Planification de trajectoires pour un robot manipulateur

Michel Pasquier

THESE

présentée par

MICHEL PASQUIER

pour obtenir le titre de **DOCTEUR**

de L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Spécialité : INFORMATIQUE

0000

PLANIFICATION DE TRAJECTOIRES POUR UN ROBOT MANIPULATEUR

0000

Date de la soutenance : 25 Janvier 1989

Composition du jury :	M.	Jean FONLUPT	Président
	MM.	Philippe COIFFET	Rapporteurs
		Alain LIEGEOIS	
		Rachid ALAMI	
		Christian LAUGIER	

Planification de Trajectoires pour un Robot Manipulateur

Michel PASQUIER
LIFIA - IMAG
46 Avenue Félix Viallet
38031 Grenoble
pasquier@lifia.imag.fr

25 Janvier 1989

Résumé

Cette thèse traite du problème fondamental que constitue la planification de trajectoires de robots manipulateurs. Une première partie précise le contexte robotique de notre travail et présente le système général de programmation automatique développé au LIFIA. L'importance de la représentation des connaissances nécessaires aux raisonnements géométriques particuliers à la planification de déplacements est ensuite analysée. Les méthodes de modélisation et les concepts de représentation que nous avons mis en œuvre sont présentés. Une deuxième partie traite de la résolution du problème de planification de trajectoires pour une structure articulée. Une méthode de planification globale par construction de l'espace des configurations est détaillée, ainsi qu'une méthode de replanification locale par application de champs de potentiels et, enfin, une méthode hybride réalisant la synthèse de ces approches complémentaires, pour lesquelles sont décrits algorithmes, résultats d'expérimentation et futurs développements.

Mots-clés : Robotique, Programmation Automatique des Robots, Modélisation Géométrique, Représentation, Géométrie Algorithmique, Raisonnement Géométrique, Planification de Trajectoires, Evitement d'obstacles.

Abstract

This dissertation deals with the drastic problem of planning collision free paths for manipulator robots. In a first part, we present the robotic context of our work and the general system for automatic robot programming developped at the LIFIA laboratory. Then we analyse the needs in geometric modelling for the reasonning techniques particular to the motion planning problem, and detail the miscellaneous representations we have implemented. In a second part, we present a path planning system for a robot arm. We first describe a global method, aimed at planning a trajectory in a completely mapped configuration space, then a local "replanning" method using potential fields. Next we describe an hybrid method which realizes a synthesis of these complementary approaches, giving algorithms, experimental results and improvements of our approach.

Keywords : Robotics, Automatic Robot Programming, Geometric Modelling, Representation, Computational Geometry, Geometric Reasonning, Path Planning, Obstacle Avoidance.

Remerciements

Ainsi qu'il est d'usage, nous tenons à exprimer ici nos plus vifs remerciements à ceux qui, peu ou prou, ont contribué à la réalisation du présent ouvrage ou des travaux qu'il expose. A nos "juges" tout d'abord :

- Monsieur *Jean Fonlupt*, Professeur à l'INPG et Directeur du laboratoire Artemis, qui nous a fait l'honneur de présider le jury;
- Monsieur *Philippe Coiffet*, Professeur à l'INSTN et Directeur de Recherche au CNRS, ainsi que
- Monsieur *Alain Liégeois*, Professeur à l'USTL, qui ont tous deux accepté d'évaluer le travail présenté dans ce mémoire puis d'y apporter leur caution;
- Monsieur *Rachid Alami*, Chargé de Recherche au CNRS, qui a également bien voulu porter un œil critique sur ce rapport;
- Monsieur *Christian Laugier*, Chargé de Recherche à l'INRIA et Responsable de l'équipe robotique du LIFIA, qui nous a permis de découvrir le monde de la "robotique intelligente" et est à l'origine de nos recherches dans le domaine, et dont l'aide maintes fois répétée a grandement contribué à mener à terme ce dur labeur.

A nos "mécènes" ensuite, l'ADI, la DRET et l'ADR, qui ont successivement financé ces travaux.

A tous ceux enfin, amis et amies, collègues et relations, dont la liste pléthorique ne saurait être énumérée ici, pour leur contribution à titres divers. *Intelligenti pauca.* (A ceux qui savent, il n'est besoin de mots.)

Le syndrome du rédacteur

Ou l'application d'élémentaires règles psychologiques à la rédaction d'un mémoire de thèse.

1. Si une erreur peut se produire, elle se produit. *(Murphy)*
2. On n'a jamais le temps d'écrire correctement un rapport de thèse, mais on l'écrit toujours. *(Meskimen)*
3. Quelque soit la date à laquelle commence la rédaction d'un rapport, elle ne finit jamais dans les délais. *(Hofstadter)*
4. Une erreur typographique apparait toujours là où elle est susceptible de faire le plus de dommage. *(Silverman)*
5. La qualité d'un texte est inversement proportionnelle au nombre de personnes qui ont contribué à sa conception. *(Watson)*
6. Lorsque l'on hésite entre deux orthographes, on finir toujours par choisir la mauvaise. *(DeVries)*
7. Une fois un rapport terminé, toute tentative espérant l'améliorer ne fera que le rendre pire. *(Finagle)*
8. Tous les problèmes faciles ont été résolus. *(Kinkler)*
9. Celui qui sourit en lisant un rapport y a certainement trouvé une erreur. *(Jones)*
10. Chaque solution apporte de nouveaux problèmes. *(O'Toole)*
11. Plus une erreur de rédaction est grave, plus on la découvre tard.
    Corollaire: si c'est une erreur capitale, on ne la découvre jamais. *(Murphy)*
12. La quantité de travail restant à fournir est proportionnelle à celle du travail déjà effectué. *(Vail)*
13. Rien n'est jamais assez bon pour qu'il n'y ait pas quelqu'un, quelque part, qui le trouve mauvais. *(Pohl)*
14. Il suffit de chercher assez longtemps pour prouver la justesse d'une théorie. *(Murphy)*
15. Il n'existe pas de réponses : seulement des références croisées. *(Weiner)*
16. Dans tout ensemble de données, le chiffre qui apparait comme le plus vraisemblable est celui qui est faux. Corollaire: si tous les chiffres semblent corrects, tout est faux. *(Finagle)*
17. Si les faits vont à l'encontre de la théorie, ils doivent être rejetés comme expériences douteuses. *(Maier)*
18. Ce qui est faux parait toujours le plus vraisemblable. *(Scott)*
19. Si vous ne pouvez convaincre, semez le doute et la confusion. *(Truman)*
20. Il y a toujours une autre erreur. *(Lubarsky)*

Sommaire

Introduction

La planification de trajectoires

Quelque soit son type - robot mobile ou bras manipulateur - ou son environnement de travail, le robot *doit* se déplacer. Pour une telle machine, déplacement est synonyme d'action, et qui dit déplacement dit trajectoire. Voilà pourquoi la planification de trajectoires est sans doute un des problèmes fondamentaux de la robotique.

Le terme même de trajectoire mérite d'être précisé : on peut en effet distinguer au moins trois niveaux d'acception de ce mot. Un premier niveau - très général - assimile tous les mouvements d'un robot à des trajectoires, sans distinction de nature (amplitude, vitesse etc) ou d'objectif (mouvements de contact, d'évitement etc). Pour des raisons de complexité et d'efficacité, le vaste problème de planification qui en découle est généralement décomposé d'un point de vue fonctionnel en sous-problèmes autorisant ainsi le développement de planificateurs dédiés et d'algorithmes spécialisés. Enfin, au niveau - le plus bas - de la commande des actionneurs (des moteurs du robot), le terme désigne le déplacement réellement exécuté entre deux positions de consigne : on parle alors plutôt de génération de trajectoire.

Nous traitons ici le problème de la *planification de trajectoires de transfert.* Une telle trajectoire est définie de manière purement géométrique comme un chemin de grande ou moyenne amplitude, spécifié sans ambiguité par un ensem-

ble discret de paramètres du système. Le déplacement du robot le long de ce chemin, intégrant cette fois les aspects cinématiques et dynamiques, est laissé au niveau de la commande. Sur le plan fonctionnel, ce chemin doit être planifié dans un espace plus ou moins encombré de telle sorte qu'il relie deux positions particulières de l'espace en garantissant l'absence de collision le long du parcours. Nous excluons ainsi les mouvements de précision devant être effectués au contact des objets afin de les saisir, les manipuler et les assembler.

Le contexte de notre travail est celui de la programmation automatique de tâches d'assemblage, dont le but est de produire un programme de commandes de robot à partir d'une description de haut niveau d'une tâche robotique. Dans le cadre du système SHARP, développé au LIFIA en tant que support expérimental des recherches dans le domaine, la complexité du problème général est réduite par l'utilisation d'une structure modulaire dont notre système constitue l'un des planificateurs. Le problème de la planification d'une trajectoire doit alors être résolu hors-ligne à partir d'un modèle géométrique - supposé exact - du robot, de son univers de travail et des positions initiale et finale contraintes par les autres sous-systèmes.

Nous nous intéressons donc particulièrement au déplacement de robots manipulateurs dans un univers a priori connu et peu évolutif. Le problème peut alors s'énoncer avec concision de la façon suivante :

> *Déterminer, à partir d'une description géométrique du robot et de son environnement, une trajectoire exempte de collision entre deux positions spécifiées.*

Il nous semble indispensable, pour la compréhension de certaines parties du texte, de préciser dès à présent quelques notions qui seront formalisées plus loin. Ainsi une *configuration* (ou vecteur de configuration) du robot est un ensemble de paramètres de cardinalité minimale permettant de spécifier la position de tous ses éléments. L'ensemble de ces vecteurs est dit *espace des configurations* et le sous-espace de ces configurations pour lesquelles il ne survient nulle collision est dit *espace libre*. Aux positions initiale et terminale demandées correspondent des configurations particulières du robot, et une *trajectoire* est alors définie comme une courbe de l'espace libre joignant ces configurations.

Difficulté du problème

La raison principale de la difficulté du problème de planification de trajectoires semble être que *les meilleures représentations du robot manipulateur et les meilleures des obstacles ne se situent pas dans le même espace.*

En l'occurence, il s'agit respectivement de l'espace des configurations du robot et de l'espace cartésien.

Ainsi l'espace cartésien est parfait pour représenter les objets de l'environnement : leur structure géométrique et les propriétés afférentes y sont parfaitement connues. De même la tâche à réaliser, décrite en termes de positions des objets constituant l'assemblage, s'y trouve naturellement exprimée. Par contre la maîtrise de la géométrie du manipulateur y est très délicate.

En effet, la manipulation des objets implique la connaissance de la position de l'outil terminal du robot (en général une pince). Or, d'une part, il existe de multiples configurations permettant à cet outil terminal d'atteindre une position donnée, entraînant ainsi des problèmes d'indétermination sur l'état du robot et, d'autre part et surtout, on ne peut se contenter pour déplacer le manipulateur de planifier la trajectoire de son préhenseur : il faut encore contrôler la position du reste de la structure articulée relativement aux obstacles. Et pour reprendre l'exemple de P.G.Ránky [1-RH85], il suffit pour s'en convaincre de se rappeler la difficulté que représente pour un humain l'accès à un buffet surpeuplé.

L'espace des configurations est en revanche mieux adapté pour représenter la position de l'ensemble des éléments du manipulateur que l'espace cartésien dans lequel évolue le robot. Un inconvénient majeur toutefois est que les contraintes d'encombrement spatial induites par les obstacles se traduisent sous la forme de régions de configurations interdites dont la topologie est très complexe et bien souvent impossible à caractériser de manière exacte. La détermination de l'espace des configurations est par conséquent le problème clef qu'essaient de résoudre les méthodes générales de planification de trajectoires.

L'approche choisie

Le développement de la robotique passe impérativement par le perfectionnement des diverses tâches de planification, en particulier par l'établissement

d'algorithmes efficaces pour le calcul de trajectoires. L'axe principal de notre travail a précisément porté sur ce point.

Nous proposons tout d'abord une méthode de planification globale (par référence à la nature des données sur lesquelles repose la recherche de trajectoires) opérant en deux points :

- la construction d'une représentation approchée de l'espace libre d'un robot manipulateur à l'aide de techniques de raisonnement géométrique, à partir d'un modèle géométrique du robot et de son environnement.
- la planification d'une ou plusieurs trajectoires dans la représentation ainsi obtenue, avec prise en compte des contraintes imposées par la tâche.

L'originalité de la méthode réside avant tout dans le procédé récursif utilisé pour construire un modèle surcontraint de l'espace des configurations du robot. Ce procédé est fondé sur une discrétisation de l'espace articulaire, et sur un mécanisme de propagation de contraintes géométriques dérivé des techniques dites de "grossissement d'obstacles".

Cette approche, qui peut être qualifiée de *constructive*, est basée sur une connaissance globale de l'environnement. Elle cherche à analyser la topologie de l'espace atteignable par le robot afin de construire une représentation du sous-espace où ce dernier est libre de se mouvoir, permettant ainsi la recherche exhaustive d'une classe de trajectoires optimisant un critère donné. Très coûteuse sur le plan calculatoire, elle ne peut être raisonnablement utilisée pour un grand nombre de degrés de liberté.

Nous étudions ensuite une méthode de génération locale de trajectoires basée sur l'application heuristique de champs de potentiel fictifs, envisagée comme complément de la méthode globale. Cette seconde approche, que nous dirons *corrective*, vise à déterminer, pour une configuration donnée du robot et en l'absence d'information globale sur l'ensemble de l'univers, un déplacement sans collision le rapprochant du but à atteindre.

Nous présentons enfin une synthèse de ces deux approches complémentaires, conduisant très naturellement à l'utilisation d'une méthode hybride : une première phase permet alors de planifier à partir d'un environnement global simplifié un ensemble de trajectoires de référence pouvant dans un deuxième temps faire l'objet d'une modification locale.

Etant donnée l'importance que nous attachons à la nature des représentations disponibles, les capacités et les performances des algorithmes développés étant étroitement liées à celles-ci, nous présentons un modèle particulièrement adapté aux besoins de la robotique et plus spécialement au calcul de trajectoires.

Structure du mémoire

Le mémoire se scinde en deux parties présentant respectivement une définition et une analyse du problème puis les méthodes de résolution que nous avons développées.

Dans la première partie, nous présentons le contexte robotique qui est le cadre de notre recherche, et plus particulièrement celui de la programmation automatique des robots d'assemblage. Nous exposons une idée de ce que doit être un système de programmation automatique et, après un bref aperçu des précurseurs du genre, nous présentons le système développé au LIFIA, dans lequel s'intègre notre travail. Nous décrivons alors la nature du problème de planification de trajectoires, sa complexité et les différentes approches existant à ce jour. Nous abordons ensuite le problème de la modélisation en tant que représentation des informations nécessaires aux raisonnements géométriques mis en œuvre, détaillant les diverses représentations adoptées et utilisées, en insistant sur celles développées spécifiquement pour notre problème.

La deuxième partie, plus technique, expose une méthode générale de raisonnement géométrique visant à la construction d'une représentation de l'espace des configurations adaptée à la recherche de trajectoires de transfert. Nous traitons ensuite de la recherche de trajectoires dans cette représentation. Puis nous développons l'étude d'une méthode locale et son utilisation complémentaire dans le cadre de la précédente méthode, définissant ainsi une méthode hybride permettant la planification de trajectoires de référence localement modifiables. Nous présentons enfin le système développé, les résultats obtenus et les perspectives futures.

Partie I

Programmation automatique, modélisation géométrique et planification de trajectoires : problématique et outils nécessaires

Chapitre 1

Principes généraux de la programmation des robots

1.1 Productique et robotique

1.1.1 La robotique : mythe et réalité

La notion de robot, à l'origine basée sur un concept anthropomorphique, a toujours conservé un aspect magique, rituel, dans l'esprit humain. Pourtant, à mesure que les robots apparaissent dans notre quotidien, ils perdent leur identité morphologique à mesure que s'accroit leur autonomie, certains diront leur "intelligence".

Tout d'abord ce furent des mécanismes ingénieux, à l'eau ou au mercure, qui permirent l'existence des premiers robots. Puis l'horlogerie accorda aux automates un réalisme étonnant qui force encore l'admiration. Cette philosophie de l'automate imitateur, réplique de l'être humain, a persisté jusqu'au début du XXe siècle, renforçant l'image anthropomorphique et la fonction de serviteur, d'esclave du robot. Son nom même est d'ailleurs issu de cette idéologie : celle du travail.

Bénéficiant des progrès fulgurants de la technique, les robots sont alors devenus utiles. Ils sont nés, comme la cybernétique, d'une modélisation de l'électronique et de l'électromécanique, et sont devenus esclaves, non de l'homme, mais de la production, maître-mot de notre siècle. Le robot en a perdu toute velléité de forme humaine : le besoin créant la forme, il n'en a plus. Ou plutôt il a celle de ses composants, assemblage de mécanismes et de capteurs destinés à remplir une fonction bien précise. Il est devenu fonctionnel, par construction.

1.1.2 Les robots et l'industrie

L'apparition de l'informatique dans l'industrie sera peut être considérée par les générations futures comme la révolution industrielle du XXe siècle. Son application à la gestion d'ateliers a motivé l'apparition d'un domaine hybride : la productique. Celle-ci, définie comme l'ensemble des techniques d'automatisation de production (conception, fabrication et gestion de production), fait implicitement appel à l'informatique comme moyen de contrôle et à la robotique comme moyen d'exécution.

Typologie des robots

Les robots industriels peuvent être classés selon leur fonctionnalité : on distingue ainsi les robots d'assemblage, de montage de composants en surface, de chargement et déchargement, de manipulation d'outils, de projection (peinture, colle) de soudage (point ou arc), de découpage, de moulage, d'usinage, de traitement thermique...Beaucoup de ceux-ci sont des machines à la morphologie et aux outils spécialisés, au possibilités limitées. Aussi est-il sans doute préférable de classer les robots selon leur degré d'évolution.

Ainsi l'AFRI [1] distingue quatre classes de robots. Ceux des deux premières catégories (bras manipulateurs manuels et automatiques) ne sont capables d'actions ou de manipulations que selon des séquences figées. Le développement de l'informatique a permis l'apparition de machines capables de traiter des séquences variables, c'est-à-dire paramétrables. La classe suivante concerne donc les robots programmables par apprentissage ou par langage symbolique. Enfin vient la classe des robots dits "intelligents", c'est-à-dire des manipulateurs automatiques programmables capables d'analyser leur environnement et de réagir en conséquence.

Seule cette dernière classe nous intéresse, puisque le robot "cible" d'un système de programmation automatique doit à notre sens posséder des caractéristiques morphologiques et mécaniques, des qualités de polyvalence et d'autonomie les plus avancées eut égard aux possibilités technologiques actuelles. Vient alors le problème de les utiliser au mieux...

[1] Association Française pour la Robotique Industrielle

L'automatisation

Un atelier de production est composé de systèmes remplissant un certain nombre de fonctions automatisables, que l'on cherche à regrouper en *îlots d'automatisation* qui devront communiquer entre eux via un réseau local industriel. L'informatique permet d'interfacer dans une unité globale les différentes familles de machines automatiques - robots manipulateurs, moyens de transport (tapis, élévateurs), automatismes divers - et de gérer la synchronisation des tâches et des mouvements, depuis la conception jusqu'à la gestion de production. L'architecture du réseau comporte donc un aspect fonctionnel macroscopique (planification et gestion du travail) et un aspect opérationnel localisé (mécanique et robotique).

Dans ce contexte, le robot apporte des capacités inexistantes sur les machines spécialisées. En effet, celles-ci, aux performances encore supérieures, sont astreintes à évoluer dans un univers connu et invariant, totalement maîtrisé. Au contraire, les robots, par leur polyvalence et leur adaptabilité, peuvent être utilisés dans un environnement évolutif sur lequel l'information détenue reste incertaine. Cette propriété implique cependant une certaine sophistication des programmes de commande, laquelle fait l'objet de l'essentiel des recherches actuelles en robotique.

Avantages fonctionnels du robot :

Un des aspects particulièrement intéressants du robot est sa capacité à fonctionner en milieu hostile (atmosphère polluée ou radioactive, espace, milieu subaquatique etc). Mais même dans des tâches plus classiques, ses qualités de fiabilité, de répétabilité et de précision en font une machine extrêmement efficace. Le robot place ses "yeux" au meilleur endroit, sa pince intégre l'outil - parfaite adéquation à la tâche proposée - et ses multiples degrés de liberté l'autorisent à atteindre les positions les plus complexes et travailler dans un volume réduit. Son mode de programmation, ainsi que sa sensibilité à l'environnement lui permettent d'aborder une gamme très étendue de problèmes.

Avantages économiques du robot :

La robotisation est donc le moyen de modernisation par excellence de l'appareil de production, ajoutant ses avantages propres aux bénéfices de l'automatisation. Sans prétendre à l'exhaustivité, on peut mentionner : augmentation des rendements et donc diminution des coûts, qualité et régularité des produits finis,

réduction et maîtrise des délais de fabrication, conditions de travail améliorées, économie de main d'œuvre, auxquels s'ajoutent - flexibilité et polyvalence obligent - diminution des en-cours, minimalisation des stocks de produits semi-finis et même réduction des délais de commande-livraison, enfin résistance à l'obsolescence. Mais l'investissement reste encore lourd, tant matériel que logiciel, et vient s'y ajouter la nécessité de former du personnel et de réorganiser l'unité de production.

Le robot de troisième génération a donc pour lui la polyvalence, et l'on s'efforce maintenant de lui donner l'autonomie. Il semble d'ores et déjà être la véritable alternative au travail manuel, et l'exemple des tâches d'assemblage de petites séries est significatif à cet égard. Le robot cesse alors d'imiter l'homme et, du point de vue de la productivité, il le dépasse même.

1.1.3 Les robots et les roboticiens

L'utilisation industrielle des robots de troisième génération reste rare. En effet, s'ils offrent potentiellement une plus grande adaptabilité et une plus grande autonomie, leur emploi pose un problème au niveau de la commande. Ainsi, il est une phase délicate mais nécessaire dans l'utilisation d'un robot : la spécification des tâches qu'il doit accomplir, phase de *programmation* dépendant de la nature de ces tâches.

Actuellement, ce travail est du ressort d'un spécialiste capable de traduire les spécifications de la tâche en une séquence de commandes, ce qui est long, pénible, et exige du programmeur une totale connaissance des conditions d'exécution du programme. Ainsi la spécification d'une trajectoire se traduit par la donnée de points de consigne déterminés "à la main" par l'opérateur. De plus, toute modification de la tâche implique une modification du programme souvent équivalente à la spécification d'une nouvelle tâche. Enfin la mise au point de ces programmes est délicate et nécessite l'immobilisation du robot devenu alors improductif. La lourdeur et la difficulté du travail de programmation va donc à l'encontre des principes de flexibilité et de performance évoqués précédemment.

On constate en pratique que l'univers d'un robot est souvent vaste, variable et peu structuré. L'hypothèse de la connaissance a priori de cet environnement étant peu réaliste, apparaît la nécessité pour le robot de pouvoir lui-même acquérir l'information manquante afin d'adapter ses actions en conséquence. L'étape suivante du raisonnement consiste tout naturellement à lui faire planifier

l'intégralité de ses actions.

1.1.4 Vers des robots "intelligents"

Par *robot "intelligent"*, nous entendons un robot capable de connaissance, de perception, de décision et d'action.

Il doit par conséquent être muni d'une base de connaissance sur l'univers, fournie par un agent extérieur (par exemple un opérateur humain), et doté de capacités de perception (capteurs divers) lui permettant de faire évoluer cette connaissance. Il doit également être doté de capacités décisionnelles, c'est-à-dire d'un ensemble de méthodes de raisonnement (algorithmes, règles, heuristiques) lui permettant de planifier et modifier son comportement sans intervention extérieure (et en particulier humaine). Enfin ses capacités mécaniques doivent lui permettre d'accomplir les actions planifiées.

Dans cette optique, l'intelligence se ramène à une forme suffisamment évoluée de la représentation et du traitement de l'information. Ainsi la communication entre le robot (système robotique + logiciel) et l'opérateur humain doit supporter un haut niveau d'abstraction; en particulier, la description des tâches doit pouvoir se limiter à la spécification de but à atteindre. L'exemple le plus classique de commande de haut niveau est sans doute la forme "Pick and Place", soit "Prendre *objet* et placer à *position*".

Déjà capable de locomotion et de perception, le robot doit donc désormais pouvoir interpréter l'information acquise et engendrer lui-même un plan d'actions en fonction des contraintes extérieures. Il doit pouvoir détecter la présence d'une situation "familière" et agir en tenant compte de résultats antérieurs (son expérience), et élaborer de nouvelles stratégies lorsque la situation rencontrée est inédite. L'autonomie est à ce prix...

1.2 Les robots et l'assemblage

1.2.1 Description d'une cellule robotique

La cellule d'assemblage

La cellule flexible d'assemblage (CFA) est considérée comme le prototype idéal de robotique de travail à poste fixe. Les thèmes abordés couvrent la totalité des problèmes actuels de la robotique : commande adaptative asservie

à la tâche, génération automatique de trajectoires (au sens large), contrôle d'exécution. Le but poursuivi est de développer les capacités de décision et augmenter l'autonomie du robot. De plus on cherche à accroître ses performances en flexibilité, en vitesse et en précision.

Une cellule robotique se compose d'un *environnement* comportant des *objets* à manipuler et assembler, d'autres à éviter ou utiliser comme support, des *robots* permettant d'agir et des *capteurs* permettant d'observer. Evitant l'assimilation de la cellule à l'atelier dans sa totalité, nous parlerons ici de cellule robotique au sens de cellule minimale composée d'un seul robot et de son espace de travail.

L'ensemble des cellules robotiques doit être géré par un *système superviseur* devant planifier et coordonner les actions, puis contrôler leur bonne exécution. La planification repose sur l'utilisation d'informations internes (modèle géométrique des objets, modèle cinématique du robot etc) et le contrôle à l'aide d'informations externes (capteurs de toutes sortes). Bien sûr, les deux opérations ne sauraient être indépendantes...

Les robots d'assemblage

Les robots utilisés en assemblage sont des manipulateurs articulés à base fixe, possédant un nombre de degrés de liberté suffisant pour les tâches qui leur sont confiées. Leur motorisation et le contrôle des asservissements qui permettent leurs déplacements font l'objet de recherches en mécanique et en automatique.

Les capteurs utilisés sont essentiellement de deux types, bien que l'on puisse considérer une troisième classe, celle des capteurs de sécurité. Ainsi, les capteurs *proprioceptifs* permettent de décrire l'état interne du robot par les grandeurs qu'ils mesurent. Ce sont quasi exclusivement des capteurs de position, de vitesse et, plus rarement, d'accélération. Les capteurs *extéroceptifs* assurent une fonction d'observation de l'environnement du robot. On distingue les capteurs tactiles (capteur de force, main à retour d'effort) dont la caractéristique est d'opérer en contact avec un objet, et les capteurs visuels, opérant à distance par rayonnement, dits actifs lorsque la source rayonnante est dans le capteur, et passifs sinon. Le rayonnement peut être thermique, sonore (onde de pression, ultrasons), ou électromagnétique (ondes radio, infra-rouges, visible - stéréoscopie, télémétrie laser ou plan de lumière).

Lorsque la tâche est parfaitement définie, elle peut être décomposée en grandeurs physiques que les seuls capteurs proprioceptifs suffisent à mesurer. La

seule action possible du robot consiste en effet à déplacer les éléments de sa structure dans l'espace, selon une séquence de positions indiquées en consigne. Mais lorsque la tâche est partiellement indéfinie (en particulier à cause des incertitudes sur l'environnement), elle doit être décomposée de manière fonctionnelle. L'utilisation de capteurs extéroceptifs doit alors permettre de spécifier l'information manquante afin de définir complètement la tâche. Ce type de capteurs est donc plus complexe et comprend en général, en plus du capteur proprement dit (mesurant une grandeur physique), un système de traitement du signal fourni (electronique et logiciel). Tel est le cas d'un capteur vision, où l'image (la mesure) doit être interprétée afin de fournir au système l'information de forme ou de position dont il a besoin.

1.2.2 Définition d'une tâche

La tâche d'assemblage doit être vue au plus haut niveau d'abstraction possible : seul importe alors le but à réaliser indépendamment des moyens employés pour y parvenir.

Le *but* est défini comme un état de l'univers en général distinct de l'état courant. Un *état* de l'univers est une vue instantanée de celui-ci prise par un observateur extérieur, et donc statique par nature. La description d'un but peut alors se ramener à celle de l'état souhaité, c'est-à-dire une description géométrique de l'univers résumant l'objectif du système robotique.

Le rôle du système superviseur (effectuant la planification globale au niveau du réseau productique) est de décomposer la tâche générale en tâches intermédiaires faisant intervenir certains outils dans un certain environnement. Chacune de ces tâches doit alors être résolue par la détermination d'une stratégie locale d'assemblage, effectuée au niveau de la cellule, c'est-à-dire du robot chargé de l'opération. La description fournie est alors dite de niveau tâche (cf 1.3.1).

Celle-ci doit ensuite être interprétée - et c'est précisément là le rôle d'un système de planification tel que celui que nous présentons - au niveau objet, c'est-à-dire décomposée en une séquence d'*opérations d'assemblage*, chacune définissant un objectif intermédiaire et des contraintes associées à sa réalisation.

L'approche adoptée dans le système SHARP fait reposer ces opérations sur le concept de mise en coïncidence d'entités géométriques. Le langage induit par ce type de programmation offre des constructions complètes permettant de décrire les mouvements du robot en termes de conditions d'arrêt et de con-

traintes d'exécution, à leur tour exprimées en termes de relations géométriques et de contacts. Une tâche est alors définie par la vérification de contraintes sur un ensemble de mesures du système. L'ensemble des séquences d'opérations exécutables qui vérifient ces contraintes est dit : ensemble des solutions de la tâche intermédiaire. Il existe une solution au problème général lorsque chacun de ces ensembles de solutions contient au moins un plan d'action local compatible avec le plan global.

Si le système superviseur est inexistant, c'est à l'opérateur humain de fournir une description de niveau tâche relative à chaque cellule de l'atelier robotique. Ceci nécessite un langage spécialement conçu dans ce but, à la sémantique simple et extrêmement réduite, que nous considérerons appartenir au langage naturel.

1.3 La programmation des robots

1.3.1 Langages et niveaux de programmation

Le but général de l'automatisation est de faire le lien entre les deux niveaux extrêmes de description de la tâche que sont d'une part une spécification abstraite de celle-ci, et d'autre part une séquence de mots d'état commandant les moteurs d'une machine ou d'un robot. Ce problème se décompose selon une hiérarchie de niveaux conceptuels, à laquelle correspond une hiérarchie de langage et de niveaux de programmation, et qui peut être abordée par l'une ou l'autre de ses extrémités.

Approche ascendante : la programmation explicite

Un robot est un système mécanique complexe. Faire évoluer son extémité terminale suivant une trajectoire précise implique la maîtrise de la coordination de tous les degrés de liberté en position, en vitesse et en accélération.

Pour un robot manipulateur constitué d'une chaîne cinématique à plusieurs degrés de liberté, la commande de base est la tension aux bornes de chacun des moteurs entraînant ses axes. En amont de ce vecteur de commande en tension, on trouve celui des couples moteurs, puis des couples articulaires et enfin le vecteur des variables articulaires ou variables généralisées, correspondant aux valeurs prises par les différents degrés de liberté. Une étude de l'espace de ces variables généralisées est faite au chapitre 5. Commander le robot consiste donc à spécifier un vecteur de variables généralisées, c'est-à-dire une séquence de configurations.

Ce mode de déplacement élémentaire du robot, par commande des moteurs et contrôle des asservissements, est dit de *niveau actionneur.*

Ce niveau de commande est impraticable directement par un opérateur en raison de la complexité des calculs qu'il nécessiterait de sa part. Aussi a-t-on construit des ensembles d'instructions permettant de commander plus simplement les actionneurs du robot. Il existe ainsi de nombreux langages de programmation, tous issus de ou calqués sur un langage informatique classique (procédural) augmenté de primitives spécialisées (gestion des déplacements, des acquisitions capteurs, des repères). Seuls quelques uns ont été entièrement développés et utilisés, tels AML [7-TSM82], LM [7-MM84] et VAL [7-SGS84]. Un exemple type de ce genre de primitives est l'instruction dite de "déplacement cartésien" du manipulateur, dont l'effet est de contraindre la trajectoire de son extrémité terminale à être rectiligne entre deux positions de consigne, contrôlant chacun des degrés de liberté dans ce but au lieu de les laisser varier proportionnellement au temps. De tels langages sont dits de manipulation ou de *niveau effecteur.*

Mais le niveau de ces langages est insuffisant pour décrire correctement une tâche d'assemblage, essentiellement à cause de la nécessité de prendre en compte les aléas du monde réel. En effet la position des objets et les déplacements du robot sont entachés d'erreurs de position qui rendent impossibles l'accomplissement d'une tâche donnée en termes de positions. La programmation d'une tâche d'assemblage doit donc ignorer les informations absolues (telles que les positions numériques des repères liés aux objets) pour ne considérer que les informations relatives (symboliques) qui constituent donc le *niveau objet*, où se situe d'ailleurs la programmation géométrique. Celle-ci peut être textuelle (LDGT) ou graphique (PGR [2-Les85]).

Enfin, la planification au niveau de l'atelier robotique complet, de niveau *tâche* ou *objectif*, peut tirer avantage de l'utilisation d'un langage plus proche du langage naturel (que nous noterons LN).

Approche descendante : la programmation implicite

Il semble raisonnable d'avancer que, hors toute considération technique quant à la réalisation du système correspondant, le langage le plus naturel soit de loin le meilleur que l'on puisse utiliser pour commander un robot ou un atelier robotique. D'autre part le niveau d'abstraction auquel est spécifié l'objectif permet de se suffire d'ordres très "généraux". On arrive à la limite à un niveau tel qu'une seule commande suffit de fait pour *tout* montage. Par exemple :

"Réaliser l'assemblage"

C'est ainsi la production de toute une gamme d'assemblage qui est planifiée, chaque assemblage étant à son tour décomposé en une séquence d'opérations de montage élémentaires, c'est-à-dire ne concernant que deux objets bien définis. Le système superviseur décidant alors qu'un seul robot suffit bien à la réalisation de ce montage, et lequel, le problème se trouve ramené à l'exécution de cette tâche élémentaire, qui peut être exprimée dans un langage simple. On aurait ainsi, pour l'exemple de montage présenté en figure 1.1, dans lequel la pièce de droite (dite en L) doit être insérée dans celle de gauche (dite de base B) :
"Monter la pièce en L sur la pièce B" ou plus simplement encore :

"Monter L sur B"

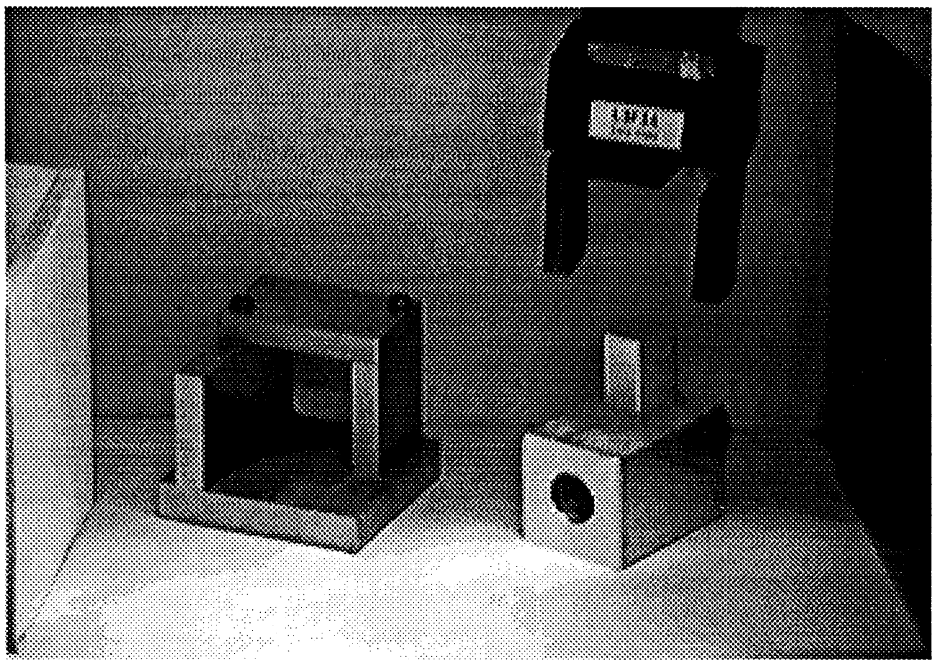

Figure 1.1: Exemple d'assemblage élémentaire.

L'ensemble d'instructions du niveau tâche doit alors être décomposé en une séquence d'instructions définie par la stratégie d'assemblage propre au manipulateur choisi. Celles-ci sont connues et ordonnées, mais seulement en fonction de leurs effets et non de leur modalité d'exécution. On obtient alors un plan intermédiaire du type "Prendre-et-placer", comportant une séquence d'instructions "générales" : "Saisir L ; Déplacer L ; Monter L sur B" .

Les contraintes spécifiées sont d'ordre purement géométrique et les instruc-

tions utilisées dans le plan final mettent en jeu les positions spatiales d'un objet relativement à un référentiel fixe ou aux autres objets de l'environnement. L'ensemble des instructions nécessaires constitue alors un langage de description géométrique de la tâche (LDGT) de niveau objet. Le rôle d'un système de programmation automatique est précisément d'effectuer le passage du niveau tâche au niveau objet. Dans l'exemple choisi, la séquence d'instructions obtenue est, dans un premier temps, le plan intermédiaire :

Saisir L ; Déplacer L ; Assembler L sur B ; qui est ensuite développé en :

```
(Réaliser-S (R-robot sur R0))                                   déplacement initial
(Saisir (R-robot sur R-prise0))                                 saisie de l'objet L

(Réaliser-S (R-robot sur R1) (via (ς0 ς1 ··· ςp)))              déplacement intermédiaire

(Réaliser-C (point-P1 sur face-B1)                              montage de L sur B
            (Suivant (translation :vect vecteur-V1)))
(Réaliser-C (arete-A1 sur face-B1)
            (Suivant (rotation :axe (make-axe :pt point-P1 :vect vecteur-V2)))
            (Maintenir (point-P1 sur face-B1)))
(Réaliser-C (face-F1 sur face-B1)
            (Suivant (rotation :axe (make-axe :pt point-P1 :vect arete-A1)))
            (Maintenir (arete-A1 sur face-B1)))
(Réaliser-C (arete-A2 sur face-B2)
            (Suivant (translation :vect vecteur-V3))
            (Maintenir (face-F1 sur face-B1)))
(Réaliser-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)))
(Réaliser-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))
```

L'étape suivante consiste à déterminer la paramétrisation des actions planifiées, c'est-à-dire à instancier le code ci-dessus. L'instruction Saisir exige la sélection d'une approche puis le choix d'une prise stable. De même le déplacement effectué par Réaliser-S nécessite la détermination d'une trajectoire, c'est-à-dire de la suite de positions amenant le robot de la position nominale fournie par l'instruction de saisie à la position demandée par la première instruction de montage... L'instanciation des différents paramètres du problèmes permet de construire un véritable programme de commande du robot, exprimé dans un

langage de niveau manipulation. Les langages de ce type les plus adaptés sont RAPT [7-PAB80] et LM-GEO [7-Maz83]. Ce dernier a été conçu comme une interface entre le langage LM et le modèle géométrique de l'univers. Bien que plus riches qu'un "simple" langage de commandes comme LM, leur capacité de description reste toutefois incomplète. C'est pourquoi le langage cible du système SHARP est LDGT et non un de ces langages.

Le code générique obtenu doit enfin être transformé en une séquence de vecteurs de commandes adressant directement les actionneurs du robot. C'est ainsi que notre programme LM sera compilé en LME, code exécutable pouvant être téléchargé sur l'armoire de commande du manipulateur.

Niveaux de :			Langage utilisé
Langage (informatique)	Commande (automatique)	Modèle cible du langage	
tâche / objectif	-	stratégie d'assemblage	LN
objet	objet	environnement	LDGT LM-GEO / PGR
effecteur	outil	géométrique et/ou cinématique	LM / LME
actionneur	moteurs	dynamique, transmissions et asservissements	CESAR & CLEOPATRE

Figure 1.2: Niveaux de commande et niveaux de langage.

Le tableau synoptique 1.2, inspiré de [1-Coi86] donne un condensé de la hiérarchie des niveaux de programmation que nous venons de décrire.

1.3.2 Techniques de programmation

La programmation par l'exemple

Dans ce contexte, un opérateur manipule le robot ou une réplique de celui-ci, un *pantin*, en effectuant les actions constituant la tâche à réaliser. Pendant cette phase de description, un calculateur mémorise la séquence des différentes positions afin de les reproduire ensuite. Cette technique de programmation, simple à mettre en œuvre, reste limitée à des tâches ponctuelles et à un environnement

invariant (soudure par points, peinture etc). D'autre part, la précision limitée de la méthode interdit son utilisation pour des tâches telles que l'assemblage. Enfin, la prise en compte de toute information complémentaire (issue de capteurs) passe obligatoirement par l'appréciation de ce même opérateur.

La programmation procédurale

Une des différences fondamentales entre un robot "évolué" et une machine "classique" est la présence sur le premier de tout un appareillage d'acquisition de l'information : les capteurs. Pour prendre en compte cette information, analogique ou numérique, il est nécessaire de disposer de commandes de calcul et de lecture de celle-ci, lesquelles viennent s'ajouter aux commandes usuelles du robot. Il suffit alors d'organiser et de séquencer ces commandes de manière procédurale pour obtenir un programme permettant d'exécuter fidèlement la tâche à accomplir. Les langages existants sont par voie de conséquence invariablement dérivés d'un langage algorithmique standard auquel ont été ajoutées des primitives spécifiques du contrôle des actionneurs et capteurs du robot.

Mais le principal inconvénient de la programmation procédurale est sa trop grande rigidité. Un programme de commandes est difficilement modifiable, et tout aussi diffilement décomposable en procédures élémentaires, indépendantes et susceptibles d'être réutilisées par ailleurs. Si la construction incrémentale d'un programme d'assemblage semble une méthode tout aussi peu adaptée que l'utilisation de squelettes de programmes, c'est que la connaissance modélisée n'est pas pertinente. Ces techniques sont pourtant celles utilisées, puisqu'à ce jour les seules utilisables de manière réaliste.

Une démarche généraliste pourrait amener à une approche par le haut de type "raisonnement symbolique", celle de la résolution de problèmes. Peu ou pas spécialisé, le système raisonne sur une base de faits à laquelle il applique divers opérateurs (correspondant aux actions possibles du robot) afin d'atteindre une situation de but décrite dans les mêmes termes.

De tels systèmes ont déjà vu le jour, tels STRIPS [6-FN71], mais sont encore trop "déconnectés" de la réalité robotique pour être pleinement utilisés. La programmation automatique ne peut être envisageable à ce niveau tant que la maîtrise du niveau objet n'est pas acquise.

1.3.3 La programmation automatique

Le problème qui se pose alors est celui de la conception, la réalisation et la mise au point des programmes de commande. La richesse du langage robotique induit une certaine difficulté à le maîtriser. Ainsi, l'accroissement en puissance impliquant l'accroissement en complexité, l'écriture d'un tel programme relève du spécialiste dont la justesse du raisonnement est largement mise à contribution.

La solution paraît simple, au moins à exprimer : elle consiste à *automatiser* le processus de programmation. La programmation explicite des tâches est confiée à un système informatique, le travail de l'opérateur de bornant à décrire la tâche de manière abstraite. Le principe consiste à construire de manière *automatique* le programme de commande du robot à partir d'une description de l'univers, issue par exemple d'une base de données CAO, et d'une description de la tâche dans les termes d'un langage évolué.

Chapitre 2

La Programmation Automatique des Robots : problèmes et approches

2.1 Le problème de la Programmation Automatique

Nous avons vu (1.1.2) que le robot est une machine particulière, puissante certes, mais restant un élément à intégrer à un système de production, un outil de l'automatisation parmi d'autres. L'intégration d'un système de programmation automatique au sein d'un système général de planification de tâches d'assemblage (2.1) nous semble être la clef de cette intégration.

La programmation automatique des robots est un problème général qui se prête difficilement à une décomposition en sous-tâches moins complexes. En effet, si l'on peut conceptuellement isoler quelques pôles fondamentaux tels, classiquement, les problèmes de *saisie*, de *transport*, de *montage*, il est impossible de s'affranchir de la forte dépendance qui existe entre ces différents modules (cf [1-Lau87]). Il semble pourtant que la résolution du problème général doive passer par une décomposition en sous-buts intermédiaires. Si la résolution de chacun de ces sous-problèmes fait encore l'objet d'une recherche à part entière, des résultats ont déjà été obtenus. En revanche, le problème d'une structure de contrôle gérant la décomposition en sous-tâches puis l'intégration des stratégies locales reste totalement ouvert. Il s'agit d'ailleurs précisément de l'objectif final du projet SHARP .

Les problèmes matériels soulevés par la mise en place d'un système robotique impliquent, quelque soit le niveau d'abstraction auquel on veut se placer

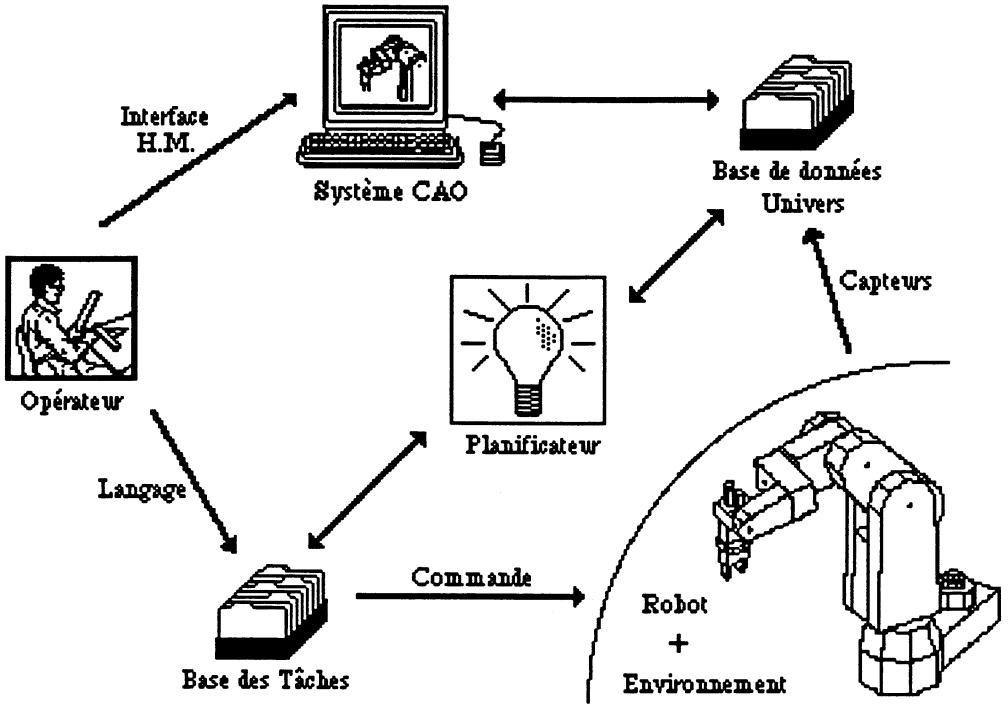

Figure 2.1: Cellule de CFAO robotique.

pour décrire la tâche à réaliser, la prise en compte d'informations et l'exécution d'actions de bas niveau. Ainsi, l'utilisation des capteurs du robot et des informations associées, l'intégration des erreurs dans la stratégie de contrôle du robot sont autant de problèmes devant être résolu par un système de programmation automatique. L'existence de ces contraintes rend très difficile la séparation verticale des niveaux de langages (cf 1.3.1), de la même manière que la séparation horizontale des différents sous-buts reste délicate.

Les données de base de la programmation automatique sont décrites en termes de moyen (ce dont on dispose pour s'informer, décider et agir) et de but (ce qu'on veut faire).

Etant donné le contexte industriel visé, il ne semble pas trop restrictif d'admettre que les moyens sont figés : l'environnement est clos et sa géométrie connue (au moins partiellement), les machines et robots disponibles et les outils afférents sont parfaitement définis. De même, quelque soit le langage utilisé par l'opérateur, il semble légitime de considérer que la tâche est parfaitement spécifiée.

Ce dernier point suppose donc résolue l'étape première qui consiste à planifier

l'utilisation des différents outils de production relativement au travail à effectuer et à déterminer l'ordre dans lequel les différentes pièces devront être assemblées, travail qui relève de la *génération de gammes d'assemblage*. La production d'un programme de commandes correspondant à la tâche demandée nécessite de trouver la manière de procéder pour réaliser la succession de manipulations définie précédemment. C'est-à-dire que pour chaque pièce impliquée dans le montage, il faut déterminer comment elle doit être saisie, manipulée, transportée et finalement assemblée, chacune de ses opérations dépendant très fortement des autres.

Un système de programmation automatique de robots d'assemblage doit pouvoir planifier *globalement* l'ensemble de ces opérations. Toutefois, il semble indispensable de subdiviser cette planification en tâches plus spécialisées faisant appel à des techniques de résolution bien distinctes, telles les trois opérations citées:

- *Saisie :* Le problème est de choisir une prise stable afin de contrôler la position spatiale de l'objet et de pouvoir l'utiliser là où sa présence est requise. Le choix de la prise doit également intégrer nombre de contraintes, certaines provenant de la morphologie de l'objet comme de celle du préhenseur ou de la géométrie de l'environnement local, d'autres étant imposées par la technique de montage choisie ou la nature des déplacements devant être effectués.
  On assimile habituellement ce problème de saisie à celui de la *manipulation :* il s'agit de toutes les opérations de reconfiguration de l'objet dans la pince ("regrasping") dues soit à l'incompatibilité des prises initiales possibles avec la réalisation du montage final, soit à l'impossibilité de transporter correctement l'objet (pour des raisons d'instabilité ou de risque de collision).

- *Transport :* Le problème est de déterminer une trajectoire évitant toute collision du manipulateur et de l'objet transporté avec l'environnement concerné par la réalisation de la tâche d'assemblage. Nous ne considérons que les mouvements de transfert, ou "grands mouvements", effectués entre les diverses positions clés de l'assemblage et insensibles (par hypothèse) aux effets des incertitudes.

- *Montage :* Alors que l'on désigne par assemblage la tâche générique devant être effectuée par le robot, le montage désigne plus spécifiquement l'opération de mise en contact d'entités géométriques visant à placer deux objets dans une position bien définie l'un par rapport à l'autre. Il s'agit

donc d'un ensemble de micro-mouvements devant être exécutés "en finesse" et utilisant des techniques de compliance afin de réaliser *pratiquement* le montage théorique rendu impossible de manière directe par les erreurs d'incertitude, tant géométriques que manipulatoires.

Cette décomposition est désormais classique et est utilisée dans tous les systèmes de planification d'assemblage existants. Le problème principal reste la forte dépendance entre ces différents modules. En effet, pour transporter une pièce il faut savoir comment elle est tenue dans la pince, donc comment elle est saisie; et pour la saisir, il faut savoir comment elle doit être montée. Il se dégage a priori de ce raisonnement une hiérarchie *Montage* $\rightarrow$ *Saisie* $\rightarrow$ *Transport*, mais celle-ci ne peut être scrupuleusement respectée. L'ensemble des prises permettant le montage peut en effet être incompatible avec celles réellement possibles ou avec les positions permettant le déplacement de l'objet...

Ces problèmes d'échecs locaux ne peuvent par conséquent être résolus que, d'une part, par l'emploi de méthodes de communication permettant l'échange d'informations vitales pour la résolution de chaque sous-problème, et d'autre part par l'utilisation d'une structure de contrôle chargée de la décomposition et la répartition des tâches spécialisées.

