$	$\emptyset$	{Normales extérieures à e_1 au point de contact.}
arête vecteur directeur $= \overrightarrow{A_2}$	$\emptyset$	$\{\overrightarrow{U}\}$ $\overrightarrow{U}$ a pour vecteur directeur $\overrightarrow{A_1} \wedge \overrightarrow{A_2}$ et son sens est tel que $\overrightarrow{U}$ se dirige vers l'extérieur de l'objet support de e_1.	{Normales extérieures à e_1 en tout point de $e_1 \cap e_2$.}
face	{Normales extérieures à e_2 au point de contact.}	{Normales extérieures à e_2 en tout point de $e_1 \cap e_2$.}	{Normales extérieures à e_1 en tout point de $e_1 \cap e_2$.}

5.3 Équations induites par les mouvements compliants

5.3.1 Mouvement d'un point sur une surface

Pour présenter l'idée directrice de cette méthode, nous allons déterminer les équations du mouvement d'un point glissant sur une surface (mouvement ponctuel à compliance). Soit P un point en contact avec une surface $S(x, y, z) = 0$ et devant se déplacer en conservant le contact.

Le mouvement d'un point dans l'espace est complètement déterminé par la donnée du vecteur vitesse en tout point et à tout instant. Comme le point P est contraint de rester sur la surface S, le vecteur vitesse du mouvement est orthogonal en tout point

de la surface à la normale extérieure de S. La mise en équation de cette contrainte puis la résolution des équations obtenues déterminera le mouvement. L'équation de la normale extérieure $\overrightarrow{N}$ en tout point de S est :

$$\overrightarrow{N}\ (x,y,z) = \left| \begin{array}{l} \frac{\partial S}{\partial x}(x,y,z) \\ \frac{\partial S}{\partial y}(x,y,z) \\ \frac{\partial S}{\partial z}(x,y,z) \end{array} \right.$$

Les mouvements élémentaires qui maintiennent localement le contact ont pour vecteur vitesse : $\overrightarrow{V}\ (x,y,z)$ avec $\overrightarrow{V}\ (x,y,z) \cdot \overrightarrow{N}\ (x,y,z) = 0$. D'un point de vue théorique, le mouvement du point peut s'obtenir en intégrant par rapport à t le système : $M(t) = \overrightarrow{V}\ (x,y,z)\ \cdot dt$. Ceci se traduit par un système d'équations différentielles difficiles à résoudre de manière formelle. Une résolution numérique est cependant possible (Cf §5.4).

5.3.2 Cas mettant en jeux plusieurs contacts

Au cours de la planification, chaque configuration est caractérisée par un ensemble C de couples d'entités en contact.

$$C = \{(e_1, e_2), (e_3, e_4), \cdots, (e_{2i+1}, e_{2i+2})\} 0 \leq i < n.$$

La phase de démontage consiste à réduire cet ensemble jusqu'à l'ensemble vide. Pour chaque couple d'entités en contact, on établit les équations de telle manière que le mouvement soit tangent aux entités (mouvement compliant). Chaque couple définit localement un plan tangent suivant lequel peuvent s'effectuer des mouvements or il n'est possible de glisser que sur un ou deux plans non-coplanaires. Le système à n équations est donc très vite surcontraint, ce qui peut conduire à l'absence de solutions. L'heuristique de base que nous appliquons (relaxation progressive des contraintes issues des contacts) conduit à considérer deux par deux les couples d'entités en contact. Il y a donc $\frac{n \cdot (n-1)}{2}$ systèmes d'équations traduisant des mouvements possibles.

L'étape suivante consiste à utiliser un filtre géométrique éliminant certaines solutions des systèmes précédents. Le but de ce filtre est de supprimer les mouvements qui font entrer en collision des entités géométriques. Soit $C_1 = (e_{11}, e_{12})$ et $C_2 = (e_{21}, e_{22})$ deux couples de C pour lesquels on veut déterminer les mouvements

possibles. La mise en équation des mouvements possibles nous fournit un ensemble de vecteur vitesse $V = \{\overrightarrow{v_1}, \overrightarrow{v_2}, \cdots, \overrightarrow{v_i}, \cdots, \overrightarrow{v_n}\}$ dont l'intégration définira un mouvement. Un vecteur vitesse $\overrightarrow{v_i}$ est compatible si :

$$\forall C_j = (e_{j1}, e_{j2}) \in C, \qquad \overrightarrow{v_i} \diamond C_j = Vrai.$$

La relation $\diamond$ est définie en fonction de la nature de e_{j1} et de e_{j2} comme suit :

$$\overrightarrow{v_i} \diamond C_j = vrai \iff \forall\, \overrightarrow{n} \in EN(C_j), \quad \overrightarrow{v_i} \cdot \overrightarrow{n} \geq 0.$$

Dans le cas d'un contact où une des entités n'est pas un point (arête, face), les spécifications du maintien du contact deviennent plus compliquées. Il existe deux manières pour maintenir la liaison :

1. En maintenant $(e_{2i} \cap e_{2i+1})$ comme zone de contact.
2. En maintenant (e_{2i}, e_{2i+1}) en contact mais en faisant varier $(e_{2i} \cap e_{2i+1})$.

Si on considère le mouvement d'un cylindre en contact avec un plan, on peut soit faire glisser l'arête du cylindre sans le faire rouler (cas 1), soit faire une rotation sans glissement autour de l'axe du cylindre (cas 2). Le choix d'un type de mouvement[1] est délicat de même que la mise en équation des contraintes. La prise en compte des mouvements du type (1), nécessite de considérer en plus les cœfficients de frottement.

5.4 Principe de résolution

La résolution des systèmes d'équations différentielles établis en considérant des couples d'entités en contact, est complexe dans le cas général. Pour un planificateur, l'important n'est pas les équations formelles du mouvement mais uniquement la position finale de l'objet après avoir glissé sur une surface donnée. Le mouvement exact pourra être restitué par la commande grâce aux mouvements compliants. Pour ces raisons, on va utiliser les techniques de l'analyse numérique pour trouver cette position finale.
Les différentes méthodes existantes permettent de trouver une solution à l'équation suivante :

[1] On fait ici l'hypothèse que l'on effectue un seul des deux types de mouvement, dans un mouvement véritable il se peut fort bien que l'on combine les deux actions de manière continue.

$$y' = f(x, y)$$

sur l'intervalle $[x_0, x_0 + X]$ avec comme condition initiale $y(x_0) = y_0$. Ces méthodes peuvent être appliquées pour $x \in \mathcal{R}^n$ et $y : \mathcal{R}^n \rightarrow \mathcal{R}^n$.

Le développement de Taylor de $y(x)$ au voisinage de x_0 nous donne :

$$y(x) \approx \sum_{i=0}^{n} \frac{y^i(x_0)}{i!}(x - x_0)^i$$

La convergence de cette série n'est assurée que pour un certain ouvert centré en x_0. Si $|x - x_0|$ est supérieur au rayon de convergence, l'intervalle $[x_0, x_0 + X]$ est partagé en segments plus petits $[x_{j-1}, x_j], j = 1, \ldots, \mathrm{N}$. La technique consiste alors à chercher de proche en proche les approximations y_j de la solution $y(x_j)$.
En effectuant un développement à l'ordre un et en posant $h = x_j - x_{j-1}$, on établit la formule d'Euler :

$$y(x + h) = y(x_0) + h \cdot f(x_0, y_0) \tag{5.1}$$

Les méthodes de résolution basées uniquement sur le développement de Taylor sont parmi les plus anciennes. Elles sont cependant souvent délaissées au profit de méthodes nécessitant moins de calculs. Considérons l'égalité suivante :

$$y(x + h) = y(x) + \int_0^h y'(x + t)dt$$

En utilisant la formule des rectangles pour résoudre numériquement l'intégrale du membre de droite, on obtient :

$$y(x + h) = y(x) + h \cdot f(x + \frac{h}{2}, y(x + \frac{h}{2})) + O(h^3) \tag{5.2}$$

Grâce à la formule d'Euler, il est possible de calculer $y(x + \frac{h}{2})$. La combinaison des équations 5.1 et 5.2 nous fournit un algorithme permettant de calculer $y(x_{j+1})$ en fonction de $y(x_j)$:

$$\begin{aligned} y_{j+1/2} &= y_j + \frac{h}{2} \cdot f(x_j, y_j) \\ y_{j+1} &= y_j + h \cdot f(x_j + \frac{h}{2}, y_{j+1/2}) \end{aligned}$$

Ce type de méthode est désigné sous le nom de méthode de Runge-Kutta. De manière plus générale, ces méthodes consistent à fixer certains nombres

$$\alpha_2, \ldots, \alpha_q, p_1, \ldots, p_q, \beta_{ij} 0 < j < i \leq q;$$

A trouver successivement :

$$\begin{aligned} k_1(h) &= h \cdot f(x,y) \\ k_2(h) &= h \cdot f(x+\alpha_2 h, y+\beta_{21} k_1(h)) \\ \vdots &= \vdots \\ k_q(h) &= h \cdot f(x+\alpha_q h, y+\beta_{q,1} k_1(h) + \cdots + \beta_{q,q-1} k_{q-1}(h)) \end{aligned}$$

Et écrire ensuite :

$$y(x+h) \approx y(x) + \sum_{i=1}^{q} p_i k_i(h)$$

.

Le choix des paramètres $\alpha_i, p_i, \beta ij$ revient à établir une formule d'intégration optimale vis-à-vis de la classe de fonctions susceptibles d'être traitées. Nous ne détaillerons pas ici le calcul de ces cœfficients mais on pourra se reporter à [Bak73]. Nous allons utiliser ces formules d'intégrations pour traiter un exemple. Les formules retenues sont les suivantes :

$$\begin{aligned} k_1 &= h \cdot f(x_n, y_n) \\ k_2 &= h \cdot f(x_n + \frac{h}{3}, y_n + \frac{k_1}{3}) \\ k_3 &= h \cdot f(x_n + \frac{2h}{3}, y_n - \frac{k_1}{3} + k_2) \\ k_4 &= h \cdot f(x_n + h, y_n + k_1 - k_2 + k_3) \end{aligned}$$

$$y_{n+1} = y_n + \frac{1}{8}(k_1 + 3k_2 + 3k_3 + k_4)$$

5.4.1 Un exemple en 3D : Le vissage

Le problème à résoudre consiste à trouver numériquement la position d'un point contraint à rester au contact de deux surfaces. Soit S_h l'équation de l'hélice et S_c l'équation du cylindre, toutes deux représentant les surfaces de contact.

$$\begin{aligned} S_h: &\quad x \cdot \sin(\tfrac{z}{h}) - y \cdot \cos(\tfrac{z}{h}) = 0 \\ S_c: &\quad x^2 + y^2 - R^2 = 0 \end{aligned}$$

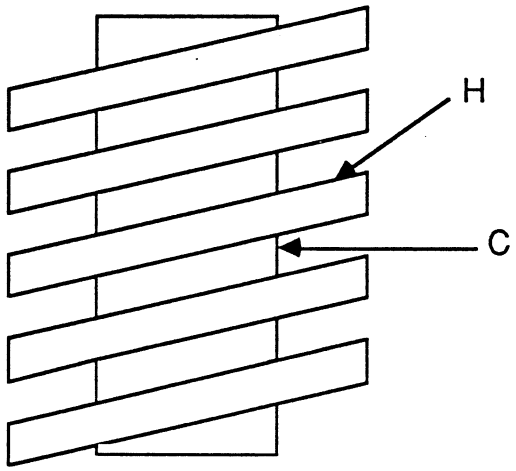


Figure 5.2: Vis

Pour notre exemple, nous choisissons $h = 1$ et $R = 1$. Les vecteurs perpendiculaires aux surfaces S_h et S_c sont les suivants :

$$\overrightarrow{T}_h : \left| \begin{array}{lcl} \frac{\partial S_h}{\partial x} & = & \sin(z) \\ \frac{\partial S_h}{\partial y} & = & -\cos(z) \\ \frac{\partial S_h}{\partial z} & = & y \cdot \sin(z) + x \cdot \cos(z) \end{array} \right. \qquad \overrightarrow{T}_c : \left| \begin{array}{lcl} \frac{\partial S_c}{\partial x} & = & 2x \\ \frac{\partial S_c}{\partial y} & = & 2y \\ \frac{\partial S_c}{\partial z} & = & 0 \end{array} \right.$$

La contrainte due à la nature tangentielle du mouvement conduit à ce que la tangente au mouvement soit perpendiculaire aux vecteurs $\overrightarrow{T}_h$ et $\overrightarrow{T}_c$. Le vecteur tangent indiquant localement la direction du mouvement peut être aisément obtenu par le produit vectoriel suivant :

$$\overrightarrow{T}_h \wedge \overrightarrow{T}_c : \left| \begin{array}{l} -2y^2 \cdot \sin(z) - 2xy \cdot \cos(z) \\ 2xy \cdot \sin(z) + 2x^2 \cdot \cos(z) \\ 2y \cdot \sin(z) + 2x \cdot \cos(z) \end{array} \right.$$

Ces équations nous fournissent immédiatement le système qu'il faut résoudre :

$$\begin{array}{lcl} x' & = & -2y^2 \sin(z) - 2xy \cos(z) \\ y' & = & 2xy \sin(z) + 2x^2 \cos(z) \\ z' & = & 2y \sin(z) + 2x \cos(z) \end{array}$$