2.2 Les travaux précurseurs

Le problème de la programmation automatique des robots n'a jusqu'à présent suscité qu'assez peu de tentatives de réalisation concrète. Par contre, les différents sous-problèmes qu'elle induit directement - saisie, transport, montage - ou indirectement - incertitudes, contrôle - ont fait, font et feront longtemps encore l'objet de recherches intenses.

Parmi les précurseurs dans le domaine, le MIT a offert une très large contribution, traduite dans la pratique par l'existence des trois systèmes que sont LAMA, TWAIN et HANDEY, sommairement décrits ci-après.

2.2.1 Le système LAMA

Ce système a historiquement été le premier à véritablement aborder dans son ensemble le problème de la planification de tâches d'assemblage. Bien que se plaçant sur un plan purement théorique, les auteurs examinent les différents aspects du problème et esquissent l'architecture des futurs systèmes de program-

mation automatique (cf [1-LP76]).

Le but du système LAMA est de produire un *plan d'assemblage* à partir de la seule description symbolique de la tâche à effectuer. Cette tâche est décrite à l'aide d'instructions élémentaires d'insertion (GRASP), de transport (PLACE) et de montage (INSERT) permettant d'exprimer les actions du manipulateur sans en donner explicitement tous les paramètres. Ceux-ci sont par contre parfaitement spécifiés au niveau du plan d'assemblage. Le plan doit ensuite être décomposé en opérations du type "Prendre et Poser". Il s'agit d'un type de manipulation générique composée d'une saisie, d'un déplacement et d'un lâcher de la pièce concernée. Enfin, le système doit produire un programme exécutable par le robot (écrit en langage LLAMA).

Déjà les auteurs préconisaient l'emploi de modèles géométriques dérivés de la CAO, soulignaient l'importance de l'aspect relationnel de la structure géométrique employée et retenaient, outre la décomposition hiérarchique saisie-transport-montage, la nécessité d'une structure de contrôle intégrant l'information issue de capteurs (force, toucher, vision etc). Les solutions entrevues restaient encore théoriques, un module de saisie faisant seul l'objet d'une implantation partielle.

Le principal intérêt du système LAMA, présenté comme un premier essai épistémologique de réalisation d'un système de programmation automatique de robots d'assemblage, est d'avoir clairement mis en évidence les principaux problèmes inhérents au domaine.

2.2.2 Le système AUTOPASS

Le système AUTOPASS (IBM) se présente comme un système d'assemblage mécanique contrôlé par ordinateur. De même que LAMA, il vise à décomposer la tâche d'assemblage en actions élémentaires de type "Prendre et Poser". La spécification de ces opérations est toutefois plus complète, puisqu'elle inclut des manipulations délicates comme l'insertion et le vissage. D'autre part, un effort particulier portant sur l'aspect syntaxique du langage de description de la tâche a conduit les auteurs [1-LW77] à développer un système d'analyse lexicographique.

AUTOPASS est le premier système à avoir proposé un planificateur de trajectoires que l'on peut estimer fonctionnel. Toutefois, seul l'analyseur syntaxique et un système de modélisation géométrique furent réellement implantés.

2.2.3 Le système TWAIN

Longtemps les problèmes rencontrés lors de l'élaboration des premiers systèmes de programmation automatique ont conduit à des développements théoriques et pratiques séparés. Le système TWAIN [1-LB85] se présente, bien des années après LAMA et AUTOPASS, comme une tentative d'intégration d'un système réel.

Les auteurs proposent un certain nombre de nouvelles approches. Ainsi, le problème de la structure de contrôle, jusqu'alors négligé, est largement traité et un mécanisme de propagation de contraintes est proposé comme moyen de résoudre l'interdépendance des différents modules de planification. Les auteurs proposent d'utiliser ceux-ci pour fournir des éventails de solution possibles, facilitant ainsi le contrôle dans la mesure où les choix proposés par chaque sous-système sont alors moins spécifiques. De même ils préconisent l'utilisation de squelettes de programmes : en fait des procédures paramétrées, c'est-à-dire en l'occurence des programmes d'assemblage "modèles" pour lesquels il reste à fixer les valeurs des divers paramètres correspondant à la réalisation de la tâche. Le travail du planificateur consiste alors à choisir les procédures et à déterminer leurs paramètres d'appel. La méthode de propagation de contraintes est précisément appliquée aux squelettes de programmes employés afin de déterminer les valeurs des paramètres qui influencent le plus d'opérations. L'idée générale consiste donc à utiliser "en parallèle" les modules de saisie, de montage et de transport, partout où cela est possible, et à les synchoniser sur les difficultés communes.

Le plan d'assemblage est là encore formé d'opérations du type "Prendre et Poser", ici plus détaillées, améliorées même par la possibilité de montage. La décomposition adoptée est : déplacement du robot vers l'objet, saisie, déplacement, montage, et enfin éloignement de sécurité.

Le module de montage a pour but l'obtention d'une séquence de mouvements compliants permettant de réaliser pratiquement la tâche d'assemblage. Les auteurs proposent d'utiliser une méthode originale, dite des *pré-images*, consistant schématiquement à construire un chemin d'assemblage en démontant le composant considéré par des mouvements élémentaires. La méthode n'a pour l'instant fait l'objet d'aucune implantation.

Le module de transport vise à construire une représentation explicite de l'espace libre du manipulateur [10-LP81] dans l'*espace des configurations*, concept désormais adopté et utilisé par un grand nombre de planificateurs de trajectoires.

Enfin, le module de saisie cherche un ensemble de prises (pour une pince à mors parallèles) de type contacts plans. L'utilisation d'un filtre géométrique permet de réduire l'ensemble des prises candidates, celles-ci étant ensuite testées en calculant - toujours dans l'espace des configurations - les collisions éventuelles entre la pince et l'objet à saisir ou ceux alentours.

Comme ses prédécesseurs, TWAIN ne fut jamais implanté, à l'exception du module de planification de trajectoires, mais fit néanmoins l'objet de nombreuses études théoriques contribuant pour une large part aux progrès de la robotique.

2.2.4 Le système HANDEY

Le système HANDEY [1-LJM*87] constitue la première implantation véritable d'un système de type "Prendre et Poser" (ou "Pick and Place"). Les limitations - inévitables - du système sont une conséquence directe de cette réussite. Le nom de HANDEY est une contraction de "hand-eye", de l'œil à la main. Le but avoué de ses concepteurs est donc de fournir un système doté de capacités de reconnaissance de l'environnement, de planification des tâches à effectuer et de manipulation des objets.

La commande de base qu'un opérateur est susceptible de donner au système est de la forme "MOVE object TO destination", soit DEPLACER objet A destination . Bien sûr, le système doit posséder un modèle géométrique de son environnement (en l'occurence sous forme d'objets polyhédriques).

Le planificateur, à l'aide d'un système de vision tridimensionnel (caméra + capteur laser) identifie dans la scène la position des objets présents. Puis il choisit une prise stable en prenant en compte la configuration d'assemblage souhaitée et la présence d'obstacles éventuels, passant si nécessaire par une étape de ressaisie effectuée dans une zone libre réservée à cet usage. Puis il planifie une trajectoire de transport permettant d'atteindre la position de saisie et une autre afin d'amener l'objet à la position finale. Enfin, en l'absence de commande spécifique, le montage proprement dit est réalisé de manière élémentaire (à ce niveau en particulier, l'absence de véritable stratégie d'assemblage impose de sévères restrictions sur la nature des tâches réalisables). Le planificateur de trajectoires du système intègre tous les déplacements du robot, même les approches de saisie ou de montage. Il constitue la première implantation réelle d'un planificateur fonctionnel intégré dans un système plus vaste.

L'architecture du système HANDEY regroupe donc une interface de modélisa-

tion, un système de reconnaissance visuelle, deux modules de planification pour la saisie et le transport, et enfin un module de contrôle de la structure articulée du manipulateur. Encore pourvu de nombreuses imperfections, il présente l'avantage de constituer à ce jour le seul prototype de système de programmation automatique "en fonctionnement".

2.3 Le système SHARP

2.3.1 Architecture générale

Le système SHARP [1] - développé au LIFIA - n'est pas une réalisation industrielle mais constitue l'ossature expérimentale d'un système de planification de tâches d'assemblage. Le contexte d'implantation est celui d'une cellule robotique comprenant un manipulateur à poste fixe et un environnement connu a priori. Le but visé est la planification de la tâche au niveau manipulation à partir de spécifications géométriques.

L'architecture fonctionnelle de SHARP est basée sur les trois modules de planification saisie - transfert - montage ([1-LT85] et [1-Lau87]). Une structure de contrôle à deux niveaux permet de gérer les interdépendances des différents modules (cf figure 2.2). Le premier niveau traite la communication entre les modules de planification et la gestion des échecs de planification (dus aux contraintes trop fortes rendant impossible l'obtention d'une solution: accessibilité ou transport). Ces opérations n'étant pas effectuées sur le site d'assemblage mais en simulation, ce niveau est dit "*hors-ligne*". Le second niveau traite les échecs rencontrés lors de l'exécution du plan d'actions, inévitables en raison des incertitudes. La vérification est nécessairement effectuée sur le site même, "*en ligne*".

Des résultats significatifs ont été obtenus au niveau des modules de planification (saisie, transport et montage), dont l'action coordonnée vise à l'élaboration d'un programme d'assemblage de niveau objet. Cette phase de synthèse est pour l'instant contrôlée de manière simple (méthode essai-erreur) jusqu'à obtention d'un programme complet. Elle est alors suivie d'une phase de vérification et de correction visant à rendre le plan de manipulation théorique compatible avec les données physiques de l'assemblage. L'exécution en ligne du programme de manipulation peut alors être effectuée, le contrôle du robot étant parfaitement opérationnel (mouvements compliants et gardés). Enfin nous disposons d'un

[1] Acronyme pour High level System for Automatic Robot Programming

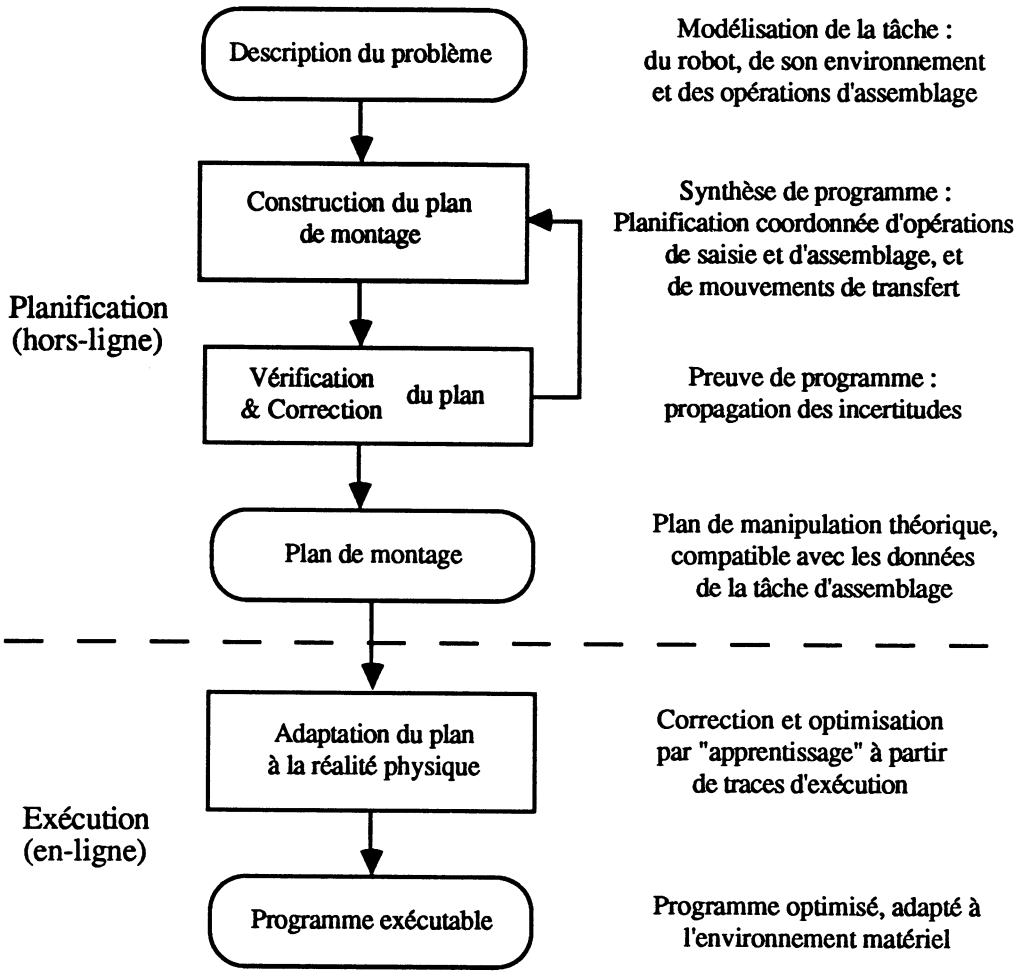

Figure 2.2: Architecture fonctionnelle du système SHARP.

système de modélisation et de simulation pour la robotique.

2.3.2 Entrées du système et paramétrisation

Un système de programmation automatique de robots d'assemblage a besoin de deux types d'information : un modèle de l'environnement dans lequel il doit opérer et une description du travail qu'il doit effectuer. Le premier est donné par l'intermédiaire d'un système de modélisation spécialement conçu pour la robotique d'assemblage, le second est spécifié grâce à un langage de description géométrique. La description des différentes représentations du modèle fait l'objet du chapitre 4. Nous nous intéressons ici à la manière dont ces représentations sont données au système.

La description de la tâche

La spécification de la tâche a été évoquée précédemment. Notre système ne possédant pas - encore - d'interface de description en langage symbolique, la spécification du but à atteindre est faite de manière purement géométrique, par la donnée des positions relatives des objets à assembler.

L'interface de modélisation

Le système de modélisation possède certaines caractéristiques imposées par l'utilisation qui doit en être faite. Le noyau est évidemment formé par les données, spécifiques aux raisonnements mis en œuvre en robotique. En fait il s'agit non seulement des données elles-mêmes mais aussi de la manière dont elles sont organisées, fonction de la nature des objets modélisés et des traitements qu'elles doivent subir. Le fait que ces données doivent être entrées dans la mémoire du calculateur impose de plus l'existence d'une interface homme-machine (IHM). Nous avons été amenés, à la recherche d'une certaine convivialité et d'une modélisation performante, à concevoir une IHM très classique du point de vue de la CAO. Nous pensons que l'orientation future de tels systèmes tend vers la programmation graphique, aussi bien pour l'aspect modélisation que pour la description de la tâche.

Respectant les principes ergonomiques qui font actuellement loi dans le domaine, notre interface repose sur une utilisation intensive de périphériques d'entrée "évolués" : souris ou tablette graphique. Les choix possibles de l'opérateurs sont présentés sous forme de menus hiérarchiques et le résultat de ces choix est visualisé de manière conviviale (affichage des solides manipulés etc).

Mais l'originalité de ce système de modélisation réside dans la nature des représentations sous-jacentes - et de l'information aussi bien numérique que symbolique qui y est stockée - lesquelles sont décrites dans le chapitre 4.

Il s'agit donc d'un module semi-automatique, acceptant en entrée les deux descriptions mentionnées ci-dessus, issues d'un opérateur humain par l'intermédiaire d'une interface spécialisée, et fournissant en sortie une base de données complète qui pourra être utilisée de manière optimale par les différents planificateurs.

Paramétrisation des données

La description de la tâche est paramétrée par la donnée des positions finales des objets qui doivent être assemblés. Le modèle de l'environnement comporte également une partie description numérique indispensable, celle des diverses entités géométriques présentes dans l'univers de travail.

De tous les modules de planification, seul le planificateur de trajectoires utilise explicitement la structure cinématique du robot. Celle-ci constitue un paramètre du système. Les seules contraintes a priori sur la nature du robot (dans l'implantation actuelle) sont d'être constitué d'une chaîne articulée "ouverte" comportant un nombre quelconque de degrés de liberté et terminée par une pince à mors parallèles, les articulations pouvant être indifféremment prismatiques ou rotoïdes. Nous verrons toutefois que l'obtention de performances "raisonnables" fixe une borne supérieure à la longueur de la chaîne cinématique du manipulateur. Par contre, pour un nombre de degrés de liberté fixé, cette paramétrisation n'influe en rien sur les performances du système, ceci étant permis par la puissance des langages utilisés : le LISP (le programme général étant une donnée auto-modifiable pour arriver à un programme dédié au robot spécifié) et C (présence d'instructions de compilation conditionnelle #define, #if etc).

2.3.3 La structure de contrôle du planificateur

Comme nous l'avons déjà dit, la forte dépendance des trois sous-problèmes induits par la décomposition hiérarchique choisie interdit une résolution strictement parallèle et séparée. La structure de contrôle doit être à même de gérer les problèmes d'incompatibilité et les échecs.

Une méthode simple - mais pas forcément très efficace - consiste à ordonner arbitrairement les modules de planification et gérer les échecs par un mécanisme de retour arrière ("backtracking"). Lorsque l'une des contraintes imposées par un module ne peut être satisfaite, celui-ci doit effectuer une replanification. Quand celle-ci n'aboutit pas à un meilleur résultat, il devient nécessaire de modifier les données d'entrée du module, c'est-à-dire de propager l'échec au module amont qui devra à son tour effectuer une replanification.

On est donc amené à associer naturellement à chaque module un certain nombres de contraintes, correspondant à ce qu'il lui est impossible de réaliser. Les contraintes issues de chaque planification locale peuvent être considérées comme des règles impératives dans un contexte figé, que le système de contrôle doit

prendre en compte afin d'assurer au mieux l'exécution de la tâche globale.

Les modules de saisie et de montage ont en commun, du point de vue de la planification, d'opérer à partir de considérations morphologiques pour réaliser une situation géométrique basée sur des contacts, ceci de manière totalement indépendante et, du point de vue de la communication inter-modules, de fournir une position de référence unique. En revanche, le module de transport (trajectoires de transfert) ne s'intéresse pas à la position des éléments de l'assemblage, sinon comme obstacles supplémentaires à éviter.

La position finale des objets étant la seule contrainte incontournable dans le bon accomplissement de la tâche, il nous paraît raisonnable de considérer les contraintes liées au module de montage comme prépondérantes. En effet, si la détermination d'une prise correcte paraît tout aussi contrainte, on peut toujours, lorsqu'elle ne peut être effectuée directement, reconfigurer la prise de l'objet par une ou plusieurs opérations dites de "regrasping". On peut donc envisager à la limite l'emploi d'une stratégie visant à trouver une position de saisie initiale quelconque, totalement indépendante de la position finale, puis à passer de l'une à l'autre par une succession de resaisies. Dans cette optique, les contraintes du montage doivent être prioritaires sur ceux de la saisie, qui à leur tour doivent s'imposer au module de transport, le plus "souple". Cette flexibilité du planificateur de trajectoires s'explique essentiellement par la nature moins contraignante de la tâche qui lui est assignée : dans le cas de l'assemblage, on cherche le plus souvent *une* trajectoire sans se préoccuper de sa forme. D'autre part, il constitue un module intermédiaire dont les contraintes de but dépendent surtout des autres planificateurs, et très peu (voire pas du tout) de la nature des objets à assembler.

2.3.4 Les modules de planification intermédiaire

La structure de contrôle gère donc la décomposition de la tâche en sous-buts et défère leur résolution à un ensemble de planificateurs spécialisés. S'il s'agit toujours de calculer des trajectoires - le robot n'étant après tout capable que de déplacements - la nature de celles-ci et la diversité conceptuelle des algorithmes utilisés dans ce but impose cette décomposition tripartite.

Le module de montage

Il s'agit d'un module de planification automatique, acceptant en entrée une configuration d'assemblage à réaliser et fournissant en sortie la position spatiale à laquelle l'objet concerné doit être amené. Cette position peut alors constituer un but du planificateur de trajectoires.

La nécessité d'un tel module provient de la présence inévitable d'incertitudes en position lors de la manipulation d'assemblage. En effet, la commande de mise en position finale, engendrée à partir des seules données théoriques issues d'une représentation géométrique, est virtuellement vouée à l'échec. Il est alors indispensable pour mener à bien le montage, d'une part de pouvoir reconsidérer la position des pièces concernées grâce à l'utilisation de capteurs, et d'autre part de pouvoir effectuer des mouvements relativement aux objets eux-mêmes et non à leur position nominale.

On se place dans le cas de l'utilisation d'un seul manipulateur, ce qui a pour conséquence directe de limiter la manipulation au montage de deux objets, l'un supposé fixe alors que l'autre est tenu par le préhenseur. L'objet manipulé est amené à une position initiale, dite position d'approche, "raisonnablement" proche de la position finale et réalisable avec certitude (relativement à l'ensemble du manipulateur) malgré les incertitudes de position qui caractérisent le montage.

Le but des mouvements fins de montage est précisément de réduire l'incertitude de position des objets. Ainsi la réalisation d'un contact entre deux entités géométriques a pour effet de supprimer l'incertitude relative suivant la direction du mouvement, l'incertitude absolue étant soit réduite, soit conservée. Par conséquent, la stratégie du planificateur de mouvements fins est essentiellement de créer ou maintenir des contacts entre les éléments à assembler.

Plus spécifiquement, la méthode utilisée vise à construire un plan de démontage fictif puis à en déduire par chaînage arrière un plan de montage correct cf [2-The88].

Le module de saisie

Il s'agit d'un module de planification automatique, acceptant en entrée la position spatiale d'un objet à saisir et fournissant en sortie une prise et la direction d'approche associée. La position extrémale ainsi obtenue constitue alors un but

possible pour le planificateur de trajectoires.

L'action de saisie présente de nombreuses analogies avec celle de montage. En effet, il s'agit de réaliser également une situation de contact entre deux objets (l'objet mobile étant ici le préhenseur lui-même) à partir d'une position d'approche. La différence réside dans l'ignorance a priori des contacts à réaliser dans la situation terminale.

L'objectif poursuivi - la détermination d'une prise - doit respecter deux types de contraintes : celles, incontournables, inhérentes à la morphologie de l'objet à saisir et du préhenseur (accessibilité, stabilité) et celles qui sont issues des spécifications de la tâche (attitude d'un outil ou d'un objet à monter).

Le module de saisie a pour but la résolution des deux sous-problèmes induits : d'une part la détermination d'un ensemble de prises possibles de l'objet et les configurations associées du manipulateur, et d'autre part le choix d'une position d'approche avant saisie, utilisable en particulier par le module de transport, ainsi qu'une position de retrait après saisie répondant au même critère. Les déplacements entre positions d'approche et de retrait et position de prise sont des translations garantissant l'absence de collision dans le voisinage de la pièce à saisir (pour une prise à mors parallèles seulement).

La méthode retenue ([2-LP83] et [2-PT86]) opère en deux phases : l'une dite d'analyse d'accessibilité locale et de stabilité (A^2LS), l'autre d'analyse d'accessibilité globale (A^2G).

La phase A^2LS a donc pour but de déterminer un ensemble de prises potentielles, soit de paramètres de préconfiguration de l'outil terminal du manipulateur. Ce calcul est basé sur une étude topologique de la pièce à saisir en fonction de la morphologie du préhenseur. Les problèmes de stabilité et d'incertitudes sont traités par l'utilisation de filtres géométriques appropriés.

Lors de la phase A^2G, le système vérifie la compatibilité des prises retenues par rapport à l'environnement de manipulation, ce qui permet de fixer les paramètres de prise encore indéterminés. Il s'agit en fait de la recherche locale d'une trajectoire rectiligne exempte de collision avec les objets du voisinage. Les contraintes de déplacement imposées par le parallélisme du contact entre la pince et l'objet permettent de réduire la complexité de cette recherche à une translation dans l'espace à deux dimensions des configurations de l'ensemble pince - objet manipulé.

Le module de transport

Il s'agit d'un module de planification automatique, acceptant en entrée deux positions spatiales (ou deux configurations) de départ et de but, et fournissant en sortie un trajectoire exempte de collision qui les relie.

Contrairement aux deux modules précédents, celui-ci doit planifier une séquence de mouvements évitant absolument tout contact avec les objets de l'environnement. Les contraintes du problème sont figées, dans la mesure où l'environnement n'est pas manipulable et les positions initiale et terminale sont impératives.

La recherche d'une trajectoire s'effectue dans l'espace des configurations du manipulateur, dans lequel elle est équivalente à la recherche d'un chemin dans un graphe. Cette représentation de l'espace des configurations traduit les contraintes dues à l'encombrement spatial de l'environnement sur les mouvements possibles du robot. Elle doit donc être initialement construite à partir de la description géométrique de l'univers puis être mise à jour lorsque celui-ci est modifié.

C'est sur ce module que porte l'essentiel de notre contribution.

2.3.5 Sortie du système

Le langage cible

La programmation automatique d'une tâche d'assemblage, en tant que processus de planification hors-ligne, s'arrête donc à la création du programme de niveau objet. Celui-ci est écrit dans un langage cible (LDGT) au nombre d'instructions volontairement limité, que nous avons déjà évoqué et que nous décrivons sommairement ci-après. Toutefois l'exécutabilité de ce programme ne peut être totalement garantie, compte tenu des impératifs de l'environnement physique.

Le langage comporte quatre classes d'instructions fondamentales, que l'on peut présenter de manière synthétique en disant qu'elles visent à réaliser respectivement les actions de *saisie*, *contact*, *position* et *correction*.

Instructions de saisie

De manière générale, il s'agit de réaliser une action de préhension. Un premier niveau de séparation distingue deux actions complémentaires : la saisie et le

lâcher. La première consiste à déplacer en mode cartésien la main du robot de la position courante à la position spécifiée pour y actionner effectivement la pince. A l'inverse, la seconde permet d'actionner la pince afin de libérer l'objet qui y est tenu puis d'effectuer un mouvement cartésien jusqu'à une position de retrait. Cette distinction paraissant suffisante, nous nous contenterons donc des deux instructions Saisir et Lâcher.

L'intégration dans ces instructions mêmes du déplacement local effectué relativement aux objets manipulé (aussi bien la phase d'approche préliminaire que celle de retrait ultérieur) a été justifié par la décomposition du problème de planification.

Instructions de contact

Nous distinguons ici deux types d'instructions : celles permettant de réaliser un contact entre deux entités géométriques et celles permettant d'en supprimer un, notées Réaliser-C et Supprimer-C. Toutes deux doivent autoriser l'exécution d'un guidage compliant et gardé, c'est-à-dire maintenant un ou plusieurs contacts annexes jusqu'à réalisation de la position de garde (le but à atteindre étant défini par une description des contacts terminaux).

Ce type de déplacement permet essentiellement de réaliser les mouvements de précision indispensables à l'opération de montage en s'affranchissant des incertitudes sur la position des objets.

Instructions de positionnement

Le positionnement peut être défini comme la réalisation d'une relation géométrique, c'est-à-dire comme la mise en correspondance de deux repères : celui spécifiant la position du préhenseur du robot et celui identifiant la position spatiale à atteindre. La réalisation d'une telle situation géométrique peut être spécifiée à l'aide d'une seule instruction, notée Réaliser-S. Celle-ci constitue la commande de base utilisée par le planificateur de trajectoires lors de l'étape finale de production d'un programme de commande exécutant le déplacement requis.

Les paramètres nécessaires en sont la position finale à atteindre par le manipulateur et la liste des positions intermédiaires impératives. Entre chacune de ces positions, le déplacement se fait en mode cartésien. Dans l'optique d'une planification de trajectoires sans collision, et compte tenu de la méthode que

nous avons développé, ces positions peuvent avantageusement être spécifiées en mode articulaire avec une discrétisation adaptée à la précision des trajectoires souhaitées. Ceci permet de contrôler les positions du robot et d'éviter ainsi de coûteuses reconfigurations.

Instructions de correction :

Ici encore une seule instruction s'avère nécessaire : **Corriger**. Cette instruction permet la correction incrémentale d'une relation géométrique. En effet, diverses erreurs et incertitudes viennent immanquablement perturber l'exécution du plan d'amendement nominal (incertitude sur la position réelle des objets, imprécision du déplacement du manipulateur...) et il est indispensable de pouvoir y remédier. Cette correction peut d'ailleurs être vue comme une boucle de régulation visant à maintenir le déplacement réel aussi proche que possible de la consigne émanant du plan d'actions originel, la mesure de l'erreur étant alors effectuée par divers capteurs.

Syntaxe des instructions de déplacement

Nous ne présentons ici que l'instruction de positionnement utilisée par le module de planification de trajectoires, et les éléments de syntaxe associés qui nous intéressent. On pourra trouver une spécification complète des six instructions évoquées précédemment dans [2-The88].

(Réaliser-S <situation> Via <chemin>)

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

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

<position articulaire> ::= $(\theta_1 \ldots \theta_N)$; $\theta_i \in \mathcal{R}$ et N = nombre de d.d.l. du robot.

Chapitre 3

La planification de trajectoires

3.1 Définition du problème

3.1.1 Contraintes spatiales

Identification du problème

Les données du problème de planification de trajectoires se résument, en plus des contraintes de la tâche, au robot et à son environnement, disponibles sous la forme d'un modèle essentiellement géométrique. La clé de sa résolution réside donc dans notre aptitude à raisonner sur ce type de représentations, à les manipuler et les transformer afin d'obtenir finalement une version à partir de laquelle on puisse aisément faire apparaître une solution.

Le problème général de la planification de trajectoires est fort complexe et ne semble pas soluble tel quel. Il se décompose naturellement en classes de sous-problèmes dont certains, moins difficiles, peuvent trouver une solution. Cette subdivision de la complexité se trouve alors immanquablement liée à la dimension de l'espace de travail, au nombre et à la nature des objets mobiles constituant le robot.

Sur ce dernier point, il importe de souligner les problèmes spécifiques inhérents aux deux archétypes de robots existants, le robot manipulateur et le robot mobile. Il est toutefois nécessaire, pour pouvoir opérer la distinction entre les deux, de définir quelques notions supplémentaires.

Rappels de mécanique analytique, définitions

Un *solide parfait* est tel que la distance entre deux quelconques de ses points est constante. Nous considérons que tous les objets sont des solides parfaits

(éliminant les objets souples et déformables). Un *système mécanique* est défini comme un ensemble de solides parfaits non libres, c'est-à-dire liés entre eux par des *liaisons intérieures.* Le système complet est alors soit libre, soit lié à un référentiel par des *liaisons extérieures.*

Le *mouvement* d'un solide peut être défini par la donnée à tout instant de sa position spatiale, et le mouvement d'un système mécanique par la donnée à tout instant du N-uplet de ses coordonnées généralisées.

On appelle *coordonnées généralisées* d'un système mécanique les N paramètres indépendants q_n qui déterminent complètement la configuration du système (c'est-à-dire la position de *tous* ses points relativement à un référentiel). Le nombre de *degrés de liberté* du système est défini comme le nombre S de mouvements indépendants possibles.

Un système mécanique est dit *holonome* si les équations qui traduisent les liaisons intérieures entre ses composants ou éventuellement ses liaisons extérieures ne comportent pas les dérivées des points du système. Pour un système holonome, on a $S = N$; pour un système non holonome, $S = N - K$, où K est le nombre de liaisons non holonomes.

Nous pouvons donc définir maintenant un *robot* comme un système mécanique général doté de certaines fonctionnalités particulières (déplacement autonome, préhension etc).

Muni de ces définitions, on appelle généralement *robot manipulateur* à N degrés de liberté un système mécanique à N coordonnées généralisées composé de plusieurs solides en liaison holonome et lié à un référentiel par une liaison constante. On appelle identiquement *robot mobile* un système mécanique composé d'un solide assujetti à l'environnement par des liaisons extérieures généralement non holonomes.

L'holonomie admise du robot manipulateur permet de ne considérer ses liaisons que dans leur aspect géométrique, contrairement au robot mobile pour lequel l'aspect cinématique est fondamental.

Les divers aspects de la planification de trajectoire

Le robot manipulateur est donc constitué d'une structure articulée dont la base est fixe et dont l'extrémité terminale est généralement munie d'un outil de préhension permettant de manipuler des objets. Le robot mobile est un véhicule

progressant de façon autonome dans un environnement quelconque. Les deux peuvent être combinés : on obtient alors un robot autonome capable à la fois de déplacement, d'observation et d'action.

Le problème général de planification de trajectoire revient donc à déterminer le mouvement d'un système mécanique à N coordonnées généralisées. Sa complexité dépend de ce nombre et de la nature de ses liaisons extérieures.

Ainsi pour un robot manipulateur, on est confronté à un problème tridimensionnel dont la complexité dépend beaucoup du nombre d'articulations. Le problème est alors essentiellement de nature géométrique, et peut être formulé spécifiquement de la façon suivante :

> Déplacer un solide dans l'espace (la pince du manipulateur par exemple) en évitant les objets présents, avec la contrainte supplémentaire d'éviter toute collision à tout point de la structure articulée qui relie ce mobile à une base fixe.

A l'inverse, la complexité du problème de détermination de trajectoires pour un robot mobile est essentiellement due à sa cinématique. En effet, la morphologie du robot est alors extrêmement "sympathique" (relative simplicité du volume, et surtout *rigidité* de la structure), et la géométrie du problème est généralement assez simple (déplacement bidimensionnel d'un solide libre dans le plan). Mais le caractère non holonome de ce déplacement impose des contraintes supplémentaires extrêmement délicates à prendre en compte.

Le problème de planification de trajectoires repose donc sur une base géométrique, celle du déplacement d'un solide au milieu d'autres solides. L'intérêt de l'espace des configurations est d'offrir la possibilité de caractériser l'existence d'une trajectoire réalisant ce déplacement par l'existence d'une composante connexe contenant les positions initiale et finale. Notons que cette caractéristique n'est valable a priori que pour des systèmes holonomes : dans ce cas seulement, toute configuration adjacente à une autre est atteignable depuis cette dernière sans sortir du voisinage défini par leur union.

Ce problème de déplacement peut déjà se révéler délicat suivant la nature des solides manipulés ou de l'espace de travail. Il peut ensuite être compliqué en augmentant le nombre de solides concernés (cas typique du manipulateur) ou en imposant des contraintes sur la nature du déplacement (cas du robot mobile).

On peut enfin ajouter des contraintes relatives à l'environnement (déplacement des obstacles, problème multi-mobiles) ou à la trajectoire (optimalité, maintien de contact ou suivi de surface).

Notre travail porte exclusivement sur l'aspect géométrique, dans son cas pratique le plus général, celui d'un robot manipulateur évoluant dans l'espace tridimensionnel (en l'occurence dans le contexte de la robotique d'assemblage).

3.1.2 Contraintes temporelles

Si la planification dont nous parlons est a priori essentiellement spatiale, la prise en compte du temps peut y être faite à plusieurs niveaux.

L'hypothèse d'un environnement statique, figé n'est en effet pas réaliste dans le contexte de la robotique industrielle. Même en supposant que le robot soit le seul agent présent dans son domaine d'évolution, il est nécessaire de prendre en compte les modifications dues à ses propres actions. Dans ce cas l'univers du robot évolue suivant un temps qui lui est propre, et la planification d'une trajectoire dans un état donné se fait nécessairement en statique, puisque l'évolution ne se produira qu'avec le déplacement d'un objet, conséquence de cette planification même.

Par contre, si le domaine spatial atteignable par le manipulateur l'est également par un autre robot, l'hypothèse d'un état immuable n'est plus valide : d'une part les objets peuvent être déplacés par le robot étranger et, d'autre part, celui-ci constitue lui-même un objet de l'environnement en constant déplacement. Dès lors, le modèle sur lequel opère le planificateur de trajectoires est amené à évoluer dans le temps au rythme des mouvements de l'autre robot. Entre deux de ces mouvements, on est ramené au cas d'un univers statique dans lequel le manipulateur surnuméraire n'est plus qu'un obstacle parmi d'autres. Le problème de la coopération de deux manipulateurs peut donc être vu comme un problème découplé de planification à un manipulateur augmenté de fortes contraintes temporelles. Les algorithmes utilisés doivent alors souvent être remis en question pour d'évidentes raisons de performance.

Nous ne traitons pas dans ce mémoire cet aspect temporel de la planification de trajectoires. Aussi, dans la suite, et sauf mention contraire, le terme de *robot* désigne un manipulateur constitué d'un nombre quelconque de solides en liaison articulaire, évoluant dans un espace cartésien tridimensionnel partiellement encombré.

3.1.3 Méthodes de résolution

La démarche générale de la résolution d'un problème peut être résumée en quatre étapes fondamentales : tout d'abord saisir et comprendre les données de ce problème, établir une représentation idoine permettant la découverte d'une solution, calculer effectivement celle-ci et enfin rétablir la solution dans la représentation originelle où l'on peut vérifier sa validité.

Les données du problème

Dans notre cas, les données du problème se "résument" à la description de l'univers, c'est-à-dire du robot et de son environnement, et d'une tâche à effectuer. Au niveau du système de planification de trajectoires, et grâce à la planification conjointe des autres sous-problèmes (saisie, montage), celle-ci n'est plus décrite de manière abstraite, mais comme un ensemble de positions spatiales à atteindre.

Par conséquent, les données de notre problème se décomposent en constantes (le modèle du robot et de son environnement) et variables (positions du robot et des objets, configurations à atteindre). Les premières étant donc connues a priori peuvent faire l'objet d'un prétraitement. C'est précisément l'objectif du changement de représentation constituant la deuxième étape.

Nous émettons de plus l'hypothèse que les données du problème sont *fermées*, c'est-à-dire complètes (on dispose de toutes les informations nécessaires pour résoudre le problème) et non ambigües (ces informations ne sont pas contradictoires). Enfin la redondance éventuelle des données n'est pas gênante, pourvu qu'elles soient cohérentes.

La représentation de travail

Cette deuxième étape est à notre sens la clef du problème de planification de trajectoires et plus généralement du raisonnement géométrique : il s'agit de construire une représentation de l'univers pertinente relativement au but à atteindre, en l'occurrence la détermination de trajectoires. Dans notre cas, il paraît naturel de considérer cette représentation comme un ensemble de configurations du robot auxquelles sont associées des informations d'ordre statique. Cet ensemble ne peut être obtenu directement mais doit être dérivé de la représentation géométrique de l'univers qui constitue une donnée du problème.

La représentation élaborée doit de plus intégrer diverses contraintes de l'environnement non spécifiées explicitement dans les données initiales, comme les imprécisions du modèle fourni, les incertitudes sur la localisation des objets et les perturbations introduites par la dynamique du robot.

Du fait de la complexité géométrique d'une structure articulée, la construction de cette représentation constitue un problème difficile et un sujet de recherche à part entière, dont la phase de modélisation initiale est la pierre d'achoppement.

La recherche d'une solution

Une fois cette représentation disponible, la recherche d'une trajectoire proprement dite fait appel à des techniques de planification de deux types : la recherche d'un parcours effectif dans la représentation courante (laquelle se se ramène à la détermination d'une séquence de configurations de cette représentation satisfaisant les contraintes de la tâche) et la mise à jour locale de cette représentation (dans le cas où aucune solution satisfaisante ne peut être trouvée).

Cette recherche d'une trajectoire peut être enrichie de celle d'une enveloppe de trajectoires laissant quelque liberté dans le choix de la commande finale, afin de pouvoir prendre en compte aussi bien les incertitudes dans le déplacement du robot que, éventuellement, une modification mineure des configurations initiale et terminale. Dans tous les cas, le système peut soit conclure à l'échec (il n'existe pas de trajectoire) soit fournir une ou plusieurs trajectoires correctes.

La trajectoire planifiée

La trajectoire calculée doit être alors transformée en une séquence de positions spatiales respectant les contraintes cinématiques et dynamiques du robot. Les restrictions ainsi imposées par la nature des mouvements réalisables contraignent alors le choix final d'une trajectoire parmi l'éventail proposé. Cependant, il n'est pas de notre propos de traiter dans ce mémoire des problèmes que pose la commande du robot : si les trajectoires finalement retenues par notre planificateur doivent être "plausibles", leur exécution reste cependant dépendante du contrôleur utilisé.

La trajectoire calculée peut alors être traduite sous la forme plus exploitable qu'est celle d'une procédure de commandes du robot. Celle-ci est écrite en un langage de niveau objet (LDGT), par compatibilité avec celles issues des autres planificateurs.

3.2 Complexité du problème de planification de trajectoires

3.2.1 Résolution de problème

Résoudre un problème revient à exhiber un objet d'un ensemble connu, au besoin construit à dessein, et satisfaisant les contraintes imposées par les hypothèses de départ. On peut distinguer sommairement trois méthodes permettant d'atteindre ce but : l'application d'une formule explicite, l'utilisation d'une définition récursive ou d'un algorithme convergent et l'énumération de tous les cas possibles.

La méthode explicite est évidemment la plus satisfaisante. Malheureusement, la résolution analytique du problème de recherche de trajectoires se révèle pour l'instant impossible, sauf dans certains cas pour des mobiles en déplacement dans le plan [5-Boi87]. La complexité du problème est alors celle du calcul de la formule.

La forme de résolution récursive ne peut être applicable qu'à condition de connaître ou de pouvoir construire une fonction de résolution pouvant être appliquée au problème. De la même manière, la méthode itérative utilisant un algorithme convergent n'est possible que si l'on est, d'une part, capable de passer d'un stade de résolution donné à un stade plus avancé et d'autre part capable de résoudre une situation initiale (la récursivité devenant alors récurrence).

La méthode par énumération reste donc la seule applicable dans une large majorité de problèmes et en particulier celui de la recherche d'une trajectoire : tous les cas possibles sont testés jusqu'à obtention d'une solution. Il s'agira donc d'établir la liste de toutes les positions physiquement possibles du robot, c'est-à-dire n'engendrant aucune collision entre le robot et son environnement.

3.2.2 Notions de complexité algorithmique

Un écueil supplémentaire dans la résolution de notre problème provient de la nature même de l'ordinateur en tant que support de résolution : il faut tenir compte des contraintes d'espace et de temps. Il faut non seulement trouver une solution mais en plus qu'elle soit calculable avec une place mémoire et un temps raisonnables. La complexité de la résolution est ainsi définie comme *le nombre d'opérations qu'elle recquiert exprimée en fonction de la taille des*

données, ceci bien sûr en utilisant le meilleur algorithme connu (qui n'est pas forcément optimal). La complexité exacte d'un algorithme étant souvent impossible à déterminer précisément, on l'assimile généralement à sa complexité maximale, qui est la borne supérieure de la précédente.

Ainsi la complexité idéale est constante : elle ne dépend pas des données d'entrée. Mais, si certains problèmes sont "simples" (de complexité linéaire, soit en $O(n)$ ou en $O(n^2)$), la majorité est nettement plus difficile. Cette notion de difficulté regroupe les problèmes en trois classes dites P, E et les autres. La première catégorie (P) est celle des algorithmes *polynomiaux* : la complexité peut être évaluée à un polynôme de degré constant connu, indépendant des données. Les problèmes plus difficiles sont dits *exponentiels* par nature (E) : la complexité est en f^n où f est lui même un polynôme en n. Enfin il existe des algorithmes n'appartenant à aucune de ces deux classes : leur complexité reste inévaluable pour le moment.

La nature de l'implantation informatique permet de distinguer ensuite deux types de machines algorithmiques, dites *déterministes* ou *non-déterministes*. La première catégorie de machine peut être ramenée à un automate d'états qui, à un moment donné, se trouve dans un seul état bien déterminé et dont l'action ne dépend que de celui-ci. La seconde désigne de la même façon un automate muni, en plus des mêmes instructions, d'une fonction de choix (aléatoire), et dont les actions ne dépendent donc pas uniquement de l'état dans lequel il se trouve mais également du hasard.

On obtient finalement une hiérarchie comprenant trois classes de complexité : $P \subset NP \subset P$-space. La classe P comprend donc les algorithmes polynomiaux (dont la résolution demande un temps polynomial), la classe NP les algorithmes pouvant être résolus en temps polynomial sur une machine non déterministe, c'est-à-dire aussi en temps exponentiel et en mémoire polynomiale sur une machine déterministe. La classe P-space regroupe les algorithmes demandant de plus une taille mémoire polynomiale sur une machine non déterministe.

Un problème est dit *dur* pour une classe si sa complexité est du même ordre que tout problème de cette classe, et *complet* s'il est difficile et appartient à cette classe. Ainsi les problèmes NP-durs sont de complexité équivalente aux problèmes non déterministes polynomiaux, et l'intersection des deux groupes donne les problèmes NP-complets. Notons que les problèmes NP-durs sont considérés traitables seulement pour un petit nombre de variables.

3.2.3 Complexité du problème de la planification de trajectoires

Complexité estimée

L'évaluation de la complexité d'un algorithme est un point capital pour estimer l'intérêt que présente son utilisation. Ainsi la plupart de problèmes robotiques sont évalués de complexité NP ou même P-space, rendant illusoire la recherche d'algorithmes universels résolvant le problème de planification de trajectoires.

Les recherches effectuées sur le sujet par Schwartz et Sharir [10-SS83*] ont montré le résultat fondamental suivant : le problème de la planification de trajectoires d'un robot à nombre de degrés de liberté fixé est traitable (c'est-à-dire déterministe polynomial ou P-complexe).

Les hypothèses faites ne sont pas très restrictives : d'une part l'espace libre du robot doit être défini par des inégalités algébriques polynomiales exprimant les limites de collision (les objets de l'univers doivent donc être rigides, compacts et eux-mêmes limités par des surfaces algébriques d'équation polynomiale), et d'autre part les articulations du robot doivent être des translations et/ou des rotations.

Alors le problème de planification de trajectoires peut être résolu en temps polynomial en le nombre des inégalités en question. L'évaluation précise de la complexité pour un manipulateur à N degrés de liberté dans un univers défini par m surfaces est : $m^{2^{N+6}}$

Notons que la méthode conduisant à ce résultat repose sur l'algorithme d'élimination des quantificateurs de Tarski et utilise la décomposition cylindrique des ensembles semi-algébriques de Collins. Elle n'est hélas d'aucune utilité pratique, étant vraisemblablement impossible à implanter de manière efficace.

Ce résultat peut néanmoins être étendu au problème général de la planification de trajectoires, dont il a pour intérêt principal de majorer la combinatoire. De plus, il laisse supposer qu'en pratique le problème n'est pas traitable pour un grand nombre de degrés de liberté, constatation que l'expérimentation n'a jusqu'alors que faiblement démenti.

Le problème général à nombre de degrés de liberté non borné a été prouvé du type P-Espace [5-CR86]. Des résultats ont également été obtenus pour des

problèmes particuliers. Ainsi beaucoup de problèmes de déplacement ont été prouvés de complexité NP ou P-space, tel celui du déplacement dans le plan d'un ensemble de disques qui est donc "seulement" NP-dur, ou celui du déplacement coordonné de rectangles à l'intérieur d'un autre rectangle, qui est P-space complet.

Réduction de la complexité

Une technique de réduction par *projection* a été introduite par Schwartz et Sharir [10-SS83*]. Elle est basée sur le principe même introduit par la méthode algébrique citée précédemment : celui d'une décomposition récursive en cellules de l'espace des configurations et le regroupement de celles-ci en un graphe d'adjacence.

L'idée générale consiste à réduire la complexité du problème en fixant l'un de ses paramètres et en résolvant le sous-problème ainsi obtenu. Opérer ainsi de manière récursive permet de ramener le problème originel à une configuration de complexité minimale (compte tenu des données) pour laquelle une solution est effectivement calculable. Celle-ci constitue alors une solution particulière du problème global.

La recherche d'une telle solution partielle doit être effectuée de manière discrète pour un nombre fini de valeurs critiques du paramètre fixé. Le calcul de ces valeurs revient à partitionner l'espace des variables opérationnelles en régions connexes, ou cellules, que l'on utilise alors pour construire un graphe de connexité discret traduisant les relations d'adjacence de ces cellules. La recherche d'une trajectoire pour le robot est alors ramenée à celle d'un chemin dans ce graphe.

Une méthode similaire de réduction de la complexité, dite de *rétraction*, consiste à rétracter l'espace des configurations valides sur un espace de dimension inférieure de telle sorte que deux configurations de l'espace libre appartiennent à la même composante connexe si et seulement si leurs rétractions appartiennent à la même composante connexe du sous-espace considéré [5-Boi87]. La dimension du problème est ainsi réduite récursivement et la résolution du cas monodimensionnel est équivalente à la recherche d'un chemin dans un graphe.

Le chapitre 6 traite de la conception et l'utilisation d'une technique de projection pour un robot manipulateur ayant un nombre quelconque de degrés de liberté.

3.3 Les différentes approches

Une première classification distingue deux types d'approches, dites globale (ou constructive) et locale (ou corrective) selon la nature de l'information prise en compte.

Seules les méthodes globales permettent d'élaborer une représentation explicite et complète des contraintes de l'environnement (généralement l'espace des configurations valides). Elles peuvent être subdivisées selon leur complexité et la manière dont celle-ci est maîtrisée. Ainsi les méthodes *directes* permettent, dans des cas particulièrement simples, de calculer exactement une représentation de l'espace libre. Sinon, l'emploi de méthodes de réduction, par *projection* ou par *rétraction*, permet d'obtenir une représentation approchée de l'espace libre.

Toutefois, les performances de ces méthodes restent généralement assez faibles et motivent de ce fait la recherche d'autres méthodes : les méthodes *locales* ou *stratégies correctives* dont le but est de déduire, sur la base d'informations locales, une direction instantanée de déplacement évitant toute collision, de telle sorte que le mouvement engendré par itération conduise globalement au but fixé.

3.3.1 Les méthodes directes

Afin de construire une représentation exacte de l'espace libre d'un robot, il est nécessaire que la dimension du problème reste faible (2 ou 3).

C'est le cas par exemple du problème du déplacement en translation d'un objet rigide en dimension 2 ou 3, convexe ou non (le cas non convexe se ramenant au précédent par décomposition de l'objet en éléments convexes). Nilsson a essayé le premier de construire une représentation de l'espace des configurations du robot pour un robot mobile en translation (SHAKEY), pour lequel était défini un "graphe de visibilité" construit sur les sommets des obstacles dans l'espace des configurations [6-Nil69].

Une variante classique du robot mobile est celle du "déménageur de piano" ("piano mover") [10-SS83*], dans lequel l'objet mobile (le piano) doit être déplacé dans une pièce encombrée d'obstacles polygonaux (les meubles). Les problèmes de déplacement en translation ont fait l'objet d'études approfondies pour lesquelles on dispose d'algorithmes efficaces [5-Sha87]. Les mouvements de rotation ajoutent une dimension au problème : ils ont été traités de diverses manières. Brooks modélise partiellement l'espace des configurations sous formes de cônes généralisés symbolisant un chemin de passage préférentiel, ou "freeway" [10-Bro83] : l'axe

d'un cône détermine les positions valides du mobile et les orientations possibles en un point dépendent de sa section. La méthode a été étendue au cas d'un manipulateur à quatre degrés de liberté manipulant des parallélépipèdes. Mais une caractérisation exacte de l'espace des configurations lorsque le mouvement combine translation et rotation nécessite une décomposition de l'approche.

3.3.2 Les méthodes de réduction

Méthodes par projection

La technique de la réduction de complexité par projection peut être appliquée de manière exacte au cas d'un segment de droite évoluant au milieu d'obstacles polygonaux [5-LS85] ou polyhédriques [10-SS83*]. Son utilisation de manière approchée semble inévitable dans les algorithmes de construction de l'espace libre d'un manipulateur.

Widdoes [10-Wid74] et Udupa [10-Udu77] ont éte les premiers à utiliser une technique de partitionnement de l'espace articulaire d'un robot manipulateur (limité à trois degrés de liberté). C'est ensuite Lozano-Pérez [12-LW79] qui a décrit le premier algorithme de recherche de trajectoires applicable à des systèmes généraux, en donnant une formalisation du problème dans l'espace des configurations. Différentes approches ont alors émergé : discrétisation en tranches d'orientation [11-BLP83], découpage en régions homogènes ou non ([12-Don84], [12-Ger85] et [12-Loz86]).

Méthodes par rétraction

Les deux types principaux de rétraction utilisés sont la rétraction sur le diagramme de Voronoï et celle sur le bord de l'espace libre.

O'Dulaing et Yap ont les premiers utilisés une méthode de rétraction sur le diagramme de Voronoï pour planifier les déplacements d'un disque dans un environnement polygonal ([5-ODY82] et [5-OSY82]). Une variante a été élaborée pour planifier les mouvements en translation et en rotation d'un segment de droite dans un environnement polygonal [5-LS85]. Le cas d'un objet polygonal convexe quelconque en translation a été traité en utilisant une rétraction associée à la métrique convexe définie par la forme du robot [5-For86].

Un exemple de rétraction sur le bord de l'espace libre a été donné par Yap pour planifier le mouvement coordonné de deux disques à l'intérieur d'un polygone

[13-Yap84]. Cet exemple a été étendu à la coordination de robots élémentaires représentés par un segment de droite et évoluant dans le plan en translation et en rotation [13-FWY86].

3.3.3 Les méthodes locales

Le principe de base des méthodes de type "génération et test" est du à Lewis [12-Lew74]. Difficile d'emploi dans le cas général, ces méthodes sont utilisées pour calculer des portions de trajectoires limitées en amplitude et en complexité, ou pour compléter ou modifier localement une trajectoire existante.

L'orientation des composants terminaux d'une structure articulée peut être déterminée de cette façon, en particulier le préhenseur ou même l'avant-bras du robot [10-Udu77]. Les heuristiques employées sont diverses : "glissement" d'un mobile le long de plans tangents aux obstacles constituant l'environnement local [10-Fav86] où de plusieurs mobiles le long d'hyper-plans séparateurs définissant leur frontière de contact [13-Tou86].

Les méthodes de type *potentiel* conduisent à représenter les contraintes spatiales formées par les obstacles par un champs de potentiel fictif, lequel, appliqué au mobile, engendre une force de répulsion éloignant celui-ci des objets. L'ajout d'un potentiel attractif permet alors de guider le déplacement vers une position particulière.

Introduite pour un manipulateur dans le contexte de la commande dynamique [14-Kha85], cette méthode a été utilisée pour planifier les trajectoires d'un disque dans un environnement bidimensionnel [14-Tho84], principe généralisé par la suite afin de prendre en compte les contraintes cinématiques du mouvement [14-Kro84]. Des variantes en ont été dérivées, séparant par exemple les contraintes d'anti-collision (répulsives) de celles de la tâche [12-FT87].

Méthodes hybrides

Effectuer un bilan des avantages et inconvénients des différentes méthodes existantes conduit fort logiquement à l'utilisation de *méthodes hybrides*, combinant des stratégies à la fois globales et locales, dans cet ordre.

Ainsi on peut utiliser une méthode de type "potentiel" pour chercher des trajectoires dans l'espace des configurations construit explicitement par une méthode globale [10-Don84].

3.4 Besoins en modélisation géométrique

3.4.1 Nécessité d'une modélisation

Le cerveau humain ne raisonne pas sur la réalité mais sur un modèle interne de cette réalité. Ce modèle est représenté dans le système nerveux à travers les neurones et les synapses. Cette faculté de construire des représentations est d'ailleurs pour J.P.Changeux une aptitude fondamentale de l'encéphale [6-Cha85]. C'est là en tous cas quelque chose dont on peut espérer doter concrètement un robot, bien plus facilement que ce sens commun cher à H.Dreyfus, cette intuition indissociable du processus de pensée et si difficile à cerner et définir.

L'existence de cette représentation est une conséquence directe de la nécessité d'un comportement "intelligent" incluant réflexion et prévision des actions. Pour J.L. Laurière [6-Lau88], "la représentation des objets concrets ou abstraits est au cœur de l'intelligence artificielle". Plus que cela encore, elle est au cœur de toute science en général et de la robotique en particulier. Traduire un problème posé sous la forme d'une représentation manipulable est une étape clé de sa compréhension. Et mieux encore, de sa résolution.

Il n'est pas inutile de bien positionner le problème de l'utilisation d'un modèle. Et pour prendre une définition d'école, rappelons que le modèle est à l'origine "*un objet réduit et maniable qui reproduit les propriétés d'un objet de plus grande dimension, architecture ou dispositif mécanique, pouvant être soumis à des mesures, des calculs, des tests physiques non applicables à l'original*".

De là, le terme a acquis une vaste portée méthodologique pour désigner toutes les figurations ou reproductions qui servent les buts de la connaissance. A l'opposé du sens platonicien, où le modèle était le paradigme, la forme idéale, il devient la réalisation concrète au lieu de l'idée réalisable.

Mais il arrive aussi que les deux approches se rejoignent. Et c'est bien le cas ici, alors que notre modèle est à la fois l'objet "concret", forme simplifiée sur laquelle on raisonne et expérimente, en même temps que la forme idéale affranchie des erreurs engendrées par la réalité de la manipulation.

Dans notre cas, le dispositif mécanique modélisé est la structure articulée du manipulateur (modèle géométrique du robot complété par celui des objets). Il est clair que concept de programmation hors-ligne implique - avant même que cette programmation puisse être effectuée - une résolution théorique du problème posé. En particulier, dans le cas de la recherche d'une trajectoire de déplacement,

tests et mesures ne peuvent être effectués sur le robot : d'où le besoin impératif d'un modèle pouvant en toute sécurité remplir ce rôle.

Le modèle sert à fixer les lois sur un objet bien structuré, favorisant la conception et l'expérimentation; il permet la validation des raisonnements effectués, des algorithmes utilisés et des heuristiques employées afin de résoudre le problème posé par la réalisation d'une tâche. Dans le contexte du système SHARP , le problème est celui de la commande de robots d'assemblage, et le modèle n'existe qu'à travers l'implantation informatique - décrite par ailleurs - qui en est faite. Il n'est qu'une transcription abstraite, mais contrôlée par la pensée logique et mathématique, d'une réalité concrète, empirique, dont l'étude directe se révèle impossible.

Il est donc essentiel de disposer d'une représentation de l'univers du robot à la fois précise en tant que modèle d'une réalité physique (en particuliers dans ses aspects géométriques) et pertinente au niveau des raisonnements qui seront effectués.

3.4.2 Modèle et représentations nécessaires

Nous distinguons deux niveaux de modélisation, respectivement situées dans l'espace de manipulation et l'espace de planification, lesquels sont définis par la nature des représentations qui leur sont associées.

Par l'intermédiaire des représentations au niveau de l'*espace de manipulation*, on désignera la description cartésienne d'un contexte robotique selon des notions géométriques et technologiques, mais aussi fonctionnelles et relationnelles, compatible avec les outils géométriques classiques. Ces représentations constituent donc le modèle du problème de planification et les principales données d'entrée du système.

Grâce aux représentations situées au niveau de l'*espace de planification*, on désignera la description d'un même contexte sous la forme de la réunion d'ensembles de configurations valides ou invalides, compatible avec les outils de planification (outils topologiques et décisionnels) que nous avons défini.

Loin d'être un modèle abstrait, la description de l'espace de manipulation constitue déjà une interprétation de la réalité de l'univers. Donnée de base du problème de planification de trajectoires, elle peut et doit receler les informations spécifiques de sa résolution. C'est à cette condition que l'on pourra ensuite obtenir une représentation pertinente pour la planification par l'application de

techniques de raisonnement adéquates. Enfin les trajectoires correspondant à la tâche demandée seront obtenues à partir de cette dernière par application des techniques de planification.

3.4.3 Outils de modélisation nécessaires

Les systèmes classiques

Les systèmes de modélisation sont généralement orientés vers la conception et la fabrication assistées par ordinateur (CAO et CFAO). En conséquence, leurs points forts sont essentiellement leurs interfaces d'entrée (création et édition d'objets conviviales), de sortie (simulation graphique de qualité), ainsi que parfois des outils de calculs spécialisés (trajectoires de machines-outils etc). Mais la structure même du modèle représentant les objets n'est pas réellement adaptée aux algorithmes de géométrie qui constituent l'essentiel des outils utilisés pour la planification en robotique.

On peut dire que les systèmes de modélisation classiques sont conçus pour le calcul géométrique et non pas pour le raisonnement géométrique. C'est pourquoi un système de planification de trajectoires, comme un système plus général de programmation automatique, comme d'ailleurs tout système robotique de haut niveau, a besoin de représentations *différentes* de celles disponibles dans les systèmes classiques.

Brève taxonomie des représentations CAO

Nous reprenons très brièvement ici une classification due à Requicha ([4-Req80] et [4-RV82]), en regroupant de plus les représentations selon les inconvénients qu'elles présentent relativement à nos besoins.

On distingue tout d'abord les représentations génériques qui offrent une description symbolique sous forme de classes d'objets, les représentations analytiques, décrivant rigoureusement les surfaces utilisées, et celles obtenues par balayage de cônes généralisés, trop complexes pour supporter le calcul de nos prédicats géométriques.

On trouve ensuite les représentations basées sur l'énumération de l'espace occupé, celles analogues décomposant les objets en cellules et enfin les représentations par les limites (ou frontières, ou encore enveloppes externes) des objets. Cette dernière contient toutes les informations géométriques nécessaires, sauf celles de

matière (du fait de l'absence de notion d'unité volumique). Aussi, la représentation volumique, basée sur une construction ensembliste incrémentale des objets et présentant l'avantage de conserver l'objet dans son aspect volumique tout en donnant accès aux paramètres de surfaces et de contours, nous paraît la mieux adaptée à nos besoins algorithmiques.

D'autre part, les objets modélisables sont souvent limités aux solides rigides. Or il est indispensable, dans le contexte de la robotique, de pouvoir modéliser des structures mécaniques mobiles et articulées, ainsi que les relations existant entre les objets de l'univers.

Caractéristiques de notre modèle

L'aspect essentiel du modèle réside dans la richesse et l'adéquation de ses représentations aux problèmes de planification rencontrés. Comme il semble impossible de trouver une représentation unique possédant suffisamment de propriétés reflétant la réalité physique et satisfaisant les besoins en modélisation des différents planificateurs d'un système robotique, nous avons opté pour un système à représentations multiples, hiérarchiques, aussi bien symboliques que numériques.

Nous avons ainsi défini une représentation géométrique hybride complétée d'aspects relationnels, physiques et fonctionnels, dont la fonction est essentiellement de fournir toutes les données nécessaires à la résolution des multiples tâches de la programmation automatique sous la forme la plus intéressante et la plus compatible possible. Diverses représentations dédiées en sont ensuite dérivées pour les besoins des planificateurs du système.

Le but global visé est d'établir un modèle de l'univers le plus riche possible de façon à faciliter autant que faire ce peut le travail des planificateurs. Si les techniques de résolution employées par ces différents systèmes sont suffisamment disjointes pour nécessiter leur séparation, il existe en revanche une intersection très importante au niveau des outils du raisonnement géométrique. C'est donc par soucis de concision et d'efficacité que nous tendons vers cette intégration.

Chapitre 4

Modélisation de l'espace de manipulation

4.1 Les différents aspects du modèle

La nature des représentations nécessaires dépend d'une part de celle des objets devant être modélisés et d'autre part de celle des multiples raisonnements mis en œuvre pour résoudre la tâche proposée.

Ce qui nous intéresse essentiellement dans la représentation des objets, c'est leur géométrie. Nous disposons donc d'une représentation *volumique* (cf 4.2.2) dont le couplage avec une représentation *surfacique* (4.2.3) définit une représentation *topologique* des objets. La composante volumique permet d'exhiber les propriétés d'encombrement spatial des objets dans la scène, la composante topologique illustre les spécifications morphologiques de chaque pièce, et la composante surfacique permet l'application de divers prédicats géométriques (interférence, collision, débattements etc).

L'ensemble de ces trois représentations définit un modèle géométrique structurel, auquel sont alors ajoutées des représentations spécifiques en fonction des besoins des modules de planification (comme par exemple la représentation sphérique utile pour la planification locale de trajectoires).

Ce modèle structurel n'est alors pas suffisant pour décrire les liaisons permanentes définissant la chaîne cinématique d'un robot manipulateur ou celles temporaires existant entre objets. C'est pourquoi nous disposons également d'une représentation *géométrique relationnelle*, répondant à ces besoins.

A ces représentations de base viennent s'ajouter une représentation graphique utilisée pour la simulation hors-ligne, une représentation des incertitudes sur la position des objets utilisée pour la vérification du programme produit par le

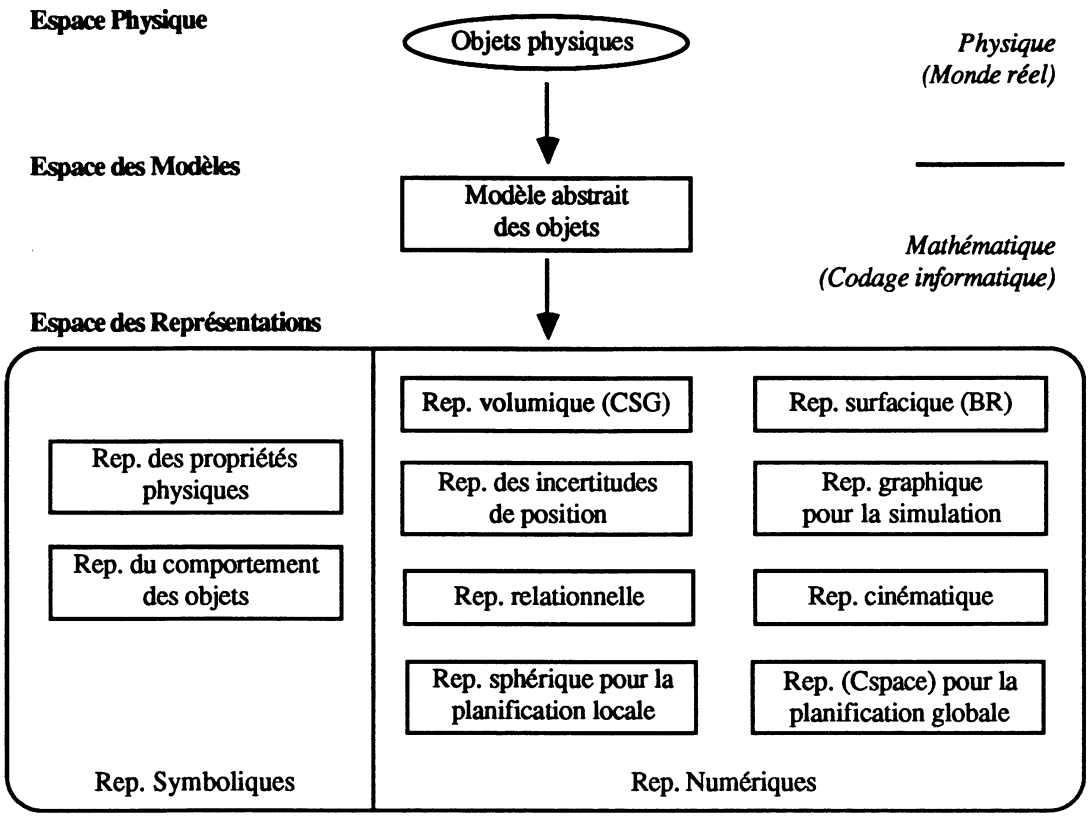

Figure 4.1: Modèle et représentations.

système. D'autres représentations peuvent également être créées par les différents planificateurs. C'est le cas par exemple de la représentation de l'espace libre du manipulateur calculée par notre système.

La modélisation des incertitudes semble devoir être désormais considéré comme un des problèmes-clés de la programmation automatique des robots. Les incertitudes de position peuvent provenir de l'environnement initial, du contrôle du robot et de la nature de ses mouvements, ou des mesures issues des capteurs. La détection et la maîtrise de ces erreurs est fondamentale pour une bonne exécution des mouvements planifiés. Dans le système SHARP , ces incertitudes sont considérées d'un point de vue global et traitées a l'aide de techniques de preuves de programmes (cf [3-PPT86] et [3-Pug88]), encore non implantées. En ce qui nous concerne, la prise en compte des incertitudes est donc effectuée au niveau de la construction de l'espace libre en sur-contraignant les approximations effectuées dans les calculs de distance aux obstacles.

La multiplicité des diverses représentations (en particulier géométriques) de notre modèle implique une très forte redondance. Celle-ci présente en tous cas l'avantage de faciliter sa mise à jour et d'offrir une grande flexibilité relativement aux problèmes algorithmiques. La philosophie sous-jacente à la réalisation de ce système de modélisation est résumée dans l'idée que toute information nécessaire à un instant donné doit être disponible de manière explicite dans le modèle". Par contre un tel niveau de redondance implique une constante vérification de la validité du modèle et de la cohérence des diverses représentations.

4.2 Modèle géométrique structurel

Propriétés du modèle

C'est donc sous le terme de représentation géométrique que l'on désigne habituellement la représentation d'objets réels du point de vue de leur structure géométrique. Le but visé est de fournir une description utilisable des formes et dimensions de chaque objet, ainsi que de sa position spatiale.

Le but de la modélisation est de fournir une représentation des objets la plus intéressante possible. Les représentations utilisées doivent donc présenter certaines propriétés. Ainsi, les objets étant des solides, leur représentation doit respecter un certain nombre de conditions inhérentes au fait qu'il s'agit d'objets fabriqués. La *validité* du modèle structurel est assurée par l'unicité de l'objet correspondant à tout modèle constructible, et sa *puissance* est définie par l'ensemble des objets modélisables. Enfin son *ouverture* est caractérisée par la diversité des grandeurs géométriques qu'elle permet d'appréhender, les facilités algorithmiques offertes par sa structure interne.

D'autre part, les opérateurs appliqués à une représentation doivent préserver ses propriétés. En particulier, leur *cohérence* assure la conservation des propriétés de solide de l'objet modélisé, l'homogénéité et la concision des données.

Hiérarchies du modèle

La première hiérarchie est dite *hiérarchie spatiale* des représentations. Parfaitement naturelle, elle exprime l'ordre dans lequel les différentes représentations géométriques sont déduites les unes des autres. Ainsi, la modélisation des objets reposant sur le concept d'assemblage virtuel d'entités volumiques, la représentation du même nom peut être considérée comme celle de base. Les représentations

topologiques et surfaciques en sont immédiatement dérivées, puis les représentations particulières d'intérêt non général (sphérique par exemple) et annexes (graphique, incertitudes).

Chaque objet de l'univers est ainsi modélisé avec la meilleure précision possible. Il est certain que cette seule représentation de l'objet suffit à nos besoins algorithmiques mais, afin d'améliorer les performances du système de planification de trajectoires, nous calculons pour chaque objet un ensemble de représentations approchées dérivées de celle d'origine, constituant une hiérarchie de solides englobants, et pour le manipulateur une deuxième hiérarchie englobante correspondant au volume atteignable par chacun de ses composants.

La *hiérarchie englobante* (en "profondeur") définit donc pour chaque objet une échelle de niveaux à précision variable, dont chaque état représente un *volume englobant* de l'objet. On trouve au sommet de l'échelle l'objet tel qu'il a été modélisé, puis une cascade d'objets "dégradés" dont la géométrie est de plus en plus simple et le volume de plus en plus gros, jusqu'à arriver à l'approximation la plus grossière possible (parallélépipède, sphère etc).

La *hiérarchie de balayage* (en "largeur") définit pour chaque structure articulée une échelle à mobilité variable, dont chaque état représente un *volume balayé* de l'objet en figeant plus ou moins de degrés de liberté. Là encore, on trouve au sommet la structure initiale, puis une cascade de structures au nombre décroissant de degrés de liberté et dont la géométrie est de plus en plus simple, jusqu'à arriver à l'approximation la plus contraignante d'un objet fixe dont le volume est la sphère de déplacement du mobile.

En pratique, chaque objet correspondant à un volume englobant ou à un volume balayé est traité comme un nouvel objet venant au moment opportun se substituer dans la scène à l'original dont il est issu. Leur représentation est donc en tout point similaire à celle des objets "normaux".

Ce qui suit est donc une description des différentes représentations utilisées dans notre système. Elles ne sont évidemment pas indépendantes : l'aspect surfacique est contraint par la représentation volumique choisie, et la structure topologique des objets n'est qu'une conséquence de l'emploi de ces deux représentations. De même la représentation sphérique, les notions de volumes englobants et de volumes balayés ne sont qu'une utilisation particulière de la représentation volumique...

4.2.1 La représentation topologique

La composante topologique décrit les relations entre les différentes entités constituant un objet. Elle est utilisée par tous les raisonnements qui mettent en jeu des contacts (détection ou contrôle). Elle combine une structuration spatiale de ces entités géométriques avec une caractérisation à tous les niveaux des positions relatives entité-matière.

Les divers éléments d'un objet sont structurés suivant une hiérarchie

$$objet \rightsquigarrow composant \rightsquigarrow face \rightsquigarrow contour \rightsquigarrow point$$

choisie de manière à pouvoir parcourir la description d'un solide de manière optimale. Ceci peut être représenté par un graphe dont les nœuds sont les entités géométriques et les arcs dénotent une relation de décomposition ou d'appartenance.

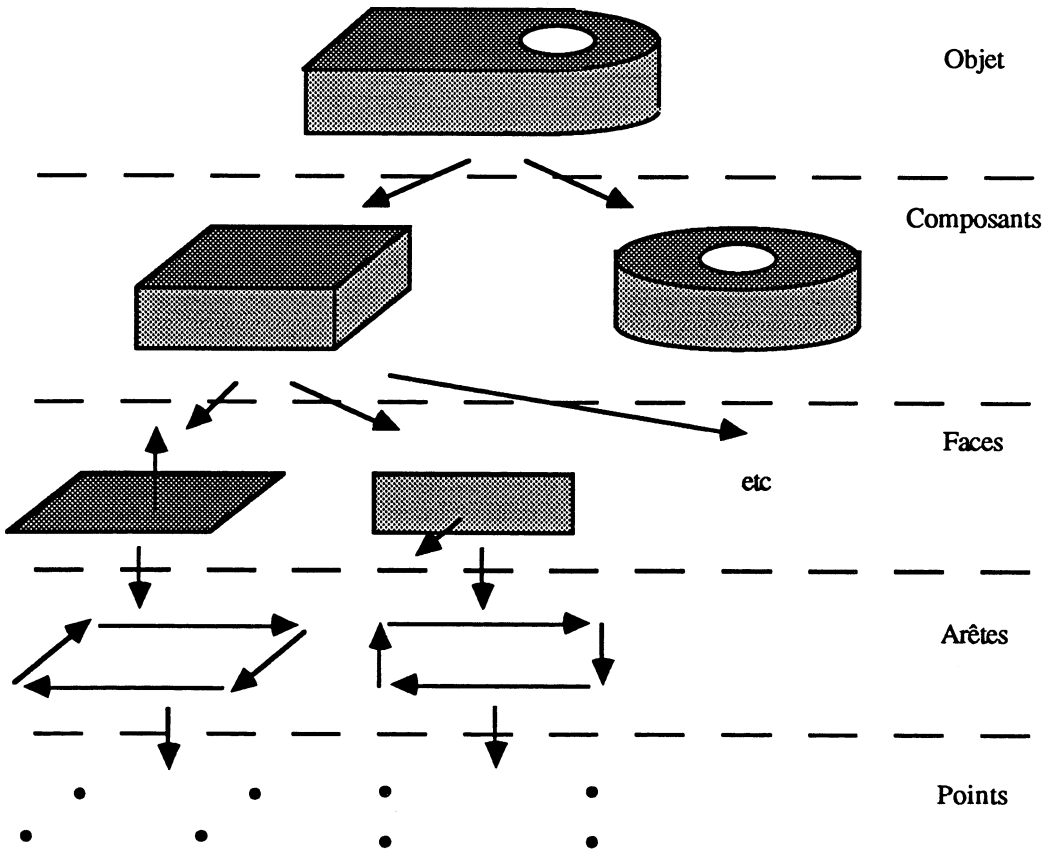

Figure 4.2: Structure topologique d'un objet.

Ces relations de décomposition et d'appartenance sont complétées par des chaînages de type "winged- edge" [4-Cam81] qui permettent de représenter ex-

plicitement les relations de voisinage dans le modèle. Cette caractérisation de la *topologie locale* est très utile pour les raisonnements géométriques mis en œuvre.

La présence ou l'absence de matière est donc référencée à chaque niveau de la structure topologique d'un objet. Ainsi, les composants sont "marqués" vide ou plein. Leurs faces sont orientées par rapport à la matière grâce à la spécification de leur normale extérieure (dirigée du plein vers le vide). Enfin, les contours sont également orientés avec la règle de Mœbius, dite également convention "matière à droite" (c'est-à-dire dans le sens rétrograde autour de la normale extérieure à la face d'appartenance). Cette orientation artificielle du sens de parcours des contours permet alors de distinguer localement intérieur et extérieur, vide et matière (propriété qui est utilisée par exemple par les prédicats géométriques de calcul d'interférence et de collision).

4.2.2 La représentation volumique

Description générale

Il s'agit d'une représentation par arbre de construction (de type CSG, pour "Constructive Solid Geometry", cf [4-RV82] et [4-Bro82]). On trouve aux feuilles de l'arbre des objets primitifs paramétrables, ou *composants*, sur les arcs des opérations, aux nœuds des composants intermédiaires et enfin au sommet l'*objet* modélisé. Ce type de représentation présente l'avantage de faciliter la définition des objets, ainsi que leur manipulation en tant que solides physiques.

Les volumes de bases utilisés sont prédéfinis dans le système et paramétrables. Pour des raisons de simplicité (voire de faisabilité) des calculs géométriques, nous n'avons retenu que les parallélépipèdes, les cylindres, les sphères et les cônes, auxquels viennent s'ajouter une certaine classe de polyhèdres.

La construction des volumes est effectuée à l'aide d'opérations booléennes (union, intersection et différence) sur ces objets élémentaires. En pratique, seule l'union a été retenue, là encore pour des raisons de simplicité. En effet les prédicats géométriques, lorsqu'ils sont appliqués à un objet, parcourent récursivement l'arbre de sa structure, opèrent sur les composants élémentaires trouvés à la base et propagent ensuite l'information obtenue en remontant la structure jusqu'à l'objet final. Or pour la majorité des prédicats, on ne sait pas dans le cas d'une intersection ou d'une différence d'objets comment effectuer cette propagation, comment combiner l'information issue des branches inférieures.

Cette limitation sur le choix des opérateurs de composition est trop contraignante, surtout dans le contexte de l'assemblage où la nécessité d'un opérateur de perçage est impérative. Aussi nous avons du ajouter aux primitives volumiques déjà citées les mêmes primitives, cette fois percées.

La classe de polyhèdres utilisés dans le système est celle des volumes constuits par extrusion, c'est-à-dire à partir d'un contour défini dans un plan et augmenté dans la troisième dimension par balayage ou révolution (partielle ou complète) autour d'un axe.

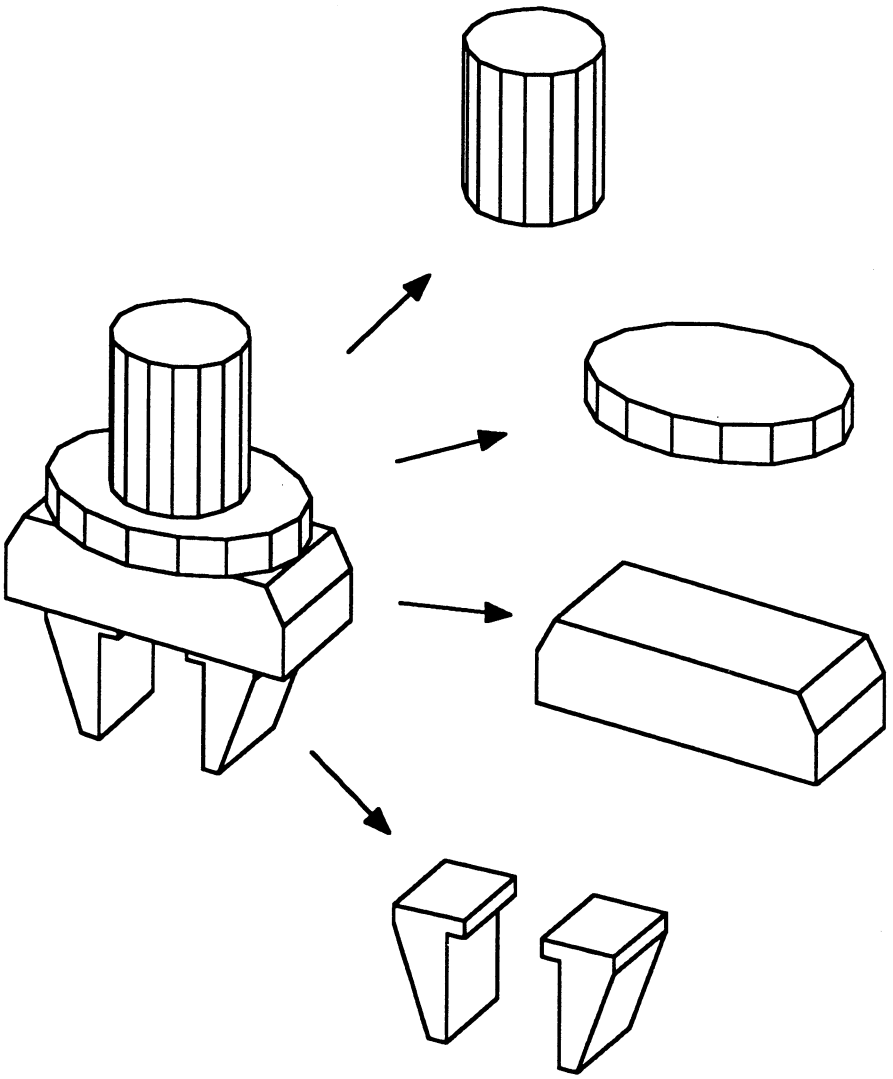

Figure 4.3: Décomposition en composants.

Afin de conserver une représentation homogène des objets vis-à-vis de certaines fonctions géométriques (calculs de "grossissement" en particulier, visualisation), le système associe une approximation polyédrique à toutes les primitives à surfaces courbes.

Les volumes englobants

La composante volumique est utilisée pour réduire la complexité du raisonnement spatial, chaque fois que les contraintes imposées par la tâche le permettent. Nous avons donc défini une hiérarchie de volumes englobants, dans laquelle chaque nœud représente un parallélépipède, et chaque feuille correspond à un composant volumique élémentaire du système (prisme, cylindre, cône, sphère ou polyèdre). Cette structure est illustrée par la figure 4.4.

Les volumes balayés

La hiérarchie de balayage définit pour une structure articulée un ensemble de volumes obtenus en déplaçant l'un de ses composants dans l'intervalle des valeurs possibles. L'encombrement de l'objet obtenu est alors équivalent au volume maximal occupé par l'élément lors de son déplacement. Il est possible de construire alors pour ce dernier une nouvelle hiérarchie de volumes englobants correspondant alors à un *manipulateur équivalent* [12-FT87].

Dans la pratique, les composants constituant les divers éléments du manipulateur équivalent sont obtenus par balayage à partir de ceux du manipulateur initial : ils peuvent donc être représentés (au besoin de manière approchée) par un composant par balayage.

La figure 4.5 montre deux exemples de volumes balayés correspondant à deux degrés de liberté de la pince du robot. Dans cet exemple, le composant obtenu par balayage des mors de la pince est un tronc de cône percé d'un cylindre. Les composants obtenus peuvent à leur tour être approximés par des volumes englobants. En ce qui concerne la pince, le volume total obtenu pour un balayage exhaustif de ses degrés de liberté est une sphère.

Décomposition structurelle d'un objet

La décomposition d'un objet en primitives élémentaires reste encore entièrement à la charge de l'opérateur. En conséquence, elle n'est pas unique et, bien

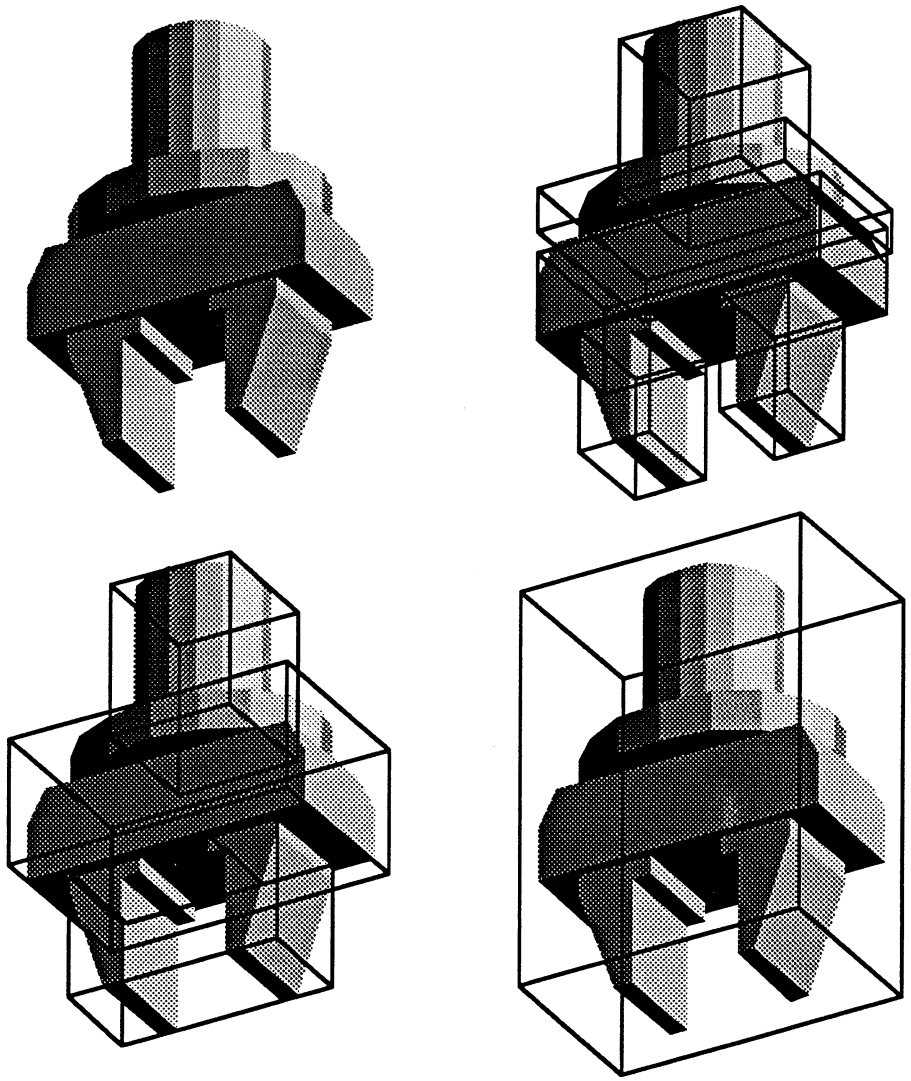

Figure 4.4: Exemple de hiérarchie de volumes englobants.

souvent, le choix du découpage est totalement arbitraire, sans que l'on puisse en apprécier la "valeur".

Bien sûr on peut essayer de dégager quelques principes de base suivis consciemment ou non par l'opérateur dans sa démarche. Par exemple, un bon critère a priori semble être la minimisation du nombre de primitives volumiques utilisées, ceci afin de ne pas surcharger la base de données du système. Mais cela implique alors que chacun de ces volumes soit plus complexe (typiquement des polyhèdres plus "découpés"), puisque de toute façon le nombre total de contours est une constante incontournable de la géométrie de l'objet. Or, la complexité intrinsèque

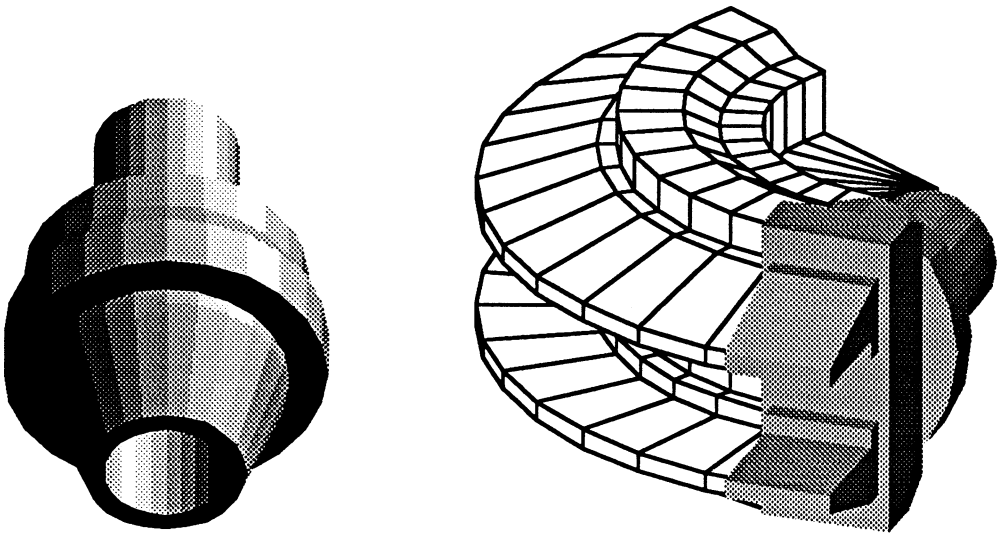

Figure 4.5: Exemples de volumes balayés.

des objets de base constitue un facteur de performance du système difficilement évaluable mais pourtant fort important. Un choix se pose entre créer des arborescences importantes comprenant beaucoup de primitives simples ou d'autres plus épurées intégrant des primitives plus élaborées. La majorité du temps de calcul étant consacré à l'évaluation des prédicats géométriques (opérant au niveau des composants de base), il nous semble préférable de se limiter à des formes simples. C'est d'ailleurs dans cet esprit que nous nous sommes contentés de primitives aussi élémentaires que le cylindre ou le cône...L'inconvénient de la méthode étant alors d'offrir un choix d'"outils" restreint et de multiplier leur emploi afin d'obtenir la même précision de modélisation.

4.2.3 La représentation surfacique

Description générale

La représentation surfacique d'un objet décrit l'ensemble des faces constituant l'objet, lesquelles sont définies par leurs surfaces de support et leurs contours, ou/et par leurs équations.

La nature des surfaces et des contours utilisables est directement liée à celle des calculs géométriques qui devront être effectués. Les surfaces complexes utilisées classiquement en modélisation CAO (approximées par des fonctions polynomiales, courbes et surfaces de Bézier, splines diverses etc) ne peuvent donc pas être

employées ici. La faisabilité et l'efficacité de nos calculs implique de se restreindre à l'emploi de surfaces simples (plans, quadriques).

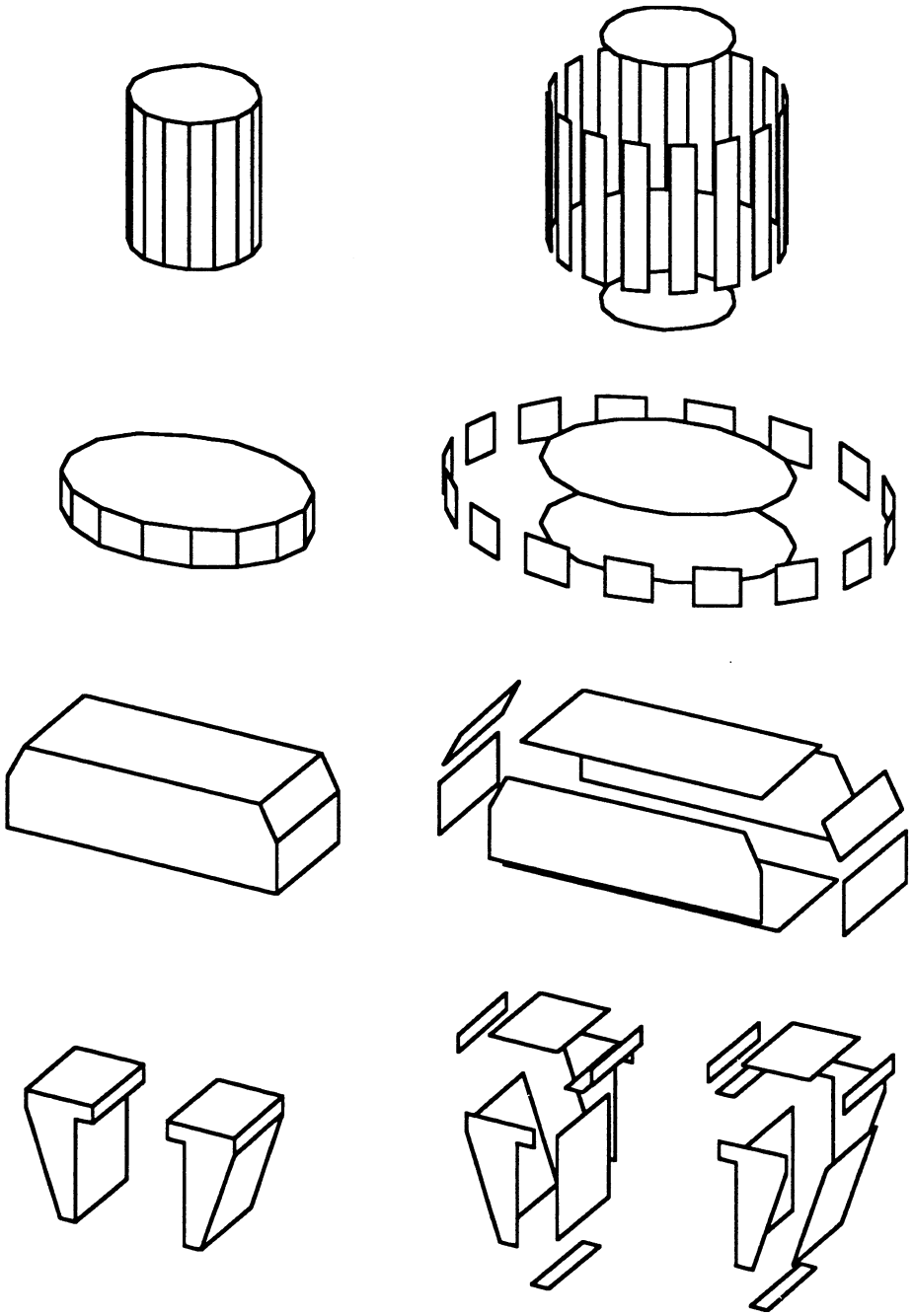

Figure 4.6: Décomposition de l'objet en faces.

L'approche mixte que nous avons retenu définit les surfaces comme limite géométrique des volumes utilisés dans la création d'un solide. Etant donné les limitations imposées sur les primitives de base, les seules surfaces possibles sont planes, circulaires, côniques ou sphériques.

Détermination des contours "vrais"

La représentation surfacique utilisée est donc directement dérivée de la représentation volumique. L'avantage pratique immédiat est que l'opérateur n'a pas à spécifier - numériquement - les contours des objets. Néanmoins cela a du être fait - de manière paramétrée - pour chacune des primitives volumiques de base. L'ensemble des faces, respectivement des contours, constituant la représentation surfacique, respectivement linéique, de l'objet est obtenu comme l'union des ensembles correspondants pour chacune des primitives formant l'objet final. Sous-entendu, l'union exhaustive. Cela vient évidemment du fait que l'ensemble des primitives constituant l'objet a précisément été défini de cette manière. L'information "volumique" présente dans la base de faits géométriques (la description des solides de l'univers) est donc redondante. Des faces se recouvrent partiellement ou totalement (cas de primitives accolées) ou sont incluses dans un volume plein (cas du recouvrement de primitives). Il en est de même de leurs contours. En pratique, cela arrive d'autant plus fréquemment que la forme de l'objet est complexe.

Il peut donc être souhaitable de voir ces problèmes détectés par le système et la représentation surfacique mise réellement à jour. Ce qui pose le problème de *l'évaluation des frontières*, où une frontière désigne de façon générale une surface ou une arête d'un solide [4-Pas85]. Seules doivent être conservées les frontières propres d'une primitive (celles réellement présente dans la primitive même et conservées dans l'objet final) et ses frontières mixtes (celles obtenues par interférence de primitives entre elles).

L'intérêt premier de cette opération est donc d'obtenir une représentation des frontières qui soit exacte et non redondante. De plus, en agissant de la sorte, on s'affranchit également du découpage arbitraire issu de la descrition volumique initiale et on rend la représentation surfacique indépendante de toute interprétation extérieure.

Nous avons développé par ailleurs une méthode d'évaluation des frontières adaptée à l'utilisation de primitives polyhédriques, dont l'intégration au système de modélisation présenté ici n'a pour l'instant pas été effectuée.

Prise en compte des incertitudes de forme

Il s'agit là d'un domaine, le *tolérancement* qui commence tout juste à être abordé dans les systèmes de CAO ([3-Req83] et [3-GSZ88]) et qui n'existe pas par ailleurs. Il peut sembler paradoxal d'avoir négligé jusqu'alors cet aspect de la modélisation, puisque cette source d'erreur potentielle se situe en amont de tout traitement informatique. Cependant, force nous est d'admettre que si cette précision est fondamentale dans la fabrication même des pièces mécaniques (la précision des machines étant adaptée), elle l'est beaucoup moins au niveau de leur manipulation, sauf dans quelques cas d'assemblage particulièrement "fins".

Du point de vue de l'assemblage robotique, ou plutôt "robotisé", on constate que les incertitudes en question sont d'un ordre de grandeur beaucoup plus faible que celles sur le placement du manipulateur et surtout que celles sur la position des objets. Leur prise en compte au niveau de la partie modélisation d'un système de programmation automatique de robots d'assemblage nous semble donc pour l'instant peu nécessaire.

La représentation graphique

Cette représentation est directement héritée de la représentation surfacique : un objet est alors vu comme un ensemble de faces polygonales (les faces non planes étant décomposées en facettes polygonales), chacunes référencées par rapport au repère local de son composant d'appartenance. Cette décomposition oublie complètement la notion d'objet et n'est utilisée que pour les opérations de visualisation de la scène et la simulation des opérations d'assemblage.

4.2.4 La représentation sphérique

La sphère fait partie des primitives de base utilisables dans la représentation volumique, mais elle reste en général réservée à des cas bien particuliers (et assez rares). Il s'agit pourtant d'un objet extrêmement intéressant par ses propriétés géométriques, que nous avons utilisé de manière systématique, présentant de multiples avantages:

- *Génération automatique* à partir d'un modèle surfacique lui même dérivé d'un modèle volumique tel celui que nous avons adopté;
- *Simplicité* du système qui ne comporte plus qu'une seule primitive volumique de base;

- *Propriétés* topologiques et géométriques intéressantes.

La sphère est l'objet géométrique de volume non nul dont la représentation paramétrique est minimale (un point, une distance) et dont la symétrie est maximale. Elle est invariante par rotation, sa projection est constante, sous forme de disque (la correspondance 2D-3D est donc élémentaire). La détermination des contours obtenus par intersection ne pose aucun problème (deux sphères s'intersectant suivant un cercle). En ce qui concerne les calculs géométriques qui nous sont nécessaires, les tests d'intersection de solides se ramènent à des comparaisons de distance (dont le calcul est immédiat), les calculs de débattements sont facilités, enfin beaucoup d'opérations de grossissement sont triviales...

Nous déterminons donc une représentation à base de sphères à partir du modèle volumique dont nous disposons (en l'occurence à partir des primitives volumiques). L'algorithme adopté effectue la décomposition d'un objet tridimensionnel polyhédrique en un ensemble de sphères [4-ORB79]. Ceci implique que les primitives non polyhédriques du modèle doivent être auparavant remplacées par une approximation polyhédrique (ce qui revient à "facétiser" cylindres et cônes). La perte de précision introduite s'ajoute à celle de la décomposition sphérique.

La précision de cette décomposition des objets en sphères est précisément le problème majeur de la méthode, influant directement sur le nombre et la taille des sphères obtenues. Etant donné l'utilisation qui en est faite (cf 8.2.2), nous avons pour notre part contraint la représentation à des sphères d'égal rayon. La hiérarchie englobante relative aux sphères est alors directement fonction de ce rayon, et le seul choix oscille aux extrêmes entre un très grand nombre de sphères modélisant assez bien l'objet ou un petit nombre ne donnant qu'une approximation grossière des objets.

4.3 Modèle géométrique relationnel

4.3.1 Structure relationnelle

Les repères

A chaque objet de l'univers du robot doit être associé un repère cartésien, dit *repère local*, situant à la fois sa position et son orientation par rapport à un référentiel absolu, dit *repère base*. La donnée de la transformation géométrique entre ces deux repères suffit alors à spécifier la position de l'objet.

Les repères constituent des structures de données à part entière, indépendantes et flexibles. Elles contiennent trois sortes d'information : le repère cartésien associé, la liste des objets dont elles référencent la position et l'orientation, et la liste des repères qui leur sont liés.

En fait de repère cartésien, l'information numérique retenue est la transformation géométrique de référence par rapport au repère station. Chaque objet pointe vers un repère, qui le référence en tant que tel, et les relations entre objets sont donc exprimées à travers les relations entre les repères associés. Parmi les diverses représentations possibles des transformations, nous avons opté pour celle de couples translations-quaternions, dont une description détaillée est donnée en annexe A.

Les relations orientées exprimant une liaison entre repères sont fixes ou paramétrées. Le premier cas correspond à une liaison rigide pour laquelle la transformation entre les deux repères est constante; le second cas est celui d'une liaison articulaire. Les relations ainsi modélisées peuvent être créées, modifiées ou détruites à tout moment.

La structure relationnelle ainsi représentée est un graphe orienté connexe, sans cycle, dont la racine est le repère base, les nœuds tous les autres repères, et dans lequel chaque arc dénote une relation particulière entre les deux objets caractérisés par les repères associés.

Objets rigides et structures articulées

Tous les composants d'un objet sont positionnés relativement à son repère propre, et les repères associés sont donc liés à ce dernier. Un objet constitue donc du point de vue relationnel une arborescence dont le repère constitue un point d'accès privilégié.

Les liaisons entre repères sont rigides et permanentes dans le cas des composants d'un objet physique de l'univers (constituant un tout inséparable). Elles sont rigides mais temporaires dans le cas du groupement d'objets rendus solidaires par l'application d'une force extérieure (force de serrage d'une pince exercée le temps d'un déplacement, force gravitationnelle de superposition etc). Ces liaisons peuvent éventuellement devenir permanentes, par exemple dans le cas d'un assemblage d'objets.

Les liaisons sont paramétrées (et permanentes) dans le cas de structures relationnelles modélisant les relations spatiales évolutives qui caractérisent par

exemple une structure articulée. Dans ce cas la transformation associée dépend de la valeur des degrés de liberté existant entre les éléments concernés. Dans le cas de la chaîne cinématique d'un manipulateur, les paramètres se limitent à la valeur de la variable articulaire définissant la position relative des composants.

4.3.2 Représentation cinématique

Le robot est représenté par une *boucle cinématique ouverte*, constituée de N composants liés entre eux par des articulations. Deux types d'articulations de base sont utilisés pour cette description: l'articulation *rotoïde* qui permet des déplacements en rotation autour d'un axe fixe, et l'articulation *prismatique* qui autorise des mouvements de translation dans une direction donnée.

Les articulations simples, très répandues, se composent d'une seule articulation indépendante. Mais il existe d'autres articulations plus complexes, qui peuvent également être décrites à l'aide de ces deux types de liaisons élémentaires (liaison sphérique ou rotule, structures à articulations dépendantes poly-prismatiques et poly-rotoïdes). Il existe également des structures à articulation couplées (par exemple la boucle cinématique de type parallélogramme). Une description exhaustive des types d'articulations et de leur représentation relationnelle peut être trouvée dans [4-Bo*83] et [4-Per84].