Résolvons ce système grâce à la méthode présentée précédemment. Pour juger la qualité des résultats obtenue numériquement, résolvons de manière formelle ce système. On choisit comme condition initiale : $(x_0 = 1, y_0 = 0, z_0 = 0)$. C'est un point qui appartient à l'intersection du cylindre et de l'hélice. La solution exacte est facile à imaginer : c'est l'équation d'une hélice de rayon 1 et de pas π. Son équation est :

$$\begin{cases} x &= \cos(t) \\ y &= \sin(t) \\ z &= t \end{cases} \qquad t \in \mathcal{R}$$

n	x_n	y_n	z_n	erreur
0	1.000	0.000	0.000	—
1	0.980	0.199	0.200	$1.07 \cdot 10^{-6}$
2	0.921	0.389	0.400	$2.14 \cdot 10^{-6}$
3	0.825	0.565	0.600	$3.24 \cdot 10^{-6}$
4	0.697	0.717	0.800	$4.33 \cdot 10^{-6}$
5	0.540	0.841	1.000	$5.41 \cdot 10^{-6}$
10	-0.416	0.909	2.000	$1.08 \cdot 10^{-5}$
20	-0.654	-0.757	4.000	$2.16 \cdot 10^{-5}$
30	0.960	-0.280	6.000	$3.23 \cdot 10^{-5}$
40	-0.145	0.989	8.000	$4.31 \cdot 10^{-5}$
50	-0.839	-0.544	10.000	$5.39 \cdot 10^{-5}$
60	0.844	-0.537	12.000	$6.46 \cdot 10^{-5}$
70	0.137	0.991	14.000	$7.52 \cdot 10^{-5}$
80	-0.958	-0.287	16.000	$8.65 \cdot 10^{-5}$
90	0.660	-0.751	18.000	$9.77 \cdot 10^{-5}$
100	0.409	0.913	20.000	$1.06 \cdot 10^{-4}$

n	x_n	y_n	z_n	erreur
200	-0.667	0.745	40.000	$2.18 \cdot 10^{-4}$
300	-0.952	-0.306	60.001	$3.21 \cdot 10^{-4}$
400	-0.108	-0.994	80.003	$4.25 \cdot 10^{-4}$
500	0.865	-0.502	100.006	$5.43 \cdot 10^{-4}$
600	0.809	0.588	120.010	$6.52 \cdot 10^{-4}$
700	-0.212	0.978	140.015	$6.06 \cdot 10^{-4}$
800	-0.980	0.200	160.021	$8.57 \cdot 10^{-4}$
900	-0.578	-0.817	180.027	$9.02 \cdot 10^{-4}$
1000	0.516	-0.857	200.034	$1.03 \cdot 10^{-3}$
1500	0.059	-0.999	300.083	$1.01 \cdot 10^{-3}$
2000	-0.391	-0.921	400.155	$2.33 \cdot 10^{-3}$
2500	-0.746	-0.667	500.246	$2.61 \cdot 10^{-3}$
3000	-0.953	-0.307	600.361	$3.83 \cdot 10^{-3}$
3500	-0.998	0.085	700.494	$2.96 \cdot 10^{-3}$
4000	-0.897	0.447	800.652	$6.22 \cdot 10^{-3}$
4500	-0.684	0.732	900.832	$9.75 \cdot 10^{-3}$

Figure 5.3: Résolution numérique d'un mouvement hélicoïdal

Le tableau de la figure 5.3 regroupe les résultats obtenus lors de la résolution. La colonne *erreur* indique la distance minimale entre le point trouvé et la courbe théorique correspondant au mouvement hélicoïdal. La résolution s'est effectuée avec un pas h=0.1. Comme on le constate sur le tableau les erreurs sont très faibles. Les images de l'annexe A.2 présentent une copie d'écran d'une représentation graphique de la solution obtenue avec plusieurs méthodes.

Remarque : La méthode retenue pour la résolution est une bonne méthode mais elle n'a pas été optimisée. Il est sans doute possible d'améliorer encore les performances en faisant une analyse sur la nature des équations à résoudre et en choisissant des cœfficients d'intégration plus adaptés.

5.5 Calcul des amplitudes de mouvement possible

Une fois la trajectoire du mouvement connue, il faut déterminer l'amplitude du mouvement réellement exécutable. Cette notion de longueur de chemin correspond à la définition mathématique d'intégrale curviligne le long de la trajectoire. Nous voulons répondre aux deux questions suivantes :

- Est-ce que le mouvement d'amplitude ε (mouvement de correction) est applicable?
- Quel est le débattement maximum d'un mouvement donné?

5.5.1 Principe du calcul

Le calcul du débattement possible d'un objet animé d'un mouvement $\mathcal{M}$ dans un univers encombré est un problème complexe. Dans le cadre de la planification des mouvements fins d'assemblage, une solution approchée ne peut nous suffire et il faut donc établir une méthode permettant de calculer l'amplitude exacte possible pour un mouvement.
Dans une prémière approche, un mouvement est limité par la réalisation d'un ou plusieurs contacts créé par :

- un point sur une face
- une arête sur une arête
- une arête sur une face
- une face sur une face

De manière plus générale, un mouvement est limité par la rupture de la situation géométrique que l'on voulait conserver. Si

$$C_p = \{(e_1, e_2), (e_3, e_4), \ldots, (e_{2i+1}, e_{2i+2})\}\ 0 \leq i < n$$

désigne l'ensemble des couples d'entités pour lesquelles il existe une propriété géométrique intéressante (contact mais aussi adjacence par exemple) pour une position $\mathbf{p}$ de l'objet mobile, le problème consiste à rechercher l'amplitude d'un mouvement amenant $\mathbf{p}$ en $\mathbf{p}'$ tel que :

- $(C_{p'} = C_p + c) \vee (C_{p'} = C_p - c)$
  où c=$\{(e_{2j+1}, e_{2j+2}),\ 0 \leq j < p\}$
- **et** c est le plus petit possible

Ce dernier point traduit le fait que l'on s'intéresse à la première modification de situation géométrique rencontrée. Nous n'allons considérer dans la suite que les propriétés de contact et d'adjacence qui se traduisent par l'intersection des supports mathématiques des entités considérées.

Les équations mathématiques des supports d'entités qu'il est possible de rencontrer sont les suivants :

point : (x_0, y_0, z_0)

arête : $\begin{cases} S_1(x,y,z) = 0 \\ S_2(x,y,z) = 0 \end{cases}$

face : $S(x,y,z) = 0$

Les équations de ces mêmes entités auxquelles a été appliqué un mouvement $\mathcal{M}$: $\varphi_x(x,y,z,t)$, $\varphi_y(x,y,z,t)$, $\varphi_z(x,y,z,t)$, prennent alors les allures suivantes :

point : $(\varphi_x(x_0,y_0,z_0,t), \varphi_y(x_0,y_0,z_0,t), \varphi_z(x_0,y_0,z_0,t))$

arête : $\begin{cases} S_1(\varphi_x(x,y,z,t), \varphi_y(x,y,z,t), \varphi_z(x,y,z,t)) = 0 \\ S_2(\varphi_x(x,y,z,t), \varphi_y(x,y,z,t), \varphi_z(x,y,z,t)) = 0 \end{cases}$

face : $S(\varphi_x(x,y,z,t), \varphi_y(x,y,z,t), \varphi_z(x,y,z,t))$

L'amplitude de mouvement possible pour une entité E_1 soumise à un mouvement $\mathcal{M} : (\varphi_x, \varphi_y, \varphi_z)$ et entrant un intersection avec une entité fixe E_2, est solution d'un système d'équations. Les différents cas sont regroupés dans les tableaux suivants :

$E_1 \longrightarrow$ E_2 $\downarrow$	*point* (x_1, y_1, z_1)
point (x_2, y_2, z_2)	—
arête $\begin{cases} S_1^2(x,y,z) = 0 \\ S_2^2(x,y,z) = 0 \end{cases}$	—
face $S^2(x,y,z) = 0$	$S^2(\varphi_x(x,y,z,t), \varphi_y(x,y,z,t), \varphi_z(x,y,z,t)) = 0$

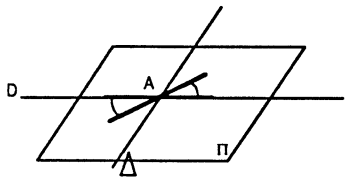
$E_1 \longrightarrow$ E_2 $\downarrow$	*arête* $\begin{cases} S_1^1(x,y,z) = 0 \\ S_2^1(x,y,z) = 0 \end{cases}$
point (x_2, y_2, z_2)	—
arête $\begin{cases} S_1^2(x,y,z) = 0 \\ S_2^2(x,y,z) = 0 \end{cases}$	$\begin{cases} S_1^1(\varphi_x(x,y,z,t), \varphi_y(x,y,z,t), \varphi_z(x,y,z,t)) = 0 \\ S_2^1(\varphi_x(x,y,z,t), \varphi_y(x,y,z,t), \varphi_z(x,y,z,t)) = 0 \\ S_1^2(x,y,z,t) = 0 \\ S_2^2(x,y,z,t) = 0 \end{cases}$
face $S^2(x,y,z) = 0$	$\begin{cases} S_1^1(\varphi_x(x,y,z,t), \varphi_y(x,y,z,t), \varphi_z(x,y,z,t)) = 0 \\ S_2^1(\varphi_x(x,y,z,t), \varphi_y(x,y,z,t), \varphi_z(x,y,z,t)) = 0 \\ S^2(x,y,z) = 0 \end{cases}$

$E_1 \longrightarrow$ E_2 $\downarrow$	*face* $S^1(x,y,z) = 0$
point (x_2, y_2, z_2)	$S^1(\varphi_x^{-1}(x,y,z,t), \varphi_y^{-1}(x,y,z,t), \varphi_z^{-1}(x,y,z,t))$
arête $\begin{cases} S_1^2(x,y,z) = 0 \\ S_2^2(x,y,z) = 0 \end{cases}$	$\begin{cases} S^1(\varphi_x(x,y,z,t), \varphi_y(x,y,z,t), \varphi_z(x,y,z,t)) = 0 \\ S_1^2(x,y,z,t) = 0 \\ S_2^2(x,y,z,t) = 0 \end{cases}$
face $S^2(x,y,z) = 0$	$\begin{cases} S^1(\varphi_x(x,y,z,t), \varphi_y(x,y,z,t), \varphi_z(x,y,z,t)) = 0 \\ S^2(\varphi_x(x,y,z,t), \varphi_y(x,y,z,t), \varphi_z(x,y,z,t)) = 0 \end{cases}$

Tous les systèmes d'équations précédents sont insuffisants pour trouver une solution en (x, y, z, t) car ils sont sous-dimensionnés. Le problème consiste en réalité à

rechercher la plus petite valeur positive de t tel que le système considéré admette une solution.

Lorsque l'un des supports mathématiques du couple d'entités considérées est infini (équation d'une droite, d'un plan), l'amplitude du mouvement peut être réduite par une intersection que nous ne devrions pas prendre en compte.



Dans l'exemple précédent, le problème consiste à rechercher l'angle de rotation possible autour de Δ avant que l'arête A de support D n'entre en contact avec la face F. Le moindre mouvement autour de Δ va rompre le parallélisme de Δ et de F et créer une intersection de leur support mathématique.
La solution à ce problème consiste à remarquer qu'il suffit de s'intéresser aux contacts éventuels entre les points extrémité de A et les supports des arêtes bordant F.

Le cas géneral résiste à une approche uniquement mathématique et il faut faire appel à un raisonnement géométrique pour transformer le problème initial en sous-problème traduisant mieux l'aspect physique des contacts établis.

Quel que soit le raisonnement mis en œuvre, il reste cependant quelques systèmes à résoudre. La résolution formelle est difficile et il est préférable de faire appel à des méthodes numériques. Les systèmes d'équations n'étant pas linéaires, il n'est pas possible de faire appel à des méthodes fournissant des solutions en un nombre fini de pas de calcul. Il faut faire appel à des méthodes itératives. L'idée directrice de ces méthodes consiste à se "rapprocher" progressivement de la solution. Le même processus est répété jusqu'à ce que la distance entre deux solutions approchées consécutives soit inférieure à une valeur fixée au préalable. Parmi les nombreuses techniques de résolution existantes, deux se prètent bien aux calculs que l'on veut faire :

- Les méthodes de Newton.
- Les méthodes dichotomiques.

Avec les méthodes de Newton, pour résoudre l'équation F(x)=0, on se donne un point X_0 départ de l'itération, puis on calcule les points $X_1, X_2, \cdots X_n$ grâce à une formule de la forme $X_{n+1} = \Phi(X_n, F(X_n), \frac{dF(X_n)}{dX})$. L'idée de la méthode consiste à obtenir le point suivant en se déplaçant suivant la tangente au point considéré.

Dans la méthode dichotomique, on oscille autour d'une solution par valeur $\frac{h}{2^n}$ ou h est le pas initial de la dichotomie. Le choix d'une méthode ne peut se faire simplement car il faut prendre en considération la nature de la fonction à résoudre. Un choix optimum ne peut être fait mais il est cependant possible de dégager certains critères pour diriger ce choix. Constatation importante : La méthode de Newton ne fournit qu'une solution et est sensible au phénomène de minima locaux alors que la méthode dichotomique peut fournir toutes les solutions mais au prix de plus de calculs car la convergence est moins bonne. Dans notre cas, nous avons besoin de plusieurs solutions dans un intervalle donné et nous aimerions que le calcul se fasse le plus vite possible. Nous choisirons donc une méthode mixte : une dichotomie pour trouver les points initiaux proches d'un zéro puis une méthode type Newton pour affiner la solution.