Les méthodes que nous avons développé pour la planification de trajectoires ne sont valides que pour un manipulateur à articulations indépendantes, c'est-à-dire - heureusement - la majorité d'entre eux.

4.4 Représentation symbolique des états de l'univers

4.4.1 Rêgles d'évolution de l'univers

L'ensemble des données symboliques et numériques présentes dans le modèle donne une description instantanée de l'état de l'univers. Cette description évolue au cours du temps en fonction des actions exécutées par le robot ou éventuellement par l'opérateur.

Les diverses représentations décrites précédemment permettent de décrire de manière assez complète un état donné de l'univers, mais ne contiennent pas encore toutes les informations de nature géométrique nécessaire à la résolution du

problème de planification de trajectoires. De plus elles n'offrent aucun contrôle de la validité de l'état courant de l'univers et ne permettent pas d'en gérer l'évolution.

Nous avons par conséquent besoin d'une part d'une *base de faits géométriques* (BFG) pouvant recenser les informations pertinentes qui font actuellement défaut, et d'autre part d'un mécanisme permettant l'accès à cette base de données symboliques, ainsi que sa mise à jour.

La BFG est essentiellement constituée des diverses représentations du système. Celles-ci recouvrent en quelque sorte l'aspect passif de l'information (géométrie des objets, description physique etc) plus l'aspect relationnel évolutif, auxquels on peut alors ajouter une information active (relations spatiales, contacts etc).

Les fonctions de calcul géométrique utilisées (en particulier pour la construction de l'espace libre) deviennent des prédicats de la BFG et doivent être complétés d'un ensemble de prédicats d'accès. L'activation automatique de certains prédicats afin d'effectuer la mise à jour de la BFG conduit alors à définir des règles d'évolution de l'univers.

4.4.2 Exemples de prédicats et de règles

On peut distinguer prédicats et règles selon le niveau d'information manipulée.

Les prédicats géométriques de plus bas niveaux sont ceux - très nombreux - qui permettent d'accéder directement à l'information stockée dans la BFG : nature et caractéristiques des entités géométriques composant la scène, position des objets... Exemple :

$$\text{Position}(\mathcal{O}) := \text{Transformation}(\text{ Repère }(\text{ Identificateur}(\ \mathcal{O}\ )))$$

Les règles de plus bas niveaux sont par exemple celles qui assurent le maintient de la cohérence des liens entre les divers repères de l'arborescence relationnelle :

$$\text{Transitivité : si } R_0 \text{ est lié à } R_1 \text{ et } R_1 \text{ à } R_2 \text{ alors } R_1 \text{ est lié à } R_2$$

$$\text{Consistance : si } R_0 \text{ est lié à } R_1 \text{ et à } R_2 \text{ alors } R_1 \text{ est lié à } R_2$$

L'action nécessaire et suffisante à l'activation de ces règles est donc l'ajout d'un lien entre deux repères. En l'occurence, les conditions sont exprimées par la nature des liens déjà existants et les actions à entreprendre constituent la mise à jour (ici additive) de l'arbre de liaison.

A l'opposé, les prédicats géométriques de plus haut niveau sont par exemple les fonctions de détection d'interférence entre objets (*INTER*) ou de calcul de débattement articulaire d'un composant du manipulateur (*DEBAT*), définies au chapitre 6. Il est clair que le résultat de ce genre de fonction ne peut être systématiquement stocké dans la base de faits : aussi leur appel entraîne l'exécution des calculs géométriques nécessaires. De même, pour savoir si une configuration ς du manipulateur est valide ou non, on peut bien sûr faire appel au prédicat de calcul correspondant (*VALID*). Mais cela n'est pas indispensable : il suffit en fait d'aller lire l'information associée contenue dans la BFG, en l'occurence dans la représentation de l'espace des configurations valides, l'espace libre EL (à condition bien sûr que celle-ci soit maintenue à jour) :

$$\mathsf{VALID}(\, cf \,) := \bigcap_{n,p} INTER(composant_n(\varsigma_n),\ objet_p)$$

$$\mathsf{VALID}(\, cf \,) := [\, ?\, \varsigma \in EL \,]$$

Enfin les règles d'évolution de plus haut niveau sont celles qui contrôlent la mise à jour de représentations comme celle de l'espace des configurations. En l'occurence, la condition d'activation est l'ajout, le déplacement ou la suppression d'un objet dans la scène et le prédicat géométrique appelé pour réaliser la mise à jour est la fonction de calcul de l'espace libre (*LIBRE*) également définie au chapitre 6.

Les règles de contrôle limitent l'application de certains prédicats en fonction du résultat d'autres prédicats interrogés à dessein. Par exemple le déplacement du manipulateur d'une position à une autre ne sera autorisé qu'après vérification de la validité de la trajectoire demandée, ou même après recherche d'une trajectoire valide si celle-ci n'a pas été spécifiée.

4.4.3 Implantation des règles d'évolution

L'univers physique est donc régi par des lois qui sont modélisées par des règles. Celles-ci, associées aux états caractéristiques de leur application, ne sont pas activées de manière systématique mais seulement lorsqu'elles doivent l'être. Elles sont de plus affectées d'une priorité d'exécution, et certaines sont de priorité maximale : les démons. Il s'agit donc de processus indépendants devant s'activer au moment opportun afin de modifier la BFG (ajout, modification ou suppression de données) et d'en préserver la cohérence. Plus spécifiquement, un démon est spécifié par un triplet de paramètres :

Conditions d'activation - Descripteur de situation - Actions à entreprendre

Les *conditions d'activation* sont les évènements devant entraîner impérativement le déclenchement de la règle. Leur gestion est implantée sous forme de *moniteur d'activation* chargé de sélectionner les règles activables à un instant donné. Il opère par mise en correspondance des expressions d'activation avec les expressions des faits de la base. Ce principe de mise en correspondance est très largement inspiré des travaux de Winston [6-Win77]. Une description détaillée du mode opératoire de ces moniteurs pourra être trouvée dans [2-The88].

Les *descripteurs de situation* définissent les états de l'univers ou les transitions entre états qui rendent les règles applicables. Par exemple, la modification d'une position spatiale, caractéristique d'un mouvement, entrainera l'activation des règles vérifiant la validité de la nouvelle position, puis de la trajectoire empruntée.

Enfin les *actions* à entreprendre peuvent être spécifiées explicitement par l'appel de prédicats géométriques ou bien appeler d'autres règles.

L'implantation de règles de contrôle régissant l'évolution de l'univers vise à automatiser la gestion de la cellule d'assemblage. Elles doivent rendre implicites les opérations qui sont encore explicitement à la charge de l'opérateur (typiquement le contrôle de validité de la position d'un objet, la mise à jour de l'espace libre etc).

Chapitre 5

Modélisation de l'espace de planification

5.1 Espace de planification

5.1.1 Notions et définitions

Espace de planification

Les données du problème de planification de trajectoires sont donc disponibles sous la forme de représentations géométriques mal adaptées à sa résolution. Nous définissons donc l'*espace de planification* E comme la représentation sur laquelle notre système doit raisonner afin de planifier une ou plusieurs trajectoires.

Plus généralement, cet espace représente *l'ensemble des états du robot qu'il est possible de réaliser avec les commandes du système, et de différencier à l'aide des moyens perceptifs disponibles* [1-Lau87]. En pratique, seules nous intéressent ici les informations sensorielles de nature proprioceptive permettant de mesurer la dynamique du robot (position, vitesse, éventuellement accélération). Pour un robot à N degrés de liberté, l'espace de planification peut donc s'exprimer par un ensemble de $3N$-uplet de coordonnées généralisées :

$$E = \{(\vartheta_1, \ldots, \vartheta_N,\ \vartheta'_1, \ldots, \vartheta'_N,\ \vartheta''_1, \ldots, \vartheta''_N)\}$$

Espace des configurations

Nous considérons à tout instant le robot comme statique. Ceci a pour effet de partitionner les états du manipulateur en classes d'équivalence telles que sont associés tous les états atteignables de même position mais de vitesse ou d'accélération distinctes. Nous appelerons une telle classe d'états une configura-

tion, notée ς de manière générique.

Une *configuration* ς est donc définie comme un N-uplet de variables généralisées (ou valeurs articulaires) de la structure cinématique $(\vartheta_1 \dots \vartheta_N)$.

L'ensemble de toutes les classes d'états équivalents est donc appelé *espace des configurations* et noté EC.

$$EC = \{(\vartheta_1, \dots, \vartheta_N)\}$$

Espace libre

L'espace des configurations peut être clairement partitionné en deux sous espaces non connexes. Le premier, que nous appellerons *espace de contact* EK, est l'ensemble de toutes les configurations du manipulateur qui impliquent une intersection matérielle entre deux objets de l'environnement, évènement alors nommé *collision.*

Nous parlons ici d'*objets* au sens général, incluant par conséquent les composants mêmes du robots, les pièces transportées, les meubles et murs de l'atelier. Si la collision la plus fréquente est celle d'un élément de la structure articulée avec un volume solide présent dans l'univers, nous devons également inclure celle, plus délicate, entre un solide fixe et un autre transporté par le robot. L'espace de contact est à son tour partitionné en un nombre fini de sous-espaces connexes correspondant à chacun des obstacles de l'environnement.

Le complémentaire de l'espace EK par rapport à l'espace EC définit donc l'ensemble des configurations valides du manipulateur, soit celles qui n'engendrent aucune collision : nous le nommerons donc *espace libre*, EL. Le but de la construction de l'espace de planification est précisément de déterminer l'espace libre du manipulateur.

Enfin la frontière entre l'espace de contact et l'espace libre constitue la *frontière de contact* EF, formée d'un ensemble de surfaces (de dimension $N-1$) usuellement appelées "C-surfaces".

$$EK \cup EL = EC, \quad EK \cap EL = \emptyset, \quad et \quad \overline{EK \cap EL} = EF$$

Le paragraphe 5.2 présente formellement l'espace des configurations.

5.1.2 Construction de l'espace de planification

Problème

Le problème de la construction de l'espace libre est équivalent à celle de l'espace de contact, celui-ci traduisant les contraintes imposées par l'environnement. Cette construction se heurte à deux difficultés majeures : l'une d'ordre conceptuel, l'autre relative à la réalisabilité de la méthode conduisant à une solution. En effet, la complexité de l'espace des configurations ne permet pas de l'exprimer de manière analytique et sa grande dimension empêche en pratique la construction d'une représentation exacte des contraintes de l'environnement (d'une complexité exponentielle). De fait, l'ensemble des configurations, s'il est en pratique fini, reste d'une taille trop importante pour pouvoir opérer par énumération.

La technique adoptée, exposée au chapitre 6, consiste donc à construire un modèle approché surcontraint de l'espace de contact, afin de garantir la validité des solutions trouvées par la suite.

Graphe des configurations

L'espace libre obtenu peut être structuré sous la forme d'un *graphe d'états* dont les nœuds sont les configurations valides et les arcs les relations d'adjacence (cf chapitre 7). Cette représentation autorise l'emploi d'algorithmes de recherche de trajectoire de complexité raisonnable. Une étape de planification se traduit donc par la recherche d'un chemin continu depuis un nœud associé à l'état initial jusqu'à un nœud définissant l'état final.

5.1.3 Propriétés de l'espace de planification

Consistance et complétude

Un processus de planification peut être caractérisé par des propriétés de consistance et de complétude.

Ainsi un système de planification de trajectoires sera dit *consistant* si les mouvements engendrés permettent d'atteindre les configurations choisies, et ce malgré les erreurs de commande et de perception. Il sera dit *complet* si l'on est certain de trouver une solution dès lors qu'il en existe une.

Le choix d'une stratégie de construction d'un modèle approché de l'espace de planification interdit au système d'être complet. En effet les approximations

appliquées lors de cette construction pour réduire la complexité algorithmique impliquent la perte - délibérée - de solutions potentielles. Par contre, ces mêmes approximations visant à accroître les contraintes du système, permettent de prendre en compte les incertitudes de contrôle et de position. Elles permettent ainsi d'éliminer les configurations dont la validité n'est pas garantie par le modèle. Le système de planification est donc consistant.

Compacité et précision

La représentation finale doit répondre à plusieurs exigences, de compatibilité avec les outils de planification disponibles, de précision, de compacité. Le critère de compatibilité implique la conservation de propriétés topologiques (distance, connexité). Celui de compacité impose à la représentation d'être utilisable en pratique et, d'un point de vue implantation, de ne pas sacrifier l'aspect mémoire à celui du temps de calcul. Enfin la précision est un facteur déterminant quand à l'obtention de trajectoires solutions et leur nature. En particulier, la construction de cette représentation devant contourner l'écueil de l'énumération par l'utilisation de fonctions de discrétisation, il est indispensable de maîtriser la perte d'information induite, afin de pouvoir éventuellement la réduire localement.

Que l'on décide de représenter l'espace libre dans l'espace cartésien même ou dans l'espace des configurations, il faut dans tous les cas choisir une mode de découpage de l'espace en question.

Représentation par octrees

La technique des *octrees* permet de découper un espace tri-dimensionnel en une grille régulière dont chaque cellule élémentaire est un parallélépipède [11-Fav84]. Cette représentation peut être étendue à un nombre quelconque de degrés de liberté, et peut donc être employée pour décrire l'espace des configurations d'un manipulateur généralisé.

Chaque cellule représente un ensemble de configurations du robot et peut se trouver dans trois états différents : *vide* lorsque toutes les configurations de la cellule sont valides, *pleine* lorsqu'aucune de l'est et *mixte* dans le cas où les deux cas se produisent. Dans ce cas, il faut subdiviser la cellule et effectuer la même opération "d'étiquetage" sur les cellules nouvellement créées, et itérer ainsi le processus jusqùà obtenir des cellules connues comme étant pleines ou vides. Cette division récursive des cellules ordonne ces dernières en une hiérarchie de taille,

et l'on obtient un arbre décrivant l'espace libre à divers niveaux de résolution.

Représentation par intervalles

L'idée est d'obtenir un pavage non uniforme de l'espace libre du manipulateur. Nous utilisons pour cela la structure hiérarchique du modèle géométrique précédemment défini. A un niveau donné, les bornes des intervalles sont calculées comme les limites de débattement des différents constituants du manipulateur (cf chapitre 6) et dépendent donc de la résolution de la représentation courante. L'intérieur des intervalles croisés constitue l'ensemble des cellules vides.

Lorsque l'on passe à un niveau de précision supérieur, l'approximation des objets est moins grossière et donc les contraintes de débattement sont moins fortes : le volume "libéré" par la prise en compte de la nouvelle représentation correspond alors aux cellules mixtes, dont l'état peut être évalué en calculant les nouvelles limites de débattement.

L'intérêt par rapport aux octrees est de fournir un arbre hiérarchique dont le nombre de niveaux est borné (par le nombre de niveaux du modèle) et dont la convergence est donc plus rapide. Le pas de discrétisation n'est plus arbitraire mais dépend de la différence de volume entre deux approximations successives d'un objet. L'inconvénient majeur est que la représentation hiérarchique obtenue est fortement dépendante de la modélisation effectuée.

5.2 Représentation de l'espace des configurations

On cherche à représenter dans l'espace des configurations EC l'espace libre EL d'un manipulateur entouré d'obstacles fixes.

Comme dans EC une configuration est un point, l'idée de base retenue est de contruire un graphe dont les nœuds sont des ensembles de tels points. Construire cet espace implique donc la détermination de l'image de tout obstacle dans l'espace des configurations.

La structure articulée du robot présentant un encombrement conséquent (volumique) qu'il n'est pas possible de traduire au niveau de la configuration, il est donc nécessaire de répercuter cette contrainte au niveau de l'obstacle. On construit en fait non pas l'espace de contact réel du manipulateur mais un espace plus important équivalent à l'espace de contact d'un manipulateur squelettisé

évoluant dans un univers plus contraint.

5.2.1 Définitions et notations

Les notations définies ici et utilisées dans toute la suite du mémoire sont en partie inspirées du formalisme introduit par Th.Lozano-Pérez [10-LP81].

Structure du robot :

Nous considérons le cas d'un système mobile constitué d'une structure articulée évoluant au milieu de solides fixes. La structure du robot est formée d'une base fixe A_0 et de N ($N \geq 1$) composants rigides A_i évoluant dans un espace $\Re^p$ ($p \leq 3$). Le terme de robot désignera exclusivement la partie mobile $\mathcal{A} = \{A_1, \ldots, A_N\}$[1].

Nous définissons également le sous-système mobile $\mathcal{A}_n = \{A_1, \ldots, A_n\}$ ($n \leq N$), constitué des n premiers éléments du robot, ignorant ainsi les composants postérieurs. Pour $n = N$, on a donc $\mathcal{A}_N = \mathcal{A}$.

Degrés de liberté, configuration :

La position de chaque composant A_i relativement à son prédécesseur A_{i-1} dans la chaîne cinématique est définie par un paramètre q_i, dit *degré de liberté* du composant. Chaque q_i varie dans les limites d'un intervalle I_i initialement défini par les butées mécaniques du robot.

Nous serons amenés par la suite à décomposer cet intervalle I_i en une partition (de cardinalité L_i) de sous-intervalles connexes notés Δ_{i,l_i}, dont ne seront retenus que des intervalles articulaires valides (voir ci-dessous), généralement non connexes, notés D_{i,k_i}, avec $1 \leq k_i \leq K_i$ et sauf exception $K_i \leq L_i$.

Une configuration ς du système $\mathcal{A}_n$, n-uplet de variables généralisés, est donc en l'occurence un n-uplet de degrés de liberté $(q_1, \ldots, q_n)$, avec $q_i \in I_i$.

Position spatiale :

Nous définissons une *position spatiale* de ce système comme un ensemble de M paramètres permettant de spécifier la position et l'orientation dans l'espace $\Re^p$ de l'extrémité terminale du manipulateur. Le M-uplet des variables opérationnelles définit donc une position spatiale. Dans le cas d'un système non-redondant,

[1] Nous notons de manière générique B tout obstacle de l'environnement et A tout objet mobile, à ne pas confondre avec l'ensemble du robot $\mathcal{A}$ ou ses composants A_i.

$M = N$ et il y a identité entre la configuration des composants et la position spatiale de l'effecteur.

Posture, Entropie :

Nous nommons alors *posture* d'un objet la donnée de sa géométrie et de sa position, notée $P_A(\varsigma)$. La posture du sous-système mobile $\mathcal{A}_n$ est donc par extension la donnée d'une configuration et de la géométrie de l'ensemble des n composants associés. Cette géométrie peut varier selon le niveau hiérarchique h de la représentation utilisée. On notera cette posture $P^h_{\mathcal{A}_n}(\varsigma)$. On distingue en particulier la posture originelle (robot dans sa représentation la plus précise, pour $h = 0$) et la posture filaire (robot squelettisé).

Nous appelons alors *entropie*[2] et notons $\mathcal{E}_A(\varsigma)$ le volume occupé dans l'espace $\Re^p$ par la posture $P_A(\varsigma)$. L'entropie d'un sous-système est donc $\mathcal{E}^h_{\mathcal{A}_n}(\varsigma)$, définie pour la posture $P^h_{\mathcal{A}_n}(\varsigma)$ du sous-système mobile considéré au niveau hiérarchique h.

Débattements valides

On appelle *débattement valide* associé à un mobile A se déplaçant suivant une seule direction articulaire q, relativement à un obstacle B, l'ensemble des valeurs D_q de q telles que $\mathcal{E}_A(q) \cap B = \emptyset$.

Le calcul de ces débattements est réalisé afin de déterminer tous les déplacements possibles d'un mobile animé d'un mouvement simple (translation ou rotation). Ainsi donc, pour un système mobile $\mathcal{A}_n$, le débattement valide relatif au déplacement suivant q_i dans la partition $\{\Delta_{i,l_i}\}$ de l'intervalle I_i, les autres degrés de liberté étant fixés en $q_0 \ldots q_{i-1}$, est la partition d'intervalles $D_{i,k_i} = [q^0_{i,k_i}\ q^1_{i,k_i}]$ telle que :

$$\forall q \in D_{i,k_i},\ \mathcal{E}_{A^n}(q_0, \ldots, q_{i-1}, q) \cap B = \emptyset$$

Objets, Obstacles, C-obstacles :

Les objets de l'environnement $\mathcal{B}$ du robot, notés B_p, avec $p \leq P$, sont modélisés comme l'union de K_p composants solides élémentaires $B^{k_p}_p$.

De manière générale, un objet B est un *obstacle* relativement à autre objet A si l'un de ses composants constitue un corps solide susceptible d'entraver le

[2]Le terme d'entropie pouvant traduire la faculté d'encombrer l'espace désigne ici le volume occupé par le système mobile.

mouvement de celui-ci. On appelle alors *C-obstacle* et on note $CO_A(B)$ l'image de B relativement à l'objet A dans l'espace des configurations de ce dernier, c'est-à-dire l'ensemble des configurations de A qui occasionnent une collision avec B.

On notera donc $CO_{\mathcal{A}_n}(B)$ le C-obstacle relativement au sous-système $\mathcal{A}_n$ dans l'espace des configurations associé, et $CO(B)$ le C-obstacle de B relativement au système complet.

$$CO_{A_n}(B) = \{\varsigma \in EC \mid \mathcal{E}_{\mathcal{A}_n}(\varsigma) \cap B \neq \emptyset\}$$

Ces C-obstacles sont donc des hypervolumes de dimension n dont la caractérisation exacte pose souvent des problèmes de complexité algorithmique difficiles à résoudre, comme nous le verrons au chapitre 6.

L'image d'un obstacle B dans l'espace des configurations $EC_{\mathcal{A}}$ est donc le C-obstacle relatif à la posture courante du manipulateur complet. Une caractérisation exacte de ces ensembles $CO_{\mathcal{A}}(B)$ doit ainsi faire appel à des outils mathématiques sophistiqués, dont l'exploitation informatique n'est pas toujours envisageable.

Nous serons amenés à travailler dans des sous-espaces EC_n de l'espace des configurations ($n \leq N$), conduisant à considérer tour à tour les sous-systèmes $\mathcal{A}_n$. Dans le cas où les trois premiers degrés de liberté définissent la position d'un composant mobile A, le sous-espace associé aux trois translations de A sera noté EC^{τ}, et les obstacles correspondants sont notés $CO^{\tau}(B)$.

Nous noterons de plus $EC_{\mathcal{A}_n,q_1,\ldots,q_{n-1}}$ (ou simplement $EC_{\mathcal{A}_n}$) le sous-espace associé au $n$ième degré de liberté du sous-système $\mathcal{A}_n$ lorsque les d.d.l antérieurs sont fixés à des valeurs $q_1, \ldots, q_{n-1}$. Les obstacles correspondants sont alors les $CO^{q_n}_{A_n}(B)$, et l'espace balayé par le $n$ième composant A_n lorsque q_n décrit un intervalle $D_n = [q_n^0\ q_n^1]$ est donc représenté par l'entropie totale

$$\mathcal{E}_{A_n}(q_0, \ldots, q_{n-1}, D_n) = \cup_{q \in D_n}\ \mathcal{E}_{A_n}(q_0, \ldots, q_{n-1}, q)$$

les obstacles résultants étant notés $CO^{D_n}_{A_n}(B)$.

Espace libre, Trajectoire :

Dès lors, l'espace de contact est défini formellement par :

$$EK_{\mathcal{A}} = \bigcup_{p=1}^{P} CO_{\mathcal{A}}(B_p)$$

Et nous obtenons donc une formulation de l'espace libre EL utilisable pour construire une représentation de l'espace de planification :

$$EL_{\mathcal{A}} = EC_{\mathcal{A}} - \bigcup_{p=1}^{P} \{\varsigma \in EC_{\mathcal{A}} \mid \mathcal{E}_{\mathcal{A}}(\varsigma) \cap B_p \neq \emptyset\}$$

Une *trajectoire* $\mathcal{T}$ sera définie comme un chemin dans $EL_{\mathcal{A}}$, c'est-à-dire comme une séquence (ensemble discret fini) de configurations de $EC_{\mathcal{A}}$.

Le problème de la planification des mouvements du système mobile est donc ramené à la recherche de chemins dans $EL_{\mathcal{A}}$. Les techniques utilisées pour ce calcul sont décrites dans le chapitre 7.

5.2.2 Positions spatiales et configurations

Les problèmes rencontrés en pratique se limitent aux cas d'un système mobile évoluant dans $\Re^2$ ou $\Re^3$ et possédant par conséquent, s'ils sont non redondants, au plus respectivement trois et six degrés de liberté. Les configurations traitées sont alors des éléments de $\Re^p \times \mathcal{O}^q$ avec p et q quelconques, où $\mathcal{O}^q$ est le groupe des rotations orthogonales sur l'espace vectoriel $\Re^p$.

Objets rigides mobiles

La position d'un objet rigide en mouvement dans le plan nécessite deux paramètres et son orientation un seul, et dans l'espace respectivement trois et trois. L'espace des configurations est alors isomorphe à celui des positions spatiales, $\Re^2 \times \mathcal{O}^1$, respectivement $\Re^3 \times \mathcal{O}^3$.

Si R_o est le référentiel absolu de l'espace considéré, une configuration ς du système s'écrira $(x\ y\ \theta)$, respectivement $(x\ y\ z\ \theta\ \phi\ \psi)$, où x, y et z sont les coordonnées du point de référence du solide relativement à R_o, et θ, ϕ et ψ définissent les rotations autour des directions Ox, Oy et Oz.

La figure 5.1 illustre ce mode de représentation dans le cas bidimensionnel, le plus courant. A gauche un mobile et sa configuration $(x\ y\ q1)$, à droite une structure articulée redondante de configuration $(q1\ q2\ q3)$.

On peut également étendre cette représentation au cas de sous-structures mobiles (en fait des structures dont certains degrés de liberté sont volontairement ignorés) ou de structures moins classiques. Le premier exemple est entre autres

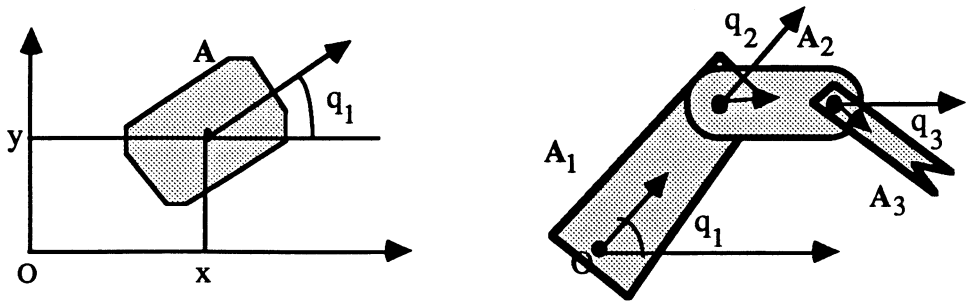

Figure 5.1: Exemples de configurations d'un robot.

celui de la planification des mouvements fins de montage dans SHARP . Le second s'applique au cas simple d'éléments à trois degrés de liberté concourants, l'espace des configurations étant un sous-ensemble de $\Re^3$ dans le cas d'un robot cartésien, de $\Re^2 \times \mathcal{O}^1$ pour un robot cylindrique ou sphérique, et $\mathcal{O}^3$ pour une main articulée (en rotation pure).

On peut également remplacer les paramètres de position par d'autres en translation (ainsi les robots cartésiens qui combinent trois articulations prismatiques et trois articulations rotoïdes ont leur EC inclus dans $\Re^3 \times \mathcal{O}^3$), ou en rotation (et en ajouter alors un nombre quelconque), ce qui nous amène à considérer le cas des bras manipulateurs.

Structures articulées

Le système mobile est dans ce cas constitué d'une chaîne cinématique ouverte à base fixe, composée de N corps rigides articulés.

Une configuration ς est alors un N-uplet de variables articulaires, débattements en translation ou rotations selon que le type de mouvement est prismatique ou rotoïde. Du fait des rotations, l'espace des configurations EC n'est généralement pas isomorphe à l'espace Euclidien $\Re^N$, et à une position spatiale donnée correspond plusieurs configurations. L'exemple typique illustrant ce problème est celui du manipulateur à deux degrés de liberté en rotation (cf figure 6.7 : les deux composants peuvent être indifféremments placés de chaque coté de la droite liant la base à l'extrémité terminale, pour toute position de celle-ci. Au cas déjà cité du robot cartésien à six degrés de liberté, nous pouvons ajouter celui tout aussi courant du robot comportant six degrés de liberté de type rotoïde. Une position du manipulateur est alors spécifiée généralement un élément de $\mathcal{O}^6$ et une configuration est un élément du produit C_1^6 (où C_1 est le cercle trigonométrique),

également connu sous le nom de "6-tore". Dans ce cas, à chaque position spatiale correspond 8 configurations possibles [1-Pau81].

Sous cette forme, et considérant la cinématique du robot, l'espace de planification est le plus homogène, le plus symmétrique, le mieux adapté à l'expression du mouvement, le plus utile enfin en regard des techniques de planification de trajectoires utilisés. Toutefois, pour la planification et la commande des autres mouvements, l'espace $\Re^3 \times O^3$, représentant l'espace des configurations de l'outil terminal du robot, est plus communément utilisé. Le passage de l'un à l'autre réclame alors l'utilisation d'outils spécifiques : les changeurs de coordonnées direct et inverse.

Partie II

Raisonnement géométrique et planification de trajectoires : méthodes et implantation

Chapitre 6

Construction de l'espace de planification

6.1 Principes de l'approche

6.1.1 Principes

Approche de base et variante choisie

Ce chapitre décrit une technique de construction de l'espace de planification du manipulateur, et plus particulièrement de l'espace libre :

$$EL = EC - \bigcup_{p=1}^{P} CO(B_p)$$

Les techniques appliquées, dites de *raisonnement géométrique spatial*[1], opèrent sur un modèle complet de l'espace de travail du robot. La prise en compte de la structure hiérarchique de cette représentation géométrique relève d'un niveau de contrôle plus élevé (cf chapitre 9) : nous nous intéressons ici au passage d'une représentation donnée de l'espace de manipulation à une représentation équivalente de l'espace de planification.

Nous décrivons par conséquent un algorithme de base, les outils géométriques qu'il nécessite et les approximations qu'il impose dans son utilisation pratique au cas d'un manipulateur. L'inextricable complexité algorithmique du cas général est en effet réduite par l'utilisation d'approximations, dont la maîtrise et la précision varient en fonction des caractéristiques globales des trajectoires cherchées.

[1] Raisonnement géométrique pour la nature des données sur lequel il opère, et spatial en raison des concepts topologiques qu'il est amené à manipuler : nature et connexité des positions, forme et volume des objets, relations spatiales...

La formulation ci-dessus conduit à calculer l'espace de contact pour l'ensemble des obstacles, puis d'en déduire par complémentarité l'espace libre du manipulateur. Une fois EK calculé, rien ne permet plus de distinguer l'origine d'une région interdite et, si un objet vient à être ajouté ou supprimé, on est obligé de tout recalculer.

C'est pourquoi nous préférons la formulation équivalente

$$EL = \cap_{p=1}^{P} EL(B_p) \quad \text{avec} \quad EL(B_p) = EC - CO(B_p)$$

qui nous amène à calculer l'espace libre relativement à chaque obstacle et prendre ensuite la conjonction des contraintes imposées par la présence simultanée de tous les obstacles. Cette dernière opération est décrite au paragraphe 7.1.1.

Nature de l'espace de planification construit

L'originalité de la méthode réside dans le procédé récursif utilisé pour construire un modèle surcontraint de l'espace libre. Celui-ci est basé sur une discrétisation de l'espace articulaire : la représentation obtenue est donc un ensemble d'intervalles de validité des différents degrés de liberté du manipulateur.

Plus précisément, on obtient une arborescence d'ensembles de sous-intervalles non connexes des intervalles de variations I_i des degrés de liberté q_i.

$$\begin{array}{ccccc}
D_{1,1} & \cdots & D_{1,k_1} & \cdots & D_{1,K_1} \\
 & \swarrow & \downarrow & \searrow & \\
D_{2,k_1,1} & \cdots & D_{2,k_1,k_2} & \cdots & D_{2,k_1,K_2} \\
 & \swarrow & \downarrow & \searrow & \\
 & & \text{etc} & & \\
 & \swarrow & \downarrow & \searrow & \\
D_{N,k_1,,k_{N-1},1} & \cdots & D_{N,k_1,,k_{N-1},k_N} & \cdots & D_{N,k_1,,k_{N-1},K_N}
\end{array}$$

où chacun des K_i intervalles $D_{i,k_i} = [q^0_{i,k_i}\ q^1_{i,k_i}]$ se voit associé K_{i+1} intervalles $D_{i+1,k_{i+1}}$ avec $1 \leq k_{i+1} \leq K_{i+1}$. Le nombre et la nature de ces intervalles dépendent de toute la séquence $(D_{0,k_0} \ldots D_{i,k_i})$, que nous nommerons *chaîne de discrétisation.* Nous noterons alors

$$EL(B) = \langle D_{i,k_i} \rangle$$

cette arborescence de partitions d'intervalles constituant une représentation de l'espace libre relatif à un objet B.

6.1.2 Algorithme de construction de l'espace libre

Forme générale de l'algorithme

Nous donnons une version condensée, sous forme de pseudo-programme, de l'algorithme $LIBRE(n, D, B)$ utilisé pour construire récursivement l'espace libre EL relativement à un obstacle B. Bien que repris et commenté en 6.3, il est présenté ici afin d'en illustrer dès à présent l'aspect récursif et d'insister sur son utilisation extensive de prédicats géométriques (ou macro-fonctions de calcul géométrique).

procédure LIBRE(n,D,B)
début
si $n = N$:
  $\{D_{N,k_N}\} := DEBAT(A_N, I_N, B)$; /* *liste de* K_N *intervalles* */
  pour $k_N = 1 \ldots K_N$: $EL = EL \cup (D \times D_{N,k_N})$;
fin si
$\{\Delta_{n,l_n}\} := PARTN(I_i)$; /* *liste de* L_n *intervalles* */
pour $l_n = 1 \ldots L_n$:
  $B := GROSS(n, D \times \Delta_{n,l_n}, B)$; /* *modification des obstacles* */
  $q_{n\,R} := QREF(\Delta_{n,l_n})$; /* *point de reference* */
  si $\neg\, INTER(A_n, q_{n\,R}, B)$: $LIBRE(n+1, D \times \Delta_{n,l_n}, B)$;
fin pour
fin

Les macros-fonctions utilisées sont les suivantes :

- $DEBAT$: permet de calculer les *débattements valides* - relativement à un ensemble d'obstacles - d'un objet en mouvement de translation ou de rotation;
- $PARTN$: permet de déterminer une *partition* d'un intervalle de débattement articulaire par une discrétisation heuristique;
- $GROSS$: regroupe essentiellement deux types d'opérations :
  - des opérateurs dits de *grossissement* permettant de calculer de manière exacte les contraintes de position imposés par les obstacles sur un mobile ne comportant que des degrés de liberté en translation, et
  - des opérateurs dits de *réduction* permettant le calcul approché des contraintes de position *et* d'orientation par discrétisation des degrés de liberté du mobile;

- *QREF* : permet de fixer un *point de référence* dans un intervalle de débattement articulaire;
- *INTER* : permet de calculer les *interférences* pouvant se produire entre objets, particulièrement entre les éléments du robot et les objets de l'environnement.

La principale difficulté liée à l'implantation de ces fonctions, découle de la complexité algorithmique qu'elles introduisent. Cette complexité est maîtrisée dans les algorithmes implantés par une limitation des types d'objets traités et l'emploi d'approximations.

Contitions initiales d'application de l'algorithme

Les conditions initiales du problème de construction de l'espace de planification se résumant à un espace libre totalement inconnu, sa résolution générale est obtenue par l'appel de

$$LIBRE(1, \emptyset, \mathcal{B})$$

Tel quel, l'algorithme ci-dessus est appliqué à l'environnement $\mathcal{B}$ tout entier. Les fonctions DEBAT, GROSS et INTER sont donc appliquées sur l'ensemble des obstacles B_p. Afin de faciliter la mise à jour, nous avons choisi d'appliquer *LIBRE* autant de fois qu'il y a d'obstacles, construisant ainsi P espaces libres $EL(B_p)$, et de les réunir ensuite pour reconstituer l'espace libre total. La résolution du problème est alors obtenue par l'appel de

$$UNION\left(\{EL(B_p)\}_{p=1}^{P}\right) = UNION\left(\{LIBRE(1, \emptyset, B_p)\}_{p=1}^{P}\right)$$

L'ensemble EL obtenu par application de l'algorithme précédent est ensuite structuré à l'aide de la fonction *ADJACENT* (cf.paragraphe 7.1.2). Cette structure permet de chaîner entre elles toutes les cellules de EL qui possèdent une frontière commune.

Les prédicats géométriques utilisés dans cet algorithme sont décrits dans les paragraphes suivants, d'abord de manière générale, puis dans leur application au cas du manipulateur. La méthode de réduction des degrés de liberté utilisée permet assez fréquemment de se ramener à un problème bidimensionnel (collision et débattement entre autres) : c'est pourquoi nous précisons ce cas particulier.

La recherche d'une trajectoire dans le graphe ainsi construit fait l'objet du chapitre suivant.

6.2 Les prédicats géométriques

Nous présentons dans les paragraphes suivants les fonctions de calcul d'interférence, de débattement, de grossissement et de réduction. Le problème de la discrétisation (fonction $PARTN$) est traité dans le chapitre 10.

6.2.1 Calculs d'interférences

Définition

Les calculs d'interférences sont par nature statiques : ils visent à déterminer si deux objets positionnés dans l'espace sont ou non en situation de collision.

Les calculs de collisions, que l'on peut par opposition qualifier de dynamiques, visent à déterminer l'amplitude maximale d'un mouvement simple (translation ou rotation) effectué par un objet rigide. Il s'agit donc de déterminer les débattements valides d'un degré de liberté prismatique ou rotoïde : celui, dans le cas d'une structure articulée, du Nième et dernier de ses constituants.

Si la configuration mécanique d'un robot est telle qu'il puisse y avoir collision entre certains de ses constituants, alors un contrôle de non-interférence doit également être réalisé entre les A_n. Toutefois, ce type de collision est en général empêché par la présence de butées logicielles ou mécaniques sur les degrés de liberté du manipulateur.

Nous notons donc $INTER(A, \varsigma, B)$ le prédicat géométrique qui s'évalue à vrai lorsque l'objet A placé dans la configuration ς intersecte B, et $\sqcap$ cette relation d'intersection. La fonction $INTER$ est utilisée pour déterminer si l'élément terminal d'un sous-système $\mathcal{A}_n$ intersecte un obstacle, soit si la configuration associée $EC_{\mathcal{A}_n}$ appartient ou non à l'espace libre $EL_{\mathcal{A}_n}$.

$$INTER(A, \varsigma, B) \equiv A \sqcap B \equiv P_A(\varsigma) \cap B \neq \emptyset$$

On peut caractériser l'intersection de deux solides A et B par l'existence d'une entité géométrique appartenant à la fois aux deux. L'entité minimale étant le point, A et B ne sont pas en situation d'interférence lorsqu'aucun point de l'un n'appartient à l'autre :

$$A \not\sqcap B \Leftrightarrow (\forall p_A \in A,\ p_A \notin B) \wedge (\forall p_B \in B,\ p_B \notin A)$$

L'opération de base pour détecter une interférence consiste donc à analyser l'intersection de tous les couples d'entité géométriques des représentations respectives de A et de B.

Le choix de ces représentations et les limitations imposées sur la nature des composants élémentaires permettent d'utiliser des méthodes moins exhaustives. En particulier, la propriété de convexité de ces composants est fondamentale.

Interférence d'objets polyhédriques

Dans le cas tridimensionnel, celui des *objets polyédriques* convexes, les entités géométriques à tester deviennent celles de dimension 2, soit les arêtes et les faces des objets. On calcule ainsi l'intersection de tous les couples croisés "arête-face".

$$A \not\cap B \Leftrightarrow \begin{array}{l} \forall f_A \subset A, \forall a_B \subset B,\ f_A \cap a_B = \emptyset \\ \forall a_A \subset A, \forall f_B \subset B,\ f_B \cap a_A = \emptyset \end{array}$$

Du fait du caractère exhaustif de la méthode, sa complexité est proportionnelle au produit des nombres d'entités géométriques concernées. Dans le cas général, si N^a est le nombre d'arêtes d'un objet et N^f le nombre de ses faces, la complexité maximum de l'algorithme est en $O(N_A^f \cdot N_B^a + N_A^a \cdot N_B^f)$.

L'utilisation de la hiérarchie spatiale des volumes englobants permet de réduire cette complexité en limitant le nombre des primitives du modèle. En effet un volume englobant étant lui-même - par construction - une primitive élémentaire, il comporte moins d'entités géométriques que l'ensemble des solides qu'il englobe. L'utilisation de ces volumes permet en moyenne de diminuer les calculs à effectuer.

Le cas bidimensionnel ramène le problème à l'intersection d'*objets polygonaux* convexes, laquelle peut être déterminée en testant l'inclusion de leurs arêtes, sommets inclus. Dans le cas général, la complexité maximum de l'algorithme est donc en $O(N_A^a \cdot N_B^a)$.

Les cas particuliers d'inclusion d'une arête ou d'une face de l'un des objets dans une face de l'autre sont toujours très délicats à détecter à cause des erreurs de calcul inévitables (problèmes d'arrondi etc). Ce genre de problème ne peut être résolu que par une étude du voisinage des entités géométriques concernées.

Interférence d'objets non polyédriques

Le cas des *composants non polyédriques* correspond aux primitives de sphère, cylindre et cône. Pour tester l'intersection de ces trois composants avec un quelconque autre objet élémentaire, il suffit d'évaluer celles des couples croisés "face-face" :

$$A \not\cap B \Leftrightarrow \forall f_A \subset A, \forall f_B \subset B,\ f_A \cap f_B = \emptyset$$

et la complexité est proportionnelle au produit $N_A^f \cdot N_B^f$ du nombre de calculs d'intersection entre quadriques, nettement plus compliqué. En pratique, l'approximation polyhédrique et l'utilisation des volumes englobants s'avèrent particulièrement fructueuses en temps opératoire.

6.2.2 Calcul de collisions et débattements

Définition

Les calculs de collisions sont utilisés pour calculer les débattements valides D_{n,k_n} d'un sous-système $\mathcal{A}_n$. D'une manière générale, ils conduisent à déterminer les contacts susceptibles de se produire entre un objet A effectuant un mouvement ψ et un obstacle B (la présence de plusieurs obstacles imposant seulement de réitérer l'opération pour chacun d'eux).

La méthode générale consiste à analyser toutes les intersections qui existent entre les éléments de B, et les lieux géométriques décrits par les entités géométriques (sommets et arêtes essentiellement) de A [5-Boy79]. Si le mouvement ψ de A est une translation, le lieu $\psi(s_A)$ décrit par un sommet est un segment de droite et celui d'une arête $\psi(a_A)$ est une surface trapézoïdale. Par contre, si ψ est une rotation, le lieu d'un sommet est un arc de cercle, celui d'une arête est un disque, un cylindre, un cône ou un hyperboloïde de révolution, éventuellement "percés".

Les intersections ainsi calculées définissent soit des points de contact entre A et B, soit des points pour lesquels A intersecte B. Dans les deux cas, ces points déterminent des configurations interdites pour A, appartenant donc à l'ensemble $CO_{\mathcal{A}_n}(B)$.

Collision d'objets polyédriques

Un polyèdre A effectuant un mouvement ψ de translation ou de rotation peut entrer en collision un obstacle polygonal B de trois manières différentes, occasionnant des contact de trois types :

I un sommet de A avec une face de B, ou

II une face de A avec un sommet de B, ou

III une arête de A avec une arête de B.

Nous ne considérons pas les cas des contacts "instables", irréalistes, tels que ceux entre sommets, sommets et arêtes...

Les contacts de type I peuvent être détectés pour chaque sommet s_A de A par un calcul d'interférence entre le lieu géométrique $\psi(s_A)$ décrit par le sommet dans son mouvement et les faces de l'objet B. De même les contacts de type II peuvent être détectés pour chaque sommet s_B de l'obstacle B par le calcul d'interférence entre le lieu géométrique qu'il décrit dans son *mouvement fictif* $\psi^{-1}(s_B)$ relativement à A et les faces de ce dernier. Enfin les contacts de type III peuvent être détectés pour chaque arête a_A de A par un calcul d'interférence entre le lieu géométrique $\psi(a_A)$ décrit par son mouvement et les arêtes de B.

Dans le cas bidimensionnel de deux polygones, la règle de détection précédente s'applique en considérant les arêtes à la place des faces (le cas III disparaissant).

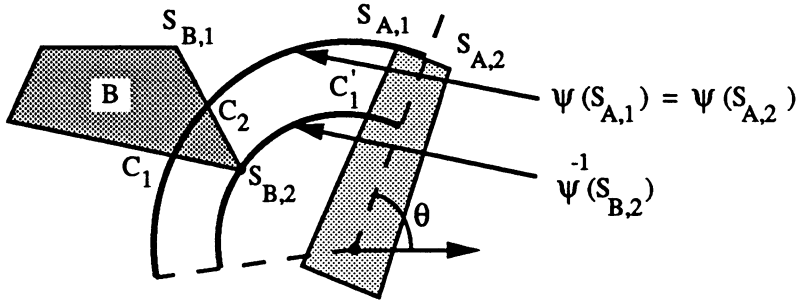

Figure 6.1: Calcul des points de contact d'un objet en rotation.

Dans l'exemple illustré par la figure 6.1, on obtient lors du mouvement en rotation du composant deux contacts de type I (sommets C_1 et C_2) et un contact de type II (sommet fictif C_1').

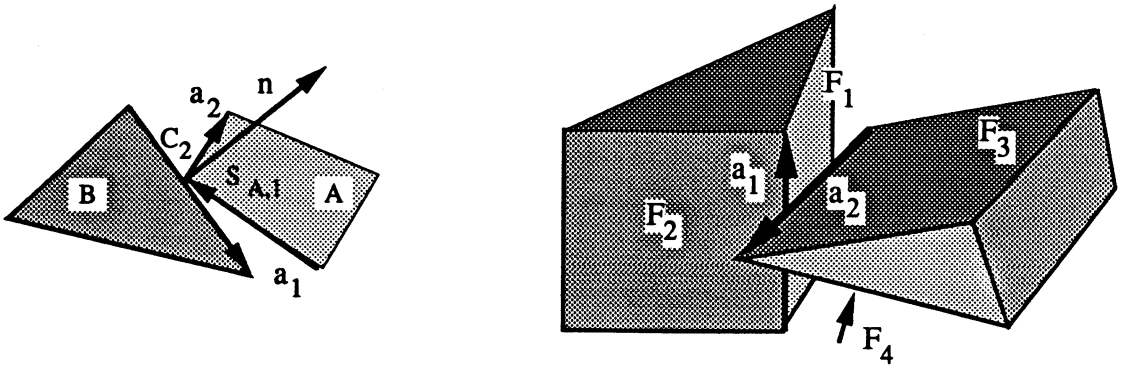

Figure 6.2: Détermination de contact à l'aide des propriétés topologiques.

Les intersections de type II et III définissent des *sommets fictifs* répartis sur des faces ou des arêtes de A. Les contacts susceptibles de se produire entre A et B mettent alors en jeu un sommet réel ou un sommet fictif de A. Cette propriété reste vraie lorsque le contact considéré n'est pas ponctuel.

L'opération suivante consiste à effectuer une analyse locale de validité des contacts potentiels détectés. En effet, ceux-ci ont éte sélectionnés sur des critères purement géométriques et l'on doit prendre en compte l'aspect volumique des composants, par la présence de matière, afin de déterminer quels contacts sont physiquement réalisables. La représentation topologique utilisée permet de résoudre simplement ce problème, du fait de l'orientation des arêtes de chaque composant.

Pour les contacts de types I ou II, le test retenu pour un sommet S est :

$$\forall a_k \mid S \in a_k, \quad a_k \cdot n \geq 0$$

où les a_k sont donc les arêtes de A (respectivement B) issues de S et orientées à partir de S, et n la normale extérieure à la face de B (respectivement A).

Pour les contacts de type III, le test retenu est :

$$(a1 \cdot n1) \cdot (a1 \cdot n2) \leq 0$$
$$(a2 \cdot n3) \cdot (a2 \cdot n4) \leq 0$$

où a_1 est l'arête de contact de A (respectivement B) orientée de manière arbitraire, a_2 l'arête de contact de B (respectivement A) orientée de manière arbitraire, n_1 et n_2 sont les normales extérieures aux faces F_1 et F_2 issues de a_1 dans A (respectivement B), et n_3 et n_4 sont de même les normales extérieures aux faces F_3 et F_4 issues de a_2 dans B (respectivement A).

La figure 6.2 (droite) illustre un exemple de contact de type III. Notons que la faisabilité réelle des solutions retenues est analysée implicitement par la méthode de construction des $CO_{\mathcal{A}}(B)$.

La complexité de cet algorithme de recherche des contacts est de l'ordre de $O(N_A^a \cdot N_B^f + N_A^f \cdot N_B^a + N_A^a \cdot N_B^a)$ (avec les mêmes notations que précédemment).

Dans le cas bidimensionnel, la plausibilité du contact est vérifiée à l'aide du test :

$$(a_1 \cdot n \leq 0) \ \wedge \ (a_2 \cdot n \geq 0)$$

où n est la normale (obtenue par une rotation de $\pi/2$) au vecteur directeur de l'arête de contact orientée dans le sens rétrograde sur A (respectivement B), et où a_1 et a_2 sont les arêtes issues de S et orientées dans le sens rétrograde sur B (respectivement A).

La figure 6.2 (gauche) illustre une application de cette propriété à l'exemple précédent (figure 6.1), précisément au point de contact C_2 du sommet $S_{A,1}$ de l'objet mobile avec l'obstacle.

Notons que cette propriété n'est pas vérifiée pour des objets concaves (mais nos représentations géométriques en sont délibérément dépourvues) ou pour des obstacles proches. Ces aspects sont alors pris en compte par l'algorithme de construction des $CO_{\mathcal{A}_n}(B_p)$ décrit plus loin.

Comme pour les calculs d'interférence, les performances de cet algorithme peuvent être améliorées de la même manière en exploitant les représentations disponibles et les propriétés du modèle (volumes englobants, classement des arêtes par orientation sur un polygone convexe etc).

Composants non polyédriques

Les propriétés précédentes ne sont évidemment pas applicables dans le cas des composants cylindriques, coniques et sphériques. Ces méthodes peuvent toutefois s'étendre aux cas des surfaces quadratiques. Mais, si le cas des mouvements de translation reste traitable, puisque se ramenant à un calcul de "butées", la complexité algorithmique du calcul des lieux géométriques et des intersections associées devient bien trop importante dans le cas des mouvements de rotation.

Une solution efficace, que nous avons retenu, consiste à substituer aux surfaces gauches une approximation polygonale. Toutefois se pose alors le problème de la qualité de la décomposition : ou elle est très précise et le nombre de faces

modélisées est très important, ou bien elle est peu précise et les objets sont très grossièrement approchés. Ce dernier inconvénient ne révèle en fait son importance que lorsqu'on utilise un des niveaux les plus précis de la hiérarchie de représentations. L'approximation polyhédrique est au contraire parfaitement cohérente avec celles effectuées lors du calcul des volumes englobants, et est donc parfaitement utilisable au moins pour les premiers niveaux de représentation.