5.5.2 Résolution d'un exemple

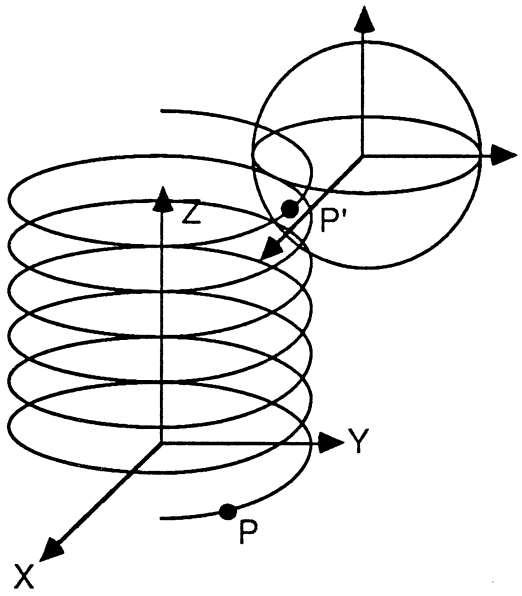


Figure 5.4: Mouvement hélicoïdal d'un point

Considérons l'exemple de la figure 5.4. Le point P(x,y,z) subit un mouvement hélicoïdal d'axe Oz et de pas h. Il risque de rencontrer la sphère S et l'on veut calculer l'amplitude du mouvement possible (éventuellement infini). Le mouvement est défini par la donnée de :

$$M: \begin{cases} \varphi_x(x,y,z,t) &= x \times \cos t - y \times \sin t \\ \varphi_y(x,y,z,t) &= x \times \sin t + y \times \cos t \\ \varphi_z(x,y,z,t) &= z + \frac{h}{2\pi} \times t \end{cases}$$

L'équation de la sphère de centre $C(C_x, C_y, C_z)$ et de rayon R est :

$$\text{S: } (X - C_x)^2 + (Y - C_y)^2 + (Y - C_y)^2 - R^2 = 0$$

Il faut remplacer X,Y,Z respectivement par $\varphi_x, \varphi_y, \varphi_z$ et résoudre ensuite. AprèS simplification, l'équation à résoudre est du type :

$$A \times \cos t^2 + B \times t^2 + C \times t + D = 0$$

La résolution explicite d'une telle équation est trop complexe pour se faire de manière formelle. Nous allons considérer ici un cas particulier pour montrer la méthode de résolution numérique. Nous voulons résoudre

$$F(t) = S(\varphi_x(x,y,z,t), \varphi_y(x,y,z,t), \varphi_z(x,y,z,t))$$

dans un cas particulier. Considérons le point $P = (\sqrt{3}, 1, \frac{1}{12})$ et la sphère de centre $C = (2,2,3)$ et de rayon R=1. Le mouvement est un vissage autour de Oz de pas h=1. L'amplitude est un zéro de la fonction :

$$F(t) = 4 \times \cos t^2 - 8\sqrt{3} \times \cos t + \frac{1}{4\pi^2} \times t^2 + \frac{35}{6} \times t - \frac{71}{144}$$

On trouvera ci-dessous les valeurs de la fonction tabulées pour permettre de se rendre compte de la manière de recherche de la solution à F(t)=0 par encadrements successifs.

t	0.1	0.2	0.3	0.4	0.5	**0.6**	**0.7**	0.8	0.9	1.0
F(t)	3.9	4.0	3.7	3.0	2.0	**0.8**	**-0.8**	-2.7	-4.9	-7.3
t	0.61	0.62	0.63	**0.64**	**0.65**	0.66	0.67	0.68	0.69	0.70
F(t)	0.61	0.46	0.30	**0.15**	**-0.01**	-0.17	-0.34	-0.50	-0.67	-0.85
t	0.641	0.642	0.643	0.644	0.645	0.646	0.647	0.648	**0.649**	**0.650**
F(t)	0.133	0.117	0.101	0.085	0.070	0.054	0.038	0.022	**0.006**	**-0.010**

En trois encadrements, nous approchons déjà la solution à 10^{-3} près. On dispose ainsi d'une bonne valeur initiale pour appliquer la méthode de Newton. Une itération plus précise donne $t = 0.649373$.

5.5.3 Conclusion

L'emploi exclusif de méthodes mathématiques ne permet pas de résoudre dans tous les cas le problème de l'amplitude possible d'un mouvement. Si les méthodes numériques permettent de trouver une solution à un système complexe d'équations, il faut souvent mettre en œuvre un raisonnement géométrique pour décomposer le problème initial en sous-problème de complexité moindre. ce même raisonnement doit aussi être utilisé pour établir les systèmes d'équations représentant le mieux les limitations physiques du mouvement.

Une solution partielle est obtenue par une étude de cas permettant d'avoir le meilleur système à résoudre pour un mouvement et un couple d'entités particuliers. Le calcul de l'amplitude du mouvement pour un solide complet reste cependant très coûteux. Dans l'implantation réalisée pour le système SHARP, les différents cas envisagés sont résolus de manière formelle pour les polyédres. Le système dispose alors pour chaque cas d'une formule donnant l'amplitude d'un mouvement donné (translation ou rotation) en un nombre fini de pas de calcul.

Chapitre 6

Le système PAMELA

Programmation
Automatique de
Mouvements
ELémentaires d'
Assemblage.

6.1 Principes généraux

La planification des trajectoires de montage est un problème très large pour lequel il n'existe pas de solution générale. L'approche que nous avons développé repose sur la réversibilité du démontage et se restreint aux mouvements de translation et de rotation.

6.1.1 Réversibilité du démontage

L'hypothèse de réversibilité consiste à supposer qu'une trajectoire de montage peut être obtenue en inversant les mouvements effectués pour le démontage. Notre système de planification prendra donc en entrée un modèle des pièces assemblées. Il tentera alors d'inférer un plan de montage à partir de l'inverse des opérations de démontage. La classe de problèmes que l'on peut résoudre avec cette hypothèse est suffisamment importante pour qu'elle ne soit pas une contrainte trop grande.

Cette hypothèse conduit cependant à éliminer tous les objets déformables (ressorts en particulier). La manipulation de ce type de pièces ne peut se planifier car nous ne disposons pas d'un modèle de leur déformation pendant la manipulation.

La classe des problèmes d'assemblage que l'on peut traiter avec cette hypothèse est légèrement plus grande que les assemblages strictement réversibles car le démontage théorique[1] d'un objet peut guider son remontage.

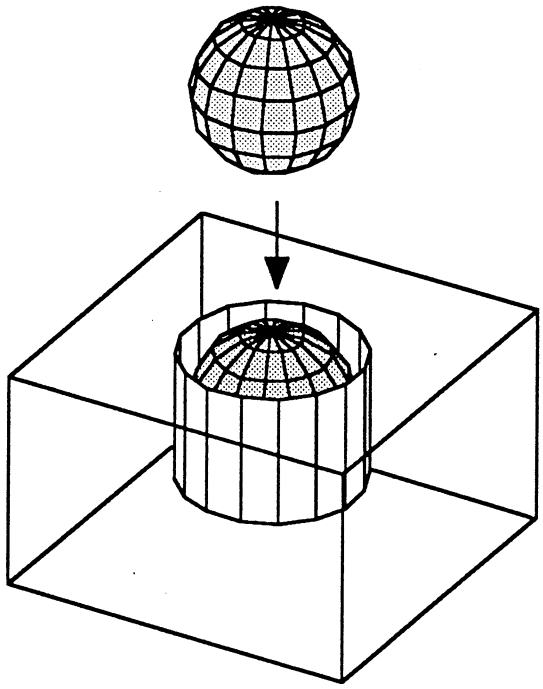


Figure 6.1: Insertion d'une bille dans un trou

Si l'on considère la bille dans le fond de son trou, on ne peut pas la sortir (la démonter) de son trou à l'aide d'une pince à deux mors parallèles. Imaginons cependant que l'on puisse. La trajectoire de sortie donne une information sur la manière d'insérer la bille dans son trou en utilisant un outil supplémentaire : la gravité. La stratégie de montage consiste alors à se placer au dessus du trou, puis à lâcher la bille.

Ce simple exemple montre que le démontage, même en faisant abstraction des

[1] Un démontage théorique consiste à ne pas prendre en compte les problèmes liés à la manipulation. On démonte par exemple en ignorant les contraintes imposées par l'outil de saisie

problèmes de la manipulation réelle, fournit des informations pertinentes pour le montage. Dans notre approche, nous ne prendrons en compte comme phénomène physique que la gravité car elle est plus simple à maîtriser.

La trajectoire de démontage, même si elle n'est pas purement réversible sert donc à **guider** la recherche d'une trajectoire potentielle de montage; La détermination d'une véritable trajectoire de montage est ensuite réalisée en introduisant explicitement les contraintes physiques de la manipulation.

6.1.2 Restriction aux translations et rotations pures

Un programme de montage est constitué d'une suite de déplacements élémentaires conduisant un objet d'une position où il est libre jusqu'à sa position finale d'assemblage. Le mode de raisonnement que nous appliquons conduit à ne considérer que des translations et des rotations pures. Ce choix à été réalisé dans le but de réduire la complexité du problème (par exemple, les stratégies de montage incluant des mouvement de type vissage sont éliminées). Cette restriction est intuitivement la plus importante car elle limite de manière significative la classe de problèmes pouvant être résolus.

La prise en compte d'un autre type de mouvement ne remet pas en cause les méthodes de raisonnement mais augmente considérablement le coût de la planification. Pour cela, il suffit d'ajouter des connaissances supplémentaires indiquant quand et comment mettre en œuvre ce nouveau type de mouvement. Les connaissances nécessaires à la planification d'autres mouvements ne seront ajoutées qu'en fonction des besoins technologiques : on n'ajoutera les vissages que si l'on doit prendre en compte des pièces susceptibles d'être vissées.

6.1.3 prise en compte de l'incertitude

La prise en compte des incertitudes est l'objet même de la planification des mouvements fins d'assemblage. Nous voulons tenir compte de l'incertain uniquement en choisissant des trajectoires de montage qui vont minimiser l'incertitude de positionnement : l'incertitude est traitée de manière implicite. Considérons le montage de la figure 6.2 qui consiste à aller placer un cube à une distance de 1cm de chacune des arêtes.

Parmi les stratégies potentiellement utilisables, considérons les deux suivantes : la première consiste à aller se placer à la verticale de la position finale puis à descendre

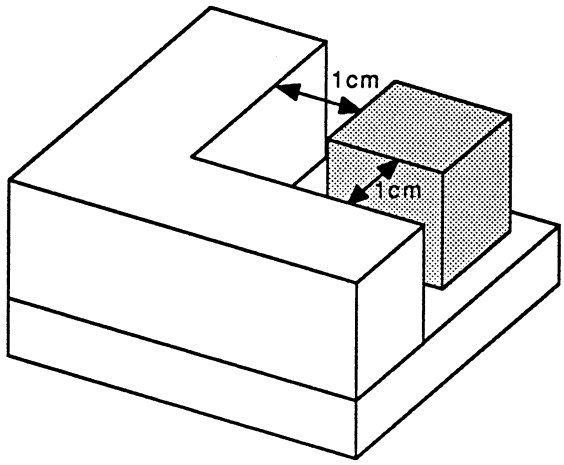


Figure 6.2: Heuristique de réduction de l'incertitude

jusqu'à réaliser un contact plan. Avec une telle méthode, l'incertitude de position relative est maximale selon les axes O_x, O_y et nulle selon l'axe O_z.

Une autre approche consiste à se placer loin du coin[2] puis à descendre jusqu'au contact. Cela réalisé, il faut rechercher un contact dans l'angle grâce à des mouvements à compliance. Ensuite, l'incertitude relative devient nulle (ou du moins de l'ordre de précision de la commande du robot) et il est possible d'effectuer le déplacement permettant atteindre le but de la manipulation. Après application de cette stratégie, l'incertitude de position est réduite selon les axes O_x, O_y à la précision du robot. C'est cette dernière approche que nous avons choisi d'appliquer dans notre système. Celui-ci privilégie à chaque fois que c'est possible une trajectoire capable de réduire les incertitudes relatives sans tenir compte des manipulations antérieures.

Deux types d'incertitudes interviennent au niveau des mouvements fins de montage : les incertitudes liées à la forme et aux dimensions des objets (qui proviennent des tolérances de fabrication), et celles liées à la position des objets (qui proviennent de la manipulation, des mécanismes de positionnement et des capteurs).

Le premier type d'incertitude conduit à imposer des jeux de montage qu'il faut impérativement respecter. Le but des mouvements fins est alors de ramener l'incertitude de positionnement à des valeurs compatibles avec les contraintes précédentes.

[2]C'est à dire à une position décalée selon chacun des axes O_x, O_y du maximum de l'erreur locale

6.2 Fonctionnement général du système

6.2.1 Principes de fonctionnement

Le système PAMELA planifie les mouvements fins liés aux tâches d'assemblage à partir des traces d'opérations fictives de démontage. Pour cela il effectue un raisonnement géométrique en utilisant les données numériques et symboliques caractérisant le montage. Les informations de nature symbolique sont inférées automatiquement par le mécanisme des démons en faisant appel à un ensemble de fonctions géométriques spécialisées.

La description de la tâche de montage s'effectue de manière implicite par l'intermédiaire du modèle qui doit représenter les différents objets de l'univers en position finale d'assemblage. La position initiale de l'objet à assembler est choisie par le système et est telle qu'elle puisse être atteinte indépendamment des effets de l'incertitude.

Le démontage théorique est réalisé en effectuant un raisonnement qui consiste à "décontraindre" progressivement l'objet assemblé. Ce raisonnement est effectué à l'aide de "règles" qui permettent de passer d'une situation géométrique à une autre. Chaque règle traduit une information dont nous disposons au sujet de l'assemblage. Ces connaissances géométriques portent essentiellement sur la manière de faire évoluer un ensemble de contacts restreignant les degrés de liberté d'un objet.

La trace de l'activation des différentes règles est conservée explicitement dans un graphe appelé: "graphe de démontage". Il représente les différentes séquences de mouvements conduisant l'objet d'une position où il est libre à sa position finale d'assemblage.

Le graphe de démontage est ensuite exploité pour produire un plan de montage optimal. Pour cela, il est parcouru systématiquement et chaque chemin, conduisant d'une position libre à la position finale d'assemblage, est noté. Ces notes sont fournies par un ensemble de conseils qui prennent en compte, pour chaque mouvement élémentaire, la stabilité des positions initiale et finale ainsi que la nature du mouvement à effectuer. Ces conseils sont aussi capables de détecter une situation singulière réclamant une étude plus fine. Dans ce cas, ils relancent le développement du graphe à partir de cette position avec un ensemble de règles plus spécialisées.

Une fois le chemin optimal trouvé, le plan de montage est produit. Il comporte autant d'instructions qu'il y a de situations intermédiaires le long du chemin retenu. Chaque instruction traduit de manière géométrique la transition à réaliser.

6.2.2 Le raisonnement par "règles"

A partir d'une situation géométrique particulière, le système effectue un raisonnement géométrique pour choisir un ou plusieurs mouvements susceptibles de relaxer les contraintes (augmenter le nombre de degrés de liberté) et ainsi démonter l'ensemble. Les différentes connaissances mises en œuvre lors de cette phase sont représentées sous forme de règles. Elles sont de deux types :

1. Des règles permettant de décrire les mouvements à effectuer pour désassembler. Elles font appel au modèle décrit précédemment.

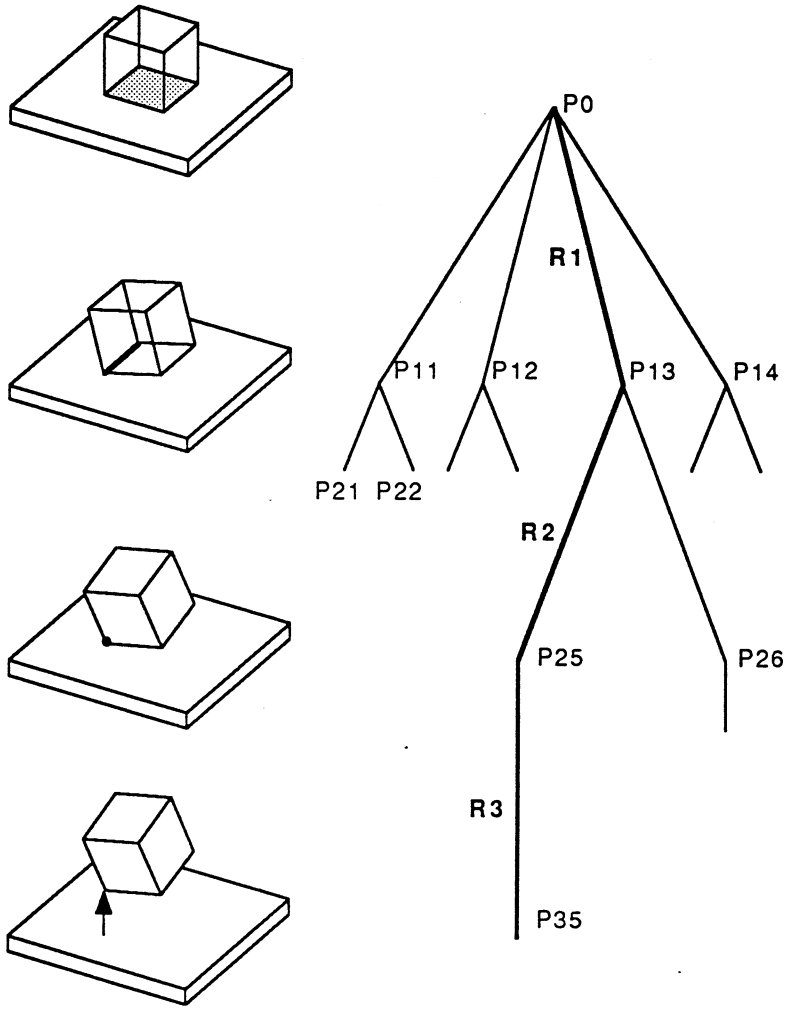
2. Des règles de contrôle.

Une règle est composée d'une condition devant correspondre (au sens de la mise en correspondance) à la situation géométrique courante et d'un certain nombre d'actions à effectuer. Les règles décrivant les actions de démontage comportent dans leur definition un poids qualifiant notre confiance vis-à-vis de cette information et une suite d'instructions paramètrables représentant le futur code engendré par le système. Le figure 6.2.2 présente un exemple d'enchainement de règles pour le "démontage" d'un cube posé sur un plan.

6.2.3 Construction et exploitation de l'ensemble solution

L'idée consiste à représenter l'ensemble des solutions dans un graphe (le graphe de démontage) de manière à avoir une vue globale des séquences d'actions possibles. Ce graphe contient les différentes situations de contact que l'on a rencontrées lors du démontage fictif (sommets du graphe) ainsi que les mouvements nécessaires à la transition entre deux situations consécutives (arcs du graphe).

Du fait de l'hypothèse de réversibilité du démontage, la recherche d'une solution se traduit par la recherche d'un chemin dans ce graphe. Ce chemin doit vérifier certains critères d'optimalité (nombre de situations intermédiaires, stabilité des contacts engendrés, ...). Ces critères sont difficilement quantifiables aussi avons nous préféré une description à l'aide de conseils. Ceux-là portent sur deux situations consécutives et sur la transition qui permet de passer de l'une à l'autre. Ils fournissent soit une note traduisant la qualité de l'action élémentaire, soit un ordre d'affinage du graphe de désassemblage à partir d'une situation singulière. Les différentes phases de la planification sont représentées sur la figure 6.4.



R1: "S'il y a un seul contact de type plan entre les faces F1 et F2, alors tenter de tourner autour d'une des arêtes de la face mobile"

R2: "S'il y a un seul contact de type linéïque entre l'arête A et la face F, alors tenter de tourner autour d'un axe passant par un point de A, perpendiculaire à A et coplanaire a F"

R3: "S'il y a un seul contact pontuel entre le point P et la face F, alors tenter une translation selon la normale extérieure à la face F"

Figure 6.3: Trace d'exécution de règles

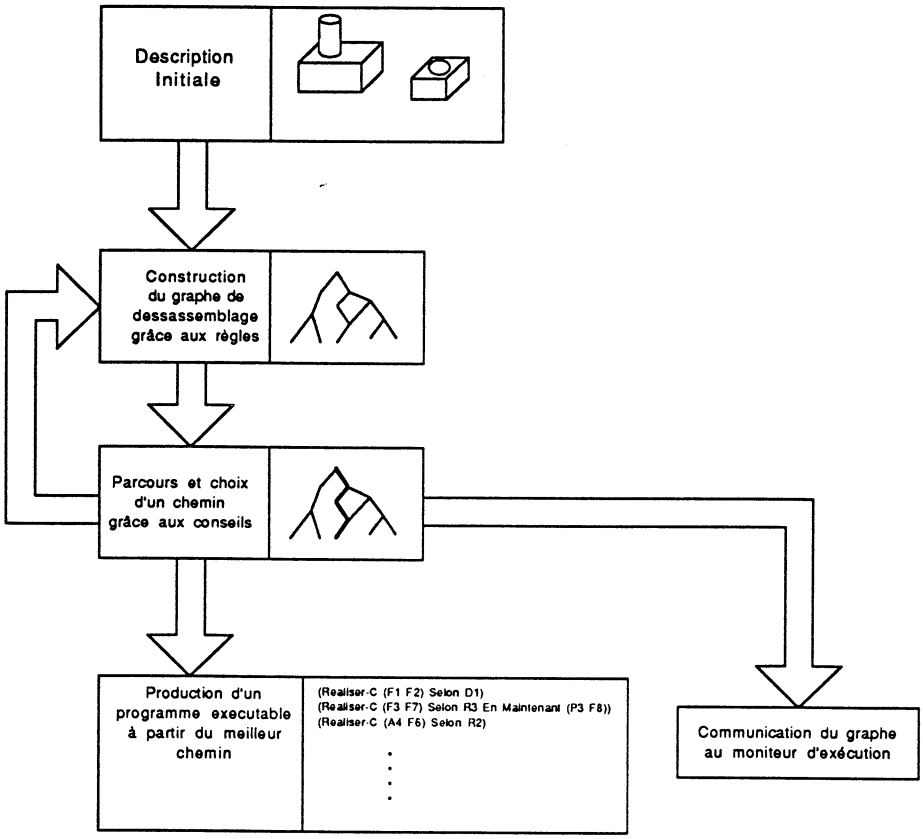


Figure 6.4: Cycle de planification

6.3 Utilisation de règles

Comme nous l'avons vu précédemment, les règles permettent d'effectuer le raisonnement géométrique nécessaire à la planification des mouvements fins de montage. Elles ont la forme suivante :

" **Si** <*situation*> **alors** <*stratégie* > "

<situation> représente une situation géométrique implicitement contenue dans le modèle et évaluée par le mécanisme d'activation des règles.

<stratégie> représente soit un choix de mouvements potentiels à appliquer, soit un ensemble de règles spécialisées à mettre en œuvre.

Les mouvements sont choisis parmi un ensemble de mouvements potentiels et leur amplitude est calculée à l'aide de fonctions appropriées. Les règles suscep-

tibles d'activer un sous-ensemble de règles spécialisées ont pour but de restreindre l'explosion combinatoire dû à la nature exhaustive du raisonnement par règles.

6.3.1 Représentation des règles

La syntaxe d'écriture est la suivante :

<règle> ::= (DEFRULE :NAME <idf>
:COMMENT <Chaîne de caractères servant de commentaires
:IF <expression d'activation>
:LET <expression lisp>)

<expression d'activation> ::= La syntaxe de l'expression d'activation est identique à celle utilisée pour les démons (voir paragraphe 4.3.4).

Remarque : La condition d'activation a pour rôle de caractériser une situation géométrique particulière pour laquelle la règle effectue un certain nombre d'actions : déclencher d'autre règles, proposer un(des) mouvement(s) de démontage. Elle porte à la fois sur les données numériques du modèle et sur les données symboliques présentes dans la base.

Considérons à titre d'exemple la condition suivante exprimée en langage courant : *"S'il y a contact entre les faces F1 et F2 et que la surface de la face F1 est supérieure à 20 alors ..."*. Sa transcription sous forme de condition est :

```
:IF (and [contact ?F1 ?f2]
         (greaterp (Surface ?F1) 20))
```

L'opérateur "and" de la condition s'applique sur des opérandes de types différents : le résultat de *[contact ?F1 ?F2]* est une liste alors que le résultat de *(greaterp (Surface ?F1) 20)* est un booléen. L'opérateur "and" utilisé ici n'est pas l'opérateur conjonctif habituel de la logique.

6.3.2 Définition des opérandes et opérateurs

Les opérandes sont de deux natures : expression lisp et résultat d'interrogation de la base de données. Le résultat d'une expression sera toujours un booléen (vrai ou

faux). L'évaluation est faite par l'intermédiaire de l'interpréteur du langage après avoir substitué aux variables leur valeur.
Le résultat d'une interrogation de la base de données est vrai ou faux si le test ne comporte pas de variables, sinon c'est l'ensemble des ensembles de valeurs vérifiant la question (Cf 4.3.4).

Exemple :

```
Base :       ( (Prd1 face-1 face2)(Prd2 face-1 face3)
               (Prd1 face-2 face-3)(Prd3 face-1 face-3))
Question :   [ Prd1 ?A ?B ]
Résultat :   (((A face-1)(B face-2)) ((A face-2)(B face-3)))
```

Remarque : Parmi l'ensemble des prédicats existants, certains sont symétriques ((contact F1 F2) ≡ (contact F2 F1)). Pour eux, les deux réponses seront fournies comme résultat.

L'expression des conditions se fait à partir de trois opérateurs {OR, AND, NOT} que nous allons étudier successivement :

NOT L'opérateur NOT s'applique uniquement à une expression où toutes les variables ont une valeur. En conséquence le résultat de l'expression à nier est un booléen. Cette restriction est importante et simplifie le problème d'un point de vue implantation aussi bien que compréhension pour le futur utilisateur.

Imaginons cet opérateur capable de faire la négation d'une question non complètement valuée. En se ramenant à un raisonnement ensembliste , nier une réponse consiste à calculer le complémentaire des points de solution fournis par l'expression.

Le modèle symbolique constitue une trace des valeurs pour lesquelles un prédicat donné est vrai. Par conséquent, quand une propriété est fausse (ou le devient) elle ne figure pas (plus) dans le modèle symbolique. Il est donc impossible avec la représentation actuelle de calculer le complémentaire d'un ensemble de données.

Avec cette interprétation de l'opérateur NOT, pour tester si une propriété est fausse il suffit de vérifier qu'elle ne figure pas dans le modèle symbolique.

Exemple :

```
Base :       ( (Prd1 face-1 face2)(Prd2 face-1 face3)
               (Prd1 face-2 face-3)(Prd3 face-1 face-3))
Question1 :  (NOT[ Prd1 face-1 face-2 ])
Résultat1 :  Faux ≡ ∅
Question2 :  (NOT[ Prd1 face-2 face-5 ])
Résultat2 :  Vrai
```

OR/AND Ces deux opérateurs peuvent s'appliquer à des expressions de type lisp ou à des expressions de type mise en correspondance. mais seuls les calculs faisant intervenir des expressions du second type nécessitent de détailler le mécanisme d'évaluation.