Calcul des débattements valides

Nous notons donc $DEBAT(A, B)$ le prédicat géométrique qui calcule les débattements valides de l'objet A relativement à l'obstacle B dans son mouvement courant. La fonction $DEBAT$ est utilisée pour calculer l'ensemble des débattements valides D_{n,k_n} du composant terminal du sous-système mobile $\mathcal{A}_n$, pour q_n pris dans un sous-intervalle Δ_{n,l_n} de I_n.

$$\{D_{n,k_n}\}_{k_n=k^0_{l_n}}^{k^1_{l_n}} = \Delta_{n,l_n} - CO_{A_n}(B)$$

On peut bien sûr l'appliquer à l'ensemble des obstacles de l'environnement ($\mathcal{B}$). On obtient alors la totalité de la partition des intervalles valides :

$$\{D_{n,k_n}\}_{k_n=1}^{K_n} = \cap_{p=1}^{P} \ (\Delta_{n,l_n} - CO_{A_n}(B_p)) = \Delta_{n,l_n} - \cup_{p=1}^{P} \ CO_{A_n}(B_p)$$

Chaque obstacle étant à son tour composé de K_p constituants élémentaires, on a en fait

$$\{D_{n,k_n}\} = \Delta_{n,l_n} - \cup_{p=1}^{P} \ \left(\cup_{k_p=1}^{K_p} \ CO_{A_n}(B_p^{k_p})\right)$$

où chaque $CO_{A_n}(B_p^{k_p})$ est, du fait de la convexité des composants de base, un unique domaine fermé de $\Re^n$ dont les bornes représentent alors les configurations extrémales du mobile, soit une situation de contact. Les D_{n,k_n} constituent donc une ensemble d'intervalles ouverts (puisque le simple contact est déjà considéré comme une collision) qui définissent chacun un domaine connexe de translation ou de rotation.

Les algorithmes de calcul des collisions n'ayant conduit qu'à déterminer des ensembles de contacts *potentiels*, il n'est pas possible de construire les $CO_{A_n}(B)$ à partir d'une simple classification de ces contacts. La méthode employée consiste alors à calculer la contrainte imposée par chaque obstacle B sur chaque sommet s du mobile A retenu à l'issue du premier traitement (ces sommets réels ou fictifs sont ceux impliqués dans les contacts potentiels retenus). $CO_{A_n}(B)$ représente alors la fermeture connexe de l'ensemble obtenu par union de ces contraintes (cf.

figure 6.3):

$$CO^s_{A_n}(B) = \{q_n \mid \psi(s, q_n) \cap B \neq \emptyset\} = \overline{\cup_{s \in A_n} CO^s_{A_n}(B)}$$

où s est un sommet réel ou fictif de l'élément mobile A et où q_n, configuration du $n$ième et dernier composant de $\mathcal{A}_n$, varie dans un intervalle D_{n,k_n} et où $\psi(s, q_n)$ est donc le lieu géométrique décrit par s dans le mouvement de A_n (droite dans le cas d'une translation ou arc de cercle pour une rotation).

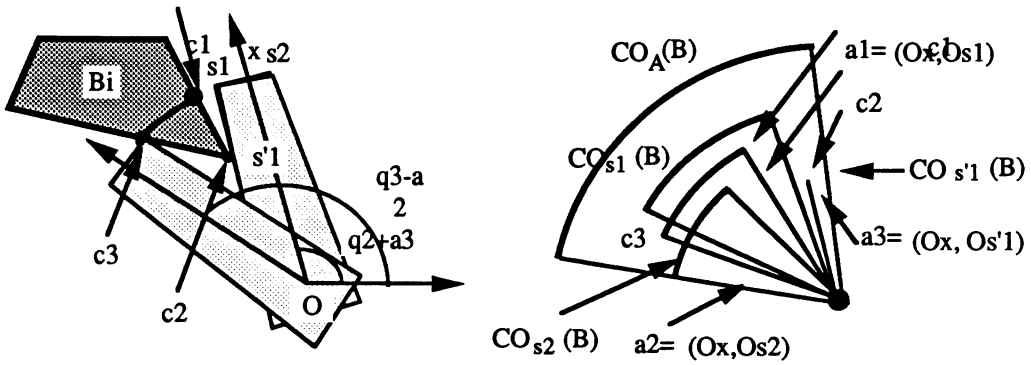

Figure 6.3: Calcul des débattements d'un composant en rotation.

La complexité de l'algorithme dans l'espace bidimensionnel est en $O((N^s_A + N^s_B) \cdot N^s B)$, où N^s est donc le nombre de sommets d'un objet. Dans l'espace tridimensionnel, la complexité devient en $O((N^s_A + N^s_B + N^a_A) \cdot N^f_B)$, où N^a_A est le nombre de d'arêtes de A et N^f_B le nombre de faces de B.

Si N^a est le nombre moyen d'arêtes d'un objet de la scène (dans $\Re^2$ ou dans $\Re^3$), la complexité de l'algorithme peut être exprimée en $O(n^{a2})$.

Cette méthode générale étant trop coûteuse, nous préferons calculer en pratique le débattement d'un segment de droite (obtenu par squelettisation du constituant mobile) par rapport à l'obstacle transformé en conséquence par une opération de grossissement uniforme.

6.2.3 Opérateurs de grossissement

Les opérateurs de grossissement sont utilisés pour construire une *représentation exacte* des obstacles dans l'espace des configurations EC_A d'un mobile A conservant une orientation fixe au cours de ses déplacements.

Somme et différence de Minkowski

Pour deux objets polyhédriques convexes A et B définis par rapport à un référentiel $\mathcal{R}_o$, la *somme de Minkowski* de A et de B est définie comme leur somme ensembliste :

$$A \oplus B = \{a + b \mid a \in A,\ b \in B\}$$

La *différence de Minkowski* $B \ominus A$ est également dite *grossissement* de l'objet B par l'objet miroir de A.

Comme l'ensemble des positions de collision entre A et B est par définition $A \cap B$, si l'on considère B comme un obstacle fixe et A comme un composant mobile se déplaçant à orientation constante, alors $B \ominus A$ est l'ensemble des positions de A (c'est-à-dire de son point de référence) qui engendrent une collision. On a donc l'égalité :

$$B \ominus A = CO^{\tau}_{A}(B)$$

Le complémentaire de $B \ominus A$ dans l'ensemble de toutes les positions réalisables (l'espace des configurations) est donc l'ensemble des positions valides (l'espace libre).

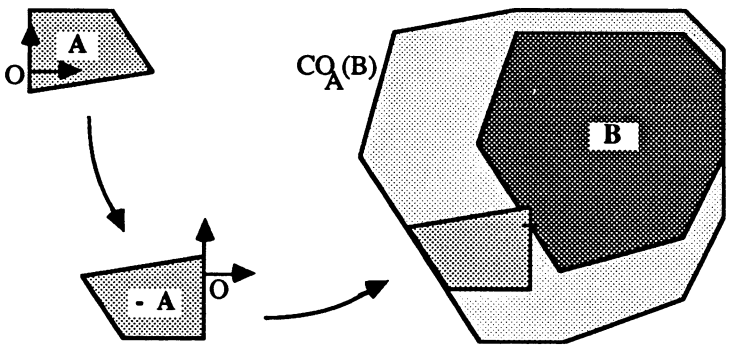

Figure 6.4: Grossissement bidimensionnel d'un obstable polygonal.

La figure 6.4 illustre un exemple de construction de la différence de Minkowski de deux objets A et B.

Calcul des ensembles de type $CO^{\tau}(B)$

Le résultat précédent peut être généralisé à des objets de dimension supérieure, pour autant que les hypothèses soient maintenues. Le procédé de construction

de ces ensembles dans le cas tridimensionnel consiste donc, en se limitant aux polyhèdres convexes en translation, à construire des différences de Minkowski.

L'objet fictif $B \ominus A$ est obtenu en "faisant glisser" le mobile A en son point de référence sur l'ensemble des surfaces de l'objet fixe B : par conséquent, l'ensemble des positions de ses sommets peut être obtenue en calculant l'enveloppe convexe des lieux géométriques des sommets de A décrits lors de ce mouvement.

Ce résultat traduit la conservation de la structure topologique des C-surfaces associées aux surfaces réelles d'un obstacle "grossi" par un objet mobile se déplaçant exclusivement en translation. L'espace EC_A^{τ} est alors isomorphe à $\Re^3$ et les ensembles $CO_A^{\tau}(B)$ sont également des polyèdres convexes. Cette propriété de conservation n'est malheureusement plus vérifiée dans le cas de mouvements de rotation : ceux-ci seront par conséquents traités différemment, comme nous le verrons au paragraphe suivant.

Dans le cas polyhédrique convexe d'un mobile A en translation dans le voisinage de B, un algorithme général pour calculer $CO_A^{\tau}(B)$ consiste à construire l'enveloppe convexe de l'ensemble des sommets des polyhèdres obtenus en plaçant $\ominus A$ sur les sommets de B [11-LP83].

$$CO_A^{\tau}(B) = EConv\left(\{s \in (\ominus A(s_B))\ s_B \in B\}\right)$$

L'enveloppe convexe de n points se calcule en $O(n \log n)$ en dimension 2 et 3. Dès lors, si N_A^s et N_B^s sont les nombres de sommets respectifs de A et de B, la complexité de l'algorithme est en $O(N_A^s \cdot N_B^s \log(N_A^s \cdot N_B^s))$. Dans le cas particulier, où A et B sont donc des polygones convexes (pas forcément coplanaires), cette borne est en fait améliorée en $O(N_A^s + N_B^s)$ en dimension 2 et $O(N_A^s + N_B^s + K)$ en dimension 3, avec au pire $K = N_A^s \cdot N_B^s$ (par exemple en utilisant un algorithme de classement par orientation des arêtes de $\ominus A$ et de B).

Cas d'objets non polyhédriques possédant une symétrie de révolution

Si le grossissement d'un objet B par un mobile A non polyhédrique entraîne a priori un fort surcroît de complexité dans la détermination de l'enveloppe convexe des sommets obtenus, il est un cas où l'on peut obtenir un bonne approximation du résultat : lorsque l'objet A possède au moins une symétrie de révolution.

L'opération de grossissement est dans ce cas homogène relativement aux orientations du mobile, et un prétraitement de l'obstacle B peut être alors effectué. Cette situation se présente par exemple lorsque A est (ou peut être approximé

à un niveau de représentation donné par) un disque, une sphère ou un cylindre se terminant par deux calottes sphériques. A condition de choisir le point de référence de A sur l'axe de révolution, on peut grossir l'obstacle B de manière isotrope dans le plan de révolution.

Notons que, afin de pouvoir conserver une représentation polyhédrique de l'objet grossi, une approximation englobante doit être effectuée. Les surfaces englobantes sont construites en prenant les plans tangents obtenus par translation "vers l'extérieur" des faces d'origine. Le calcul effectué revient à prolonger jusqu'à intersection les surfaces planes obtenues par translation des surfaces d'origine d'une distance r suivant leur normale.

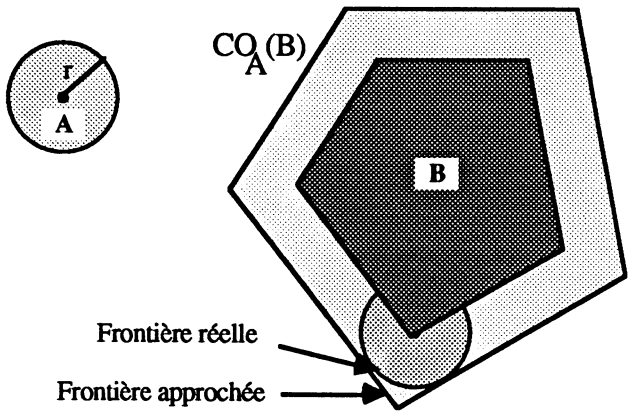

Figure 6.5: Grossissement circulaire d'un obstacle polygonal.

La figure 6.5 illustre ce procédé dans le cas simple où le mobile A est un disque de rayon r et dont le point de référence est donc le centre du disque. L'obstacle B est uniformément grossi d'une épaisseur r, et on obtient ainsi directement l'ensemble $CO_A^r(B)$. Les arêtes du contour polygonal constituant la frontière approchée sont bien obtenues par prolongement des arêtes obtenues naturellement.

On peut effectuer un traitement identique dans le cas d'un mobile cylindrique, celui-ci étant alors réduit à un segment de droite. Un calcul de grossissement supplémentaire est alors nécessaire pour déterminer l'ensemble $CO_A^r(B)$ à partir des objets équivalents obtenus (B grossi de l'épaisseur r et le segment de droite mobile).

Ce procédé de rétraction consistant à ramener un objet mobile à une entité géométrique élémentaire (point ou segment de droite) est dit de *squelettisation.*

Nous notons $Gros_r(B)$ l'obstacle obtenu à partir de B par grossissement d'un rayon r, et $Skel(A)$ le mobile A ainsi squelettisé (on pourrait dire en oubliant toute rigueur que $Skel_r(A) = Gros_{-r}(A)$).

La représentation approchée du robot obtenue lorsque tous ses composants A_n ont été ainsi réduits (dite également représentation filaire) constitue bien le squelette du manipulateur $\{Skel(A_n)\}_{i=1}^{N}$.

6.2.4 Opérateurs de réduction des degrés de liberté

Les opérateurs de réduction sont utilisés pour construire une *représentation approchée* des obstacles dans l'espace des configurations EC_A d'un mobile A comportant des degrés de liberté en rotation, ou à la fois des degrés de liberté en translation et en rotation.

Principes de la méthode

La propriété de conservation topologique des surfaces n'étant pas vérifiée à l'issu d'un grossissement par rotation, les techniques de grossissement décrites dans le paragraphe précédent ne sont donc plus directement applicables. D'une part l'opération de grossissement engendre l'apparition d'arêtes circulaires, et d'autre part les surfaces obtenues peuvent être concaves.

L'idée de la méthode est de décomposer le problème en opérant sur des sous-espaces de $EC_{\mathcal{A}}$ présentant uniquement des translations ou des rotations. Dans le premier cas, on peut calculer pour tout mobile A une approximation des obstacles $CO_A(B)$ pour différentes tranches d'orientation de A [11-LP83]. Dans le deuxième cas, plus complexe, des rotations, on peut représenter explicitement les orientations valides de $\mathcal{A}$ pour des domaines de déplacements donnés [12-Ger85]. Cette dernière méthode conduit à fixer certains degrés de liberté de $\mathcal{A}$, après leur avoir attribué des petits domaines de variation. Les C-obstacles des sous-espaces associés correspondent alors aux "projections" des portions de C-obstacles contenues dans les intervalles de débattement considérés.

Effectuer le calcul des orientations valides conduit à construire récursivement des sous-espaces de $EC_{\mathcal{A}}$ réduits à un seul degré de liberté en rotation. Nous avons donc besoin d'un algorithme permettant de calculer l'ensemble des orientations valides du $n$ième composant A_n du manipulateur lorsque q_n varie continûment dans un domaine Δ_{n,k_n}.

$$EC_{A_n}^{q_n} = Reduction\left(EC_{A_n}^{\Delta_{n,k_n}}\right)$$

Algorithme de réduction des contraintes

Lorsque tous les composants de la sous-structure $\mathcal{A}_{n-2}$ sont figés et que le composant A_{n-1} est libre de se mouvoir dans un intervalle $\Delta_{n-1,l_{n-1}}$, la position spatiale du composant mobile A_n est localement définie par une configuration $\varsigma = (p_n, q_n)$, où p_n est la position de son point de référence et q_n la valeur articulaire spécifiant son orientation. Nous notons D le domaine local de variation de p_n lorsque q_{n-1} varie dans $\Delta_{n-1,k_{n-1}}$, également dépendant de la géométrie de la liaison entre A_{n-1} et A_n.

Essayant de construire localement l'espace libre, on cherche donc l'ensemble des orientations valides de A_n relativement à un obstacle B, pour une position p donnée :

$$EL_{A_n}^{p}(B) = \{q \in I_n \mid \mathcal{E}(A_n, (p,q)) \cap B = \emptyset\}$$

Si B est un composant solide élémentaire, $EL_{A_n}^{p}(B)$ est un unique intervalle fermé inclus dans I_n. Si B est un objet (union de composants) ou un ensemble d'objets (voire tous ceux de l'environnement), $EL_{A_n}^{p}(B)$ est un ensemble d'intervalles fermés non connexes de I_n. Ces intervalles sont alors des D_{n,k_n}, ensembles de secteurs angulaires calculés à l'aide de la fonction $DEBAT$ décrite précédemment.

Mais la position de référence p varie continument dans le domaine D : on cherche donc en fait l'ensemble

$$EL_{A_n}^{D}(B) = \{q \in I_n \mid \forall p \in D,\ \mathcal{E}(A_n, (p,q)) \cap B = \emptyset\}$$

Les contraintes d'orientation de A_n sur D sont donc égales à la conjonction des contraintes en chaque position p lorsque celle-ci décrit tout le domaine D. L'ensemble des orientations valides de A_n pour toute position p de D est donc obtenu comme l'ensemble

$$El_{A_n}^{D}(B) = \bigcap_{p \in D} EL_{A_n}^{p}(B)$$

Le problème est que cet intervalle, défini comme une intersection sur un ensemble continu d'intervalles, ne peut évidemment être calculé directement. La

technique adoptée pour évaluer cet ensemble consiste à calculer *un* intervalle d'orientations valides pour une position arbitraire de A_n, relativement à un obstacle équivalent induisant par sa morphologie les mêmes contraintes positionnelles. On se ramène donc au calcul de $EL^{p_R}_{A_n}(B)$ en réduisant le domaine D au seul point de référence p_R choisi arbitrairement et en grossissant les obstacles en conséquence :

$$El^{D}_{A_n}(B) = El^{p_R}_{A_n}(B^{\dagger}) \quad \text{avec} \quad B^{\dagger} = CO^{\tau\,D}_{A_n}(B)$$

Le calcul est donc ramené à un grossissement puis un et un seul calcul de débattement valide. Sa complexité algorithmique est par conséquent la somme des complexités des deux calculs en question. La justification de ce résultat repose sur l'égalité $CO^{\tau\,D}_{A_n}(B) = B \ominus A$ [1-Lau87]. La figure 6.6 illustre le principe du calcul et l'équivalence des contraintes.

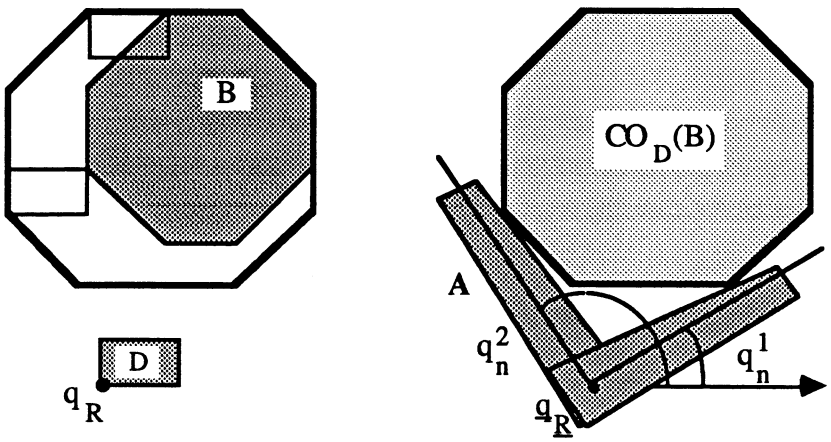

Figure 6.6: Calcul des contraintes d'orientation conjointes.

Il est à remarquer que la méthode, reposant sur un grossissement de l'obstacle, n'est applicable telle quelle au composant A_n que pour un domaine de variation D polyhédrique, ce qui n'est plus vrai lorsque le composant antérieur A_{n-1} est l'objet d' un mouvement en rotation. Dans ce cas, on peut se ramener au cas standard par approximation polyhédrique du domaine D : le grossissement effectué est alors majoré et on a $B^{\dagger} \supset CO^{\tau\,D}_{A_n}(B)$. En conséquence, l'espace libre obtenu est inévitablement surcontraint.

6.3 Application au cas d'une structure articulée

Nous utilisons maintenant les divers outils géométriques présentés afin de construire l'espace libre d'une structure articulée $\mathcal{A}$, soit calculer l'ensemble $CO_{\mathcal{A}}(\mathcal{B})$, ou plutôt les différents ensembles $CO_{\mathcal{A}}(B_p)$ correspondants aux obstacles de l'environnement.

6.3.1 Principe de construction des C-obstacles

Les ensembles $CO_{\mathcal{A}}(B_p)$ associés aux obstacles sont donc des hypervolumes de dimension N qu'il est très difficile de calculer de manière exacte. La méthode utilisée a donc pour but de les construire récursivement, en employant un procédé de partitionnement des $EC_{\mathcal{A}}$.

L'idée de base consiste à successivement prendre en compte les contraintes imposées par les B_p sur les différents degrés de liberté de $\mathcal{A}$. L'algorithme appliqué conduit à parcourir la structure $\mathcal{A}$ de sa base vers son extrémité libre. A chaque étape, le comportement du degré de liberté traité est analysé sur une *partition de son domaine de variation.* Seuls les éléments postérieurs du bras interviennent alors dans ce calcul.

Si les n degrés de liberté q_i ($1 \leq i \leq n$ avec $n \leq N-1$) sont fixés à des valeurs $q_i^{\nu_i}$, la contrainte appliquée par un obstacle B sur le sous-système $\{A_{n+1}, \ldots, A_N\}$ peut être représentée par l'ensemble :

$$CO_A^{(q_i^{\nu_i})_{i=1}^n}(B) = \{(q_{n+1} \ldots q_N) \in I_{n+1} \times \ldots \times I_N \mid (q_1^{\nu_1} \ldots q_n^{\nu_n}\ q_{n+1} \ldots q_N) \in CO_A(B)\}$$

Lorsque q_n décrit un petit intervalle Δ_{n,l_n} de I_n, la conjonction des contraintes appliquées par B sur le sous-système $\{A_{n+1} \ldots A_N\}$ est donc :

$$CO_A^{(q_i^{\nu_i})_{i=1}^{n-1}\,\Delta_{n,l_n}}(B) = \cup_{q_n \in \Delta_{n,l_n}} CO_A^{(q_i^{\nu_i})_{i=1}^{n-1}\ q_n}(B)$$

Lorsque l'on considère l'intervalle total de variation de q_n, I_n, partitionné en L_n intervalles connexes Δ_{n,l_n}, une approximation de $CO_A(B)$ peut être obtenue en calculant l'union des ensembles de type $CO_A^{(q_i^{\nu_i})_{i=1}^{n-1}\,\Delta_{n,l_n}}$:

$$CO_A(B) \approx \bigcup_{l_1=1}^{L_1} CO_A^{(q_i^{\nu_i})_{i=1}^{n-1}\Delta_{n,l_n}}(B)$$

Ce calcul permet de réduire la dimension des hypervolumes manipulés, chaque $CO_A^{\Delta_{n,l_n}}$ étant de dimension $N-n$. Le calcul effectif n'étant pas réalisable tant

que l'on a pas atteint un espace de dimension unitaire, on doit donc appliquer la méthode récursivement jusqu'à obtention des $CO_A^{(q_i^{\nu_i})_{i=1}^{N-1}\ \Delta_{N,l_N}}$, cette fois directement calculables, en l'occurence à l'aide de la fonction $DEBAT$. On obtient bien ainsi la chaîne de discrétisation annoncée au début de ce chapitre.

$$\langle D_{i,k_i} \rangle = CO_{\mathcal{A}}^{\Delta_{1,l_1} \times \ldots \times D_{N,l_N}}$$

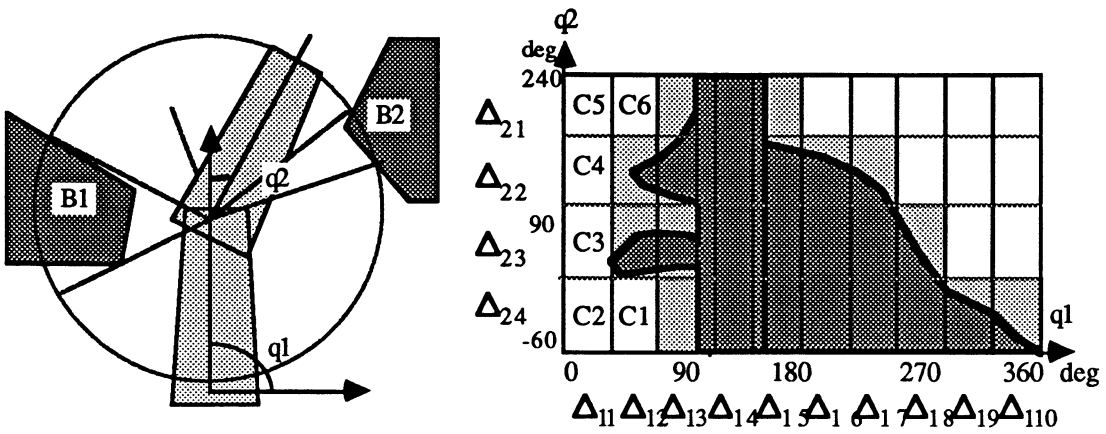

Figure 6.7: Exemple de calcul approché de $EC_{\mathcal{A}}$.

Le principe de construction approchée des $CO_{\mathcal{A}}(B_p)$ est illustré en figure 6.7. Dans cet exemple volontairement sommaire, la structure articulée est composée de deux éléments polygonaux en rotation évoluant dans un environnement composé de deux obstacles B_1 et B_2. I_1 a été uniformément partitionné en 10 intervalles $\Delta_{1,1}, \ldots, \Delta_{1,10}$ et pour chacun I_2 a été à son tour partitionné en 4 intervalles $\Delta_{2,1}, \ldots, \Delta_{2,4}$. Sur l'exemple, on peut voir pour une position fixée $q_1^{\nu_1} \in \Delta_{2,2}$ de A_1 quels sont les intervalles de débattement valide de A_2 calculés par la fonction $DEBAT$. Le calcul exact détermine en l'occurence 3 intervalles de validité $D_{2,1}$, $D_{2,2}$ et $D_{2,3}$ qui, du fait de la discrétisation plus que sommaire, se retrouvent réduits en faits à deux, correspondants sur le schéma aux cellules $C1$ et $C6$.

L'exemple choisi illustre au passage la perte d'information occasionnée par l'opération de discrétisation, soit la différence entre l'espace libre réel (dont les frontières sont représentées par les courbes en gras) et celui obtenu (les cellules non grisées).

Le raisonnement récursif décrit ci-dessus définit la base d'un support algorithmique pour la construction des C-obstacles. Toutefois, afin d'être exploitable,

cette méthode doit être pourvue d'outils algorithmiques permettant d'une part de *calculer* localement les contraintes appliquées par les obstacles B_p sur un élément A_n de la structure articulée, et d'autre part de *propager* ces contraintes vers les éléments postérieurs de la structure.

6.3.2 Calcul des contraintes

Le calcul des contraintes de position imposées par les obstacles est réalisé au moyen de la fonction $DEBAT$, et des opérateurs de *grossissement* et de *réduction* détaillés au paragraphe précédent.

A une étape n donnée de l'algorithme ($2 \leq n \leq N$), on ne considère que le sous-système $\mathcal{A}_n$, ignorant ainsi les éléments postérieurs de la structure articulée $\mathcal{A}$. La position des éléments antérieurs $A_1 \ldots A_{n-2}$ est fixée et celle de A_{n-1} est limitée au domaine de variation $\Delta_{n-1,l_{n-1}}$ attribué à A_n.

L'application de la fonction $QREF$ permet alors de choisir le point de référence $q_{n\,R}$ de A_n dans Δ_{n,l_n}. Pour des raisons de symmétrie, il est commode de le prendre sur l'axe de déplacement de l'articulation de A_n. Dès lors, le composant A_n se voit associé :

- un déplacement de $q_{n\,R}$ sur un domaine Δ_{n,l_n} créé par le débattement $\Delta_{n-1,l_{n-1}}$ associé à A_{n-1}, et
- un mouvement en translation ou en rotation suivant l'articulation de A_n réalisé dans le domaine I_n de variation de la variable articulaire q_n.

Si l'articulation de A_{n-1} est de type rotoïde, le domaine $\Delta_{n-1,l_{n-1}}$ est un arc de cercle; si elle est de type prismatique, le domaine est un segment de droite. Dans tous les cas, la position de ce domaine dans l'espace cartésien est déterminée par la configuration $(q_1 \ldots q_{n-2})$ du sous-système $\mathcal{A}_{n-2}$. On calcule alors l'espace libre "local" à l'intérieur de l'espace des configurations courant :

$$EC_{local} = \{q_1^{\nu_1}\} \times \ldots \times \{q_{n-2}^{\nu_{n-2}}\} \times \Delta_{n-1,l_{n-1}} \times I_n$$

L'opérateur de réduction permet de se ramener à un espace de type $C_{A_n}^{q_n}$ en grossissant les obstacles relativement à l'intervalle $\Delta_{n-1,l_{n-1}}$. Le calcul des débattements possibles pour q_n relativement à tout obstacle B peut alors être approximé par l'ensemble

$$El_{A_n}^{\Delta_{n-1,l_{n-1}}}(B) = El_{A_n}^{q_{n\,R}}(B^\dagger) \quad \text{avec} \quad B^\dagger = CO_{A_n}^{\tau\,\Delta_{n-1,l_{n-1}}}(B)$$

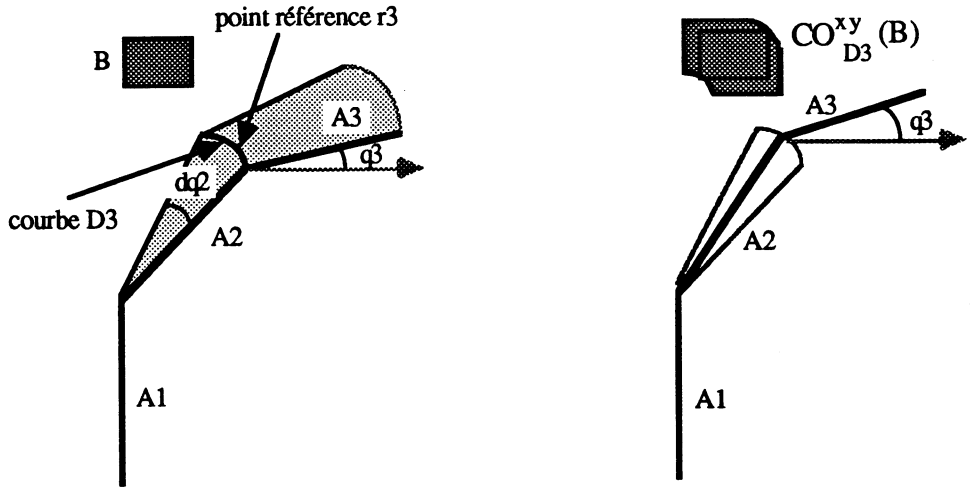

Figure 6.8: Calcul des contraintes de débattement d'un élément de $\mathcal{A}$.

La méthode conduit à des approximations qui restent raisonnables dans la mesure où les domaines de variation élémentaires choisis comme partition des I_n sont petits.

Dans la pratique, cet ensemble n'est calculé explicitement que pour le dernier degrés de liberté de la structure. Le traitement des autres variables articulaires donne lieu uniquement à l'application de l'opérateur de grossissement et à l'exécution d'un test d'interférence. La figure 6.8 [?] illustre un exemple d'un tel calcul pour une structure articulée dont les degrés de liberté sont rotoïdes. On peut voir que l'opération de grossissement engendre effectivement l'apparition de deux domaines circulaires dont l'un est concave. Ces domaines, calculés en fait dans le plan contenant le mouvement de rotation, sont donc remplacés par une approximation polygonale : un segment de droite est substitué à l'arc de cercle concave alors que l'arc extérieur est décomposé en plusieurs segments (dont le nombre dépend de l'écart angulaire associé).

6.3.3 Propagation des contraintes

Principe de propagation

Le procédé de calcul des contraintes défini ci-dessus opère de manière locale. Afin de répercuter les effets de ces contraintes sur les autres éléments de la structure articulée, les contraintes calculées doivent être propagées le long de la structure, de la base vers l'extrémité. Cette opération est rendue nécessaire par

le fait que les contraintes imposées par l'environnement sur un élément A_n sont toujours dépendantes de celles imposées sur les éléments antérieurs $A_1 \ldots A_{n-1}$. En particulier, le domaine $\Delta_{n-1,l_{n-1}}$ balayé par le point référence $q_{n\,R}$ de A_n lorsque les q_i décrivent les intervalles Δ_{i,l_i} $(1 \leq i \leq n-1)$, est une surface complexe de $\Re^3$ qu'il est difficile de représenter avec précision. Un calcul direct de l'ensemble $El_{A_n}^{\Delta_{n-1,l_{n-1}}}(B)$ n'est alors pas réellement envisageable.

Le processus de propagation de contraintes exposé ci-après permet de réaliser ce calcul en plusieurs étapes. Chacune permet, par des calculs simples, de construire itérativement des approximations de l'ensemble $CO^{\tau}_{\Delta_{n-1,l_{n-1}}}(B)$. Le principe consiste à appliquer l'opérateur de grossissement pour chaque $\Delta_{i,l_i}(1 \leq i \leq n-1)$, en prenant pour base de calcul le résultat de l'étape qui précède.

Chaque opération conduit alors à grossir les "obstacles courants" par un domaine $\Delta^i_{n-1,l_{n-1}}$ représentant le lieu décrit par $q_{n\,R}$ lorsque q_i décrit Δ_{i,l_i}. Le domaine $\Delta_{n-1,l^i_{n-1}}$ est réduit à un segment de droite dans le cas d'une articulation prismatique et à un arc de cercle pour une articulation rotoïde.

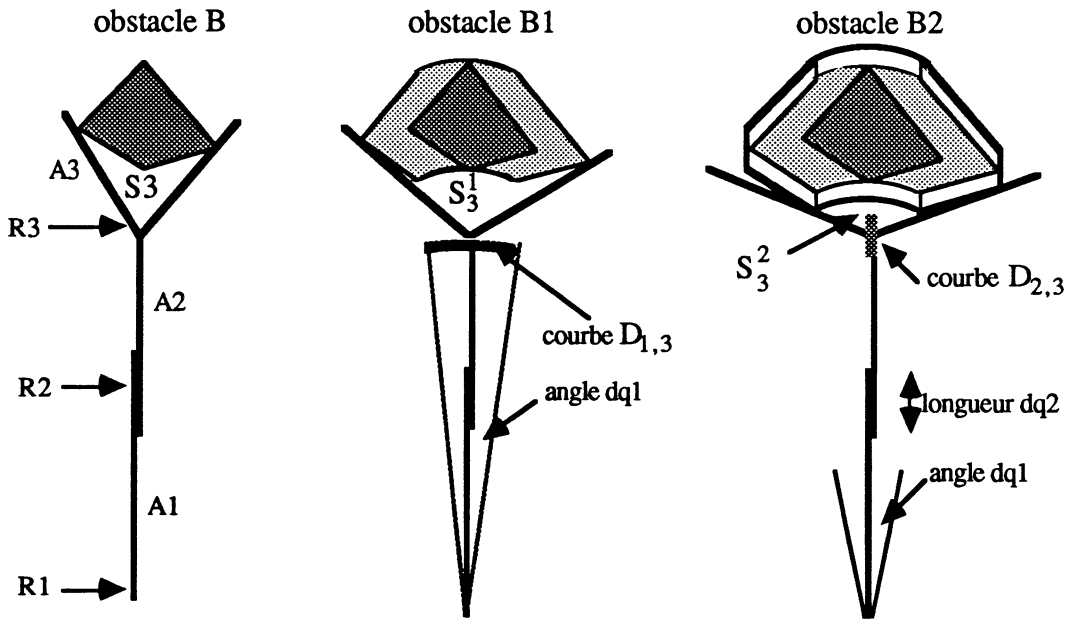

Figure 6.9: Propagation des contraintes de position d'une structure articulée.

La figure 6.9 [1-Lau87] illustre le processus de propagation des contraintes. Lorsque $q_1 = \nu_1$ et $q_2 = \nu_2$, la contrainte appliquée par B sur A_3 est représentée par le secteur angulaire S_3. Lorsque q_1 décrit Δ_{1,l_1}, donc varie sur un écart angulaire dq_1, cette contrainte est représentée par le secteur S_3^1 défini par l'obstacle

$B(1)$. Lorsque q_1 varie de dq_1 et q_2, décrivant l'intervalle Δ_{2,l_2}, varie de dq_2, la contrainte équivalente est représentée par le secteur S_3^2 défini par l'obstacle $B(2)$.

Algorithme de propagation

L'algorithme adopté pour calculer $El_{A_n}^{\Delta_{n-1,l_{n-1}}}(B)$ est le suivant :

$B(0) = B^h$
Calculer une séquence d'obstacles équivalents définis par la relation de récurrence

$$B(i) = CO_{\mathcal{A}_n}^{\tau\,\Delta_{n,l_n^i}}(B(i-1)) \quad \text{pour} \quad 1 \leq i < n$$

Puis déterminer les contraintes de débattement

$$El_{A_n}^{q_{n-1}\,R}(B(n-1))$$

dans lequel chaque obstacle est initialisé par sa représentation courante B^h (où h désigne le niveau de hiérarchie spatiale de la représentation).

Afin de pouvoir appliquer les algorithmes de grossissement précédents, les arcs de cercle sont approximés par des segments de droites. Cette approximation supplémentaire reste raisonnable du fait de la faible amplitude des débattements élémentaires. Dans le cas contraire, une approximation polynomiale peut être utilisée, conduisant à segmenter l'arc en morceaux de droite.

Nous disposons maintenant d'un algorithme de calcul et de propagation de contraintes permettant de répercuter globalement sous forme de grossissement des obstacles les contraintes positionnelles appliquées à la structure articulée.

Le prédicat $GROSS(n, D, B)$ permettant précisément de calculer une approximation des C-obstacles a donc la forme générale ci-dessus, que l'on peut écrire :

$$GROSS(n, D, B) := CO_{\mathcal{A}_n}^{q_{n-1}\,R}(B) \mid q_{n-1\,R} := QREF(D)$$

où $QREF(D)$ est donc le lieu géométrique du point de référence de A_n.

Selon les principes évoqués au paragraphe précédent, le processus de propagation de contraintes devrait être appliqué pour chaque élément A_n et pour chaque domaine de type $\Delta_{1,l_1} \times \Delta_{n-1,l_{n-1}}$. Dans la pratique, afin de minimiser les calculs à effectuer et réduire la complexité de l'algorithme, on cherche un majorant des domaines Δ_{n,l_n}^i.

Pour une chaîne de discrétisation donné $\Delta_{1,l_1} \times \Delta_{n-1,l_{n-1}}$, ce majorant correspond au plus grand déplacement cartésien δ exécuté par les points de référence $q_{n\,R}$ $(1 \leq n \leq N-1)$. Le domaine considéré peut alors être majoré de manière isotrope par une sphère de rayon δ. Ce procédé permet, au prix d'une nouvelle perte de précision, là encore raisonnable si l'amplitude des débattements élémentaires (taille des domaines engendrés par $PARTN$) reste faible, d'exécuter un seul grossissement d'obstacles par étape, et réduire ainsi la complexité algorithmique de la méthode.

$$B^{\dagger} = Gros_{\,\delta}(B) \quad \text{avec} \quad \delta = \max_{n=1\ldots N} \|\psi(q_{n\,R})\|$$

Il suffit dans ces conditions de calculer δ et le prédicat $GROSS(n, D, B)$ s'écrit alors simplement

$$GROSS(n, D, B) := Gros_{\,\delta(n,D)}\,(B)$$

Complexité algorithmique

Les paramètres du système sont :

- N, le nombre de degrés de liberté de la structure articulée A (dépendant de la nature du robot utilisé),
- N_C, le nombre moyen de composants élémentaires constituant les P obstacles de l'environnement (dépendant de la modélisation),
- N_a, le nombre moyen d'arêtes d'une de ces primitives (dépendant de la nature des composants utilisés),
- N_K, le nombre moyen d'intervalles constituant une partition de chaque intervalle de variation I_n d'un degré de liberté (dépendant de la discrétisation effectuée).

La complexité de l'algorithme complet peut alors être estimée en

$$O\left(N_K^{N-1} \cdot N_C \cdot N_a^2\right)$$

Il est clair que le facteur prépondérant est le nombre de degrés de liberté du robot, mais l'on peut voir également l'influence capitale de la discrétisation sur les performances du système.

Chapitre 7

Recherche d'une solution dans l'espace de planification

7.1 Description du graphe des configurations

7.1.1 Structuration de l'espace libre

Structuration complète de l'espace libre

Lors de la phase précédente, l'application de la fonction $LIBRE$ aux obstacles de l'environnement a permis d'obtenir soit un unique espace $EL(\mathcal{B})$ lorsque les obstacles ont été considérés ensembles, soit un ensemble de P espaces $EL(B_p)$ relatifs à chacun des obstacles lorsque ceux-ci ont été considérés séparément.

Dans ce deuxième cas, il importe maintenant de fusionner les différents $EL(B_p)$ sous forme d'une unique arborescence d'intervalles de validité constituant l'espace libre total, ceci par la fonction $UNION$.

$$EL_{\mathcal{A}} = UNION\left(\{EL(B_p)\}_{p=1}^{P}\right)$$

On a donc :

$$EL_{\mathcal{A}} = \cap_{p=1}^{P}\, EL(B_p) = \cap_{p=1}^{P}\, \langle D_{n,k_n} \rangle_p$$

Ceci peut être fait en calculant la conjonction des contraintes par intersection.

Ainsi, pour un degré de liberté q_n, on doit calculer pour chaque chaîne de discrétisation $(D_{1,k_1} \ldots D_{n-1,k_{n-1}})$ un ensemble de sous-intervalles de I_n correspondant aux valeurs admissibles de q_n en tenant compte de tous les obstacles, à partir des P ensembles d'intervalles $(D_{i,k_i^p})_p$ (avec $1 \leq k_i^p \leq K_i^p$) correspondant aux débattements articulaires valides pour chacun des obstacles B_p. Il suffit donc

de calculer leur intersection :

$$(D_{n,k_n}) = \cap_{p=1}^{P}(D_{i,k_i})_p$$

où les D_{n,k_n} constituent une sous-séquence ensembliste des D_{i,k_i^p}, au nombre maximal de $\max(K_i^p)$.

Algorithme de restructuration

Ce calcul des domaines (D_{n,k_n}) peut être effectué de manière récurrente. En effet, si l'on note

$$(D_{n,k_n})^r = \cap_{p=1}^{r}(D_{i,k_i^p})_p \quad \text{avec } 1 \leq r \leq P$$

on peut initialiser la récurrence de manière évidente en prenant exactement

$$(D_{n,k_n})^1 = (D_{i,k_i})_1$$

et chaque itération conduit alors à évaluer

$$(D_{n,k_n})^{r+1} = (D_{n,k_n})^r \cap (D_{i,k_i})_{r+1}$$

On est donc ramené au problème du calcul de l'intersection de deux ensembles d'intervalles totalement ordonnés, qui peut être résolu par un algorithme de tri analogue à un mécanisme de fusion ou d'*interclassement*, de complexité $O(K_n^r + K_i^{r+1})$.

L'idée est la suivante : dans un premier temps on fusionne les deux ensembles de manière à obtenir une séquence ordonnée de bornes d'intervalles, ceci en conservant pour chaque valeur une information de type *borne inférieure* (BInf) ou *borne supérieure* (BSup). Chaque ensemble étant une liste alternée de bornes inférieures et supérieures, l'ensemble obtenu par fusion est donc structuré en séquences alternées de bornes inférieures et supérieures (une séquence comprenant au minimum un élément). Dans un deuxième temps, on élimine alors les valeurs superflues en parcourant la séquence dans l'ordre croissant et en ne gardant que le dernier élément d'une séquence de bornes inférieures et le premier élément d'une séquence de bornes supérieures. On obtient bien à nouveau une alternance de bornes inférieures et supérieures, qui sont de plus telles que leur appariement par couple donne bien la séquence d'intervalles correspondant à l'intersection cherchée.

L'algorithme de fusion donné ci-après condense les deux phases en une seule afin de ne parcourir qu'une seule fois les ensembles donnés en entrées.

Formellement, si D_{n,k_n} est de la forme $\{[q^0_{1,k}\ q^1_{1,k}]\}$ $(1 \leq k \leq K)$ et de même D_{i,k_i} est de la forme $\{[q^0_{2,l}\ q^1_{2,l}]\}$ $(1 \leq l \leq L)$, alors les ensembles ordonnés considérés sont donc $E_1 = \{q^0_{1,k}\ q^1_{1,k}\}$ et $E_2 = \{q^0_{2,l}\ q^1_{2,l}\}$. La méthode proposée permet d'obtenir alors un ensemble E ordonné de la forme

$$E = \{(q^0_{\alpha,i})\ (q^1_{\beta,j})\} \quad \text{avec } \alpha,\beta \in \{1,2\} \ \wedge\ i,j \in \{1 \ldots K\} \cup \{1 \ldots L\}$$

d'où on extrait l'ensemble cherché (donné directement par l'algorithme) :

$$E^{\dagger} = \{\max(q^0_{\alpha,i})\ \min(q^1_{\beta,j})\}$$

procédure FUSION (E_1, E_2)
début
/ p est un pointeur* **global** *initialement de valeur NIL et de Type = BInf */*
si $(E_1 = \emptyset$ **ou** $E_2 = \emptyset)$: **retourner** $\emptyset$;
si $(\min E_1 < \min E_2)$: $e := \min E_1$; $A := E_1 - \{e\}$; $B := E_2$;
si $(\min E_1 = \min E_2)$: $e := \min E_1$; $A := E_1 - \{e\}$; $B := E_2 - \{e\}$;
si $(\min E_1 > \min E_2)$: $e := \min E_2$; $A := E_1$; $B := E_2 - \{e\}$;
si $(Type(p) = BInf$ **et** $Type(e) = BSup)$:
retourner $\{p\} \cup \{p := e\} \cup FUSION(A,B)$; */* bornes valides */*
sinon
$p := e$; **retourner** $FUSION(A,B)$;
fin si
fin

La figure 7.1 illustre sur un exemple bidimensionnel l'équivalent graphique de cette conjonction : les "masques" représentant les deux espaces libres $EL(B_1)$ et $EL(B_2)$ sont superposés puis combinés par une opération de **et** logique afin d'obtenir l'espace libre correspondant à la scène complète.

7.1.2 Définition, adjacence et hiérarchie du graphe de connectivité

Cellules et configurations

Nous représentons l'espace de planification par un *graphe de connectivité* $\mathcal{G}$. Celui-ci constitue une structure de données privilégiée, permettant de représenter facilement l'ensemble des configurations de l'espace libre ainsi que leurs liens d'adjacence.

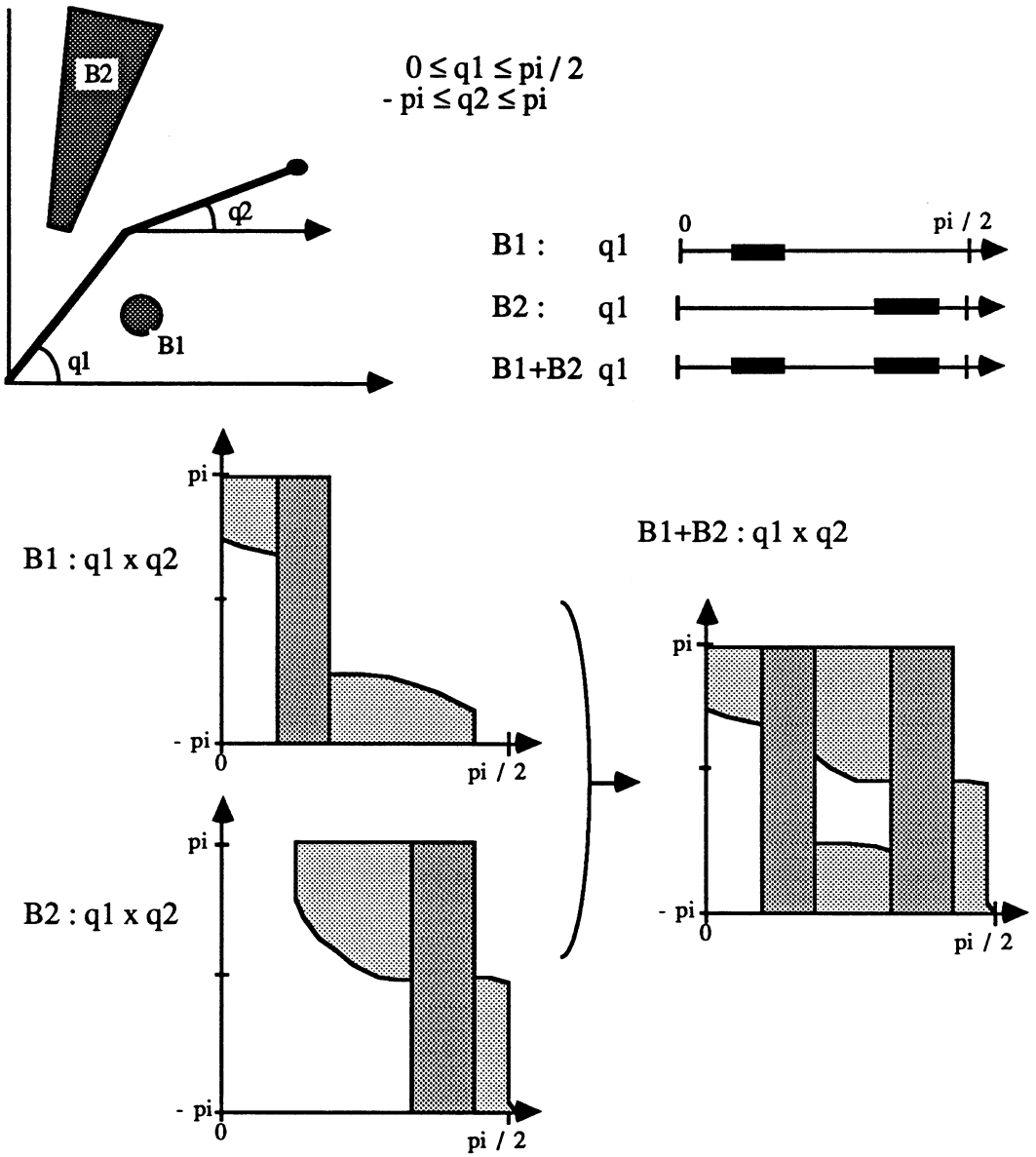

Figure 7.1: Exemple de réunion d'espaces libres individuels.

Une *cellule élémentaire* σ du graphe de connectivité est identifiée à une chaîne de discrétisation complète de l'espace libre EL, soit un N-uplet d'intervalles de validité

$$\sigma = (D_{1,k_1} \dots D_{N,k_N})$$

Une *configuration* $\varsigma = (q_1 \dots q_N)$ du manipulateur est alors un élément parti-

culier d'une cellule :

$$\varsigma \in \sigma \iff \forall n \in [1, N],\ q_n \in D_{n,k_n}$$

Nous retenons la définition d'une *configuration médiane* $\overline{\varsigma}(\sigma)$ dont la valeur de chaque degré de liberté q_n est centrée relativement à D_{n,k_n}, soit :

$$\text{si } D_{n,k_n} = [q^0_{n,k_n} q^1_{n,k_n}],\ \overline{\varsigma}(\sigma) = \left(\frac{1}{2}(q^0_{n,k_n} + q^1_{n,k_n})\right)$$

Il est nécessaire, afin de pouvoir estimer le coût du passage d'une configuration à une autre, de définir une métrique sur l'ensemble des configurations. Nous définissons donc une *distance articulaire* ∇_q simplement comme la norme de l'écart articulaire :

$$\nabla_q(q_{n,1}, q_{n,2}) = \mid q_{n,1} - q_{n,2} \mid$$

Il vient alors naturellement une *distance d'intervalle* ∇_D ainsi définie :

$$\nabla_D(D_{1,n,k_n}, D_{2,n,k_n}) = \min\left(\nabla_q(q_{n,1}, q_{n,2}) \mid q_{n,1} \in D_{1,n,k_n}, q_{n,2} \in D_{2,n,k_n}\right)$$

et une "distance médiane" (de calcul plus rapide mais moins "exact") $\overline{\nabla_D}$:

$$\overline{\nabla_D}(D_{1,n,k_n}, D_{2,n,k_n}) = \mid \frac{1}{2}(q^0_{1,n,k_n} - q^0_{2,n,k_n} + q^1_{1,n,k_n} - q^1_{2,n,k_n}) \mid$$

La norme de configurations, définie comme la norme quadratique de l'écart articulaire, ne donne qu'une indication globale de leur différence, sans préciser sur quel degrés de liberté celle-ci porte en particulier. Aussi définissons-nous la fonction ∇_ς retournant la liste des couples *(variable articulaire . distance articulaire)* :

$$\nabla_\varsigma(\varsigma_1, \varsigma_2) = (\ (n.\nabla_q(q_{1,n}, q_{2,n}))\ )_{1 \le n \le N}$$

Adjacence des cellules du graphe

Nous définissons maintenant une *relation d'adjacence*, notée $\bowtie$, entre cellules du graphe.