Grâce à l'interprétation ensembliste des opérateurs, le calcul d'un OR (respectivement d'un AND) se ramène à l'union (respectivement à l'intersection) d'ensemble de points.
Si l'intersection d'ensembles est simple car il suffit de vérifier qu'un point figure dans tous les ensembles, l'union nous pose un problème pratique qu'un simple exemple va mettre en évidence. Soient deux prédicats $P1$ et $P2$ de deux variables et leur ensemble caractéristique.

P1 $\longrightarrow ((x_1\ y_1)(x_2\ y_2)(x_3\ y_3))$

P2 $\longrightarrow ((x_1\ y_1)(x_3\ y_3)(x_5\ y_5))$

Calculons le résultat de (OR [P1 ?x ?][P2 ?x ?y]).

P1 ?x ? $\longrightarrow (((?x\ x_1))((?x\ x_2))((?x\ x_3)))$

P2 ?x ?y $\longrightarrow (((?x\ x_1)(?y\ y_1))((?x\ x_3)(?y\ y_3))((?x\ x_5)(?y\ y_5)))$

L'évaluation de [P1 ?x ?] nous fournit des solutions que l'on pourrait comparer à des "droites" de solutions (une valeur fixe et tous les seconds paramètres possibles) alors que [P2 ?x ?y] nous fournit seulement des "points". Dans la pratique, les résultats d'une interrogation de la base ne sont pas forcément de même arité. Pour pallier ce problème, l'union d'ensembles est ramenée à une simple concaténation des ensembles solutions fournis par chacun des prédicats.

opérandes

Type	Résultat
[Pred $?v_1$, $?v_2$, $\cdots$, $?v_n$]	{Ensemble des *points* vérifiant le prédicat}
(Exp-lisp)	Résultat de l'évaluation : `Vrai` ou `Faux`

opérateurs

AND	[Pred $?x_1$, $?x_2$, $\cdots$, $?x_n$]	(Exp-lisp)
[Pred $?y_1$, $?y_2$, $\cdots$, $?y_n$]	$[\cdots] \cap [\cdots]$ Ensemble des *points* communs	- ϕ si (Exp-lisp) = `Faux` - [Pred $?y_1$, $?y_2$, $\cdots$, $?y_n$] si (Exp-lisp) = `Vrai`

OR	[Pred $?x_1$, $?x_2$, $\cdots$, $?x_n$]	(Exp-lisp)
[Pred $?y_1$, $?y_2$, $\cdots$, $?y_n$]	$[\cdots] \cup [\cdots]$ Concaténation des ensembles solutions	- [Pred $?y_1$, $?y_2$, $\cdots$, $?y_n$] si (Exp-lisp) = `Faux` - T si (Exp-lisp) = `Vrai`

Figure 6.5: Tableau des opérandes et opérateurs

Les tableaux de la figure 6.5 résument les différents types d'opérandes reconnus pour l'écriture d'une expression d'activation, ainsi que les opérateurs pouvant être utilisés.

Remarque sur la propagation des variables :

La notion de propagation de variables est liée à l'utilisation d'expressions où figurent des opérandes dont les valeurs sont inconnues. Le but est alors de rechercher les valeurs que peuvent prendre ces variables de telle manière que l'expression en question soit vraie. Pour cela, lorsque l'évaluateur fixe des valeurs possibles pour une variable, il faut que celles-ci soient *propagées* à travers l'expression complète. Ce problème contraint légèrement l'écriture des expressions d'activation car il fait perdre la propriété de commutativité des opérateurs OR et AND.

Exemple :

(1) (AND [Face ?F1](> (surface ?F1) 20.0))

(2) (AND (> (surface ?F1) 20.0)[Face ?F1])

Supposons que l'évaluation se fasse de gauche à droite. L'expression (1) est évaluable car l'évaluation du premier terme détermine un ensemble de valeurs pour ?F1, qui est utilisé pour évaluer le second terme. En revanche, et pour la même raison, l'évaluation de (2) est impossible.

6.4 Graphe de démontage

La construction explicite de ce graphe permet une approche incrémentale du problème : Le système commence par proposer un certain nombre de trajectoires de démontage obtenues à partir d'un nombre restreint de règles puis lors de l'étude du graphe en vue de produire un programme de montage, un mécanisme (Cf 6.5) spécialisé propose éventuellement d'affiner des parties intéressantes du graphe. On contrôle ainsi l'explosion combinatoire dûe à l'emploi des règles.

6.4.1 Représentation du graphe

Les nœuds du graphe représentent les situations rencontrées lors du démontage fictif. Ils contiennent les informations suivantes :

- **position:** Position de l'ensemble en train d'être démonté, par rapport à sa position finale de montage.
- **contact:** Liste des couples d'entités géométriques en contact.
- **qualité:** Poids qualifiant notre confiance vis-à-vis de la possibilité d'amener l'ensemble à cette position.

Les arcs du graphe représentent les mouvements qu'ils faut effectuer pour passer d'une situation géométrique à une autre (d'un nœud à un autre). Ils contiennent les informations suivantes :

- **type:** Nature du mouvement de démontage : translation, rotation, ...
- **transformation:** Transformation entre les positions reliées par l'arc. La transformation ne tient pas compte de la nature de la trajectoire.
- **qualité:** Poids qualifiant la robustesse du mouvement.

6.4.2 Construction du graphe

Construire le graphe consiste à conserver une trace des différentes opérations fictives de démontages envisagées. On conserve dans les nœuds et les arcs les caractéristiques des situations géométriques rencontrées et des mouvements virtuellement effectués.

Principe de construction

Pour une configuration donnée, les règles de démontage proposent un ensemble de mouvements potentiels permettant de décontraindre l'objet manipulé. Parmi ces mouvements certains doivent être rejetés pour des raisons diverses :

- amplitude de mouvements incompatible avec les incertitudes.
- direction de mouvements tendant à faire interpénétrer les différents objets.
- autres ...

L'élimination des mouvements impossibles se fait par l'intermédiaire de filtres géométriques variés :

- comparaison de l'amplitude souhaitée avec l'amplitude maximale possible
- étude du vecteur vitesse du mouvement et des plans tangents aux différents contacts.
- ...

Ces différents filtres sont regroupés dans la fonction TRY. Elles est utilisée par les règles pour proposer un mouvement de démontage et accepte des arguments permettant de spécifier la nature du mouvement à effectuer et le code exécutable correspondant à l'opération de montage.

Exemple :
```
(TRY :move 'rotation
     :amplitude 50
     :axe (make-axe  :pt '?p
                     :vect (arete-to-vect '?a))
     :quality (longueur '?a)
     :code        (realiser-C ((contact '?f1 '?f2))
                     suivant (rotation :pt '?p
                                       :vect '?a)
                     en maintenant ((contact '?a 'f2))))
```

Le champ `:code` contient une description générique d'une instruction réalisant la situation géométrique. Pour obtenir le code complet, la valeur de chaque variable, calculée lors de l'activation de la règle, est substitué dans l'expression.

Le champ `:amplitude` permet de décrire l'amplitude du mouvement que l'on veut faire. Deux cas se présentent :

- L'argument est numérique : le mouvement proposé est un mouvement correctif.
- L'argument est $\star$: le mouvement de démontage est maximal. C'est la plus grande amplitude de mouvement possible qui ne change pas l'ensemble des propriétés géométriques.

Algorithme de construction

Le graphe est construit en conservant la trace des différentes situations rencontrées. Le parcours s'effectue *en profondeur d'abord*. La construction complète s'effectue en appellant la fonction DESAS de la manière suivante :

DESAS(<situation initiale>,{R_0,R_1, ..., R_n},$\emptyset$)

Où
- <situation initiale> caractérise l'ensemble à démonter (virtuellement) en position assemblé.
- $\{R_0, R_1, \ldots, R_n\}$ représente l'ensemble des règles utilisées pour guider le démontage.
- Le troisième argument est la liste des situations déjà envisagées. Elle est nulle lors du premier appel.

procédure DESAS (situation, {règles}, {déjà traité})

1. {règles utilisées} ← {règles};
2. {mouvements} ← ∅;
3. **tant que** {règles utilisées} ≠ ∅ **faire**
   (a) Sélectionner la première règle de {règles utilisées} et la supprimer de cet ensemble;
   (b) **Si** sa condition d'activation est vérifiée **alors**
   - Exécuter sa partie "instructions";
4. {nouvelles situations} ← ∅;
5. **pour chaque** mouvement **dans** {mouvements} **faire**
   (a) nouveau-nœud ← générer-nœud(situation,mouvement);
   (b) nouvel-arc ← générer-arc(mouvement);
   (c) établir les liens entre situation, nouvel-arc et nouveau-nœud;
   (d) **Si (not** (position(nouveau-nœud) ∈ {déja traité})) **alors**
      i. {nouvelles situations} ← {nouvelles situations} ∪ nouveau-nœud;
      ii. mettre à jour l'ensemble {déja traité};
6. **pour chaque** situation **dans** {nouvelles situations} **faire**
   (a) DESAS(situation,{règles},{déjà traité});

6.5 Recherche d'une solution

6.5.1 Parcours du graphe

Principe

La recherche d'une solution optimale en parcourant un graphe est un problème classique de l'informatique. Des méthodes permettent de déterminer une solution sans avoir à effectuer un parcours complet du graphe. L'utilisation de ces algorithmes de parcours, nommés A*, est impossible dans notre cas car on ne dispose pas d'une estimation du poids de la solution optimale. Les poids proposés ne sont pas significatifs des différences entre la position étudiée et la solution optimale mais traduisent notre confiance vis-à-vis de situations géométriques et de mouvements élémentaires. La détermination de poids permettant de mettre en œuvre des algorithmes de recherche de type A*, nécessite d'établir une mesure de la complexité de l'assemblage, ce qui n'existe pas actuellement.

En raison de cette difficulté, le système doit effectuer un parcours complet du graphe de désassemblage. Pour qualifier la solution et choisir ainsi la meilleure, il dispose d'un ensemble de **conseils** qui fixent les poids des nœuds et des arcs à partir des caractéristiques géométriques de la position et de la nature de la transition entre positions. Le parcours est effectué en utilisant la méthode du parcours *"en profondeur d'abord"*.

D'un point de vue théorique, seul un parcours complet du graphe garantit que la solution trouvée est optimale vis-à-vis de nos critères. En revanche, d'un point de vue pratique, si on se contente de ne développer pour chaque nœud que les trois meilleurs arcs, il s'avère que la solution est excellente. Plusieurs interprétations peuvent être faites :

- Les quelques cas traités n'ont pas permis de rencontrer le plus mauvais cas de démontage.
- Les conseils fixant les poids des arcs sont peu nombreux et sont sans doute très sélectifs. Leur augmentation risque de prendre en défaut cette constatation.

Exemple de conseil : L'utilisateur veut traduire la connaissance géométrique suivante *"Soit x le nombre de contacts de la situation courante, si le nombre de contact de la situation suivante est supérieur à x alors ce n'est pas une très bonne trajectoire*

de montage". Cette information conduit à relaxer progressivement les degrés de liberté.

Sa traduction sous forme de conseil est la suivante :

```
(defconseil
   :situation-initiale (setq x Nb-contact)
   :situation-finale (> Nb-contact x)
   :quality -10 )
```

Algorithme de parcours

procédure PARCOURS (nœud,)

1. meilleure note ← -∞
2. meilleur chemin ← ()
3. tant que {chaînage(nœud)} ≠ ∅
   (a) lien ← premier élément de {chaînage(nœud)}
   (b) retirer lien de {chaînage(nœud)}
   (c) arc ← arc successeur désigné par lien
   (d) position atteinte ← nœud successeur de l'arc
   (e) appliquer l'ensemble des conseils
   (f) {note,chemin} ← PARCOURS(position atteinte)
   (g) si note + poids(arc) + poids(position atteinte) > meilleure note alors
      i. meilleure note ← note
      ii. meilleur chemin ← chemin
4. retourner {meilleur nœud, nœud ⊕ meilleur chemin}

6.5.2 Affinage du graphe

La phase la plus coûteuse de la planification est la construction du graphe car elle repose sur le calcul des propriétés géométriques d'une situation à partir d'un

modèle. Afin de disposer d'un graphe suffisamment précis sans toutefois accroître le temps de calcul, le graphe est d'abord développé en choisissant les mouvements qui paraissent être les plus adaptés. Il est ensuite affiné en cas d'échec ou si une configuration singulière est rencontrée lors du parcours du graphe, les conseils ont donc deux tâches :

- Ils contribuent à fixer les poids des nœuds et des arcs du graphe de démontage.
- Ils permettent de développer plus finement un chemin du graphe, en préconisant l'emploi d'un ensemble de règles plus spécialisées.

Au fur et à mesure que l'on affine le graphe, on prend en compte un ensemble de règles de plus en plus important et donc de plus en plus coûteux en temps machine. Heureusement, ce coût supplémentaire ne s'applique qu'à un nombre limité de situations.

La syntaxe La syntaxe d'écriture des conseils est la suivante :

<conseil> ::= (DEFCONSEIL
:situation-initiale <expression lisp>
:situation-finale <expression lisp>
[:actions (DESAS <situation><règles>)]
[:quality-init <expression numérique entière>]
[:quality-arc <expression numérique entière>]
[:quality-fin <expression numérique entière>]

<situation> ::= /nœud du graphe correspondant à une position/

<règles> ::= /ensemble de règles de démontage (Cf 6.3)/

L'activation L'activation d'un conseil se fait simplement en calculant la valeur des expressions lisp correspondant aux situations initiale et finale. Si elles sont toutes les deux vraies, alors deux effets peuvent se produire :

- Les poids des nœuds et/ou de l'arc sont modifiés en ajoutant la note de qualité.
- Le graphe est affiné à partir d'une situation jugée intéressante.

Cette possibilité particulière, donnée aux conseils, d'affiner davantage le graphe permet une étude plus fine tout en limitant l'explosion combinatoire. Les conseils relancent le démontage par l'intermédiaire de la fonction DESAS en utilisant un ensemble de règles plus spécialisées. Dans l'implantation actuelle du système, le cycle (règles $\longleftrightarrow$ conseils) ne s'exécute qu'une fois.

6.6 Synthèse du plan de montage

La synthèse du programme exécutable s'effectue à la fin du cycle parcours $\longleftrightarrow$ affinage. A ce stade, le système dispose d'un chemin dans le graphe où figure toutes les situations géométriques retenues (les nœuds) et tous les mouvements élémentaires intermédiaires (les arcs) permettant de passer d'une situation à une autre. Il est donc possible de générer un ensemble de mouvements compliants et gardés réalisant le montage souhaité.

Soient les situations S1 et S2, et la transition T de S1 vers S2. La but est de produire l'instruction permettant de passer de S1 à S2 selon T. S1 et S2 sont caractérisées par l'ensemble des couples d'entités en contact :

$$\mathbf{S1} : ((e_{1,1}, e_{1,2})(e_{1,3}, e_{1,4}), \ldots (e_{1,2p+1}, e_{1,2p+2}))$$

$$\mathbf{S2} : ((e_{2,1}, e_{2,2})(e_{2,3}, e_{2,4}), \ldots (e_{2,2q+1}, e_{2,2q+2}))$$

Le mouvement doit réaliser une compliance sur les entités de S1 $\cap$ S2 avec comme garde S2-S1. La synthèse complète du mouvement utilise les informations contenues dans les champs de l'arc pour déterminer le type de mouvement à effectuer et les directions privilégiées pour accomplir sa tâche.

Le mouvement est décrit à l'aide d'un langage spécialisé que nous avons développé pour répondre aux besoins spécifiques de la programmation automatique (Cf §7). Ce langage a accés au même modèle que le planificateur. Cela lui permet de calculer

les paramètres qui lui manquent pour réaliser effectivement le mouvement (valeur du torseur de force, par exemple).

6.7 Implantation et expérimentation

6.7.1 Implantation

Les différents modules nécessaires au système PAMELA et plus généralement au système SHARP ont été développés sur des stations de travail SUN (3/260 et 3/75). Les langages utilisés sont Lucid Common Lisp et C. Le langage C a été utilisé pour réécrire certaines parties dont les performances étaient pénalisées par Lisp (module d'affichage et bibliothèque de fonctions géométriques entre autres).

Le robot utilisé pour valider nos résultats est un Scemi 6 axes, muni d'un capteur de force, dans le poignet, lui permettant de connaître les six composantes du torseur de forces. Sa programmation s'effectue à l'aide du langage LM. L'ordinateur de commande est un HP1000.

La communication entre la station de planification (SUN 3/260) et l'ordinateur de commande est réalisée à l'aide d'une liaison série de type RS232. Le langage de commande que nous utilisons (Cf §7) est implanté à la fois en Lisp pour ce qui concerne le calcul des paramètres nécéssaires à l'exécution des instructions, et en LM pour ce qui concerne le contrôle explicite du robot.

6.7.2 Expérimentation

Les différentes expériences ont eu lieu après avoir effectué un calibrage minutieux entre le plateau où sont disposés les composants à assembler et le robot. La correspondance entre les positions réelles des composants et les positions du modèle s'effectue manuellement.

La validation du système PAMELA s'est effectuée en planifiant l'assemblage d'une pièce en forme de L. L'annexe A.4 montre le programme généré pour ce montage ainsi que des traces graphiques des différentes configurations. D'autres exemples ont été traités de manière théorique en produisant le programme que l'on escomptait.

Afin de réduire le coût de la planification, les exemples traités sont tous constitués d'objets polyédriques mais celui-ci reste cependant élevé en raison des nombreux calculs géométriques nécéssaires.

Chapitre 7

Langage cible

7.1 Nécessité d'un langage spécialisé

De nombreux langages de manipulation existent déjà (AL, LM, autre ...) mais leur utilisation pour la description d'une tâche d'assemblage reste complexe. Ces langages sont issus d'un langage informatique classique (PASCAL, LISP, ...) auxquels ont été ajoutées des primitives spécialisées adaptées à la robotique (gestion de repères, déplacements, capteurs, ...). L'élément central de l'interpréteur est un **changeur inverse** permettant de transformer des coordonnées cartésiennes en coordonnées articulaires [Pau81].

Si l'univers d'évolution est parfaitement défini et si les effets de l'incertitude sont négligeables, une simple description des positions à atteindre est suffisante. Cet univers parfait ne correspond cependant pas à la réalité et le robot ne peut se contenter d'exécuter des commandes figées. Il faut qu'il puisse percevoir le monde qui l'entoure et réagir en fonction des situations rencontrées. Le robot peut compléter sa connaissance du monde par l'intermédiaire de capteurs extérioceptifs variés : capteur de forces, de toucher, utilisation de la vision, ...

Dans le cadre de la robotique d'assemblage, le capteur privilégié est le capteur de forces qui fournit en un point précis les valeurs du torseur de forces (trois forces + trois moments). Les différentes technologies utilisées (jauges de contrainte, effet piezo-électrique, ...) consistent toutes à transformer les effets des forces en une tension électrique. Cette mesure électrique est ensuite communiquée à l'ordinateur commandant le robot par l'intermédiaire d'un convertisseur analogique/digital. Les

informations alors utilisables par le programme d'assemblage sont constituées par des grandeurs numériques correspondant aux forces mesurées.

Si tous les langages de manipulation permettent d'utiliser des mesures en provenance de capteurs divers, leur emploi en revanche n'est pas aisé. Chacun d'eux permet de lire une(des) valeur(s) numérique(s) caractérisant la mesure transmise par le capteur mais l'interprétation est laissée à la charge du programmeur.

Les différents langages de programmation de robot ne permettent donc que de décrire la tâche à effectuer qu'en terme de positions à atteindre (positions articulaires ou positions cartésiennes), et en terme de grandeurs à mesurer. Ces différentes informations (positions et valeurs de capteur) constituent un modèle de l'univers du robot. Celui-ci est cependant trop pauvre pour décrire facilement une tâche d'assemblage. Les langages de la robotique doivent utiliser des modèles plus riches et permettre ainsi une description en termes de situations géométriques à réaliser. Les paramètres physiques de l'exécution de la tâche sont alors calculés automatiquement à partir du modèle de l'univers du robot.

7.2 Notre langage

Dans le cadre du projet SHARP, nous avons développé un langage capable de travailler sur le même modèle que le planificateur. Il comporte six instructions dédiées à chacun des modules de planification :

1 $(Saisir < situation >)$
2 $(Lacher < situation >)$
} module de saisie.

3 $(Realiser - S < situation > Via < chemin >)$ } module de transport.

4 $(Realiser - C < Contact >$ $Suivant < d >$ $[Maintenir < Contact >])$
5 $(Supprimer - C < Contact >$ $Suivant < d >$ $[Maintenir < Contact >])$
6 $(Corriger < situation >$ $Suivant < d >$ $Amplitude < a >)$
} module de planification de mouvements fins

Le langage informatique support de l'implantation est LISP. Toutes les instructions précédentes peuvent êtres incluses dans les structures de contrôle du langage hôte (while (<condition>)(<instruction>)*, ...). La syntaxe exacte des instructions propres à la robotique est décrite dans l'annexe A.3.

7.2.1 Les instructions (Saisir <...>) et (Lacher <...>)

Le but de ces instructions est de réaliser (cesser) une action de préhension. L'instruction `Saisir` consiste à se déplacer en mode cartésien de la position courante à la position décrite comme argument, puis à actionner la pince à mors parallèles pour réaliser la saisie. L'instruction `Lacher` effectue les même opérations mais en sens inverse : suppression de la liaison pince-objet puis déplacement à la position de replie.

L'intégration d'un mouvement d'approche (de "déproche") dans une commande de saisie permet de prendre en compte dans une même instruction les problèmes propres à la saisie comme le choix de surfaces d'appuie pour les mors (phase A^2LS de la saisie automatique) et les problèmes liés à l'environnement : possibilité de réaliser la saisie sans collision avec les objets environnants (phase A^2G de la saisie automatique).

7.2.2 L'instruction (Realiser-S ...)

Cette instruction permet de décrire les grands mouvements que doit réaliser le bras manipulateur. Les paramètres de l'instruction sont la position à atteindre et la description du chemin permettant de passer de la position courante à la position objectif.

Dans le cadre de la programmation automatique, la planification s'effectue en raisonnant en terme de configuration de manipulateur plutôt qu'en terme de position à atteindre. Dans ce but, la description de la `<situation>` et du `<chemin>` peut se faire grâce aux coordonnées articulaires du robot.

7.2.3 Les instructions (Realiser-C ...) et (Supprimer-C ...)

Ces deux instructions sont propres aux mouvements fins d'assemblage. Ce sont des instructions réalisant des mouvements compliants et/ou gardés. La garde est définie par la description des contacts à réaliser ou à supprimer, la compliance est spécifiée par la liste des contacts à maintenir pendant l'exécution du mouvement.

Ces instructions n'ont pas besoin d'arguments numériques pour décrire complètement le mouvement. Elles calculent les données nécessaires à partir du modèle géométrique de l'univers où évolue le robot. Grâce à cette facilité, une tâche d'assemblage peut être décrite comme une succession de configurations géométriques définies en termes de degrés de liberté.

où <insertion> représente la relation à réaliser, <chanfrein> est un prédicat indiquant s'il y a contact avec le chanfrein et <aligne-axe> est une relation représentant l'alignement des axes du goujon et du chanfrein. Le paramètre I représente l'erreur maximale associée à la relation d'alignement des axes.

7.3 L'implantation

Une version expérimentale de ce langage a été réalisée afin de pouvoir planifier puis exécuter un montage complet. Les instructions les plus difficiles à implanter sont celles réclamant la réalisation de mouvements compliants gardés (Réaliser-C Supprimer-C). Elles ont été réalisées grâce au travaux de C.Gandon et [Gan86] permettent de décrire des relations de contact entre objets polyédriques. Les différents cas implantés sont tous utilisés dans l'exemple canonique décrit dans l'annexe A.4.

Les différentes manipulations envisagées ont permis de mettre en évidence la facilité de description des tâches d'assemblage mais aussi la difficulté inhérente au calcul des paramètres de mouvements. Des formules, permettant ce calcul très complexe, ont été établies dans le cas des objets polyédriques. Pour que l'on puisse réellement parler de "langage de programmation", la classe des objets manipulables doit être étendue.

Le système PAMELA utilise ces deux instructions pour décrire les différentes configurations que doivent vérifier les composants en cours d'assemblage. Cette description à partir des degrés de liberté (des contact) permet de s'affranchir de la description de l'incertitude acceptée pour la réalisation d'une situation géométrique.

7.2.4 L'instruction (Corriger ...)

En raison de l'incertitude, il n'est pas toujours possible de réaliser une situation géométrique de manière exacte. En réalité, une situation physique réelle se compose d'une relation géométrique et d'une contrainte sur l'incertitude :

<situation> ≡ <relation géométrique> + <contrainte d'incertitude>

La réalisation d'une situation ne peut se faire qu'en effectuant un processus itératif conservant la relation géométrique et réduisant l'incertitude à chaque itération. La réduction de l'incertitude n'est plus alors limitée que par les contraintes physiques de l'univers : précision du manipulateur, des capteurs divers, incertitudes de position crées par l'incertitude géométrique des objets, ...

Ce processus itératif est réalisé par l'intermédiaire d'une boucle constituée d'une condition d'arrêt portant sur les valeurs de l'incertitude et d'un corps de boucle décrivant les mouvements correctifs à effectuer pour diminuer l'incertitude.

processus de réduction de l'incertitude ≡ $\begin{cases} \textit{condition d'arrêt} & \rightarrow & \textit{test sur l'incertitude} \\ \textit{corps de boucle} & \rightarrow & \textit{action correctives} \end{cases}$

L'instruction `Corriger` est utilisée dans le corps de boucle pour décrire les mouvements permettant de converger vers la situation géométrique recherchée. Afin de maintenir la cohérence du modèle symbolique, la relation géométrique concernée par la correction, est explicitement citée dans les paramètres de l'instruction.

Exemple : L'exemple suivant décrit la statégie de correction axiale utilisée lors d'une insertion cylindrique.

```
(realiser-s <insertion> (garde <chanfrein>))
(while (not <insertion>)
       (if <chanfrein> (corriger <aligne-axe> (suivant (F_x F_y 0))
                                              (amplitude I/2))))
```

Chapitre 8

Conclusion

Les recherches effectuées dans le cadre de la programmation automatique de tâche d'assemblage nous ont conduit à reflechir sur la méthodologie de l'assemblage et la réalisation d'un planificateur (PAMELA) nous a permis de tester en vraie grandeur les hypothèses que l'on effectue habituellement en ne s'appuyant que sur le *"bon sens"*.

Notre contribution a été tout d'abord de définir puis de développer un système de modélisation ouvert : SIMONS. Les multiples représentations nécessaires à la planification (représentation numérique, symbolique, modèle cinématique du robot), nous ont conduit à définir un mécanisme (les démons) assurant une communication facile entre les différentes représentations. Ce mécanisme est aussi utilisé pour assurer le maintien automatique de la cohérence entre les différentes représentations car toute modification d'une des représentations a des répercussions sur l'ensemble du modèle. Enfin ce mécanisme a été utilisé avec profit pour modéliser les lois physiques de l'univers, rendant ainsi notre modèle plus réel. Grâce à lui, un objet qu'on lache, tombe et la collision de deux objets est signalée automatiquement.