Deux cellules sont dites adjacentes s'il existe au moins une configuration appartenant à toutes deux. Lorsque tel est le cas, il est clair que le manipulateur peut passer de l'une à l'autre sans engendrer de collision.

Soit, formellement, si $\sigma_1 = (D^1_{n,k^1_n})$ et $\sigma_2 = (D^2_{n,k^2_n})$ sont deux cellules distinctes de $\mathcal{G}$, alors :

$$\sigma_1 \bowtie \sigma_2 \iff \forall n \in [1, N],\ D^1_{n,k^1_n} \cap D^2_{n,k^2_n} \neq \emptyset$$

La définition peut évidemment être utilisée pour un sous-système $\mathcal{A}_n$ en substituant N par n.

La fonction $ADJACENT$ permettant de mettre à jour la liste des voisins de chaque cellule détermine alors de manière récursive les relations d'adjacence entre cellules du graphe :

$$(D_{i,k_i})_n \bowtie \left(D_{j,k_j}\right)_n \iff \begin{array}{l} (D_{i,k_i})_{n-1} \bowtie \left(D_{j,k_j}\right)_{n-1} \\ \wedge \\ D_{i,k_i} \cap D_{j,k_j} \neq \emptyset \end{array}$$

L'adjacence "en profondeur" est donnée par la construction même des chaînes de discrétisation. Il reste donc à déterminer l'adjacence "en largeur", ce qui est fait à l'aide de la fonction $ADJACENT$ en parcourant de manière récursive l'arborescence de EL. Cette opération est de complexité exponentielle en le nombre de cellules du graphe.

Dans le cas d'une discrétisation uniforme (ce qui est notre cas pour un niveau de représentation donné), les relations d'adjacences entre cellules sont définies implicitement par la donnée des $\{D_{n,k_n}\}$ et du pas de discrétisation.

Le graphe de connectivité représente l'ensemble des cellules libres structuré au moyen de la relation d'adjacence. Chaque nœud du graphe représente une cellule libre : il est caractérisé par une position articulaire de référence (la configuration médiane) et un N-uplet d'intervalles de validité associés aux q_n (les D_{n,k_n}). Chaque arc représente une adjacence de cellules.

Deux nœuds du graphe sont reliés par un arc si et seulement si ils sont associés à deux cellules libres adjacentes; la relation étant symétrique, le graphe obtenu est donc non-orienté.

Hiérarchie du graphe

Nous avons vu que la représentation géométrique des objets de la scène est hiérarchisée par l'utilisation de H_N niveaux de modélisation. Pour chacun de ces niveaux peut être calculé un espace libre EL^h caractérisé par la précision de sa définition, et le graphe de connectivité associé est donc également hiérarchisé.

On dispose par conséquent d'une séquence de graphes imbriqués

$$\mathcal{G}^1 \subset \ldots \subset \mathcal{G}^{H_N}$$

Toute cellule définie pour un niveau h donné appartient à tous les graphes $\mathcal{G}^h$ pour $h \leq H$ et possède une liste d'adjacences hiérarchisée, c'est-à-dire une liste des cellules adjacentes de même niveau, complétée d'un liste des cellules de niveau $H+1\ldots$ jusqu'à une liste des cellules de niveau maximal H_N.

7.1.3 Chemins et trajectoires

Dès lors, un chemin entre divers nœuds du graphe représente une enveloppe de trajectoire possible du manipulateur, soit un ensemble connexe $(\sigma_1 \ldots \sigma_{NP})$ de cellules libres dans lequel le robot peut se déplacer sans risque de collision, depuis une configuration initiale incluse dans la cellule σ_1 à une configuration finale incluse dans la cellule σ_P.

Le passage entre deux cellules consécutives σ_i et σ_{i+1}, $(1 \leq i \leq NP)$, est effectué en des configurations communes choisies dans l'ensemble $\sigma_i \cap \sigma_{i+1}$. La transition à l'intérieur d'un même cellule est définie par un trajet continu entre les positions extrémales choisies.

Le graphe des configurations définit en fait l'ensemble des chemins possibles pour le robot au niveau de résolution choisi. Il peut être facilement modifié par adjonction de nouvelles cellules, ceci se produisant lorsque ce niveau de résolution est augmenté pour un état de l'univers donné ou lorsque l'on change d'état.

La détermination d'une trajectoire effective se décompose en trois étapes : la recherche d'un chemin dans le graphe, la détermination d'une enveloppe de trajectoires incluse dans la séquence de cellules libres proposée, et enfin le choix d'une trajectoire particulière à l'intérieur de cette enveloppe. Nous verrons plus loin que les options prises influent de beaucoup sur l'allure des trajectoires planifiées.

7.2 Parcours du graphe de connectivité

7.2.1 Méthodes de parcours de graphes

Notre préoccupation essentielle dans la détermination d'une trajectoire de transfert pour le robot est de trouver une solution satisfaisant les conditions de positions initiales et finales. Cela est d'autant plus vrai dans le contexte de la programmation automatique que, comme nous l'avons dit précédemment, le module de planification de trajectoires est contraint par les autres planificateurs

intermédiaires. L'optimalité de la solution ne nous semble donc pas une contrainte impérative du système tant qu'elle ne fait pas explicitement partie des conditions d'accomplissement de la tâche d'assemblage.

Et d'ailleurs, selon quel critère doit-on véritablement optimiser une trajectoire? Le temps, la distance parcourue, la distance aux obstacles sont autant de réponses possibles mais antagonistes. Dans le cas d'un manipulateur en particulier, la minimisation de la distance parcourue par le préhenseur n'est pas du tout synonyme de minimisation du temps de parcours et il paraît impossible a priori de trancher en faveur de l'une ou l'autre...

Aussi, lorsqu'aucune contrainte particulière ne vient orienter le choix d'une trajectoire, une méthode de type "breadth-first search" nous satisfait a priori pleinement, puisqu'elle garantit de trouver une solution, qui plus est optimale en terme de chemin dans le graphe, c'est-à-dire de déplacements articulaires. La stratégie acceptée est alors de minimiser les mouvements du robot, ce qui semble un comportement tout à fait correct.

Cependant, si la nature de la trajectoire n'est pas réellement imposée, il peut être alors intéressant de minimiser son temps de calcul. L'aspect trop exhaustif de la méthode précédente nous conduit alors à utiliser des algorithmes plus performants, en l'occurence un algorithme de type A^*. Les temps de parcours du graphe sont alors meilleurs, mais pas de façon véritablement décisive. D'autre part, leur emploi implique un inconvénient réel : celui de *devoir* choisir un ensemble cohérent de fonctions de coût, lesquelles doivent posséder un certain nombre de propriétés mathématiques. Ce choix peut effectivement être vu comme une contrainte lorsque la notion de coût d'une solution est vide de sens. Aussi dans ce cas, et toutes les fois que les contraintes (ou l'absence de) portant sur la nature de la trajectoire le permettent, nous avons pris l'option d'utiliser l'algorithme le plus simple à notre disposition.

7.2.2 Algorithme A^* et paramétrisation

Principe de l'algorithme

On doit donc choisir une fonction d'évaluation estimant le coût minimal du chemin passant le nœud courant n. On note $k(n_i, n_j)$ le coût minimal du chemin entre deux nœuds, $g(n) = k(n_I, n)$ le coût du chemin depuis le nœud initial jusqu'au nœud courant et $h(n) = k(n, n_F)$ le coût de ce dernier au nœud final. On note encore g^* l'estimée de g à une étape de l'algorithme (soit le plus court

chemin de n_I à n parmi ceux déjà calculés) et h^* l'estimée de h au même instant (déterminée de manière heuristique). La fonction d'évaluation utilisée vaut alors $f(n) = g(n) + h(n)$, et elle peut être estimée à $f^*(n) = g^*(n) + h^*(n)$.

Ainsi, lors de l'expansion d'un nœud du graphe, le choix du successeur devant être traité sera effectué en minimisant f^*. La liste des nœuds non encore examinés est dite *ouverte* (et notée $\mathcal{O}$) et celle des nœuds déjà traités est dite *fermée* (et notée $\mathcal{F}$). La trace du chemin optimal est conservée grâce à un pointeur depuis le nœud développé vers son meilleur prédécesseur.

Remarquons que si $\forall n$, $g(n) = d(n)$ et $h(n) = C$ où d est la profondeur du nœud n depuis la racine n_I et C une constante, l'algorithme A est alors ramené à un "breadth-first search". Toutefois ceci n'est équivalent à une minimisation de la distance articulaire que lorsque les incréments de ces derniers sont constants et identiques sur chaque articulation.

Lorsque le graphe est fini, ce qui est le cas du graphe des configurations (par construction), l'algorithme A garantit de trouver une solution lorsqu'elle celle-ci existe. Enfin, si h est une borne inférieure de h^* (soit $\forall n, 0 \leq h(n) \leq h^*(n)$) la solution trouvée devient optimale en terme de chemin, et l'algorithme est alors dit *admissible* et nommé A^*.

La seule condition requise pour la fonction de recherche heuristique h est d'être une mesure de distorsion. Avec la restriction de monotonie, définie par l'inégalité triangulaire $h(n_i) \leq h(n_j) + k(n_i, n_j)$, la condition devient pour h d'être une distance. Dans ce cas, la propriété $\forall n$, $g(n) = g^*(n)$ est vérifiée, ce qui se traduit par le fait qu'il n'est plus nécessaire de vérifier si le successeur d'un nœud choisi par A^* est bien dans $\mathcal{O}$: celui-ci y est forcément, car l'expansion est alors effectuée de manière optimale.

Pseudo-programme A^*

Nous présentons ici une description synthétique de la version de l'algorithme A^* [6-Nil80] utilisée pour planifier une trajectoire dans le graphe de connectivité - dans le cas où h est une distance.

```
procédure TRAJ(G, n_I, n_F)
début
O := (n_I) ; F := ∅ ; calculer f*(nI) ;
tant que O ≠ ∅ :
    choisir n ∈ O tel que ∀n_j ∈ O, f*(n) < f*(n_j) ;
    déplacer n de O dans F ;
    si n = n_F : retourner trajectoire := liste-des-pointeurs(n ··· n_I) ;/* succès */
    S := liste-d'expansion(n) ;
    pour n_i ∈ S :
        f_i := g*(n) + k(n, n_i) + h*(n_i)
        si n_i ∉ O et n_i ∉ F :
            ajouter n_i dans O ; f*(n_i) := f_i ; prédécesseur(n_i) := n ;
        sinon
            si f*(n_i) > f_i :
                f*(n_i) := f_i ; prédécesseur(n_i) := n ;
                si n_i ∈ F : déplacer n_i de F dans O
            fin si
        fin si
    fin pour
fin tant que
retourner trajectoire := ∅ ;                                    /* échec */
fin
```

Appliquer cet algorithme sur le graphe de connectivité revient donc à construire sur celui-ci un graphe de recherche de taille inférieure, dans lequel les relations de précédence définissent le chemin optimal entre une configuration donnée et celle de départ.

Choix des fonctions de coût

Les fonctions de coût utilisées en pratique sont généralement basées sur la somme des distances articulaires ∇_q ou la norme de cellules $\| \varsigma_1, \varsigma_2 \|$ précédemment définies. En l'absence de consigne particulière, nous utilisons la distance médiane $\overline{\nabla_D}$ qui permet d'obtenir les meilleures performances.

La distance Euclidienne appliquée à la position du préhenseur est plus rarement utilisée. Cela arrive par exemple lorsqu'une des contraintes de la tâche exprime la minimisation d'une distance à un objet (suivi de contours).

Il existe quelques variantes de l'algorithme, portant essentiellement sur la gestion des listes $\mathcal{O}$ et $\mathcal{F}$. Pour notre part, cette dernière n'est d'ailleurs pas utilisée mais est remplacée avec profit par un mécanisme de marquage. Des ex-

tensions ont été proposées, visant à accroître encore ses performances : recherche bidirectionnelle, obtention d'autres solutions (non-optimales), modification des fonctions de coût g et h pendant le parcours du graphe même...

Sur ce dernier point, nous avons introduit une pondération dans la fonction de coût, dépendant de la proximité au but :

$$f^*(n_i) := \alpha\ g^*(n) + (1 - \alpha)\ h^*(n_i)$$

L'idée est de privilégier de plus en plus la partie rigoureuse de l'évaluation du coût (g^*) à mesure que l'on se rapproche du but (la partie heuristique h^* devient alors moins importante). Au départ, pour $\alpha = 0$, la méthode est équivalente à celle du gradient; pour $\alpha = \frac{1}{2}$, on a bien sûr un A^* pur, et pour $\alpha = 1$ on rejoint une méthode d'énumération [6-Nil80].

La méthode employée consiste à planifier une trajectoire avec un A^* pur, déterminer sa longueur (par exemple en termes de déplacement articulaire), et recommencer en modifiant α en fonction de cette information. On prendra par exemple au début $\alpha = \frac{1}{3}$ puis, lorsqu'on aura estimé avoir parcouru un tiers du parcours, on prendra $\alpha = \frac{1}{2}$ et aux deux tiers du chemin on choisira $\alpha = \frac{2}{3}$, et ainsi de suite...

L'emploi de cette technique peut s'avérer fort intéressant dans le cas où le système est amené à effectuer plusieurs recherches de trajectoire sur le même graphe, c'est-à-dire dans le même environnement robotique. Les valeurs de α peuvent alors être mémorisées et optimisées, certains diraient "apprises", afin de réduire le temps de recherche d'une trajectoire.

7.3 Choix d'une trajectoire

7.3.1 Principes du choix d'une trajectoire

Etant donné la nature même des cellules utilisées, on dispose à l'issue de la phase de recherche précédente d'un ensemble de trajectoires potentielles. Le choix de l'une d'entre elles parmi toutes celles réalisables implique la détermination d'une suite continue de configurations aptes à définir un mouvement du robot. Chaque cellule étant un hyper-parallélépipède de l'espace des configurations, il est possible de joindre en ligne droite n'importe quel point de sa frontière. Choisir une trajectoire à l'intérieur d'une cellule est donc équivalent à choisir les valeurs d'entrée et de sortie de la cellule, lesquelles assurent la transition avec les cellules précédente et suivante.

Le choix le plus immédiat - mais tout à fait arbitraire - consiste à prendre les configurations médiane et extrémales (minimum ou maximum). Les déplacements ainsi planifiés sont malheureusement souvent trop artificiels, avec des changements de direction (articulaire) brusques conduisant à une génération d'une trajectoire saccadée.

Aussi préférons nous diriger ce choix par l'application d'une fonction de coût locale. Par exemple, pour deux cellules adjacentes σ_i et σ_{i+1}, on retient comme configuration de passage celle ς^o_{i+1} qui minimise la distance avec la dernière configuration choisie ς^o_i ($\|\varsigma^o_i\ \varsigma^o_{i+1}\|$).

Une autre solution, que nous n'avons pas retenu en raison de son coût élevé, est d'améliorer le choix des configurations de transitions en utilisant le résultat des fonctions de coût qui ont permis la détermination d'une séquence de configurations optimale. Le critère alors appliqué (la minimisation de la distance d'intervalles ∇_D par exemple) permet, pour peu qu'on ait gardé trace des évaluations de coût effectuées, de retenir pour chaque cellule les configurations véritablement optimales. L'inconvénient de cette méthode réside évidemment dans le calcul des ∇_D.

7.3.2 Prise en compte des contraintes physiques

Contraintes statiques

Nous entendons par contraintes statiques celles portant sur la nature géométrique des cellules déterminées par l'algorithme de planification de trajectoire. Ainsi, même en calculant "au mieux" les configurations de transition entre cellules, il est fréquent que la trajectoire obtenue soit quelque peu heurtée... Ceci est essentiellement du au choix d'une trajectoire "rectiligne" à l'intérieur d'une cellule.

L'utilisation d'une distance de type ∇_D permet d'améliorer considérablement la trajectoire, mais le coût de calcul associé pénalise lourdement la planification d'un chemin. De plus la minimisation des distances parcourues reste locale à chaque cellule et ses voisines. Une solution pour remédier à cela serait de calculer une trajectoire lissée à l'aide de fonctions spline, et passant par les points fixes précédemment déterminés aux frontières des cellules (une méthode des moindres carrés peut également être utilisée). Une telle méthode n'est donc possible que si l'on a déjà déterminé les points de passage. L'idéal serait de pouvoir calculer une trajectoire à l'aide de splines d'ordre 2 avec pour seules contraintes de passer

par les intervalles frontières des cellules. Malheureusement, un tel résultat ne peut être obtenu de manière analytique, et même une tentative de résolution numérique ne serait possible qu'à condition d'accepter des limitations importantes sur la nature des fonctions splines utilisées...

Contraintes dynamiques

Nous entendons par contraintes dynamiques celles portant sur la nature de la commande devant être utilisée pour déplacer le robot le long de la trajectoire planifiée. Si a priori la recherche d'une trajectoire peut amener l'ensemble des degrés de liberté à varier simultanément, la commande associée n'est pas toujours disponible au niveau du contrôleur utilisé.

Ainsi dans le cas de la cellule d'assemblage utilisée pour notre expérimentation, nous ne pouvons que commander isolément les degrés de liberté du manipulateur. Un mouvement simultané de plusieurs articulations doit alors être artificiellement décomposé en déplacements monoaxiaux qui tendent à "hâcher" la trajectoire. Il n'est pas non plus possible de transformer les commandes articulaires en commandes en position car, outre la complexité accrue de la tâche (du fait des appels alors systématiques à un changeur de coordonnées), on se heurterait à un inconvénient majeur : à savoir que les mouvements seraient exécutés en mode libre et donc sans contrôle. Tout mouvement autre que d'amplitude faible serait donc formellement proscrit. Cet inconvénient se retrouve en fait également dans la commande simultanée de plusieurs axes.

Il en résulte que l'aspect saccadé des trajectoires ne peut être éliminé.

La figure 7.2 illustre un exemple de trajectoire calculée pour l'évitement d'un obstacle élémentaire par un manipulateur à trois degrés de liberté. La vue 3 (en bas à droite) montre la scène et le robot en position initiale et finale. La figure 10.3, au dernier chapitre, permet de mieux apprécier les différentes configurations de la trajectoire.

L'espace des configurations tridimensionnel est représenté "à plat", q_1 et q_2 étant portés en abscisse et ordonnée discrètes (par intervalles de 15 degrés), et q_3 étant alors représenté à l'intérieur de chaque tranche de q_1. Les configurations interdites correspondent à la partie vide. La vue 1 (en haut à gauche) donne l'espace des configurations simplifié avec les configurations initiale (0 75 60) et finale (−90 70 40).

La trajectoire "lissée" obtenue en utilisant sans restriction les configurations médianes conduit à faire varier un ou deux degrés de liberté à la fois (vue 4, en bas à gauche). Comme la commande de déplacement articulaire disponible ne permet pas d'exécuter cette trajectoire, seule nous reste la trajectoire "brute" obtenue en limitant les variations des degrés de liberté à un unique mouvement articulaire (vue 2, en haut à droite), et dont l'aspect saccadé apparaît nettement.

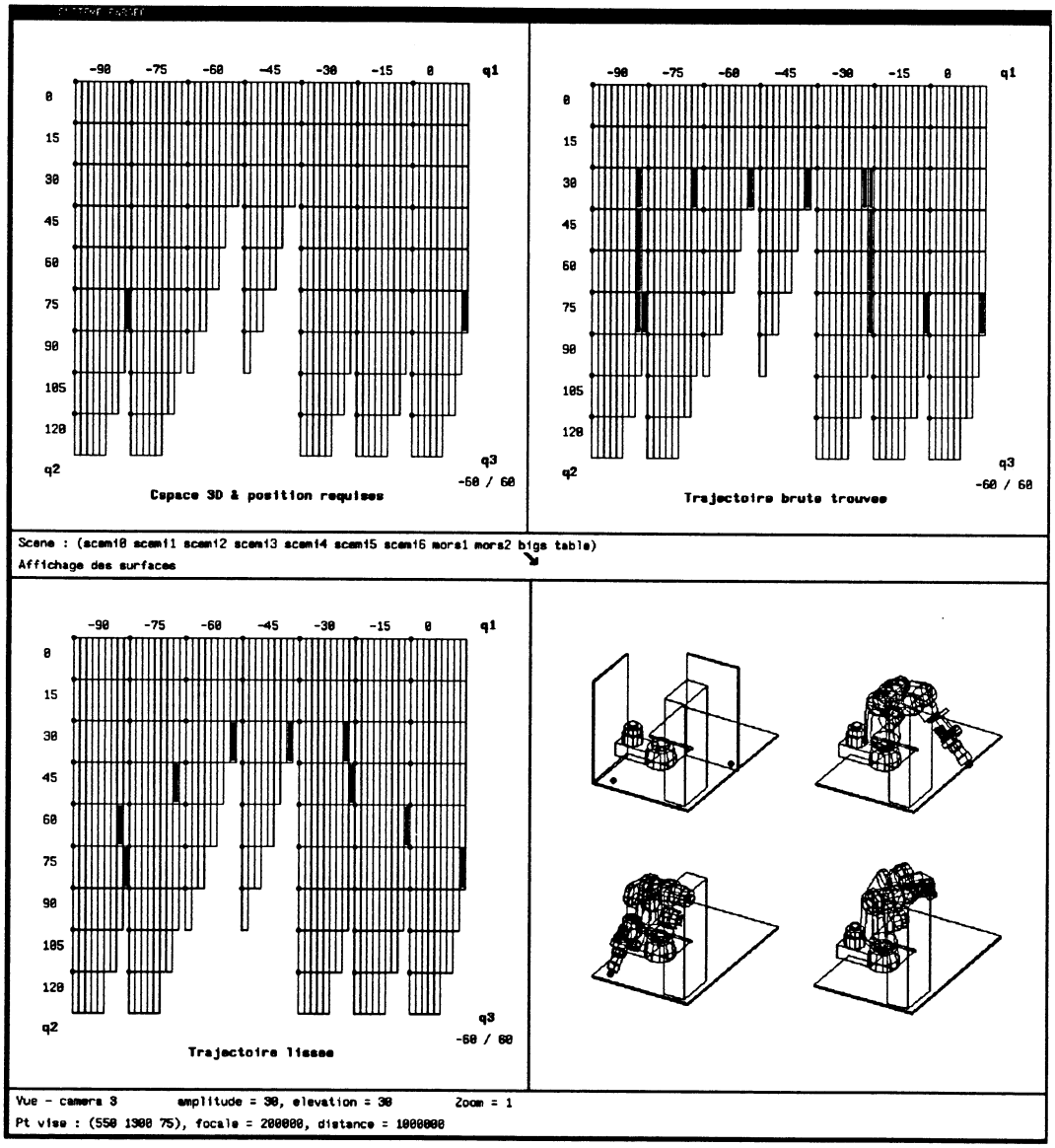

Figure 7.2: Exemple de détermination d'une trajectoire.

Chapitre 8

Méthode locale de planification de trajectoires

L'approche globale se révélant inapplicable au cas général, nous avons donc développé une méthode mixte basée sur la précédente et faisant appel à une méthode locale. Celle-ci, dite des champs de potentiel, se présente comme très délicate à maîtriser. Ce chapitre a pour but de l'étudier isolément afin de mieux appréhender les problèmes de paramétrisation qu'elle pose invariablement.

8.1 La méthode du champ de potentiel

8.1.1 Description de la méthode

Cette méthode a été initialement utilisée dans le contexte de la commande dynamique des robots manipulateurs par O.Khatib [14-Kha85]. Le but poursuivi était de pouvoir disposer au niveau de la commande de l'effecteur de fonctionnalités de déplacement intégrant les contraintes d'évitement d'obstacles, permettant d'alléger ainsi d'autant les niveaux de commande supérieurs (niveau manipulation, voire objet).

L'idée de la méthode est de considérer que *le robot évolue dans un champs de forces conditionnant la nature de ses déplacements.* Les forces appliquées sont issues de potentiels fictifs associés à l'environnement, potentiels répulsifs pour les obstacles et attractif pour la position à atteindre.

Dans cette optique, les obstacles doivent être décrits sous une forme analytique permettant la déduction du potentiel associé. Le gradient de ce potentiel est alors interprété comme une force de répulsion et la force résultante obtenue par

additivité, du fait de l'ensemble des obstacles de l'environnement, est considérée comme une commande en force venant s'ajouter à la commande nominale.

Au niveau du système de commande adaptatif qui pilote le robot, l'implantation "matérielle" de la méthode se traduit par l'adjonction de boucles de rétroaction dont l'entrée est constituée par les variables opérationnelles et dont la sortie agit sur la commande comme une perturbation additive.

Bien sûr, la planification de trajectoires devant en ce qui nous concerne être effectuée hors ligne, la méthode ne peut qu'être employée exclusivement en simulation. Si dans le cas dynamique, la force assignant à la position finale le rôle d'un pôle attractif est donc contenue dans le vecteur de commande, elle doit être dans notre cas imposée de manière arbitraire. D'autre part le gradient d'un potentiel ne doit plus alors être interprété comme une force mais comme une vitesse ou une accélération.

8.1.2 Formalisation de la méthode

Nous rappelons ici quelques éléments de mécanique analytique constituant la base d'une théorie générale utilisable en commande dynamique, pour en dériver ensuite les équations spécifiques de notre problème.

Equation générale

La théorie Lagrangienne des champs relève d'un principe de moindre action qui exprime que l'intégrale d'une densité (le Lagrangien $\mathcal{L}$, fonction des champs et de leurs dérivées) est stationnaire pour des conditions aux limites appropriées.

Si χ est une variable opérationnelle du système, le Lagrangien s'exprime comme la différence de l'énergie cinétique $T(\chi, \dot{\chi})$ et de l'énergie potentielle généralisée $U^*(\chi)$ du système :

$$\mathcal{L} = T(\chi, \dot{\chi}) - U^*(\chi) \quad \text{avec} \quad T(\chi, \dot{\chi}) = \frac{1}{2} \dot{\chi}^T \Lambda(\chi) \dot{\chi}$$

où $\Lambda(\chi)$ est la matrice d'énergie cinétique. L'application du principe ci-dessus nous permet d'écrire l'équation de Lagrange pour un système holonome sollicité par des forces potentielles ou non :

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\chi}}\right) - \frac{\partial \mathcal{L}}{\partial \chi} = 0$$

et si F est le vecteur de forces généralisées appliquées à ce dernier, il vient

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\chi}}\right) - \frac{\partial T}{\partial \chi} + \frac{\partial \mathcal{U}}{\partial \chi} = F$$

L'équation fondamentale de la dynamique ainsi obtenue décrit le mouvement du robot, sous la forme plus classique :

$$F = \Lambda(\chi)\,\ddot{\chi} - \frac{1}{2}\,\dot{\chi}^T \frac{\partial \Lambda(\chi)}{\partial \chi}\,\dot{\chi} + \frac{\partial U(\chi)}{\partial \chi}$$

Le potentiel total U est ici obtenu comme somme des potentiels répulsifs associés aux obstacles et du potentiel attractif choisi pour traduire la contrainte du but à atteindre. Ce faisant, nous négligeons le potentiel gravifique devant les potentiels fictifs introduits; il suffit pour justifier ce choix de définir ces derniers suffisamment grands.

Cas statique

En statique, l'équation ci-dessus indique bien que la force à appliquer au robot est égale au gradient du potentiel auquel il est soumis.

$$F = \frac{\partial U(\chi)}{\partial \chi} = grad\ U(\chi)$$

Chaque objet de l'univers étant modélisé par l'union de primitives élémentaires, l'environnement est donc globalement représenté par un ensemble de P composants bien définis, et le potentiel total considéré vaut :

$$U(\chi) = G(\chi) + \sum_{i=1}^{P} U_i(\chi)$$

où G est donc le potentiel associé au but et les U_i ceux associés à chaque primitive géométrique. La force d'attraction-répulsion globale vaut donc

$$F = F_G(\chi) + \sum_{i=1}^{P} F_i(\chi) \quad \text{avec} \quad \begin{array}{rcl} F_G(\chi) & = & grad\ G(\chi)\ \ et \\ F_i(\chi) & = & grad\ U_i(\chi) \end{array}$$

Il reste donc à définir ces potentiels pour savoir comment "diriger" le robot. Ainsi, à un instant donné, pour un environnement donné, l'évaluation des potentiels appliqués permet de définir une direction et une intensité (ou une vitesse[1]) de déplacement.

[1] Nous préférons le terme d'*intensité* de déplacement par cohérence avec l'intensité du potentiel total auquel est soumis le robot. Il s'agit bien de la norme de la vitesse instantanée, égale pour une unité de temps à la distance à parcourue.

8.1.3 Caractéristiques des potentiels fictifs

Les potentiels répulsifs

Les caractéristiques imposées aux potentiels fictifs sont les suivantes : chaque fonction $U_i(\chi)$ doit impérativement

- être définie non négative,
- être continue, dérivable en tout point de son espace de définition,
- s'annuler en χ=χ_G (lorsque le but est atteint),
- tendre à l'infini à la surface S_i de son obstacle associé.

D'autre part, bien que n'étant pas indispensable, il est assez logique d'ajouter la condition de monotonie : plus on est prêt de la source et plus le potentiel est élevé.

Dans le cas des potentiels répulsifs, nous estimons utiles de restreindre le potentiel U_i d'un composant à une zone, dite *zone d'effet*, délimitée par un point particulier $\chi_{L,i}$. Ceci sera argumenté plus loin. Ainsi la forme générale d'un potentiel devient :

$$\begin{cases} U_i(\chi) = \frac{1}{n}\kappa_i \left(\frac{1}{f_i(\chi)} - \frac{1}{f_i(\chi_{L,i})}\right)^n & \text{si } f_i(\chi) \leq f_i(\chi_{L,i}) \\ U_i(\chi) = 0 & \text{si } f_i(\chi) > f_i(\chi_{L,i}) \end{cases}$$

où n est un entier non nul et f_i une fonction de coût (une distance par exemple) relative à la géométrie du composant, et qui respecte les conditions aux limites suivantes :

$$\lim_{\chi \to S_i} f_i(\chi) = 0 \quad et \quad \lim_{\chi \to \chi_{L,i}} f_i(\chi) > 0$$

de telle sorte que l'on ait bien :

$$\lim_{\chi \to S_i} U_i(\chi) = \infty \quad et \quad \lim_{\chi \to \chi_{L,i}} U_i(\chi) = 0$$

La force de répulsion associée est alors la suivante :

$$\begin{cases} F_i(\chi) = -\kappa_i \left(\frac{1}{f_i(\chi)} - \frac{1}{f_i(\chi_{L,i})}\right)^{n-1} \frac{f_i'(\chi)}{f_i^2(\chi)} & si\ f_i(\chi) \leq f_i(\chi_{L,i}) \\ F_i(\chi) = 0 & si\ f_i(\chi) > f_i(\chi_{L,i}) \end{cases}$$

Le potentiel attractif

Les conditions imposées aux potentiels répulsifs ne s'appliquent pas identiquement au potentiel attractif G, mais doivent être reconsidérées. En effet, si ce dernier est défini de façon purement arbitraire, il doit néanmoins être compatible avec les conditions précédemment établies.

Ainsi la monotonie (globale ou établie) du potentiel doit être inversée : plus on est loin du but et plus le potentiel attractif doit être élevé. La fonction G doit donc être définie négative.

De plus les conditions aux limites deviennent différentes. L'annulation du potentiel attractif lorsque le but est atteint entraîne en effet des difficultés de convergence : au voisinage de la position finale, l'attraction - quasiment nulle - peut être facilement contrebalancée par une force de répulsion rémanente. Il est donc indispensable d'ajouter une composante continue, sous la forme d'un potentiel additif constant.

D'autre part, il n'est pas de position définie où G doive tendre à l'infini. En effet si la contrainte impérative de non-collision avec les obstacles impose la présence d'une barrière de potentiel à leur contact - l'éloignement immédiat devenant alors une priorité absolue -, il n'existe aucune contrainte équivalente pour le potentiel attractif. Par contre, la limite infinie doit être obtenue lorsque le système est en trajectoire d'"échappement". En effet en un lieu où le potentiel répulsif des obstacles est de loin prépondérant sur l'attraction du but, le système doit pouvoir se diriger à l'opposé de la direction souhaitée.

Le choix de la fonction potentiel G est beaucoup plus vaste que pour les U_i. Nous retenons la forme générale suivante :

$$G(\chi) \;=\; -\frac{1}{n}\kappa_G \left(\gamma + f_G(\chi)\right)^n$$

où n est un entier non nul, γ une constante strictement positive, et où f_G est une distance analogue aux f_i relative à la position finale à atteindre. Les conditions aux limites décrites précédemment sont donc respectées :

$$\lim_{\chi\to\infty} G(\chi) \;=\; \infty \;\; et \;\; \lim_{\chi\to\chi_G} G(\chi) \;=\; Cte \;=\; (\frac{\gamma^n}{n})$$

La force d'attraction associée est alors la suivante :

$$F_G(\chi) \;=\; -\kappa_G \left(\gamma + f_G(\chi)\right)^{n-1} f'_G(\chi)$$

8.2 Représentation des obstacles et potentiels utilisés

8.2.1 Représentation analytique des surfaces des obstacles

La représentation géométrique de base utilisée dans notre modèle est une arborescence de primitives volumiques assemblées par l'unique opération d'*union*. Ceci présente, dans le cadre de la méthode des champs de potentiel, un double intérêt. D'une part cela nous permet d'estimer effectivement le potentiel global comme la somme de potentiels associés à chacune des primitives élémentaires (ce qui serait absolument impossible avec une autre représentation), et d'autre part la nature même de ces primitives permet d'établir une équation analytique de ces potentiels.

Une première solution consiste à associer à chaque surface $S_{i,j}(i = 1 \ldots s)$ d'une primitive B_i une fonction potentiel $U_{i,j}$. Alors la fonction de coût $f_{i,j}$ associée peut être simplement l'équation $\mathcal{S}_{i,j}(\chi)$ de la surface $S_{i,j}$.

Ainsi par exemple, si la variable opérationnelle χ est la position de l'effecteur d'un bras articulé, on aura pour chaque surface plane une équation du type :

$$\mathcal{S}_{i,j}(\chi) \equiv ax + by + cz + d = 0 \quad avec \quad \chi = (x\ y\ z)$$

De la même manière, on peut définir les équations des surfaces cylindriques, côniques ou sphériques présentes dans le modèle.

En pratique, cette décomposition des primitives en surfaces élémentaires se révèle inadéquate, et ce pour deux raisons. Tout d'abord, elles ne permettent pas de prendre en compte les limites de l'objet. Ainsi une face plane est strictement délimitée dans l'espace, ce que ne traduit pas son équation de surface, laquelle devrait alors être alors complétée d'inéquations algébriques.

D'autre part, il est difficile de déterminer comment doivent être combinés les potentiels surfaciques ainsi définis. Dans l'exemple de la figure 8.1, les potentiels sont représentés sur le schéma de gauche par les équations de surface du parallélépipède, et sur celui de droite par une surface analytique approximant celle de l'objet. Dans le premier cas, si le choix des potentiels répulsifs est assez naturel dans les régions A et B (f_1 et f_2 s'imposant exclusivement), il est par contre plus délicat dans la région C. Un choix possible est de prendre une fonction f égale à

une combinaison linéaire de f_1 et f_2 de telle sorte que les conditions aux limites (dans le prolongement des surfaces de l'objet) soient respectées. Par exemple, si α désigne l'angle entre d'une part la droite délimitant A et C et d'autre part celle liant le coin $A\ B\ C$ à la position courante, on a :

$$f = \frac{2}{\pi}\left((\frac{\pi}{2} - \alpha)f_1 + \alpha f_2\right)$$

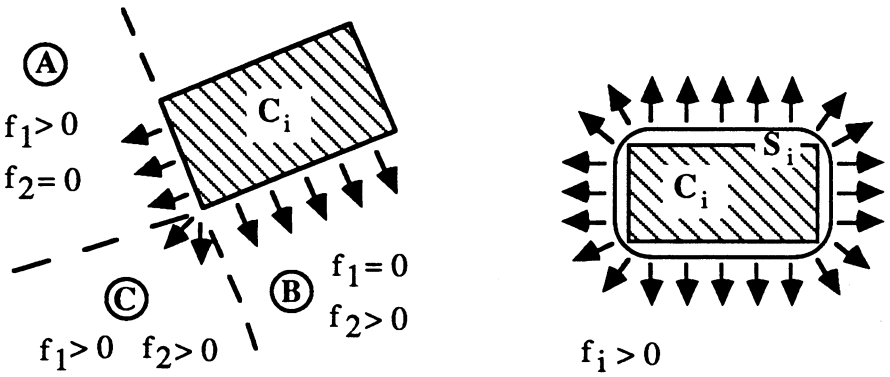

Figure 8.1: Choix des surfaces de potentiel.

La définition et la manipulation des équations de surface reste cependant délicate, en particulier lorsque l'on doit combiner les potentiels issus de plusieurs objets.

Dans le deuxième cas de figure, l'objet est approximé par une seule surface. Le potentiel agit donc de manière homogène dans toutes les directions normales à la surface analytique choisie comme modèle. La représentation gagne alors en concision et en simplicité de calcul (donc en performances).

Ainsi, une autre solution, plutôt que de représenter un objet élémentaire par la somme de ses surfaces, consiste à l'approximer par une unique surface polynômiale.

Le parallélépipède peut alors être approché par un n-ellipsoïde d'équation générale :

$$\left(\frac{x}{a}\right)^{2n} + \left(\frac{y}{b}\right)^{2n} + \left(\frac{z}{c}\right)^{2n} = 1$$

Le cylindre peut de même être approché par un n-cylindre d'équation :

$$\left(\frac{x}{a}\right)^{2} + \left(\frac{y}{b}\right)^{2} + \left(\frac{z}{c}\right)^{2n} = 1$$

De la même façon le cône peut être approché par un n-cône d'équation :

$$\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 + \left(\frac{1}{2}\left(\left(\frac{z}{c}\right)^8 + \left(\frac{z}{c}\right)^7 - \left(\frac{z}{c}\right) + 1\right)\right)^2 = 1$$

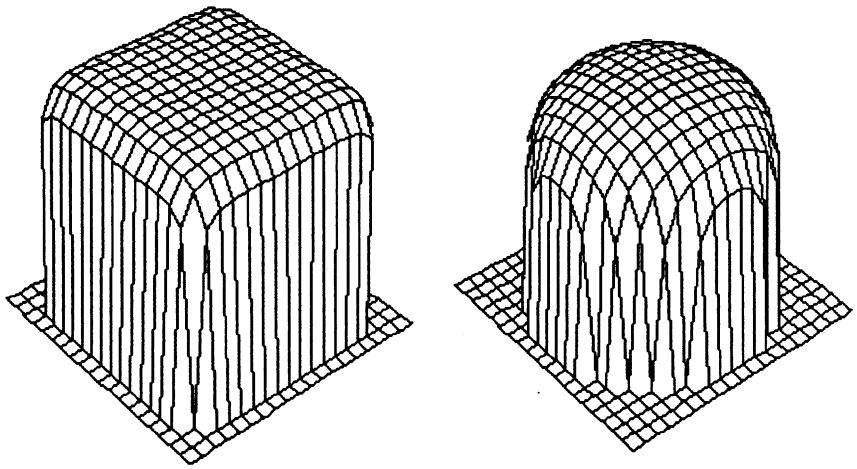

Figure 8.2: Exemple de surfaces approximant les primitives volumiques du modèle.

La figure 8.2 montre un n-ellipsoide et un n-cylindre obtenus pour $n = 4$. Le problème majeur de cette représentation réside dans le calcul des distances f_i. Il faut effet, pour avoir une approximation raisonnable des volumes, choisir $n \geq 4$. Ce choix entraîne que l'ordre des équations polynomiales de surface est supérieur ou égal à 8.

Dans ces conditions, le calcul de $f_i(\chi)$ se ramène à la résolution d'un système du même ordre (seulement possible de manière numérique, mais inapplicable ici pour des raisons de performances). Dès lors, le calcul de $\mathcal{S}(\chi)$ ne peut plus être employé pour évaluer efficacement la distance à la surface. D'autre part, une telle représentation analytique ne permet pas de représenter facilement les primitives obtenues par balayage.

L'utilisation d'équations approchant de manière plus ou moins exacte les surfaces des objets de l'univers ne se révélant pas satisfaisante, nous nous orientons vers l'utilisation d'une représentation sphérique venant se substituer aux primitives utilisées jusqu'alors.

8.2.2 Représentation sphérique des obstacles et potentiels associés

Intérêt de la représentation sphérique

Un des intérêts majeurs de la représentation sphérique est que l'équation de surface de la sphère est celle d'un 2-ellipsoïde, et la fonction de surface peut cette fois en être directement dérivée :

$$\mathcal{S}_i(\chi) \equiv \left(x^2 + y^2 + z^2\right) - R^2 = 0$$

Autre avantage important : le calcul de la distance $f_i(\chi)$ est extrêmement facile et rapide.

Choix du potentiel

L'analogie de la commande dynamique offre l'exemple de l'asservissement proportionnel, le plus simple, qui conduit à une correction de la commande de consigne sous la forme suivante :

$$F_i(X) = \kappa_i(X - X_o)$$

et donc au choix d'un potentiel de la forme :

$$U_i(X) = \frac{1}{2}\kappa_i(X - X_{L,i})^2$$

Ceci nous ramène à la forme générale précédente avec $n = 2$ et entraîne cette fois :

$$\begin{cases} U_i(\chi) = \frac{1}{2}\kappa_i \left(\frac{1}{f_i(\chi)} - \frac{1}{f_i(\chi_{L,i})}\right)^2 \mid f_i(\chi) \leq f_i(\chi_{L,i}) \\ F_i(\chi) = -\kappa_i \left(\frac{1}{f_i(\chi)} - \frac{1}{f_i(\chi_{L,i})}\right) \frac{f_i'(\chi)}{f_i^2(\chi)} \mid f_i(\chi) \leq f_i(\chi_{L,i}) \end{cases}$$

Il n'est pas possible de choisir une valeur telle que $n = 1$ (par analogie cette fois avec la loi d'attraction universelle), car un potentiel du premier ordre (analogue au champs de gravitation) entrainerait une force d'attraction/répulsion qui ne s'annule pas en $\chi_{L,i}$.

$$\begin{cases} U_i(\chi) = \kappa_i \left(\frac{1}{f_i(\chi)} - \frac{1}{f_i(\chi_{L,i})}\right) \mid f_i(\chi) \leq f_i(\chi_{L,i}) \\ F_i(\chi) = -\kappa_i \frac{f_i'(\chi)}{f_i^2(\chi)} \mid f_i(\chi) \leq f_i(\chi_{L,i}) \end{cases}$$

8.2.3 Problèmes et solutions spécifiques

Permanence et superposition de potentiels

Sans la condition d'annulation du potentiel U_i en $\chi_{L,i}$, celui-ci s'appliquerait en tout point de l'espace des variables opérationnelles. L'influence d'un obstacle ou de l'un de ses composants serait alors ainsi permanente, même lorsque le robot en est physiquement très éloigné.

A l'opposé, le choix d'une limite très contraignante revient à créer de grandes zones "libres" où seule s'exerce le potentiel attractif de la position terminale. L'intérêt principal des zones d'influence est surtout d'éviter ainsi le phénomène de *superposition de potentiel* dépendant de la décomposition initiale de l'objet et a fortiori lors de la décomposition en sphères.

L'exemple de la figure 8.3 illustre ce phénomène dans le cas typique ou un obstacle (ici C_j) occulte totalement un ou plusieurs autres obstacles (ici C_i). Dans un cas (schéma de gauche), tous les potentiels s'appliquent en tous les points de l'espace considérés : on n'a en l'occurence que deux objets élémentaires et la commande instantanée vaut $F = F_i + F_j + F_G$. Celle-ci aura alors tendance à éloigner très fortement l'effecteur de sa position finale. Dans le deuxième cas (schéma de droite), l'influence des potentiels répulsifs est sévèrement limitée par la donnée de zones limites : seul le potentiel de l'objet le plus proche influe sur le déplacement de l'effecteur et la commande vaut simplement $F = F_j + F_G$. Le déplacement du robot tend bien à s'éloigner de l'obstacle proche, mais seulement celui-là.

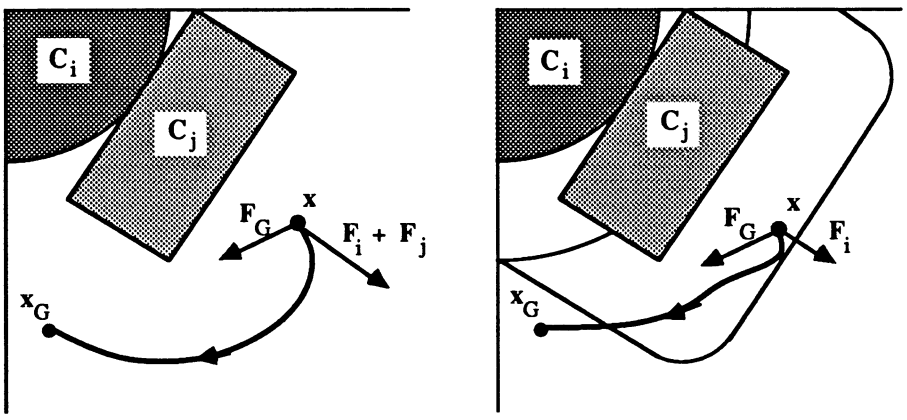

Figure 8.3: Intérêt des zones d'effet du potentiel.

Le choix de $\chi_{L,i}$ est donc un paramètre important et difficile, et il semble en

tout état de cause que la valeur $\chi_{L,i}=\chi_G$ soit une borne maximale pour laquelle le potentiel s'applique jusqu'au but à atteindre. En effet, s'il s'exerçait au delà, il tendrait à éloigner le robot de sa position terminale plutôt que de l'en approcher.

Notons que le choix d'une zone d'influence tendant vers zéro revient à donner une stratégie "de la ligne droite" selon laquelle le mobile se déplace toujours en direction du but et, lorsqu'il vient à rencontrer un obstacle, essaye de le contourner. En fait, le mobile effectue une sorte de suivi du contour de l'objet, à une distance ϵ garantissant l'absence de contact. Ceci marche d'ailleurs assez bien, surtout dans le cas des sphères où le suivi de la surface se fait toujours en se rapprochant du but. Mais cette stratégie se révèle totalement inefficace dans le cas des "culs-de-sac" (voir ci-après).

Répulsion infinie, "rebonds"

Les équations de potentiel des obstacles sont telles que la force de répulsion tend à l'infini lorsque l'on s'approche infiniment près de leur surface. Ceci peut être cause d'un phénomène de "rebond" (illustré par la figure 8.4) : la consigne d'éloignement d'un obstacle C_1 envoie le mobile sur un autre obstacle C_2 avec une force telle que ce déplacement l'amène au voisinage de ce dernier, lequel à son tour engendre une force répulsive très grande renvoyant sur l'obstacle d'origine, et ainsi de suite...

Une solution simple pour remédier à ce problème consiste à borner la consigne de déplacement F. Ce qui nous intéresse essentiellement dans le calcul de la force d'attraction-répulsion résultante, c'est la *direction* à suivre pour se rapprocher du but à atteindre. Aussi, borner l'intensité de chaque force répulsive ne présente aucun inconvénient. Les incréments de déplacements engendrés comme consigne sont ainsi bornés : si l'on s'approche a priori moins vite du but, on évite par contre les trajectoires sinueuses intempestives pour se recaler au mieux (par une sorte d'effet d'amortissement) sur la trajectoire que l'on suivra de toute façon globalement. La figure 8.4 illustre ces différents aspects.

Convergence vers le but

Le premier problème rencontré est celui de la convergence asymptotique. Il peut arriver que l'on oscille aux alentours de la position finale sans toutefois l'atteindre exactement. L'utilisation d'une *boule de proximité* permet alors de détecter cette situation, reconnaître le succès du déplacement en temps voulu et

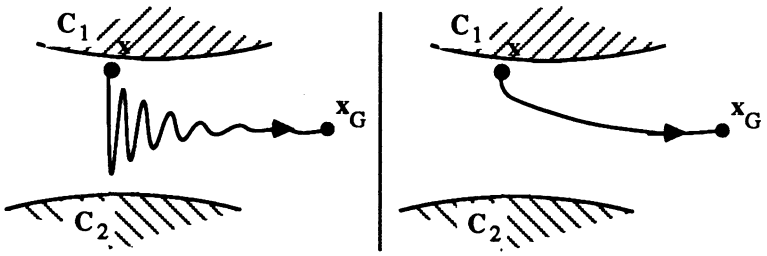

Figure 8.4: Illustration du phénomène de "rebond" potentiel.

par conséquent arrêter la recherche. Il ne s'agit en fait que de définir la précision à respecter dans la réalisation du but à atteindre.

Soit $\mathcal{B} = (O_{\mathcal{B}}, R_{\mathcal{B}})$ une telle boule définie par son centre O et son rayon ϵ_G, la détection du but se fait simplement par l'ajout, à chaque itération de la trajectoire, d'un test d'appartenance de la configuration finale à une boule de précision donnée (ϵ_G) :

$$\chi \equiv \chi_G \iff \chi \in \mathcal{B}_{O,\epsilon_G}$$

"Deadlocks" ou Culs-de-sacs

Il s'agit là du problème majeur de la méthode, illustrant parfaitement les limites d'une telle méthode locale. En l'occurence, le problème résulte de l'existence de puits de potentiels relatifs à la géométrie de l'environnement et les divers paramètres des fonctions de potentiel utilisées.

Il se traduit inévitablement par un certain bouclage de la trajectoire, conduisant le mobile à rester bloqué dans une zone de puits sans moyen de pouvoir en sortir par lui-même.

La résolution éventuelle du problème du cul-de-sac se présente en deux temps : détection puis remède.

Détection d'un cul-de-sac

La détection du cul-de-sac peut être effectuée grâce à l'utilisation de boules de proximité relatives à la position courante. Si χ_n désigne la position du mobile à l'étape n, nous définirons le cul-de-sac généralisé par l'obtention de la condition :

$$\exists N, \epsilon \mid \forall n > N, \left(\chi_n \in \mathcal{B}_{N,\epsilon}\right) \wedge \left(\chi_G \notin \mathcal{B}_{N,\epsilon}\right)$$

Un test pratique consiste à définir régulièrement le long de la trajectoire des

boules de proximité relatives à la position courante et vérifier dans quelle mesure ces boules s'intersectent ou non. Le problème de la paramétrisation s'avère là encore crucial : il s'agit de trouver l'adéquation entre la dimension des boules de proximité choisies et la géométrie de l'environnement afin de ne détecter aucun cul-de-sac "imaginaire" ni à l'opposé rester ignorant d'une situation de blocage.

Le fait de borner la fonction F permet de garantir l'obtention d'un cul-de-sac stable. La détection pratique du problème revient à celle d'une séquence de boules de proximités d' intersection non vide. Le calcul d'intersection des boules de proximité B_{m,ϵ_m} et B_{n,ϵ_n} correspondant à deux positions χ_m et χ_n se ramène au test :

$$|\chi_m - \chi_n| < \epsilon_m + \epsilon_n$$

Le rayon ϵ_n de la boule de proximité associée à l'étape n de la trajectoire peut être constant ou varier entre deux limites extrêmes en fonction de l'encombrement local de l'espace.