Sur cette base que constitue notre modèle, nous avons alors pu construire un planificateur de mouvements fins d'assemblage. Grâce à la bi-représentation de notre modèle (représentation numérique et symbolique), nous avons pu aborder la planification des mouvements fins comme un problème de raisonnement géométrique. Notre travail a alors consisté à "mécaniser" ce raisonnement pour pouvoir le reproduire sur un calculateur.

Cette phase de réflexion sur la meilleure manière de réaliser un assemblage à l'aide d'un robot nous a conduit à établir un certain nombre de "principes" garantissant la robustesse d'un programme de montage. Notre réflexion s'est portée essentiellement sur les objets polyhédriques. Notre approche a consisté à raisonner en terme de degrés de liberté car ces derniers permettent d'évaluer la complexité d'un assemblage : plus on doit supprimer de degrés de liberté en une seule opération, plus elle est complexe.

Le module de raisonnement géométrique que nous avons réalisé utilise, en plus du modèle, ces principes d'assemblage pour produire un programme robuste. La représentation de ces connaissances est faite à l'aide de règles afin qu'on puisse les traiter comme une donnée banalisée et ainsi l'exclure de la sémantique du programme de planification.

L'expérimentation sur un manipulateur des programmes générés nous a conduit à aborder le problème des langages de programmation pour les robots d'assemblage. Sans apporter une solution complète à ce problème, nous avons défini, à partir des lacunes des langages existant, notre propre langage. La solution proposée autorise une programmation qui ne s'effectue plus en fonction des effets des actions du robot mais qui décrit directement les causes de ces effets.

L'ensemble de ces recherches et de ces expérimentations nous ont permis de montrer qu'il est possible de planifier automatiquement des mouvements fins d'assemblage.

L'orientation future de nos recherches va consister à remettre en question l'hypothèse de réversibilité. De même qu'il est possible pour un être humain ayant des "connaissances" en mécanique de remonter un assemblage inconnu, un système doué de raisonnement géométrique doit être capable d'effectuer le même travail. Grâce une telle démarche, la manipulation d'objets déformables (ressort entre autres) ne nécessitera plus alors de modèle exacte de leur déformation. En revanche, une telle approche réclame une étude de l'assemblage en terme de liaisons technologiques. Plus simplement, il faut mettre en évidence les "connaissances" qu'utilise quotidiennement un mécanicien puis les transposer sous une forme exploitable par un module de raisonnement géométrique. Comme nous le voyons, le problème de la planification des mouvements fins d'assemblage reste ouvert.

Bibliographie

[ACB80] N. Ahuja, R.T. Chien, and N. Bridwell. Interference detection and collision avoidance among three dimensional objects. *Annual National Conference on Artificial Intelligence*, Août 1980.

[Bak73] N. Bakhvalov. *Méthodes Numériques.* Edition MIR, Moscou, deuxième éditions edition, 1973.

[Bro82] R.A. Brooks. Symbolic error analysis and robot planning. *International Journal of Robotics Research*, vol 1(no 4), Hiver 1982.

[Bro83] R.A. Brooks. Planning collision free motions for pick and place operations. *First International Symposium on Robotics Research*, August 1983.

[Cam81] S. Cameron. *Bridget: a geometric body modeller.* Technical Report, University of Edinburgh, July 1981. DAI Draft Working Paper.

[Can86] J. Canny. A computational approach to edge detection. In *IEEE Conf on PAMI*, pages 679–698, 1986.

[CL84] J. Pertin-Troccaz C. Laugier. Graphic simulation as a tool for debugging robot control programs. In *1st International Symposium on Design and Synthesis*, Tokyo, July 1984.

[Duf83] B. Dufay. *Apprentissage par induction en Robotique : Application à la*

synthèse de programmes de montage. Thèse de 3ème cycle, INPG, Juin 1983.

[Gan86] C. Gandon. *Introduction de la compliance dans la programmation des robots.* Thèse de 3ème cycle, INPG, Octobre 1986.

[Ger85] F. Germain. Génération statique de trajectoires : un algorithme paramètrable par le contexte. In *5ème Congrès AFCET Reconnaissance des Formes et Intelligence Artificielle*, Grenoble, Novembre 1985.

[Gru81] A. Grumbach. *Synthèse automatique de programmes d'assemblage pour un robot.* Technical Report, CERCI, Juin 1981.

[JD55] R.S. Hartenberg J. Denavit. A kinematic notation for lower-pair mechanisms based on matrices. *Journal of Applied Mechanics*, ASME, vol 77 , series E, June 1955.

[Lau82] C. Laugier. *LISP-3D : logiciel graphique pour la manipulation et la visualisation de scènes tridimensionnelles.* Technical Report no 328, IMAG, Septembre 1982.

[Lau87] C. Laugier. *Raisonnement géométrique et méthodes de décision en robotique. Application à la programmation automatique des robots.* Thèse d'Etat, INPG, Décembre 1987.

[LB85] T. Lozano-Perez and R.A. Brooks. *An approach to automatic robot programming.* Technical Report, Artificial Intelligence Laboratory, Avril 1985.

[Lel80] J. Lelong Ferrand. *Cours de mathématiques supérieures , 3ème édition.* 1980.

[LJM*87] T. Lozano-Perez, J.L. Jones, E. Mazer, P.A. O'Donnell, L. Grimson , P. Tournassoud, and A. Lanusse. Handey: a robot system that recognizes, plans and manipulates. In *IEEE*, pages 843–849, 87.

[Loz76] T. Lozano-Perez. *The design of a mechanical assembly system.* Technical Report AI-TR-397, Artificial Intelligence Laboratory, Cambridge, Décembre 1976.

[Loz81] T. Lozano-Perez. Automatic planning of manipulator transfer movements. In *IEEE Trans. on Systems, Man and Cybernetics (SCM-11)*, pages 681–698, 1981.

[Loz83] T. Lozano-Perez. Spatial planning: a configuration space approach. In *IEEE Trans. on Computers (C-32)*, pages 108–120, 1983.

[LP83] C. Laugier and J. Pertin. Automatic grasping: a case study in accessibility analysis. In *International Conference on Advanced Software in Robotics*, publié dans "Advanced Software in Robotics, Liège, Mai 1983. édité par A.Danthine et M.Géradin, North Holland, 1984.

[LP85] C. Laugier and J. Pertin-Troccaz. S.H.A.R.P.: a system for automatic programming of manipulation robots. In *3rd International Symposium of Robotics Research*, Gouvieux (France), Octobre 1985.

[LT86] C. Laugier and P. Theveneau. Planning sensor-based motions for part-mating using geometric reasoning. In *ECAI'86*, Brighton, Juillet 1986.

[LW77] T. Lozano-Perez and P.H. Winston. Lama: a language for automatic mechanical assembly. In *5th International Joint Conference on Artificial Intelligence*, M.I.T., Cambridge, Août 1977.

[Mas82] M.T. Mason. *Manipulator grasping and pushing operations.* Technical Report, Artificial Intelligence Lab., Cambridge, Juin 1982.

[Maz81] E. Mazer. *Réalisation d'un support expérimental de recherche pour le projet robotique PANDORE. Définition et implantation du langage LM.* Thèse de 3ème cycle, Institut National Polytechnique, Grenoble, Janvier 1981.

[Maz82] E. Mazer. *An algorithm for computing relative positions between two objects from symbolical specifications.* Technical Report no 297, IMAG, 1982. Rapport de Recherche.

[Mer86] J.P. Merlet. *Contribution à La Formalisation De La Commande Par Retour D'efforts En Robotique. Application A La Commande De Robots Paralleles.* Thèse de 3ème cycle, Université Paris 6, Juin 1986. Thèse de 3ème cycle.

[Mir81] J.F. Miribel. *Un langage pour la représentation de relations géométriques entre objets.* Technical Report N^0: 251, IMAG, Juin 1981. Rapport de recherche.

[MM85] E. Mazer and J.F. Miribel. *Le langage LM.* Collection Techniques Avancées de l'Informatique, Janvier 1985.

[NW80] J.L Nevins and D.E. Whitney. Assembly research. In *The Industrial Robot*, pages pp27–43, Mars 1980.

[Pas88] M.B. Pasquier. *Programmation automatique des robots : Système d'évitement de collision.* Thèse de l'Institut National Polytechnique de Grenoble, LIFIA, Decembre 1988. A paraître.

[Pau81] R.P. Paul. *Robot manipulators: mathematics, programming, and control.* The MIT Press, 1981.

[Per84] J. Pertin-Troccaz. *S.M.G.R. : un système de modélisation géométrique et relationnelle pour la Robotique.* Technical Report no 422, IMAG, Juin 1984. Rapport de recherche.

[Per86] J. Pertin-Troccaz. *Modélisation du raisonnement géométrique pour la programmation des robots.* Thèse de l'Institut National Polytechnique de Grenoble, LIFIA, Mars 1986.

[Per87] J. Pertin-Troccaz. On-line automatic robot programming: a case study in grasping. In *IEEE*, pages pp 1292–1297, 1987.

[Pug85] P. Puget. *Problèmes de prise en compte d'incertitudes en Robotique d'assemblage.* Technical Report, LIFIA/IMAG Laboratory, Juin 1985. Rapport de DEA.

[PW82] E. Pervin and J.A. Webb. *Quaternions in computer vision and robotics.* Technical Report, CMU reports, 1982.

[Rl82] A.A.G. Requicha and H.B. Vœlcker. Solid modelling: a historical summary and contemporary assessment. In *IEEE, Computer Graphics and its applications*, Mars 1982.

[SGS84] B.E. Shimano, C.C. Geschke, and C.H. Splalding. Val-IÏ a new robot control system for automatic manufacturing. In *IEEE International Conference on Robotics*, Atlanta, Mars 1984.

[SS83] J.T. Schwartz and M. Sharir. *On the piano Movers Problem I,II,III,IV,V.* Technical Report Reports 39,41,52,58,83, Courant Institute Technical, 1981-1983.

[Tou87] B. Faverjon P. Tournassoud. A local based method for path planning of manipulators with a high number of degrees of freedom. In *International Conference on Robotics and Automation IEEE*, Raleight, April 1987.

[TP88] P. Theveneau and M. Pasquier. A geometric modeler for an automatic robot programming system. In *NATO Advanced Research Workshop on "CAD-Based Programming for Sensor Based Robots"*, Il CIOTTO, Juin 1988.

[Val85] J.M. Valade. *Raisonnement géométrique et synthèse de trajectoire d'assemblage.* Thèse de Docteur-Ingénieur, Laboratoire d'Automatique et d'Analyse des Systèmes, Toulouse, Janvier 1985.

[Win77] P. Winston. *Artificial Intelligence.* Addison-Wesley Publishing Company, 1977.

Annexe A

A.1 Insertion d'un goujon

Nous nous proposons de réaliser l'insertion d'un goujon à l'aide d'un robot. En A.1.1 la tâche est décrite à l'aide d'un langage classique (LM) tansdis que en A.1.2 elle est réalisée grâce au langage géométrique servant de langage cible au système SHARP.

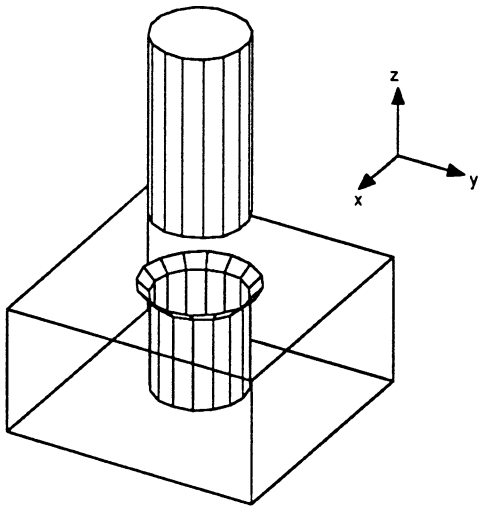


Figure A.1: Insertion d'un goujon

A.1.1 Insertion décrite en LM

```
procedure insert(repere surftrou);
co Cette procédure a pour but d'inserer un goujon
   dans un alesage chanfreine;
co Declarations;
   repere init,trou,premier,deuxieme;
   vecteur v1;
   entier i;
   reel rayon,fmz,fmx,fmy;
co Debut des instructions;
   debut
   trou := surftrou*translat(vz,-50.0);
co Positionnement du goujon au dessus du trou;
   deplacer robota surftrou;
   fixer vitesse robot a 0.2;
co Tant que l'on est dans le chanfrein;
   deplacer robot a trou jusqu'à (fz+fz+fz)≥24.0;
co Recherche du bord du chanfrein;
   tantque distance(robot,trou) ≥ 30.0 faire
         fmx:=(fx+fx+fx)/3; fmy:=(fx+fy+fy)/3;
         v1:=vect(fmx,fmy,0.0);
         Co Memorisation du premier contact;
         premier:=robot;
         Co recherche du bord oppose;
         lier trou a robot;
         deplacer robot de translat(v1,10.0) jusqu'à
               ((fx+fx+fx)*fmx≤0.0) ou ((fy+fy+fy)*fmy<+0.0);
         fmx:=(fx+fx+fx)/3;fmy:=(fy+fy+fy)/3;
         co correction si nécessaire;
         si (abs(fmx)+abs(fmy))≥5.0 alors
               deuxieme:=robot;
               rayon:=distance(premier,deuxieme)/2;
               deplacer robot de translat(v1,-rayon);
         finsi;
         delier trou de robot;
```

```
        deplacer robot a trou jusqu'à (fz+fz+fz)≥24.0;
    finfaire;
co tant que l'on n'est pas suffisamment enfonce dans le trou;
    tantque distance(robot,trou)≥12.0 faire
        fmx:=(fx+fx+fx)/3; fmy:=(fy+fy+fy)/3;
        v1:=vect(fmx,fmy,0.0);
        lier trou a robot;
        deplacer robot de translat(v1,0.5);
        delier trou de robot;
        deplacer robot a trou jusqu'à (fz+fz+fz)≥30.0;
    finfaire;
c0 fin de l'insertion;
    ouvrir pince;
    deplacer robot de translat(vz,100.0);
    retour;
    fin;
```

A.1.2 Insertion décrite de manière géométrique

L'exemple suivant décrit le même assemblage en utilisant le langage cible du système SHARP. La liste des commandes est extraite de [Lau87].

```
(realiser-s <insertion> (garde <chanfrein>))
(while (not <insertion>)
       (if <chanfrein> (corriger <aligne-axe>
                                 (suivant (F_xF_y 0))
                                 (amplitude I/2))))
```

où <insertion> représente la relation à réaliser, <chanfrein> est un prédicat indiquant s'il y a contact avec le chanfrein et <aligne-axe> est une relation représentant l'alignement des axes du goujon et du chanfrein. Le paramètre I représente l'erreur maximum associée à la relation d'alignement des axes.