Résolution du problème

Un cul-de-sac se caractérise essentiellement par l'existence d'une concavité dans l'environnement, dont le point de sortie se trouve situé à l'opposé du but à atteindre. Ainsi, le potentiel répulsif des obstacles constituant une barrière totalement infranchissable, le mobile oscille entre rapprochement (lorsque l'attraction se fait prépondérante) et éloignement (lorsqu'à leur tour les obstacles prennent le dessus), pour se stabiliser à une position d'équilibre (en fait dans une boule de stabilité) correspondant à une résultante F nulle.

Une solution éventuelle est de supprimer radicalement la concavité existante, afin d'interdire l'entrée du mobile dans la zone de cul-de-sac et forcer un contournement à la base, par un effet de glissement sur la surface potentielle artificiellement ajoutée à cette fin.

Nous proposons pour cela une technique de modification des potentiels basée sur un grossissement local des obstacles dont la réunion constitue une concavité physique. Il s'agit de substituer localement à chaque obstacle impliqué dans le cul-de-sac un obstacle grossi auquel est associé un potentiel particulier calculé de telle sorte que, d'une part, il soit égal au précédent en tout point de l'espace "assez loin" de l'obstacle et que, d'autre part, il tende à l'infini à proximité de sa surface. Le but poursuivi est de modifier le moins possible la topologie du champ de potentiels global tout en créant au voisinage de la concavité un

potentiel suffisamment élevé pour en interdire l'entrée.

Un obstacle C_i, de potentiel associé U_i, est alors grossi localement d'une distance arbitraire R, et on crée un potentiel correctif V_i venant s'ajouter au sien propre.

$$U_i(\chi) = \tfrac{1}{2}\kappa_i \left(\frac{1}{f_i(\chi)} - \frac{1}{f_i(\chi_{L,i})}\right)^2 \mid f_i(\chi) \leq f_i(\chi_{L,i})$$
$$V_i(\chi) = \lambda_i \left(\frac{1}{(f_i(\chi)-R)} - \frac{1}{(f_i(\chi_{L,i}))}\right)^\alpha \mid f_i(\chi) \leq f_i(\chi_{L,i})$$

Par addition, le potentiel équivalent obtenu est donc :

$$U_i(\chi) + V_i(\chi) = \tfrac{1}{2}\kappa_i \left(\frac{1}{f_i(\chi)} - \frac{1}{f_i(\chi_{L,i})}\right)^2 + \lambda_i \left(\frac{1}{(f_i(\chi)-R)} - \frac{1}{(f_i(\chi_{L,i}))}\right)^\alpha$$
$$lorsque \quad f_i(\chi) \leq f_i(\chi_{L,i})$$

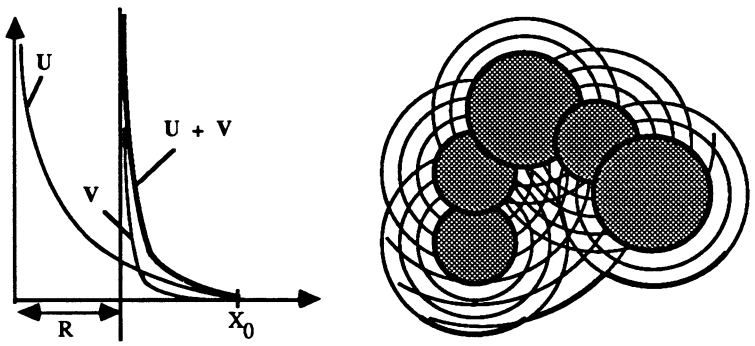

Figure 8.5: Exemple de réduction d'un cul-de-sac.

En choisissant α grand (supérieur à 5 par exemple), le terme V_i tend très rapidement vers zéro loin de la surface de l'objet grossi, de telle sorte que la modification du potentiel U_i reste faible. L'application du potentiel $U_i + V_i$ (et de la force de répulsion dérivée) aux obstacles de la concavité revient donc à leur substituer des obstacles équivalents grossis et, globalement, à "résorber" la concavité.

La méthode doit en général être appliquée récursivement : ayant détecté un cul-de-sac en une position χ_n, on revient à une position antérieure χ_{n-m} estimée hors de la concavité, on modifie alors les potentiels des obstacles environnants χ_n et on relance la recherche de trajectoire; en cas de nouvel échec, on reitère le processus.

La figure 8.5 illustre cette notion de grossissement des obstacles et de potentiel équivalent. Au bout de quelques itérations (dont le nombre dépend de la taille de

la concavité et du rayon de grossissement R choisi), on parvient ainsi à éliminer le cul-de-sac. Ceci ne fonctionne hélas pas dans le cas d'un environnement *très* contraint, lorsque la modification locale des obstacles, trop importante, entraîne la disparition de tout espace libre.

8.3 Algorithmes utilisés

8.3.1 Cas du déplacement d'un mobile

L'algorithme de base $POTENTIEL_0$ consiste donc, à partir d'une position initiale χ_I, à déplacer le mobile selon une direction et d'une amplitude dépendant du calcul des forces résultantes F_i associées aux obstacles et au but χ_G. Il opère sur un environnement $\mathcal{S}$ composé de N_S sphères S_i. L'algorithme de base est le suivant :

procédure POTENTIEL$_0(\mathcal{S}, \varsigma_I, \varsigma_G)$
début
χ=χ_I ; $F := 0$;
tant que $\chi \notin \mathcal{B}_{\chi_G,\epsilon_G}$:
  pour $i = 1 \ldots N_S$:
    $F := F + F_i(\chi)$; χ=$\chi + \max(F, F_M)$;
  fin pour
fin tant que
fin

Cet algorithme intègre la prise en compte des zones d'influence (dans la définition même des F_i), et contrôle les phénomènes de "rebond" (F bornée par F_M) et de convergence vers le but (boule de proximité $\mathcal{B}_{\chi_G,\epsilon_G}$). Il est toutefois totalement vulnérable aux blocages potentiels engendrés par les culs-de-sac.

La prise en compte des culs-de-sac donne la procédure $POTENTIEL_1$. Il ne s'agit que d'une heuristique visant à améliorer le comportement du planificateur, mais ne garantissant pas la résolution du problème. L'algorithme est alors transformé comme suit :

```
procédure POTENTIEL1(S, ς_I, ς_G)
début
χ=χ_I ; F := 0 ;
tant que χ ∉ B_{χ_G,ε_G} :
    pour i = 1...N_S :
        F := F + F_i(χ) ; χ=χ + max(F, F_M) ;
    fin pour
    B_n := B_{χ,ε_n} ; NB_∩ := 0 ;
    pour j = 1...n − 1 :
        si B_j ∩ B_n ≠ ∅ :
            NB_∩ := NB_∩ + 1 ;
            si NB_∩ > NB_{∩,MAX} :
                χ:=χ_{n−m}
                pour S_{i_n} ∈ Voisinage(χ) :
                    S'_{i_n} := S_{i_n}[U_{i_n} + V_{i_n} ←U_{i_n}]
                fin pour
                POTENTIEL1(S − {S_{i_n}} + {S'_{i_n}}, ς_I, ς_G)
            fin si
        fin si
    fin pour
fin tant que
fin
```

8.3.2 Application à une structure articulée

Dans le cas d'une structure articulée, il faut non seulement chercher une trajectoire pour son extrémité terminale (le but étant bien d'amener celle-ci à une position donnée), mais également pour toute la structure, tout en s'assurant de l'absence de collision de tout constituant.

L'idée de base consiste à appliquer les forces dérivées des potentiels attractifs et répulsifs au préhenseur, tout en contrôlant la position effective de chaque composant à chaque étape de la recherche. Afin de simplifier cette phase de contrôle, le manipulateur est réduit par squelettisation à une chaîne articulée composée de segments de droite, les obstacles étant grossis en proportion (opération très simple dans le cas des sphères).

Le calcul d'intersection entre un composant réduit et un obstacle élémentaire est donc ramené à celui d'un segment de droite $[pq]$ avec un disque (dans le plan défini par cette droite et le centre de la sphère). On obtient donc une équation du second degré $S_i(\lambda\ p + (1-\lambda)\ q) = 0$ dont le nombre et la nature des solutions indique la présence ou l'absence d'une intersection ainsi que son type éventuel.

Lorsqu'une collision est ainsi détectée, il devient nécessaire de l'empêcher. Ceci peut alors être fait en appliquant au composant impliqué une force de répulsion relativement à l'obstacle proche, induisant ainsi un mouvement d'éloignement du composant. Ce mouvement doit alors être répercuté sur le reste de la structure articulée. Comme la base du manipulateur est fixe, on traite les constituants de sa structure depuis la base jusqu'à la pince en propageant les mouvements effectués.

L'inconvénient de cette méthode est que chaque composant du manipulateur tend à avoir un comportement autonome (relativement aux obstacles qui l'entourent), ce qui perturbe le déplacement de l'ensemble du bras vis-à-vis du but à atteindre.

On peut améliorer la méthode en appliquant les potentiels sur toute la structure du manipulateur. Ne pouvant le faire exactement en tout point, il est alors nécessaire de sélectionner un ensemble de points auxquels doivent être appliquées les forces induites. Pour un composant, réduit à un segment de droite, on choisit les points extrémaux plus quelques points sur le segment. Si ces points sont suffisamment nombreux, le contrôle de collision devient inutile, du fait qu'il est implicitement géré au niveau des potentiels répulsifs. La force appliquée à l'extrémité libre du composant est obtenue comme la résultante de la force d'attraction calculée au niveau du préhenseur et de la force de répulsion moyenne appliquée sur les points du composant. Du fait des liaisons entre composants, la direction de déplacement calculée doit être projetée sur la direction de déplacement réel (donc perpendiculairement au composant pour une rotation et tangentiellement pour une translation).

Le but global poursuivi est en quelque sorte "d'aligner" les composants de la structure articulée dans la direction du but. Cependant, l'alignement effectif ne se produit que lorsque le mouvement global du manipulateur tend à le déplier. La méthode marche identiquement lorsque le mouvement tend au contraire à rétracter celui-ci, même si le but poursuivi est alors moins évident.

Les forces directionnelles sont donc calculées puis appliquées au manipulateur en partant de sa base jusqu'à son extrémité terminale, entrainant tour à tour le déplacement de chacun des composants.

L'algorithme $POTENTIEL_2$ résume le principe de cette méthode. On note χ_n la position du point de référence du $n$ième composant de la structure articulée ($1 \leq n \leq N$). $F_n(\chi_n)$ désigne la résultante des forces de répulsion appliquées au

$n$ième composant contraint par sa liaison articulaire : c'est donc la projection de la force calculée comme la moyenne des $F_{N,i}$, forces de répulsion résultantes en chaque point considéré du composant. Pour un degré de liberté rotoïde (respectivement prismatique), F_n est perpendiculaire (resp. tangentielle) au composant et entraîne une rotation (resp. translation) de ce dernier, déplaçant ainsi le point de référence χ_{n+1} du composant postérieur approximativement de la différence $\delta(\chi_n)$ (par convention, on a $\delta(\chi_0 = 0)$). Les déplacements élémentaires de chaque composant se répercutent donc sur tous les suivants et, pour éviter un trop grand déplacement du préhenseur, il est nécessaire de choisir la borne F_M petite.

Notons que dans le cas d'un manipulateur, les zones d'effet des potentiels ont avantage à être réduites, afin d'éviter la présence de forces trop nombreuses et antagonistes sur toute la structure.

Enfin, on peut également compléter cette procédure par le traitement des culs-de-sac (portant alors sur la position du préhenseur).

procédure POTENTIEL$_2(\mathcal{S}, \varsigma_I, \varsigma_G)$
début
pour $n = 1 \ldots N$: $\chi_n = \chi_{n,I}$; $F_n := 0$;
$F_N := F_N(\chi_N) + F_G(\chi_N)$; /* *Initialisations* */
tant que $\chi_N \notin \mathcal{B}_{\chi_G, \epsilon_G}$:
pour $n = 1 \ldots N$:
$F_n(\chi_n) = \frac{1}{N_n} \sum_{i=1}^{N_n} F_{n,i}(\chi_n)$; /* *Répulsion moyenne* */
$F_n := F_n + F_n(\chi_n) + F_G(\chi_N)$;
$\chi_n = \chi_n + \delta(\chi_{n-1}) + \max(F_n, F_M)$; /* *Position de référence* */
fin pour
fin tant que
fin

Le problème majeur de cette méthode reste le choix des nombreux paramètres utilisés. Ceux-ci sont actuellement fixés de manière arbitraire : les fonctions de potentiel sont imposées (cf 8.2.2), et les paramètres spaciaux (introduits au paragraphe 8.2.3) sont calculés en fonction de l'encombrement de l'espace de travail.

Si la méthode corrective présentée est utilisable seule pour un objet mobile, elle reste difficilement applicable à un manipulateur complet, particulièrement dans un environnement très contraint. C'est pourquoi nous ne l'utilisons en pratique que comme complément de la méthode globale précédemment exposée.

Chapitre 9

Méthode hybride de planification de trajectoires

9.1 Utilisation de la méthode globale

9.1.1 Planification de base

L'algorithme de base résolvant notre problème s'exprime sous la forme

$$PARSEC_0 \triangleright TRAJ\left(ADJACENT\left(LIBRE(1,\emptyset,\mathcal{B})\right),\varsigma_I,\varsigma_F\right)$$

Introduisant la séparation de l'espace libre en P espaces libres élémentaires associés à chacun des obstacles, nous obtenons l'algorithme $PARSEC_1$ ci-dessous :

procédure PARSEC$_1(\mathcal{A},\mathcal{B},\varsigma_I,\varsigma_G)$
début
pour $p=1\ldots P$: $EL_{\mathcal{A}}(B_p)=LIBRE(1,\emptyset,B_p)$;
$\mathcal{G} := ADJACENT\left(UNION\left(\cap_{p=1}^{P} EL_{\mathcal{A}}(B_p)\right)\right)$;
$\mathcal{T} := TRAJ(\mathcal{G},\varsigma_I,\varsigma_G)$;
fin

Notons que toutes les procédures $PARSEC_n$ proposées dans ce chapitre ont pour fonction de déterminer une trajectoire à partir d'une description géométrique du robot $\mathcal{A}$ et de l'ensemble des obstacles $\mathcal{B}$, ainsi que des configurations initiale ς_I et terminale ς_G. Toutefois, une fois constuit le graphe $\mathcal{G}$, l'étape de recherche d'une trajectoire dans le graphe (appel de $TRAJ$) peut être répétée pour autant de couples $(\varsigma_I,\varsigma_G)$ que nécessaire. C'est d'ailleurs là un des intérêts de la méthode.

En effet lorsque, dans le cadre de la programmation automatique d'une tâche d'assemblage, le module de planification de trajectoires est appelé par un autre

planificateur, il exécute l'algorithme ci-dessus, ou l'une de ces variantes $PARSEC_n$. L'espace des configurations correspondant à l'environnement est calculé, puis une trajectoire est planifiée et produite comme solution du problème posé. Si la solution globale produite par l'ensemble du système se révèle inapplicable, une nouvelle trajectoire peut être planifiée - à moindre coût - dans l'espace de planification construit antérieurement.

9.1.2 Planification hiérarchique

Si nous prenons maintenant en compte les différents niveaux de représentation géométrique de l'environnement, nous pouvons introduire une boucle de planification hiérarchique. De plus l'utilisation des manipulateurs équivalents définis par la hiérarchie de balayage introduit à son tour une boucle hiérarchique que l'on peut imbriquer dans la précédente.

A une étape donnée de l'algorithme, on cherche une trajectoire au niveau courant h pour un robot $\mathcal{A}^l$ dans le graphe de connectivité associé $\mathcal{G}^{h,l}$. Pour chaque niveau de représentation en partant du niveau le moins précis ($h = 0$), on augmente la précision sur le manipulateur en diminuant progressivement le nombre l de degrés de liberté figés (dont les constituants sont remplacés par un volume obtenu par balayage). On construit ainsi incrémentalement l'espace complet, en s'arrêtant dès qu'une trajectoire correspondant aux critères demandés a été trouvée. La procédure $PARSEC_2$ utilisée est la suivante :

procédure PARSEC$_2(\mathcal{A}, \mathcal{B}, \varsigma_I, \varsigma_F)$
début
$h := 0$; $l := NL$; $\mathcal{T} = \emptyset$;
tant que $\mathcal{T} = \emptyset$ **et** $h \neq H_N$:
  pour $p = 1 \ldots P$: $EL^{h,l}_{\mathcal{A}}(B_p) = LIBRE(1, \emptyset, B_p)$;
  $\mathcal{G}^{h,l} := ADJACENT\left(UNION\left(\cap_{p=1}^{P} EL^{h,l}_{\mathcal{A}}(B_p)\right)\right)$;
  $\mathcal{T} := TRAJ(\mathcal{G}^{h,l}, \varsigma_I, \varsigma_F)$;
  $l := l - 1$;
  si $l = 0$: $h := h + 1; l = NL$;
fin tant que
fin

Dans le cas le pire, le système construit l'espace complet avec le niveau de précision maximal (rejoignant alors la méthode de base). Mais il est clair que, si l'espace n'est pas trop contraint, ce calcul ne sera fait que sur un espace plus

concis. Notons enfin que l'utilisation de ces hiérarchies implique une construction partielle de l'espace libre, ce qui entraîne la perte de certaines propriétés telle que l'optimalité de la solution trouvée.

9.1.3 Application pratique et cas particuliers

Longueur de la chaîne cinématique

L'algorithme général décrit précédemment permet en théorie de planifier les mouvements d'un manipulateur constitué d'un chaîne cinématique ouverte à un nombre quelconque de degrés de liberté. Toutefois, sa complexité, qui est exponentielle en ce nombre, le rend en pratique inutilisable pour plus de trois ou quatre degrés de liberté.

Planifier les déplacements d'un robot manipulateur "classique" (possédant typiquement six degrés de liberté) implique l'emploi partiel de cette méthode, alors complétée par l'emploi d'autres algorithmes ou heuristiques. C'est en général la structure du robot qui permet de décider à combien de degrés de liberté appliquer la méthode. Dans le cas d'un manipulateur à articulations rotoïdes (tel celui que nous avons utilisé - cf 10.1), on peut distinguer deux parties dans sa morphologie : celle comportant le bras et l'avant-bras (pour le positionnement de l'outil terminal), et celle incluant le poignet et la pince, autorisant une grande richesse d'orientations. Nous avons donc choisi naturellement d'utiliser l'algorithme de construction de l'espace des configurations pour la première partie bras - avant-bras, c'est-à-dire sur les trois premiers degrés de liberté de la structure.

Mouvements du poignet

A l'issue de la phase de construction de l'espace libre, nous disposons donc d'une représentation où peuvent être planifiées les trajectoires du bras et de l'avant-bras. Cette construction peut être effectuée soit en considérant le volume qui englobe l'espace balayé par le poignet (pince et objet transporté inclus) lors des mouvements de réorientation, soit en considérant que celui-ci est figé dans une orientation arbitraire.

Il est clair que la première solution est trop contraignante et la seconde relativement irréaliste. Nous avons donc choisi d'utiliser une solution intermédiaire qui consiste à minimiser le volume englobant. Il s'agit de ne considérer que le

"volume minimal" englobant les positions initiale et finale du poignet. On obtient ainsi typiquement des portions de disques et autres quarts de sphère... Mais ce volume peut être encore trop important, dans le cas de réorientations importantes du poignet.

La méthode finalement adoptée consiste à construire en fait trois versions de l'espace libre du manipulateur. La première est calculée pour une position "figée" du poignet correspondant à sa position initiale (EL_I), la seconde est calculée de même pour sa position finale (EL_G), et enfin la troisième pour le "volume minimal" englobant le poignet. Les configurations de cet espace libre, EL_τ, correspondent alors à des zones de réorientation possible du poignet. Il suffit pour trouver une trajectoire correcte de planifier dans EL_I un chemin depuis la configuration initiale jusqu'à une configuration de EL_τ, puis de même dans EL_G depuis la configuration finale jusqu'à une autre configuration de EL_τ, et enfin un chemin entre les deux configurations de EL_τ calculées.

Cette technique est en fait basée sur l'observation expérimentale que la plupart des trajectoires de transfert mettent en jeu soit de petites rotations du poignet, soit des opérations de "réorientation". Dans le premier cas, les configurations initiales et finales du poignet seul sont relativement proches et le passage de l'une à l'autre ne pose pas de problème réel. Dans l'autre cas, il est presque toujours possible d'exécuter les opérations de réorientation dans des régions de l'espace peu encombrées (EL_τ).

Il est sommes toutes assez peu satisfaisant de devoir calculer un espace libre pour un manipulateur dont le poignet doit être figé dans telle ou telle position, sans que l'on sache bien laquelle peut être "la bonne". Aussi vient l'idée de déterminer un espace libre pour un robot sans son poignet, et de résoudre d'une toute autre manière le déplacement de ce dernier : par l'emploi d'une méthode locale.

9.2 Utilisation d'une méthode locale pour la replanification

9.2.1 Idée générale d'une approche mixte

L'utilisation d'une *méthode mixte*, c'est-à-dire utilisant tour à tour une méthode globale (comme celle décrite aux chapitres 6 et 7) et une méthode locale (comme

celle présentée au chapitre 8), vise à cumuler les avantages des deux approches. En revanche apparaît une difficulté intrinsèque : le problème de la communication entre les deux méthodes.

Du point de vue de la méthode globale, l'approche mixte permet de réduire son champs d'application, visant ainsi essentiellement une réduction de la complexité et par conséquent une amélioration des performances. Du point de vue de la méthode locale, elle offre une possibilité de "guidage vers le but", évitant ainsi les problèmes spécifiques de convergence (extrema locaux en particulier).

L'idée de base de la replanification et donc d'opérer en deux étapes : tout d'abord on planifie une trajectoire dite *de référence* dans l'espace libre EL_π calculé pour un nombre restreint de degrés de liberté (par exemple les trois du bras et de l'avant bras) puis, tout le long de cette trajectoire, on positionne le poignet à l'aide d'une méthode locale de manière à lui faire éviter toute collision. Il s'agit en fait de substituer aux diverses heuristiques d'orientation du poignet une méthode locale moins contraignante.

9.2.2 Utilisation de la méthode des potentiels

Conditions d'application

Il s'agit donc de déterminer les orientations valides de la pince le long de la trajectoire de référence. Les contraintes sont d'une part que la base du poignet décrive la trajectoire de référence, et d'autre part que sa posture à tout instant évite toute collision avec l'environnement. La position initiale fait coïncider l'orientation du poignet avec celle demandée comme position de départ, et le but à atteindre correspond de même à la configuration terminale spécifiée.

C'est donc cette configuration finale qui impose au poignet un potentiel attractif cherchant à l'orienter dans une direction particulière. Les potentiels répulsifs contrôlent implicitement l'absence de collision, et la trajectoire de référence joue alors un rôle de guide pour la méthode.

Description de la méthode

On peut distinguer deux niveaux d'application de la méthode locale, suivant qu'il s'agit de compléter ou de modifier les résultats obtenus antérieurement.

Dans le premier cas, la trajectoire de référence n'est pas remise en cause. On détermine alors une trajectoire complémentaire pour les éléments auparavant

négligés (le poignet complet) qui soit compatible avec celle déjà calculée. L'autre cas se produit lorsqu'une telle trajectoire complémentaire ne peut être trouvée. La trajectoire de référence donnée doit alors être remise en cause. Elle est dans ce cas soit modifiée localement, soit entièrement recalculée par la méthode globale.

La trajectoire de référence issue de la première phase - méthode globale - est représentée par un ensemble de cellules libres définissant une enveloppe de trajectoires, et par une instance de trajectoire issue de cet ensemble (cf chapitre 7).

La position extrémale du bras est celle de la base du poignet. Nous convertissons donc la trajectoire dans l'espace cartésien afin de faire le lien avec la méthode locale. Cette trajectoire est alors linéaire par morceaux, et elle peut être échantillonnée afin d'obtenir une trajectoire de référence $\mathcal{T}_R$ adaptée à l'utilisation de la méthode des potentiels.

La méthode appliquée consiste alors à utiliser une variante de la procédure $POTENTIEL_2$ à la structure articulée formée par le poignet, et ceci en tout point de $\mathcal{T}_R$. A la différence de la méthode locale seule où la base du manipulateur était fixe, celle χ_0 du poignet est maintenant astreinte au suivi de la courbe $\mathcal{T}_R$. Nous étudions ci-après les deux cas possibles.

Cas I : Invariance de $\mathcal{T}$

La trajectoire de référence $\mathcal{T}_R$ est supposée pour l'instant invariante. L'algorithme $MODIF - TRAJ$ calcule, pour une trajectoire complétée par les configurations de départ et d'arrivée χ_I et χ_G, la séquence de configurations devant être prises par le poignet. Les objets sont utilisés à travers leurs représentations sphériques $\mathcal{S}$ grossie, le poignet comportant N composants préalablement squelettisés, dont les positions initiale et finale sont donc imposées. Comme précédemment, les contraintes dues aux liaisons articulaires se traduisent par la projection des forces de répulsion sur la direction de déplacement, y compris pour la base du poignet en mouvement sur $\mathcal{T}$. La procédure appliquée est la suivante :

procédure MODIF-TRAJ $(\mathcal{T}_R, \chi_I, \chi_G, \mathcal{S})$
début
pour $n = 0 \ldots N$: $\chi_n = \chi_{n,I}$; $F_n := 0$;
$F_N := F_N(\chi_N) + F_G(\chi_N)$; /* *Initialisations* */
tant que $\chi_N \notin \mathcal{B}_{\chi_G, \epsilon_G}$:
$\chi_0 = \chi_0 + \max(F_N, F_M)$
pour $n = 1 \ldots N$:
$F_n(\chi_n) = \frac{1}{N_n} \sum_{i=1}^{N_n} F_{n,i}(\chi_n)$;
$F_n := F_n + F_n(\chi_n) + F_G(\chi_N)$;
$\chi_n = \chi_n + \delta(\chi_{n-1}) + \max(F_n, F_M)$;
fin pour
fin tant que
fin

Cas *II* : Modification de $\mathcal{T}$

Il peut arriver, lorsque l'environnement est trop contraint à proximité de $\mathcal{T}$, que le poignet soit empêché d'atteindre la configuration souhaitée. Il est alors nécessaire de modifier la trajectoire de référence.

Le blocage rencontré est soit une barrière de potentiel (totalement infranchissable), soit un "cul-de-sac" (difficile à contourner). Il est par conséquent détectable par l'utilisation des boules de proximité décrites précédemment. La position $\chi_{0,D}$ de la base du poignet, qui correspond à la position de blocage, permet d'identifier la cellule libre impliquée. Cette cellule (ou son voisinage) représente le point où une modification locale doit être faite, sachant que la direction du déplacement à effectuer à l'intérieur de cette cellule ou de ses voisines est indiquée par la résultante des forces en $\chi_{0,D}$.

Dans un premier temps, on peut alors se référer à la représentation courante de l'espace libre pour localement déformer la trajectoire de référence $\mathcal{T}_R$ dans la direction calculée. Ceci est toujours possible dans les limites des cellules qui définissent la trajectoire (par construction du graphe des configurations). On obtient donc une nouvelle trajectoire de référence pour laquelle on peut appliquer à nouveau la procédure $MODIF - TRAJ$. Si ceci amène un nouvel échec (le poignet ne "passe" toujours pas), il devient nécessaire de dépasser la limite des cellules courantes. Cette deuxième phase implique un changement de cellule et donc un contrôle de la validité du mouvement du bras : soit par l'existence d'une cellule adjacente autorisant le mouvement, soit par un calcul d'interférence localisé.

Le cas où il n'existe pas de cellule voisine, et celui où celles disponibles ne conviennent pas (c'est-à-dire conduisent au même type d'échec), sont significatifs d'un espace suffisamment libre pour permettre le calcul d'une trajectoire de référence mais trop contraint cependant pour autoriser le passage de l'ensemble poignet-pince. Il est alors nécessaire de replanifier complètement la trajectoire $\mathcal{T}_R$ dans un espace libre dont les configurations fautives ont été supprimées (les cellules testées étant simplement éliminées), garantissant ainsi l'obtention d'une trajectoire radicalement différente.

Remarque : L'utilisation corrective de la méthode des potentiels pose un problème de paramétrisation délicat à résoudre. Les paramètres sont nombreux, et leur influence n'est pas toujours facile à percevoir. Ainsi le choix même des potentiels associés aux obstacles reste très difficile, essentiellement du fait des variations de comportement importantes en fonction de la nature de l'environnement.

Chapitre 10

Un exemple d'implantation : le système PARSEC

L'implantation informatique qui fait l'objet de ce chapitre est à la fois le support de notre modélisation et la validation des algorithmes utilisés. Le système PARSEC[1] est un logiciel permettant de planifier les mouvements d'un robot manipulateur SCEMI-6 axes dans un univers encombré d'obstacles.

La version complète intègre une interface de modélisation permettant de définir et positionner les objets de la scène ainsi qu'une interface de simulation graphique permettant la visualisation "hors-ligne" des trajectoires calculeés. De fait, le logiciel PARSEC peut être considéré, du point de vue implantation, comme la base de SHARP , puisqu'il suffit d'y adjoindre les modules de planification de saisie et de montage pour obtenir le système complet.

10.1 Configuration matérielle du système

10.1.1 Architecture de la cellule d'expérimentation

La cellule flexible d'assemblage constituant notre domaine d'expérimentation se résume pour la partie robotique à un manipulateur de type SCEMI à six degrés de liberté indépendants en rotation. Il est muni d'une "simple" pince à mors parallèles, équipée d'un capteur de forces, qui nous semble suffire largement pour la réalisation de la plupart des assemblages industriels.

Les programmes de commande de niveau effecteur sont écrits en langage LM, et exécutés sur un ordinateur HP-1000 directement relié à l'armoire de commandes du robot. Le système PARSEC fonctionne sur une station SUN 3/260 reliée au HP-

[1] Acronyme pour : "Programmation Automatique des Robots : Sytème d'Evitement de Collision"

1000 par liaison série. Il a été implanté en FranzLisp, Kyoto Common Lisp et enfin Lucid Common Lisp pour la version courante, certains aspects géométriques et graphiques étant réimplantés en langage C.

10.1.2 Caractéristiques cinématiques du robot SCEMI

La structure du manipulateur

Ce type de robot possède les principales caractéristiques nécessaires à la réalisation de tâches complexes. D'un point de vue morphologique, il possède un préhenseur capable d'aggriper, tenir puis lâcher un objet, et sa structure articulée lui permet de se mouvoir en trois dimensions et d'orienter cet outil dans n'importe quelle direction dans l'espace de travail.

Comme de plus les composants du manipulateur sont particulièrement volumineux, le rapport entre le volume occupé et l'espace accessible est assez défavorable et limite considérablement les mouvements possibles dans un environnement contraint. De plus, la présence de cables souples externes, non représentés dans le modèle géométrique et donc ignorés par le système, nous a causé de désagréables surprises lors des expérimentations "en ligne" de trajectoires engendrées "hors ligne".

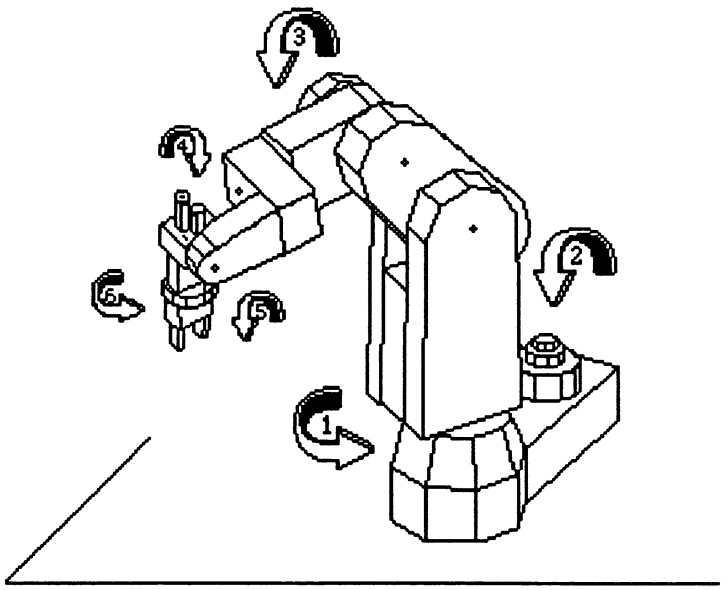

Figure 10.1: Degrés de liberté du robot SCEMI.

La chaîne cinématique du manipulateur (cf figure 10.2) comporte donc six

degrés de liberté qui peuvent être séparés fonctionnellement en deux groupes : les trois rotations principales constituant le "bras" proprement dit (1 à 3 sur la figure 10.1), et les trois rotations concourantes constituant le "poignet" (4 à 6). Cette structure permet d'atteindre tous les points de l'espace de travail avec le maximum d'orientations différentes (en l'occurence huit).

La commande

Chaque axe est asservi en vitesse et en position. La boucle de vitesse est réalisée par variateurs à découpage et génératrices tachymétriques pour les axes. Leur intérêt est d'autoriser une très grande précision de mouvement ainsi que des vitesses très lentes, rendues nécessaires par la résolution en position du manipulateur. La boucle de position est assurée pour chacun des six axes par une carte à microprocesseur 16 bits reliée à un capteur incrémental optique.

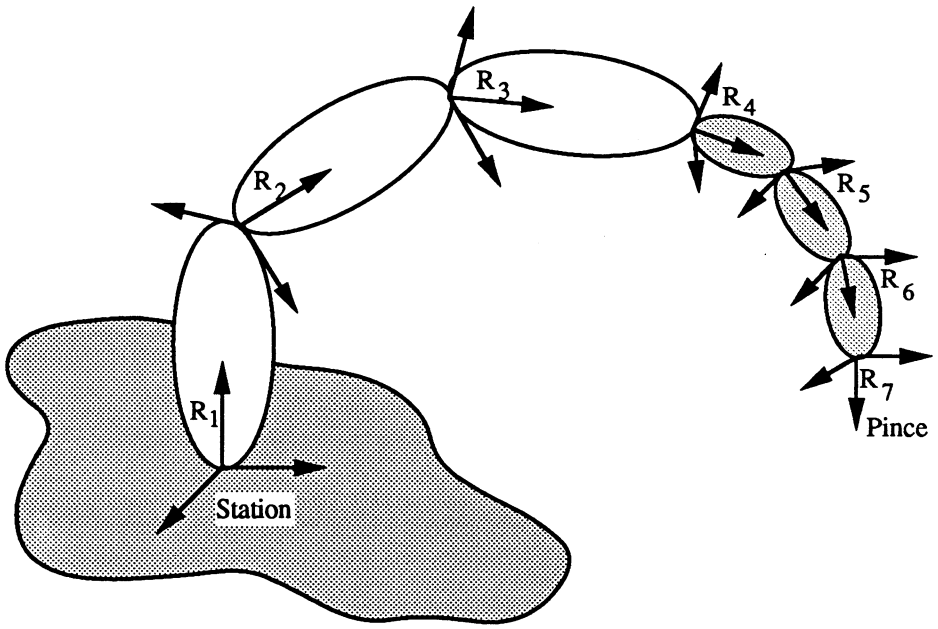

Figure 10.2: Chaîne cinématique du robot SCEMI.

La précision de commande obtenue est de $\pm 0.06mm$ en position "moyenne" et de $\pm 0.1mm$ en position "déployée". La résolution (valeur de déplacement minimal possible) est d'un incrément angulaire. La dispersion - théorique- de la position en répétabilité est de $\pm 0.04mm$. La précision mécanique du robot est donc bien bien meilleure que celle des calculs effectués lors de la modélisation de l'espace libre (d'un facteur 10) et la précision demandée par le planificateur est

donc largement garantie.

L'espace de travail

L'espace de travail réellement disponible se trouve assez restreint, étant entendu que l'espace en question n'est pas l'ensemble des points de l'univers physiquement atteignables par la pince mais celui, infiniment plus réduit, des points atteignables par une configuration *quelconque* de la pince. Cette exigence correspond au besoin réaliste de pouvoir, lors d'un montage par exemple, amener un objet dans une position quelconque relativement à l'assemblage, lequel est généralement fixe.

L'amplitude de chaque degré de liberté est une donnée importante concernant les intervalles de débattements angulaires, puisqu'elle définit les bornes de validité de l'intervalle I_n initial. Le tableau ci-dessous donne les valeurs de butées mécanique du robot SCEMI-6. Le nombre total de positions possibles au pas de discrétisation minimal (celui des incréments angulaires des moteurs) est de l'ordre de 2 10^{17} . Là encore, la précision des moteurs garantit (et de loin) l'exécution de toute commande en provenance du planificateur. De fait, le pas de discrétisation utilisé en pratique au niveau des calculs de l'espace des configurations est supérieur d'un facteur 20 à 100 au pas minimal physiquement réalisable.

	axe 1	axe 2	axe 3	axe 4	axe 5	axe 6
amplitude	135/-135	90/-35	118/0	360/0	135/-135	360/0
incrément	16"	16"	16"	20"	20"	20"
positions	1012	468	442	1080	810	1080

10.2 Les modules de modélisation et de simulation

Le module de modélisation géométrique implanté permet de gérer les représentations précédemment décrites. Les objets sont décrits dans leur représentation volumique via l'interface utilisateur, et les autres représentations sont toutes calculées à partir de cette dernière.

L'implantation actuelle de ce module, entièrement en Lisp, est destinée à être remplacée par une version en C, le système ELIOT, actuellement en cours de développement. Par son aspect "éditeur graphique", ce système est évidemment très fortement lié à l'interface graphique, ou de simulation.

Diverses interfaces de simulation ont été développées dans notre laboratoire à mesure de l'évolution technologique du matériel disponible et de l'accroissement des besoins en ce domaine. Nous avons successivement utilisé puis développé les interfaces suivantes :

L'interface graphique Tektronix

Ce module fonctionne sur tout terminal permettant une émulation Tektronix 4014. Basé sur le logiciel Lisp-3D [4-Lau82], il a d'abord été porté de MacLisp en FranzLisp, puis en Common Lisp.

Ce module permet l'utilisation du système PARSEC (et donc SHARP) avec n'importe quel calculateur, pourvu que lui soit connecté un terminal du type adéquat. Pour notre part, nous l'avons utilisé avec un MacIntosh+ (APPLE) connecté à un VAX 750 (DEC). La figure 10.3 illustre un exemple de trajectoires obtenu avec la première version du système PARSEC (méthode globale seule) dans cette configuration matérielle.

L'interface graphique Sun

Il s'agit d'un module nommé GraphSun utilisant les capacités graphiques de Lucid Common Lisp et que nous avons développé séparément puis totalement intégré aux systèmes PARSEC et SHARP . Les fonctionnalités que nous avons implanté calquent celles de Lisp-3D (paramétrisation complète du point de vue, de la perspective etc) avec le bénéfice d'une plus grande ergonomie (utilisation de menus, souris etc), d'une vitesse d'éxécution bien meilleure (moins de "couches" logicielles dans l'implantation, exploitation des caractéristiques matérielles des stations SUN), mais également aussi au détriment de la portabilité (fonctionnement exclusif sur station SUN).

Nous avons également implanté en C et connecté au système général un utilitaire de synthèse d'images par lancer de rayons (technique dite "RayCast", cf [4-Pas85]), permettant la visualisation de scènes plus réalistes ainsi que la création d'images de profondeur (où la distance est codée en niveaux de gris). Si le résultat est intéressant par sa qualité, les temps de calcul nécessaires à la création d'une image sont trop importants pour en permettre une utilisation courante (de 20 à 40 mn suivant la complexité de la scène, qui dépend linéairement du nombre de faces).

Les faibles performances d'affichage des deux modules graphiques écrits en lisp s'expliquent essentiellement par la nature du langage utilisé et par la superposition de multiples couches logicielles conduisant à l'affichage final. A titre d'exemple, la visualisation d'une scène "vide" (comprenant le robot complet plus son support, soit 46 composants élémentaires quand même) prend plus d'une minute, c'est-à-dire beaucoup plus de temps que pour calculer une trajectoire, otant ainsi tout réalisme temporel à la simulation. Pour comparaison, la même scène ne nécessite que quelques secondes pour être identiquement affichée par notre version C du simulateur graphique (Aff-R, qui sera bientôt substitué à l'interface Lisp).

Afin de limiter mieux encore les temps nécessaires au calcul et à l'affichage des scènes, nous avons récemment choisi d'effectuer les traitements graphiques sur une machine spécialisée travaillant en parallèle. Ainsi le calculateur principal se trouve déchargé de toute tâche de simulation graphique et peut se consacrer entièrement à l'aspect planification. Le système spécialisé en question est décrit sommairement ci-après.

L'interface graphique Getris

Le système de synthèse et d'affichage d'images que nous utilisons est le système GETRIS.

Il se compose d'une architecture matérielle spécialisée, à base de technologie câblée, et d'une partie logicielle pour la commande et le contrôle. La chaîne comprend un calculateur pilote, une unité graphique, une unité vidéo et enfin une unité de visualisation. Le logiciel de contrôle est implanté sur un microcalculateur dédié, en l'occurence un HP Vectra.

La couche logicielle de base permet la manipulation et la visualisation d'entités géométriques bi-dimensionnelles élémentaires (rectangles, disques, polygones). Le développement actuel comprend donc l'extension à la troisième dimension des fonctionnalités du système, et l'intégration des mêmes représentations géométriques utilisées par PARSEC. A cette partie géométrique viendra s'ajouter la gestion des communications entre le calculateur principal et le calculateur local. Celui-ci doit en effet gérer une copie de la base de faits géométriques du système. Toute mise à jour effectuée sur le modèle original est alors répercutée (via une liaison série) au niveau de ce calculateur, permettant ainsi à tout instant une visualisation rapide de la scène. En théorie, la vitesse de traitement des différentes unités de Getris devrait même permettre la simulation en temps réel des déplacements

du manipulateur. Cette connexion n'est cependant pas encore opérationnelle.

10.3 La planification de trajectoires

10.3.1 Méthode globale

Le module de planification des mouvements de transfert a été initialement implanté en Kyoto Common Lisp sur un calculateur Vax-750, puis développé par la suite en LUCID-LISP sur machine SUN 260. Des expérimentations ont d'abord été faites en simulation sur des robots planaires à deux, trois et quatre degrés de liberté. D'autres ont ensuite été faites en 3D avec le robot SCEMI six axes décrit précédemment.

La figure 10.3 donne un exemple de trajectoire planifiée par le système pour le manipulateur à trois degrés de liberté avec poignet figé dans une position arbitraire (en l'occurence dans l'axe du dernier composant mobile). Pour cet exemple, la construction de l'espace libre a nécessité 28mn de CPU. Le graphe de connexité obtenu comporte 480 cellules. Il est basé sur un découpage de l'espace articulaire, comportant des secteurs angulaires de 5, 10 et 15° (les plus petits débattements étant associés aux premiers degrés de liberté du bras). La recherche d'une trajectoire dans ce graphe prend un temps moyen de 4s.

Le même exemple en ajoutant un degré de liberté supplémentaire conduit, pour une discrétisation équivalente des domaines à un temps de construction de 67mn. Le graphe obtenu comporte cette fois 720 cellules et le temps moyen de recherche d'une trajectoire est de 5,7s. La différence de temps pour la construction de l'espace libre illustre bien le problème fondamental de la méthode.

La figure 7.2 du chapitre 7 montre une section de l'espace des configurations tridimensionnel correspondant à l'exemple précédent, ainsi que la trajectoire planifiée correspondant à celle visualisée dans la figure 10.3.

10.3.2 Méthode locale

Déplacement d'un mobile

Nos expérimentations de la méthode des potentiels se sont limité à l'espace bidimensionnel, pour des raisons strictement pratiques (comme la possibilité d'une visualisation informative de la trajectoire, beaucoup plus délicate en 3D).

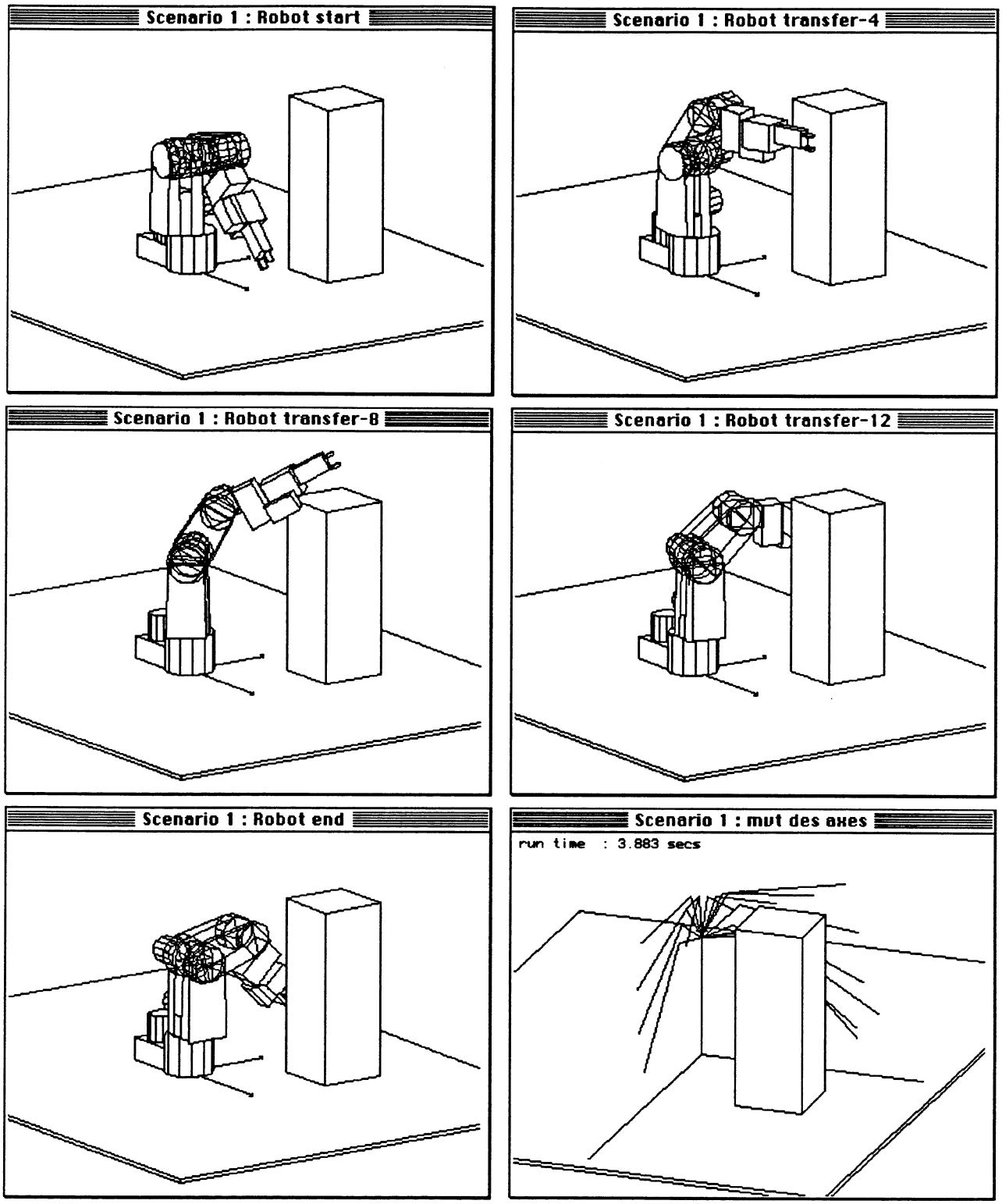

Figure 10.3: Trajectoire pour un robot à 3 degrés de liberté et poignet "figé".

D'autre part l'extension des algorithmes à la troisième dimension ne présente aucune difficulté (la seule différence étant alors la dimension des vecteurs sur lesquels portent les calculs).

De très nombreux essais de paramétrisation nous ont permis de véritablement prendre conscience de la complexité du problème. La figure 10.4 montre pour un environnement donné à potentiel attractif constant la nature des lignes de champs (image 2) et des équipotentielles (image 3). L'image 1 illustre la notion de point instable : il suffit de changer d'une décimale l'intensité du potentiel attractif pour voir la trajectoire du mobile changer totalement de direction... L'image 4 montre différentes trajectoires obtenues en différents points de l'espace. En particulier au point numéro 3, on voit que selon l'intensité du potentiel attractif se crée ou non une barrière de potentiel entre les obstacles d37 et d39, décidant par là même de l'existence d'un passage ou non.

Déplacement d'un manipulateur

La figure 10.5 illustre le déplacement d'une structure articulée à trois degrés de liberté. Les images 1 et 2 donnent les positions initiale et finale du manipulateur dans l'environnement sphérique (pour lequel les objets à l'origine polygonaux sont très grossièrement approchés). L'image 3 donne l'environnement équivalent obtenu par grossissement des obstacles et squelettisation du robot. Enfin l'image 4 donne un exemple de trajectoire obtenue par la méthode exposée précédemment. Le temps de calcul de cette trajectoire est environ de 3s. Il est à noter que, pour la méthode des potentiels, le temps de calcul est extrêmement variable selon la configuration des objets constituant la scène, puisque cela va de la demi-seconde lorsque l'espace est très peu contraint à ... l'infini, lorsque aucune solution n'est trouvée et que le système boucle indéfiniment (cas des puits de potentiel "non-réductibles").

10.3.3 Méthode mixte

Les figures qui suivent illustrent un exemple complet de planification de trajectoire utilisant tout d'abord la méthode globale seule puis une méthode hybride.