A.2 Traces graphiques d'une résolution numérique

Les trois exemples suivants représentant une trace graphique de la résolution du système :

$$\left\{\begin{array}{rcl} x' &=& -2y^2\sin(z) - 2xy\cos(z) \\ y' &=& 2xy\sin(z) + 2x^2\cos(z) \\ z' &=& 2y\sin(z) + 2x\cos(z) \end{array}\right. \quad avec \quad \left|\begin{array}{rcl} x_0 &=& 1 \\ y_0 &=& 0 \\ z_0 &=& 0 \end{array}\right.$$

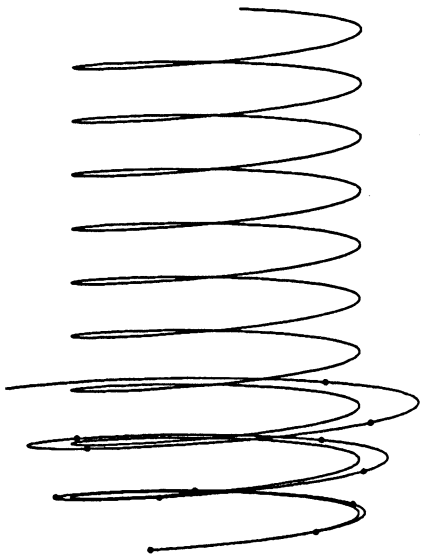
La solution théorique de ce système est une hélice d'équation :

$$\left\{\begin{array}{rcl} x &=& \cos(t) \\ y &=& \sin(t) \\ z &=& t \end{array}\right.$$

A.2.1 Utilisation de la formule d'Euler

Soit à résoudre : $Y' = f(X,Y)$ avec comme condition initiale $Y(X_0) = Y_0$.

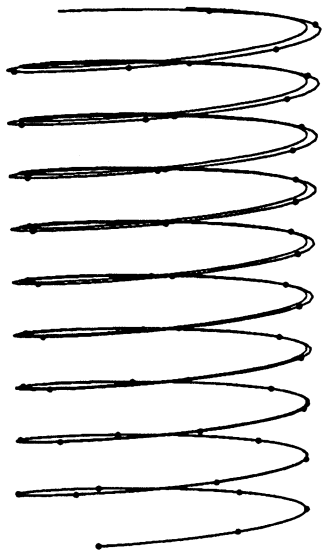
$$formule\ d'Euler : \quad Y_{i+1} = Y_i + h \cdot f(X_i, Y_i)$$



Dans l'exemple ci-contre, la résolution est effectuée avec un pas $h = 0.01$. Les marques • figurant sur la courbe solution sont espacées de 30 fois la valeur du pas d'itération.
Comme on peut le constater, les valeurs trouvées divergent très vite de la solution théorique.

A.2.2 Utilisation de la formule de Runge-Kutta d'ordre 2

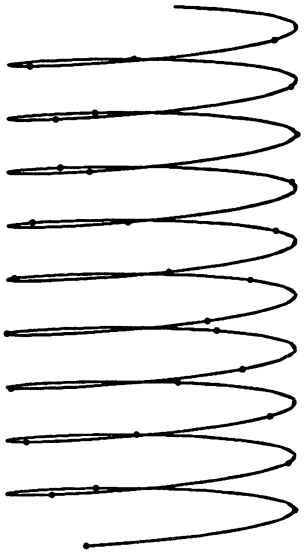
$$
\begin{aligned}
\textit{formule :} \qquad k_1 &= h \cdot f(X, Y) \\
k_2 &= h \cdot f(X + \tfrac{2h}{3}, Y + \tfrac{2k_1}{3}) \\
Y_{n+1} &= Y_n + \tfrac{1}{4}(k_1 + 3k_2)
\end{aligned}
$$



Le pas de résolution est de 0.05 et les marques sont placées tous les 5 pas d'itération. La convergence est bien meilleure mais peut encore être améliorée.

A.2.3 Utilisation de la formule de Runge-Kutta d'ordre 4

$$
\begin{aligned}
\textit{formule :} \qquad k_1 &= h \cdot f(X_n, Y_n) \\
k_2 &= h \cdot f(X_n + \tfrac{h}{3}, Y_n + \tfrac{k_1}{3}) \\
k_3 &= h \cdot f(X_n + \tfrac{2h}{3}, Y_n - \tfrac{k_1}{3} + k_2) \\
k_4 &= h \cdot f(X_n + h, Y_n + k_1 - k_2 + k_3) \\
Y_{n+1} &= Y_n + \tfrac{1}{8}(k_1 + 3k_2 + 3k_3 + k_4)
\end{aligned}
$$



Le pas de calcul est maintenant de 0.1.
La solution trouvée est meilleure tout
en réclamant moins d'étapes de calcul.

A.3 Syntaxe des instructions de notre langage cible

1. (Saisir <situation>)
2. (Lacher <situation>)
3. (Realiser-S <situation> Via <chemin>)
4. (Realiser-C <Contact> Suivant <d> [Maintenir <Contact>])
5. (Supprimer-C <Contact> Suivant <d> [Maintenir <Contact>])
6. (Corriger <situation> Suivant <d> Amplitude <a>)

<situation> ::= <position articulaire> // <transformation> // <relation géométrique>

<position articulaire> ::= $(\theta_1 \ldots \theta_n)$; $\theta_i \in \mathcal{R}$ et n = nombre de degrés de liberté du robot.

<transformation> ::= (T,Q) ; T $\in \mathcal{R}^3$ est une translation et Q $\in \mathcal{R}^4$ est un quaternion représentant une rotation.

<relation géométrique> ::= (<attribut géométrique><entité géométrique> <entité géométrique>)

<attribut géométrique> ::= aligné, coplanaire, contact

<entité géométrique> ::= <entité géométrique réelle> // axe, repère

<entité géométrique réelle> ::= nom de face // nom d'arête // nom de point

<contact> ::= (contact <entité géométrique réelle><entité géométrique réelle>)

<chemin> ::= { <situation> }$^{+}$

<d> ::= (V_x, V_y, V_z) ; vecteur orienté spécifiant
soit une direction de translation soit un sens de rotation.

A.4 Exemple de manipulation planifiée par le système PAMELA

```
(Realiser-S (R-robot sur R0)
     (via (C0 C1 C2 ...Cn)))

(Saisir (R-robot sur R-prise0))

(Realiser-S (R-robot sur R1))

(Realiser-C (point-P1 sur face-B1)
     (Suivant (translation :vect vecteur-V1)))

(Realiser-C (arete-A1 sur face-B1)
     (Suivant (rotation :axe (make-axe :pt point-P1
                                       :vect vecteur-V2)))
     (Maintenir (point-P1 sur face-B1)))

(Realiser-C (face-F1 sur face-B1)
     (Suivant (rotation :axe (make-axe :pt point-P1
                                       :vect arete-A1)))
     (Maintenir (arete-A1 sur face-B1)))

(Realiser-C (arete-A2 sur face-B2)
     (Suivant (translation :vect vecteur-V3))
     (Maintenir (face-F1 sur face-B1)))
```

(**Realiser-C** (face-F2 **sur** face-B2)
(**Suivant** (rotation :axe (make-axe :pt point-P2
:vect arete-A2)))
(**Maintenir** (face-F1 **sur** face-B1)(arete-A2 **sur** face-B2)))

(**Realiser-C** (point-P3 **sur** face-B3)
(**Suivant** (translation :vect arete-A1))
(**Maintenir** (face-F1 **sur** face-B1)(face-F2 **sur** face-B2)))

(**Lacher** (R-robot **sur** R3))

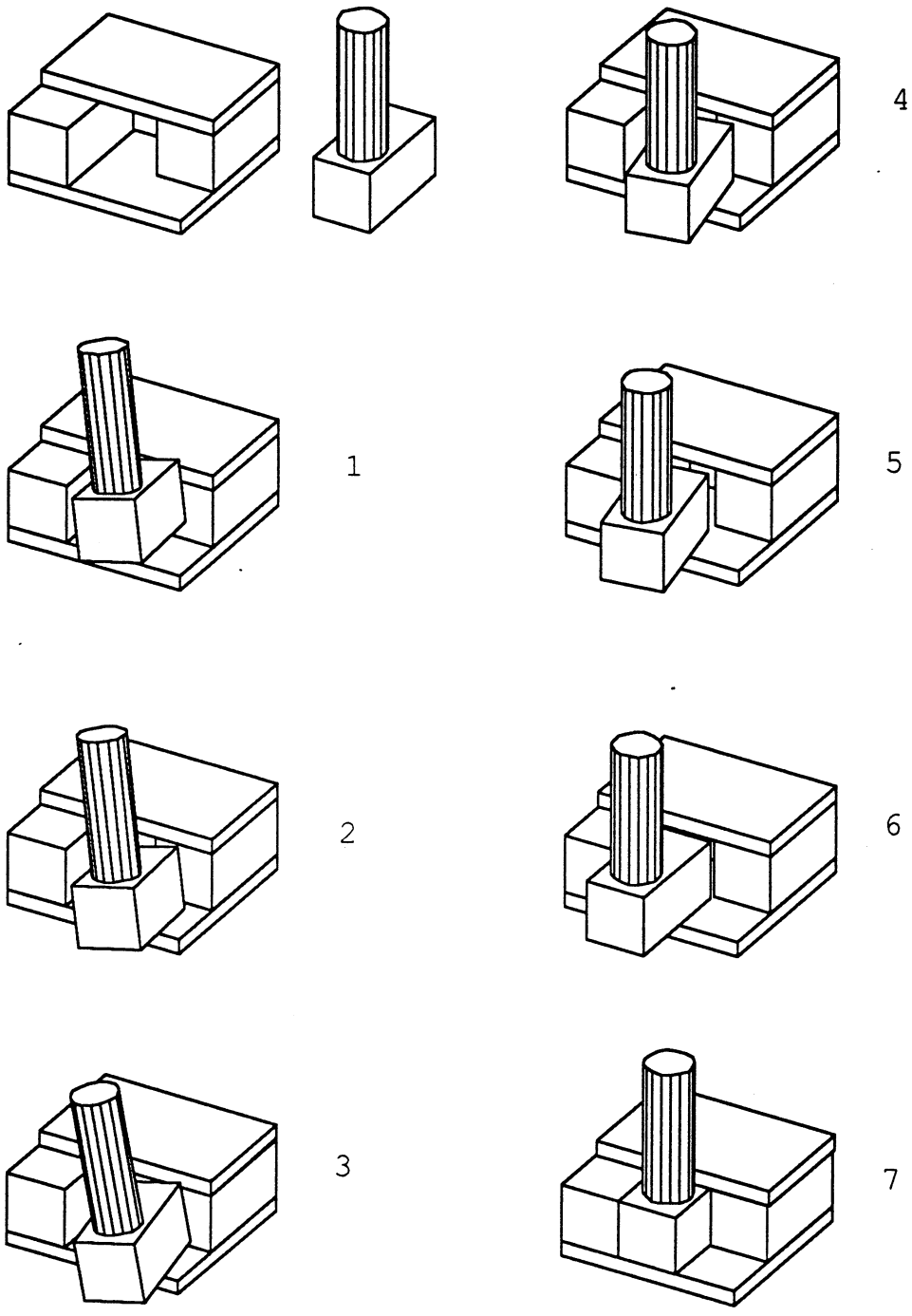


Figure A.2: Etapes du montage d'une pièce en forme de L

Résumé

L'utilisation d'un robot pour effectuer un assemblage est un travail difficile en raison de la complexité du monde réel où il évolue. La programmation automatique permet de simplifier la mise en œuvre des robots en synthétisant des programmes de manipulation à partir de descriptions symboliques de haut niveau des tâches à exécuter. Ceci nécessite de résoudre de nombreux problèmes de nature géométrique et physique. Ceux abordés dans ce mémoire concernent la **planification des mouvements fins d'assemblage**, c'est à dire les petits mouvements capables de mener à bien un assemblage en présence d'incertitudes géométriques.

Après avoir montré les limitations d'une approche purement numérique, nous proposons une méthode basée sur le raisonnement géométrique. Nous développons pour cela un modèle permettant d'effectuer facilement les opérations propre au raisonnement géométrique. Nous présentons ensuite notre système qui permet d'engendrer automatiquement des stratégies de mouvements fins. Une description de l'implantation réalisée est donnée dans ce mémoire. Elle est complétée par la présentation d'une tâche d'assemblage planifiée par le système et exécutée par le robot.

MOTS CLÉS : Robotique, Intelligence Artificielle, Modélisation géométrique, Raisonnement géométrique, Programmation automatique des robots, Planification de mouvements, Mouvements fins d'assemblage.