L'exemple choisi est celui d'un robot manipulateur planaire à trois degrés de liberté en rotation. Il s'agit en l'occurence de la projection bidimensionnelle de notre SCEMI dans un plan correspondant à une orientation arbitraire de son premier degré de liberté. La figure 10.6 montre la scène réelle, comprenant unique-

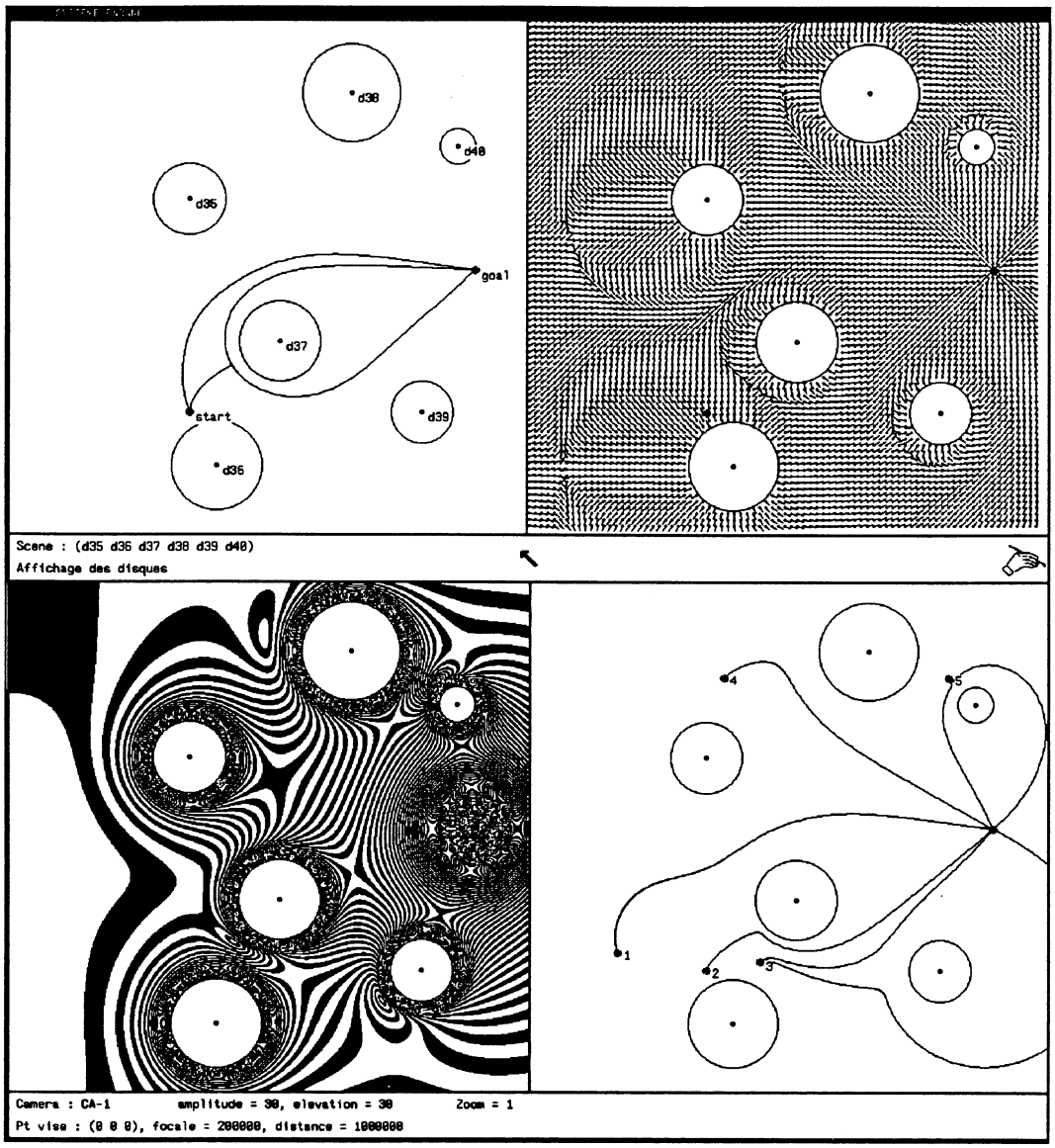

Figure 10.4: Essais de paramétrisation du mouvement d'un mobile.

ment des obstacles polygonaux, au nombre de cinq (sans compter les murs), et le manipulateur. Les positions affichées sont celles de départ et d'arrivée, d'ailleurs parfaitement interchangeables. L'aspect très contraint du problème est visuellement perceptible, l'obstacle D rendant extrêmement délicat le mouvement du robot...

Dans les vues suivantes, nous montrons les obstacles dans leur représentation sphérique (grossie de manière isotrope) et le robot dans sa représentation filaire, ceci afin de pouvoir superposer plusieurs positions du manipulateur ainsi

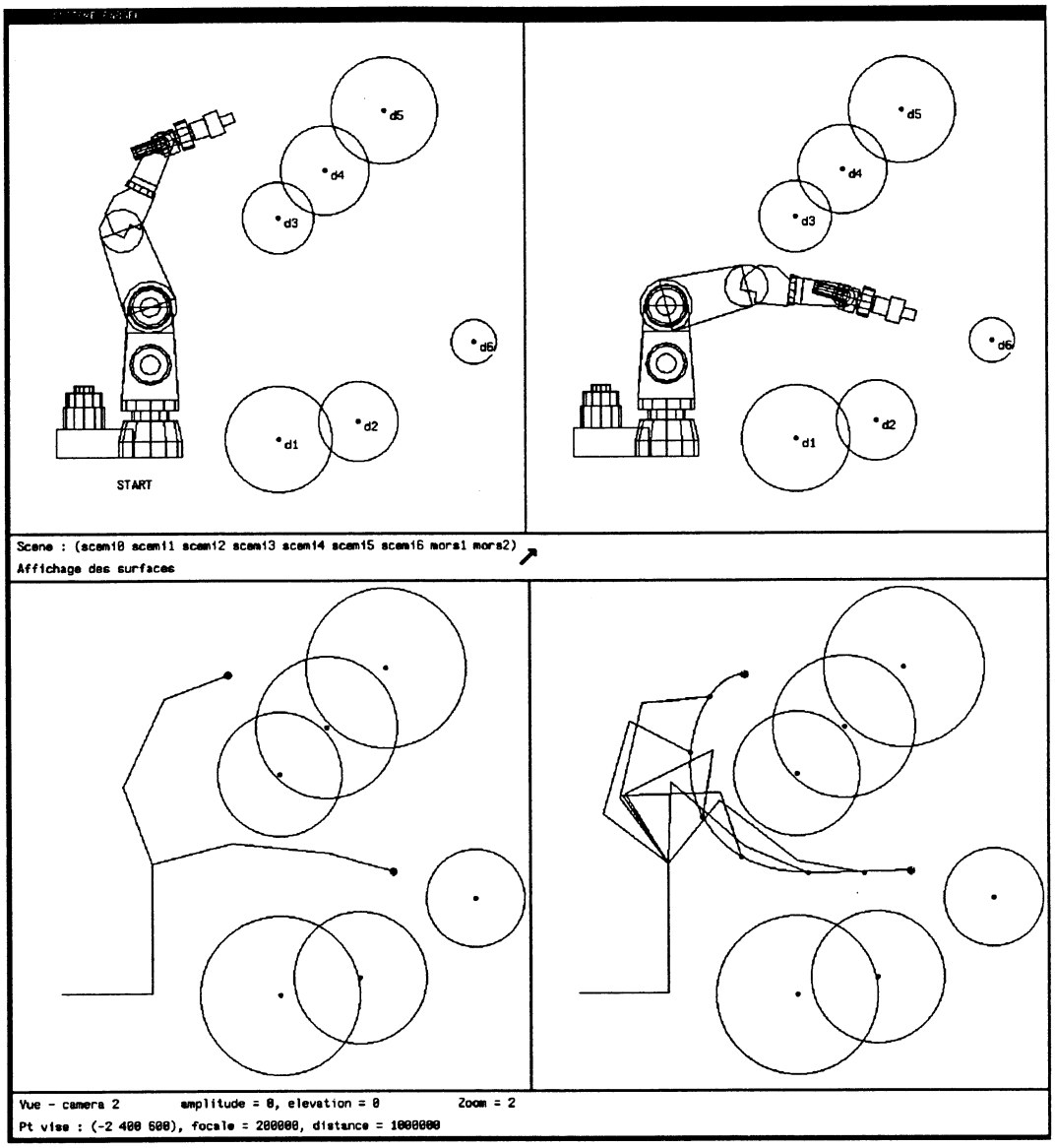

Figure 10.5:

squelettisé.

La figure 10.7 montre les espaces de configurations obtenus dans cet exemple pour chacun des cinq obstacles de l'environnement. Comme précedemment, les degrés de liberté sont représentés "à plat" : le premier q_1 varie de 0 à 90 degrés avec une précision allant de 10 degrés au niveau de représentation le plus simple jusqu'à 2.5 degrés au maximum. Il en est de même pour q_2 qui varie entre ses limites mécaniques de -60 à 120 degrés. Le troisième degré de liberté est représenté graphiquement à l'intérieur des tranches de q_1, avec une précision maximale de

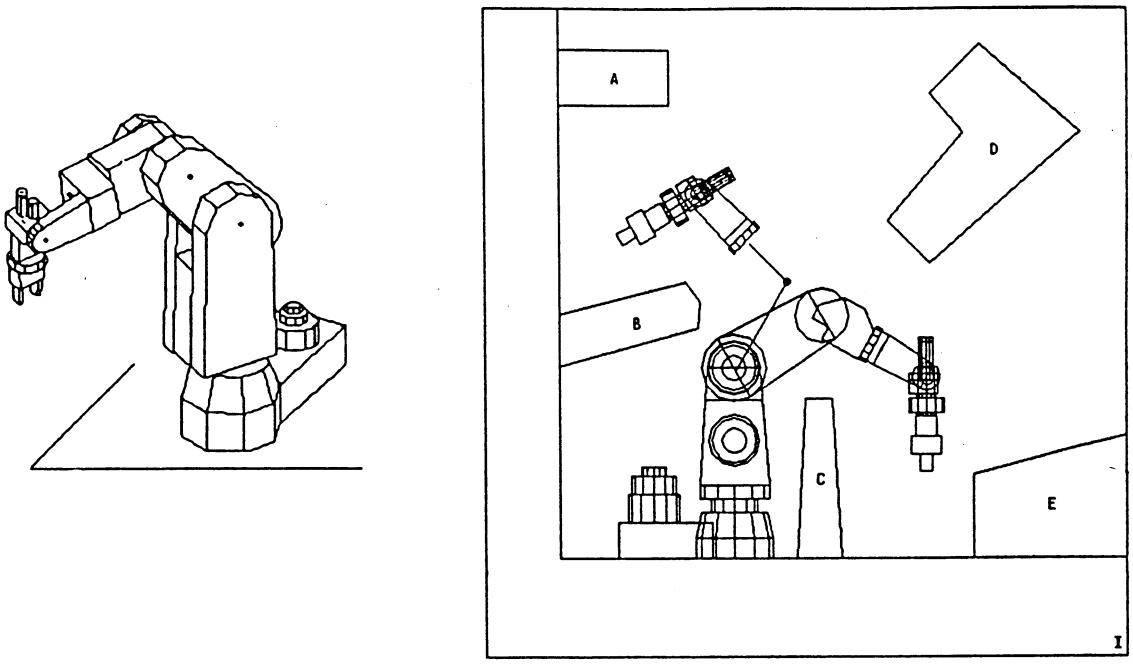

Figure 10.6: Manipulateur en environnement encombré.

5 degrés entre ses butées naturelles, de -135 à 135. Les configurations interdites sont éliminées et correspondent donc aux "vides". Comme prévu, la présence de l'obstacle D induit la plus forte contrainte et sépare nettement l'espace libre en deux régions. Il subsiste toutefois un "pont" pour les réunir en une seule composante connexe, garantissant ainsi l'existence d'une trajectoire bien que très contraignant, D n'est pas rédhibitoire.

La présence des autres obstacles ne semble pas, au vu de leurs espaces libres associés, restreindre beaucoup l'ensemble des solutions potentielles. En fait, c'est seulement lorsque l'on calcule l'espace libre complet (cf figure 10.8) que l'on voit, par comparaison avec l'espace libre associé à D seul, à quel point ces obstacles "périphériques" limitent les possibilités de manœuvre de part et d'autre de D. Il n'y a toujours qu'une seule composante connexe, mais les points limites sont cette fois très nombreux.

Le calcul de l'espace des configurations complet (espaces individuels plus unification) a pris 16 minutes; il comporte près de 600 cellules élémentaires. La trajectoire obtenue a été planifiée en 3,5 secondes.

La figure 10.9 montre la trajectoire calculée par le planificateur. La configuration initiale choisie, "Start", correspond aux triplet de valeurs articulaires

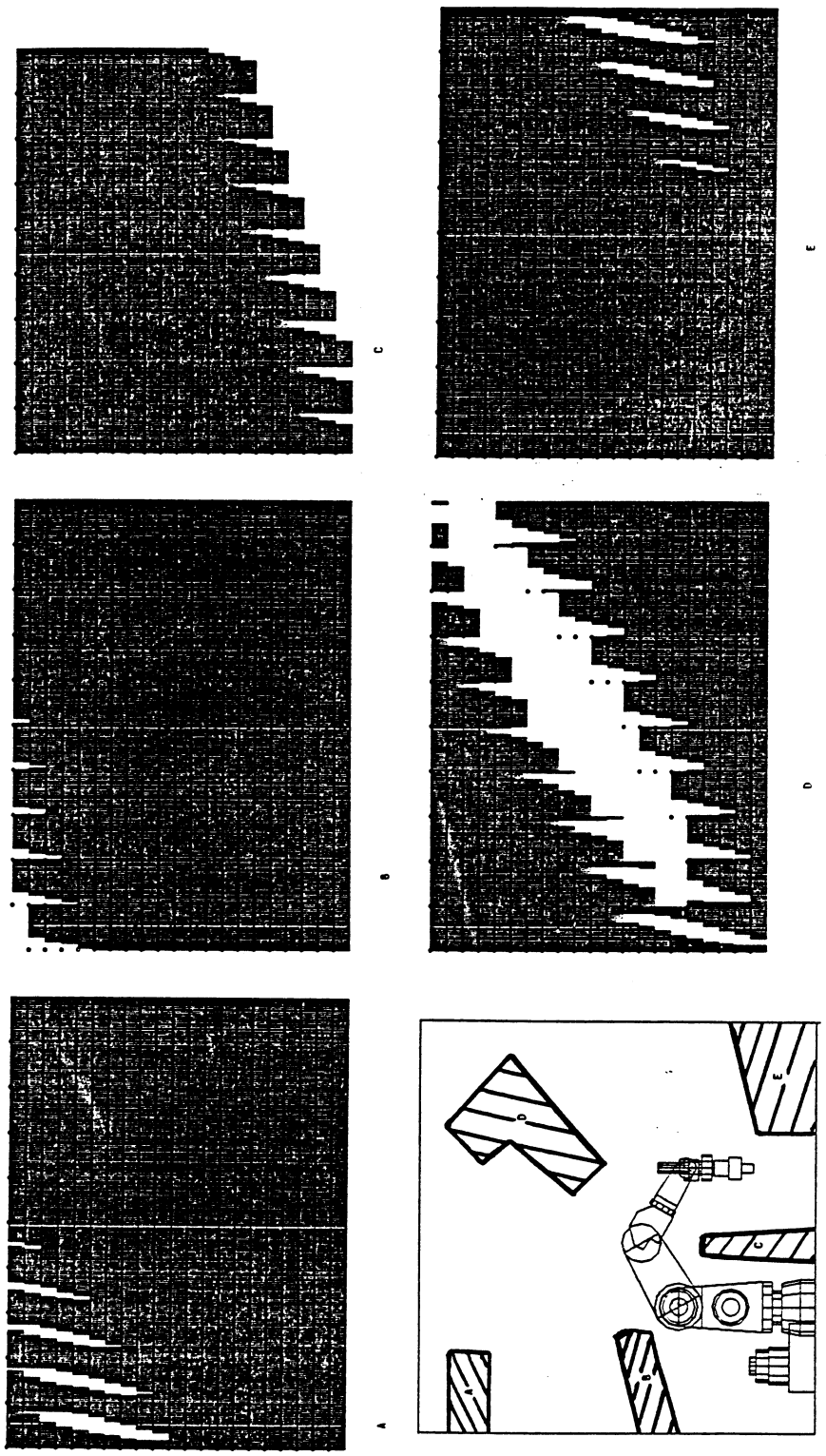

Figure 10.7: Espace des Configurations élémentaires.

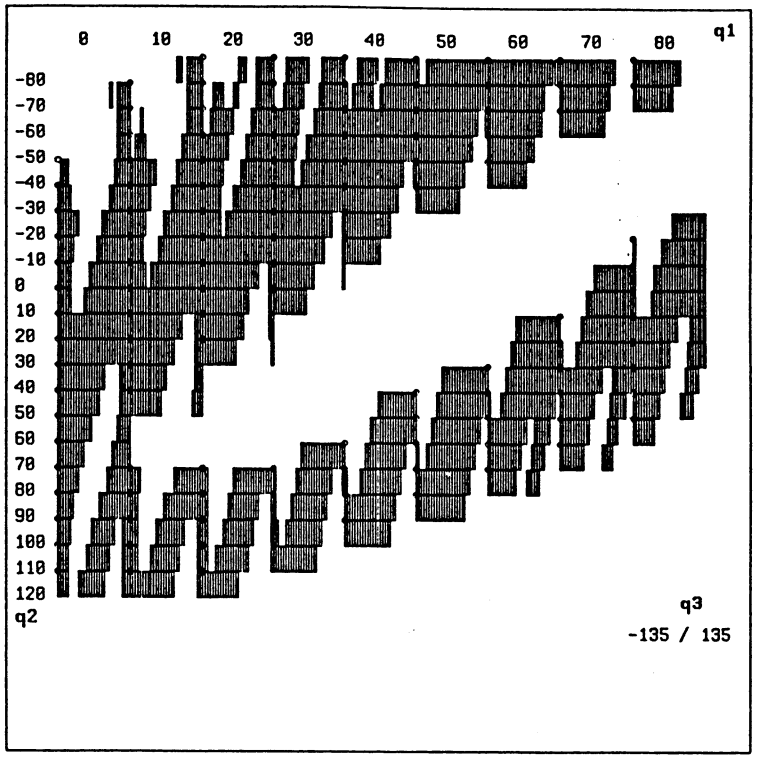

Figure 10.8: Espace des Configurations complet (3ddl).

(30 -75 -75) et la configuration finale, "Goal", à (60 60 65). Les traits horizontaux correspondent à une variation articulaire de q_3 (et sont donc limités à un bloc où q_1 est constant), les traits verticaux à une variation de q_2, et les flêches à une variation de q_1. De par la nature même de l'espace libre, la trajectoire en question est très hachée.

La figure 10.10 montre un échantillon des configurations de la trajectoire obtenue (répartie sur quatre vues). En 1 (en haut à gauche) on peut voir le robot s'extirper du goulot entre les objets A et B et commencer une complète réorientation du troisième constituant. En 2 (en haut à droite), il amorce son repli afin de négocier le difficile passage entre les obstacles B et D, lequel est effectué sur la vue 3 (en bas à droite) et suivi enfin en 4 d'un déploiement vers la configuration terminale.

La figure 10.11 montre la décomposition adoptée pour le manipulateur tridimensionnel, à 6 degrés de liberté. Les 3 degrés de liberté du poignet étant des rotations à axes concourants, son équivalent pour la méthode potentielle est un

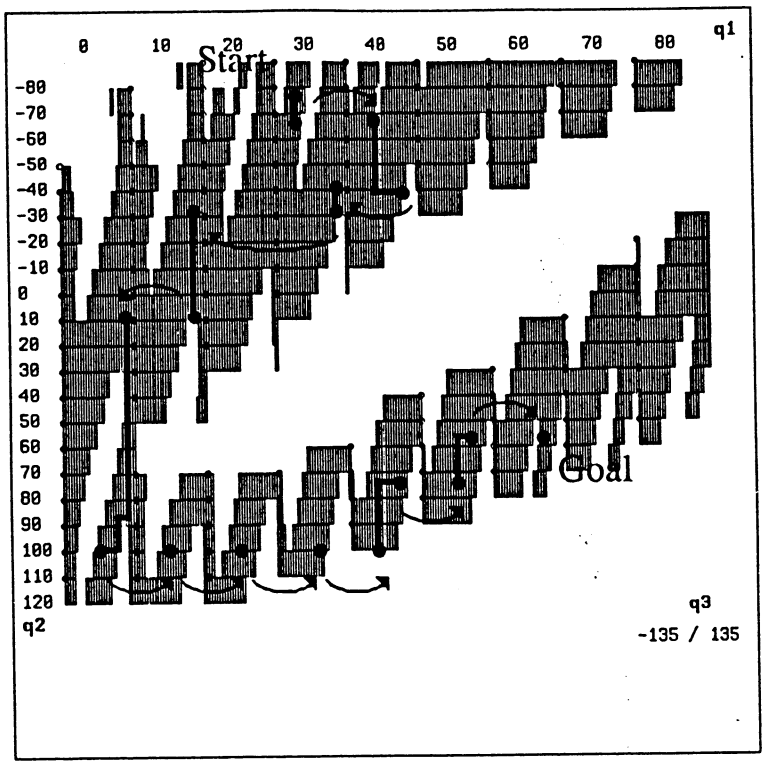

Figure 10.9: Trajectoire planifiée.

segment de droite en rotation libre autour d'un point fixe (le point d'attache du poignet).

On applique donc cette fois la méthode globale au bras seul, qui, dans l'exemple 2D, n'a plus que deux degrés de liberté, le mouvement de rotation de la main constituant le troisième. La figure 10.12 montre l'espace libre associé, qui n'est autre que l'espace libre précédent dans lequel la main du robot n'apporte plus de restriction sur les mouvements des constituants antérieurs. Cet espace est assez peu contraint, et la trajectoire planifiée est cette fois plus directe.

La création du graphe de connectivité réduit a pris 6 minutes et la planification de la trajectoire obtenue a duré environ 1,2 seconde.

La figure 10.13 montre la trajectoire physique correspondante, empruntée par le bras du robot. L'ensemble des positions du point d'attache du poignet constitue la trajectoire de référence sur laquelle celui-ci est astreint à se déplacer, guidé maintenant par la méthode potentielle. Cette trajectoire est matérialisée en figure 10.14 (à gauche).

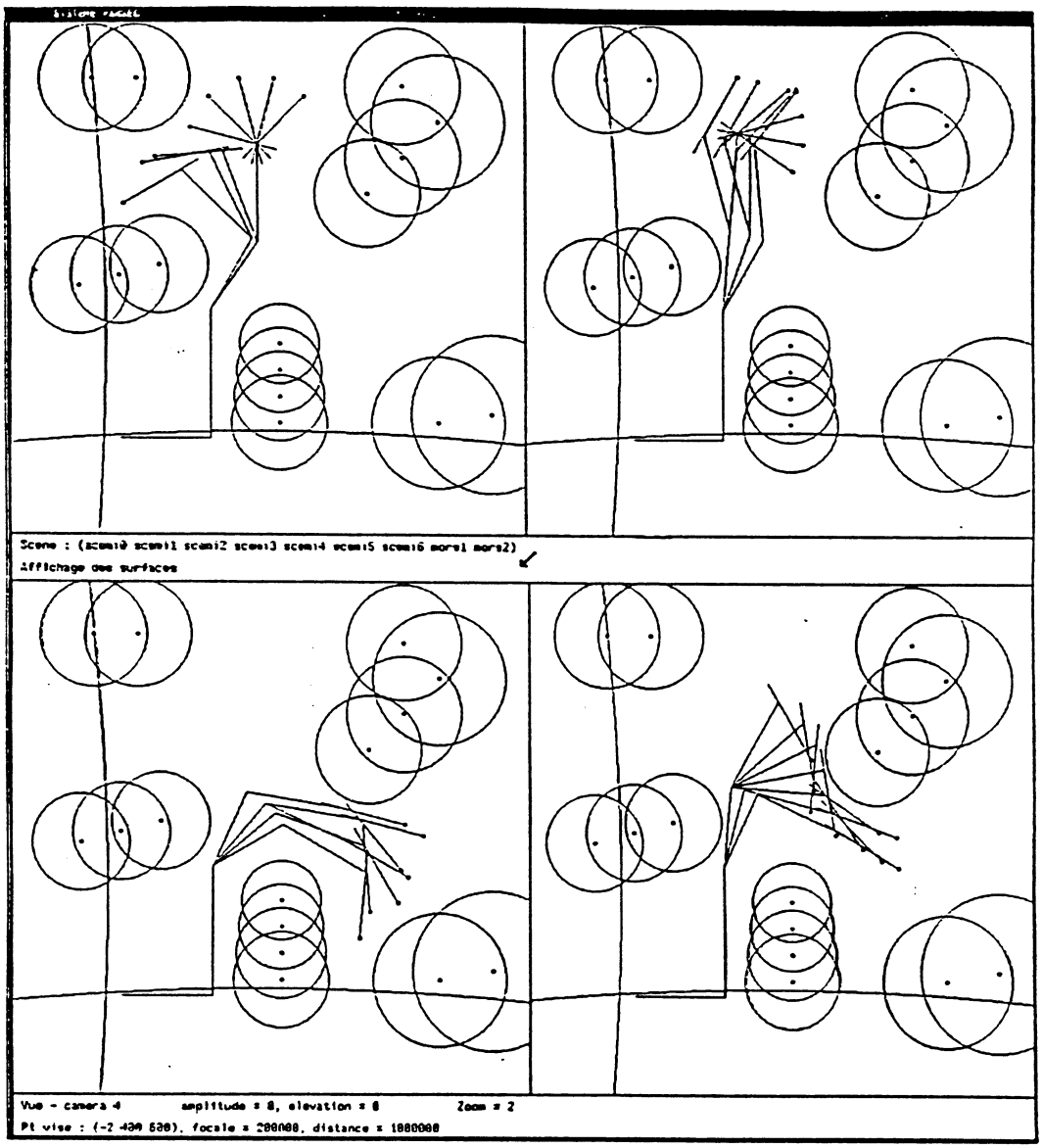

Figure 10.10: Trajectoire exécutée par le manipulateur.

Sur cette même figure, la vue de droite montre un échantillon des orientations du poignet calculées le long de cette trajectoire (les positions associées du bras ne sont pas affichées par soucis de clarté). Il est à noter que le trajet effectué par le poignet n'est qu'assez peu éloigné de celui obtenu par la méthode globale seule, ce qui est du en fait à l'aspect extrêmement contraint du problème choisi (il n'y a quasiment pas d'autre solution possible). La visualisation du résultat brut masque malheureusement les problèmes rencontrés lors du calcul de la trajectoire du poignet, et en particulier les longues hésitations (environ 5 secondes) du

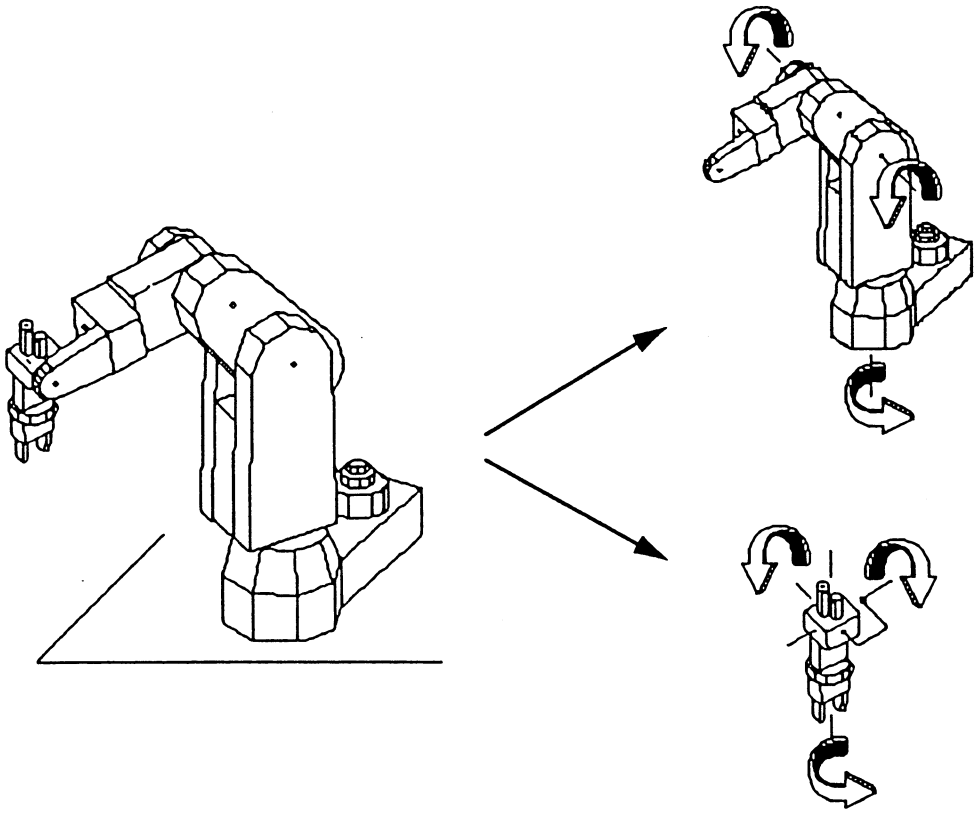

Figure 10.11: Décomposition du manipulateur.

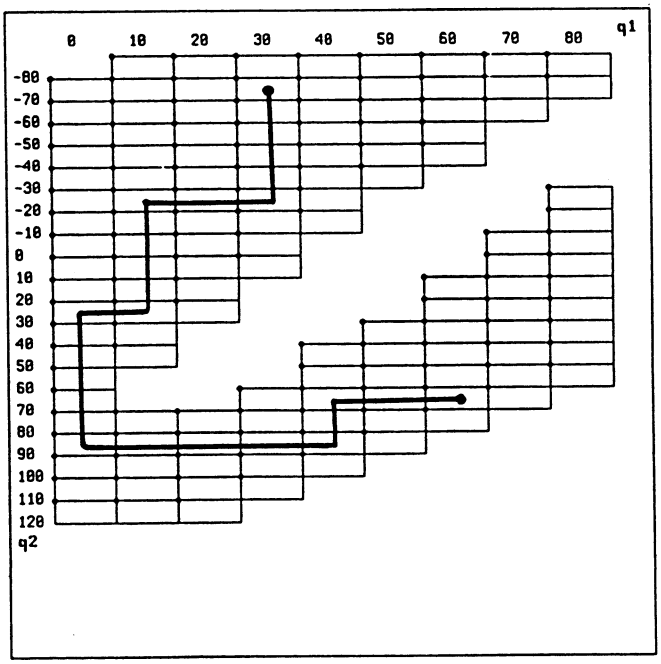

Figure 10.12: Espace des Configurations réduit (2 ddl).

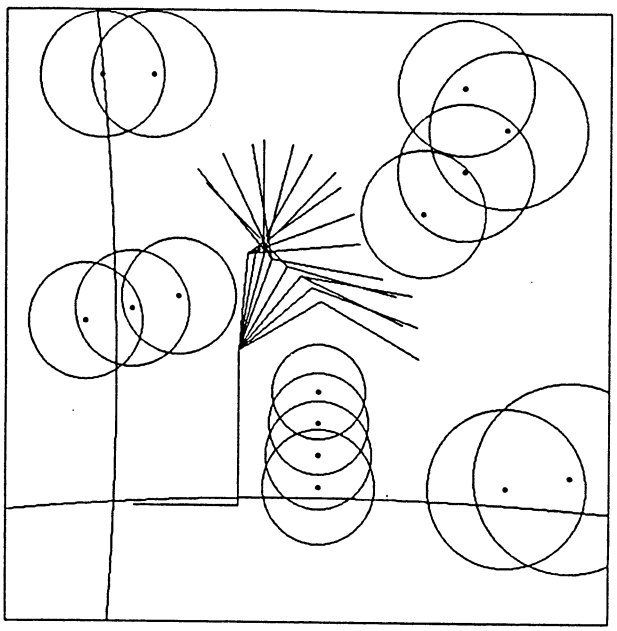

Figure 10.13: Trajectoire de référence pour le bras.

système au niveau de la rotation complète entre les obstacles A et B, tour à tour avançant puis reculant le long de la trajectoire de référence. Le temps de calcul total pour cette trajectoire est de 12 secondes environ.

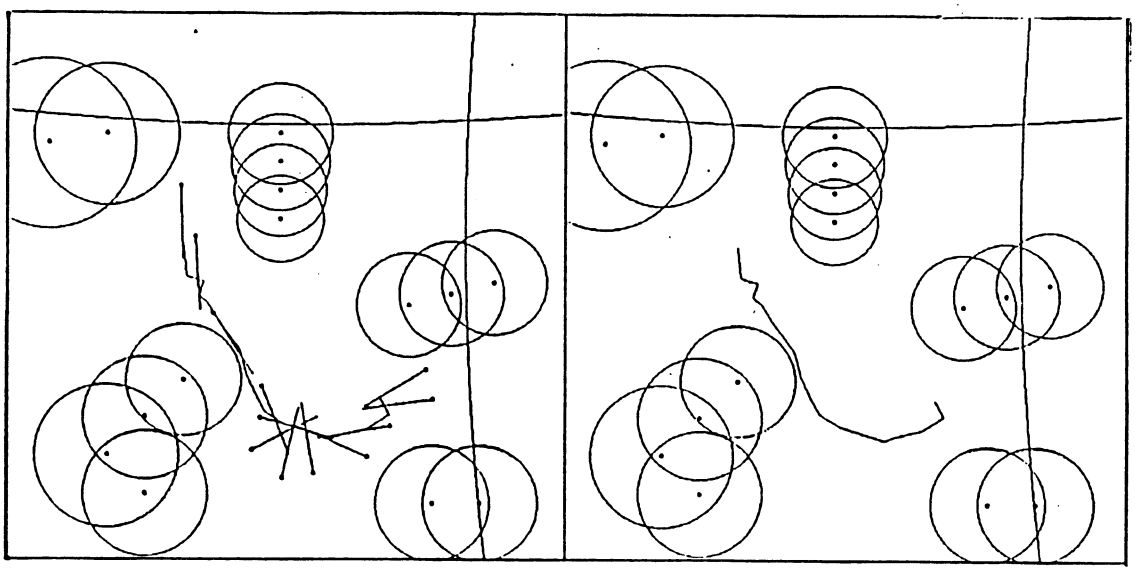

Figure 10.14: Orientations du poignet le long de la trajectoire de référence.

Si l'on compare les performances des deux approches, la méthode hybride se

révèle - comme on l'espérait - très intéressante par rapport à la méthode globale complète. Toutefois, sa fiabilité est tout de même beaucoup moins bonne. Ainsi nous avons recommencé l'expérience en déplaçant imperceptiblement l'obstacle D en direction du robot, de manière a contraindre encore plus le problème. L'espace des configurations obtenu est alors quasiment identique, à cela près que le passage entre les deux composantes connexes est encore plus ténu. Dans ces conditions, les performances de la méthode globale ne subissent aucun changement, aussi bien pour la génération du graphe que pour la détermination de la trajectoire. Par contre la détermination des orientations du poignet se révèle plus difficile et le temps de calcul est passé à 46 secondes. Mais il reste une "marge" de 10mn, et ce résultat reste donc bien meilleur que le calcul de l'espace des configurations complet. Le problème est que si l'on déplace encore l'obstacle de telle sorte que le déplacement du robot complet devienne impossible mais que celui du bras seul le soit toujours, on est alors dans un cas d'échec très difficile à détecter avec la méthode des potentiels. Ou du moins, il est difficile de savoir si cet échec est inhérent à la géométrie de la tâche demandée ou bien à la paramétrisation de l'algorithme de recherche même. La méthode globale au contraire, dans la limite des approximations effectuées, est infaillible sur ce point. Ce cas où la trajectoire de référence obtenue est "trop proche" des obstacles pour permettre le passage de l'ensemble main-poignet est réellement délicat à quantifier.

Conclusion

Au moment de terminer cet ouvrage, il est à craindre qu'en dépit du travail qu'il comporte il ne mette également en lumière quelques lacunes, études inachevées et autres terrains d'expérience fertiles méritant d'être abordés plus avant...le temps nous a par trop manqué. Si la véritable connaissance est de savoir ce que l'on ne sait pas, la rédaction d'un tel rapport permet à son auteur de progresser dans la connaissance et la compréhension du domaine et révèle toujours plus d'orientations nouvelles que l'on aimerait pouvoir prendre.

Cette rédaction fut donc particulièrement prolixe en enseignements, enrichissant inlassablement notre approche du problème général de nouvelles valeurs tour à tour contradictoires ou agonistes, la laissant au bout du compte inévitablement inachevée. Mais, devant l'alternative : écrire maintenant ou jamais, nous avons opté pour la première solution.

Nous avons donc longuement étudié une méthode de planification globale, et plus particulièrement le problème fondamental dans le contexte de la planification de trajectoires qu'est celui de la construction d'une représentation de l'espace libre du robot. Les résultats expérimentaux viennent malheureusement confirmer sans rémission les analyses de complexité les plus pessimistes. D'une part, la maîtrise théorique du problème n'est pas acquise, puisqu'il nous est impossible, au moins pour un manipulateur possédant des degrés de liberté en rotation, de construire une représentation exacte de l'espace libre. D'autre part,

même en sacrifiant aux approximations les plus draconiennes, il est très difficile d'obtenir rapidement une solution.

Une telle approche reste pourtant fort séduisante, tant par son aspect constructif que pour les possibilités assez riches qu'elle offre au niveau de la planification de trajectoires même (prise en compte des contraintes de la tâche). Elle est par contre relativement peu adaptée à un univers évolutif (et pourtant l'hypothèse d'un monde clos et immuable n'est vraiment pas réaliste), puisque le calcul de l'espace libre doit se faire à partir d'un état statique de l'univers, sans pouvoir simplement prendre en compte une variation élémentaire de celui-ci. L'emploi exclusif d'une méthode globale pour un manipulateur complet semble donc à proscrire, au moins dans l'état actuel de nos connaissances.

Il importe toutefois de bien distinguer les limitations algorithmiques de la méthode des limitations sommes toutes matérielles dues à la puissance des calculateurs utilisés. Si les premières semblent très difficiles à franchir, ces dernières au contraire sont chaque année repoussées plus loin. C'est ainsi un facteur 10 qui a été gagné sur ce seul critère entre la version initiale de PARSEC et la version en date. De plus, la méthode de construction de l'espace des configurations (par tranches) est parfaitement adaptée à une implantation en parallèle et, par exemple, l'utilisation des mêmes algorithmes sur une machine à N processeurs donnerait un gain net en temps d'un facteur N sur l'ensemble des calculs...

Nous avons ensuite étudié une méthode de génération locale de trajectoires basée sur l'application heuristique de champs de potentiel fictifs. Plus généralement, nous avons essayé de cerner les limites d'utilisation de la méthode globale et tenté de lui trouver un substitut efficace. Nous avons ensuite proposé une méthode hybride utilisant autant que possible la méthode globale (en l'occurence pour les mouvements du bras du manipulateur), et utilisant de manière locale la méthode des potentiels pour diriger l'orientation du poignet du robot. Les résultats obtenus nous semblent encourageants, les deux approches semblant bien se compléter même s'il n'est pas toujours simple de les faire communiquer et si cette méthode correctrice reste très difficile à paramétriser correctement.

Enfin, il nous est apparu une fois encore, grâce aux recherches dans le domaine "orienté" de la programmation automatique, que la stratégie la plus simple est parfois la plus satisfaisante. Ainsi les stratégies de type correctif nous semblent dans un bon nombre de cas préférables aux stratégies constructives, en particulier dans le contexte très appliqué de la robotique. Engendrer du premier coup un

plan correct est en effet un procédé extrêmement coûteux, long et finalement peu garanti, alors que produire un plan incorrect et imprécis mais tendant dans la bonne direction est une chose aisée.

Plus généralement, considérant l'ampleur et la complexité du problème de planification des mouvements d'un robot, nous sommes convaincus que les méthodes de raisonnement appliquées doivent être, tout comme les représentations utilisées, *multi-résolution.* Une méthode globale ne doit donner qu'une idée globale de la solution, et un problème bien spécifique doit être traité en conséquence. C'est pourquoi l'emploi de méthodes hybrides, utilisant algorithmes spécialisés et représentations dédiées, nous semble être la direction de recherche la plus prometteuse dans le domaine.

L'évolution du problème semble donc devoir reposer essentiellement sur une meilleure maîtrise des algorithmes de raisonnement géométrique et sur l'emploi de représentations totalement adaptées. Mais la vraie difficulté, plus encore que de concevoir les uns ou définir les autres, est alors de pouvoir décider quand et comment les utiliser. Et finalement, cette capacité de décision ne doit pas être du domaine de l'expertise humaine mais faire partie intégrante du système robotique. Alors, peut-être, pourra-t-on enfin parler à juste titre de *machine intelligente.*

Bibliographie

[Remarque] Le titre d'une publication n'étant pas toujours révélateur de son contenu exact, nous avons préféré offrir au lecteur soucieux de compléter son information une liste ordonnée de références relatives au sujet traité dans les pages qui précèdent. Les articles, thèses et livres cités sont donc classés, indépendamment de toute autre considération, selon leur apport principal à la résolution du problème de plannification de trajectoires.

Robotique et programmation automatique

[1-Bie76] A.W.Biermann : *"Approaches to Automatic Programming"*, Advances in Computers, vol.15, New York, 1976.

[1-Coi86] P.Coiffet : *La Robotique : Principes et Applications*, Ed. Hermes, Paris, 1986.

[1-Don83] B.R.Donald : *"The Mover's Problem in Automated Structural Design"*, Harvard Computer Graphics Conference, Cambridge, 1983.

[1-Fah74] S.E.Fahlman : *"A Planning System for Robot Construction Tasks"*, Artificial Intelligence 5, 1974.

[1-GP81] A.Giraud, R.Prajoux : *"La Problématique de l'Assemblage Automatique en Robotique"*, Journées Scientifiques et Techniques de la Production Automatisée, ADEPA, Toulouse, Juin 1981.

[1-Lau84] Ch.Laugier : *"Robot Programming Using a High-level Language and CAD Facilities"*, Robotics Europe Conference, Brussels, June 1984.

[1-Lau87] C.Laugier : *"Raisonnement Géométrique et Méthodes de Décision en Robotique. Application à la Programmation Automatique des Robots"*, Thèse de Doctorat es Sciences, INPG, Décembre 1987.

[1-LB85] T.Lozano-Pérez, R.A.Brooks : *"An Approach to Automatic Robot Programming"*, MIT Artificial Intelligence Laboratory, A.I. Memo 842, April 1985.

[1-LJM*87] T.Lozano-Pérez, J.L.Jones, E.Maser, P.A.O'Donnell, L.Grimson, P.-Tournassoud, A.Lanusse : *"HANDEY : A Robot System that Recognizes, Plans and Manipulates"*, IEEE, 1987.

[1-Loz82] T.Lozano-Pérez : *"Robot Programming"*, MIT, AI Memo 698, Dec.1982.

[1-LP76] T.Lozano-Pérez : *"The Design of a Mechanical Assembly System"*, Technical Report AI-TR-397, Artificial Intelligence Laboratory, Cambridge, December 1976.

[1-LT85] C.Laugier, J.Pertin-Troccaz : *"SHARP : A System for Automatic Robot Programming of Manipulation Robots*, 3rd Int. Symposium of Robotics Research, Gouvieux - France, Octobre 1985.

[1-LW77] L.I.Liebermann, M.A.Wesley : *"AUTOPASS : An Automatic Robot Programming System for Computer Controlled Mechanical Assembly"*, IBM Journal of Research and Development, 4-21, 1977.

[1-Maz87] E.Mazer : *"Handey: Un Modèle de Planificateur pour la Programmation Automatique des Robots"*, INPG, Décembre 1987.

[1-Pau81] R.P.Paul : *"Robot Manipulators : Mathematics, Programming and Control"*, MIT Press, 1981.

[1-RH85] P.G.Ránky, C.Y.Ho : *"Robot Modelling, Control and Applications with Software"*, IFS Publ., Springer-Verlag, Berlin, 1985.

[1-Tay76] R.H.Taylor : *"Synthesis of Manipulator Control Programs from Task-Level Specifications"*, AIM 228, Artificial Intelligence Laboratory, Stanford University, July 1976.

Sous-problèmes de la programmation automatique

[2-Buc87] S.T.Buckley : *"Planning and Teaching Compliant Motions Strategies"*, MIT, 1987.

[2-Gan86] Ch.Gandon : *"Introduction de la Compliance dans la Programmation Automatique des Robots"*, Thèse de 3ème cycle, LIFIA -INPG, Grenoble, 1986.

[2-Les85] P.Lespinard : *"Problèmes Liés à la Programmation Automatique des Robots"*, Rapport de DEA, LIFIA-IMAG, Sept.1985.

[2-LMT84] T.Lozano-Pérez, M.T.Mason, R.H.Taylor : *"Automatic Synthesis of Fine-Motion Strategies for Robots"*, Int. Jour. of Robotics Research, vol.3-1, Spring 1984.

[2-LP83] Ch.Laugier, J.Pertin : *"Automatic Grasping : A Case Study in Accessibility Analysis"*, International Conference on Advanced Software in Robotics, Liège, Mai 1983.

[2-Mas84] M.T.Mason : *"Automatic Planning of Fine Motions: Correctness and Completeness"*, IEEE Int. Conf. on Robotics and Automation, Atlanta, Mar.1984.

[2-PT86] J.Pertin-Troccaz : *"Modélisation du Raisonnement Géométrique pour la Programmation des Robots"*, Thèse de Doctorat, LIFIA - INPG, Grenoble, Mars 1986.

[2-The88] P.Theveneau : *"Utilisation du Raisonnement Géométrique pour la Planification en Robotique d'Assemblage"*, Thèse de Doctorat, LIFIA - INPG, Grenoble, 1988.

Incertitudes et tolérancement

[3-Bro82] R.A.Brooks : *"Symbolic Error Analysis and Robot Planning"*, The International Journal of Robotics Research, vol. 1-4, Winter 1982.

[3-Don87] B.R.Donald : *"Error Detection and Recovery for Robot Motion Planning with Uncertainty"*, MIT, July 1987.

[3-Dur87] H.F.Durrant-Whyte : *"Uncertain Geometry in Robotics"*, IEEE Int. Conf. on Robotics and Automation, Raleigh, Mar.1987.

[3-Erd84] M.A.Erdmann : *"On Motion Planning with Uncertainty"*, MIT Artificial Intelligence Laboratory, Technical Report 810, 1984.

[3-GSZ88] D.C.Gossard, H.Sakurai, R.P.Zuffante : *"Representing Dimensions, Tolerances and Features in MCAE Systems"*, IEEE Computer Graphics and Applications, March 1988.

[3-PPT86] J.Pertin-Troccaz, P.Puget : *"Contrôle dans le Système de Programmation Automatique SHARP : Gestion des Interdépendances Liées aux Contraintes d'Accessibilité et d'Incertitudes"*, Rapport de Recherche LIFIA no 50, June 1986.

[3-Pug88] P.Puget : *"Utilisation de Techniques de Preuve de Programme pour la Programmation Automatique des Robots"*, Thèse de Doctorat, LIFIA - INPG, Grenoble, 1988.

[3-Req83] A.G.Requicha : *"Toward a Theory of Geometric Tolerancing"*, Int. Journal of Robotics Research, vol.2 num.4, Winter 1983.

[3-SC83] R.C.Smith, P.Cheeseman : *"On the Representation and Estimation of Spatial Uncertainty"*, Int. Jour. of Robotics Research, vol.5-4, Winter 1986.

Modèles et représentations

[4-Bar79] A.Barr, C.Eastman, M.Henrion : *"Geometric Modeling : A Survey"*, Computer Aided Design, 11.5, 1979.

[4-Bo*83] P.Borrel et al : *"The Robotic Facilities in the CAD/CAM CATIA System"*, Brooks, IFS Publications, 1983.

[4-Bro81] Ch.M.Brown : *"Some Mathematical and Representational Aspects of Solid Modeling"*, IIIE Transactions on Pattern Analysis and Machine Intelligence, vol.sPAMI-3, July 1981.

[4-Bro82] Ch.M.Brown : *"PADL-2: a Technical Summary"*, IEEE Computer Graphics and Application, 2, Mars 1982.

[4-Cam81] S.Cameron' : *"BRIDGET: a Geometric Body Modeller"*, Dpt of Artificial Intelligence, Un. of Edimburgh, DAI Draft Working Paper, July 1981.

[4-Lau82] Ch.Laugier : *"LISP-3D: Logiciel Graphique pour la Manipulation et la Visualisation de Scènes Tridimensionnelles"*, Rapport de Recherche IMAG 328, Septembre 1982.

[4-Lau88] J.L.Laurière : *"IA : Représentation des Connaissances"*, Ed. Eyrolles, 1988.

[4-MO88] C.Mirolo, E.Pagello : *"A Solid Modeling System for Robot Action Planning"*, IEEE Computer Graphics and Applications, 1988.

[4-ORB79] J.O'Rourke, N.Badler : *"Decomposition of Three-Dimensional Objects into Spheres*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-1 3, 1979.

[4-Pas85] M.Pasquier : *"Modélisation et Représentation 3D pour la robotique"*, DEA Report, INP Grenoble, France, Sept.1985.

[4-PT88] M.Pasquier, P.Theveneau : *"A Geometric Modeller for an Automatic Robot Programming System"*, NATO Advanced Research Workshop on "CAD Based Programming for Sensors Based Robots", Milan, July 88.

[4-Per84] J.Pertin-Troccaz : *"S.M.G.R. : Un Système de Modélisation Géométrique et Relationnelle pour la Robotique"*, Rapport de recherche IMAG no 422, June 1984.

[4-Req80] A.A.G.Requicha : *"Representations for Rigids Solids : Theories, Methods and Systems"*, ACM Computing Surveys, 12, 4, December 1980.

[4-RV82] A.A.G.Requicha, H.B.Vœlcker : *"Solid Modeling : A Historical Summary And Contemporary Assessment"*, IEEE, Computer Graphics and its applications, Mars 1982.

[4-Wen79] M.J.Wenninger : *"Spherical Models"*, Cambridge University Press, 1979.

Géométrie algorithmique

[5-ACB80] N.Ahuja, R.T.Chien, N.Bridwell : *"Interference Detection and Collision Avoidance among Three Dimensional Objects"*, 1st Annual National Conference on Artificial Intelligence, Stanford University, August 1980.

[5-AHU74] .V.Aho, J.E.Hopcroft, J.D.Ullmann : *"The Design and Analysis of Computer Algorithms"*, Addison Wesley, 1974.

[5-Boi87] J.D.Boissonat : *"Complexité Géométrique et Robotique"*, Rapport Projet PRISME, INRIA, Juin 1987.

[5-Boy79] J.W.Boyse : *"Interference Detection among Solids and Surfaces"*, Communication of ACM 22, 1, January 1979.

[5-Buc85] C.Buckley : *"The Application of Continuum Methods to Path Planning"*, PhD Thesis, Dept. of Mechanical Engineering, Stanford University, 1985.

[5-Can84] J.F.Cany : *"Collision Detection for Moving Polyhedra"*, A.I. Memo 806, MIT Artificial Intelligence Laboratory, 1984.

[5-CR86] J.F.Cany, J.Reif : *"New Lower Bound Techniques for Robot Motion Planning Problems"*, 1986.

[5-DH55] J.Denavit, R.S.Hartenberg : *"A Kinematic Notation for Lower-pair Mechanisms Based on Matrices"*, Journal of Applied Mechanics, 22, Trans. ASME, 77, series E, June 1955.

[5-DK80] D.P.Dobkin, D.G.Kirkpatrick : *"Fast Detection of Polyhedral Intersections"*, Department of Electrical Engineering and Computer Science, Princeton University, 1980.

[5-For81] K.D.Forbus : *"A Study of Qualitative and Geometric Knowledge in Reasoning about Motion"*, MIT Artificial Intelligence Laboratory, Cambridge, 1981.

[5-For86] S.J.Fortune : *"Fast Algorithms for Polygon Containment"*, 12th Col. on Automata, Languages and Programming, Lecture Notes in Computer Science, Springer Verlag, 1986.

[5-Ham69] W.R.Hamilton : *"Elements of Quaternions"*, 3rd edition, Chelsea Pub. Co., New York, 1969.

[5-Hes84] D.Hestenes : *"Clifford Algebra to Geometric Calculus"*, Reidel, Dordrecht, 1984.

[5-Hil99] D.Hilbert : *"Foundations of Geometry"*, Original 1899, Repr. by Open Court, La Salle, IL, 1971.

[5-HW84] J.Hopcroft, G.Wilfong : *"On the Motion of Objects in Contact"*, Cornell University Computer Science Dept., Technical Report 84-602, New York, 1984.

[5-Kal82] Y.E.Kalay : *"Determining the Spatial Containment of a Point in General Polyhedra"*, Computer Graphics and Image Processing, 19, 1982.

[5-KM63] M.G.Kendall, P.A.P.Moran : *"Geometrical Probability"*, Hafner, New York, 1963.

[5-Lem02] E.Lemoine : *"Géométrographie"*, C.Naud, Paris, 1902.

[5-LS85] D.Leven, M.Sharir : *"An Efficient Motion Planning Algorithm for a Ladder Moving in a Two-dimensionnal Space amidst Polygonal Barriers"*, 1st ACM Symp. on Computational Geometry, 1985.

[5-MP78] D.E.Muller, F.P.Preparata : *"Finding the Intersection of Two Convex Polyhedra"*, Theoretical Computer Science, 7(2), 1978.

[5-ODY82] C.O'Du'nlaing, C.Yap : *"The Voronoi Diagram Method of Motion Planning : The Case of a Disc"*, Courant Institute of Mathematical Sciences, 1982.

[5-OSY82] C.O'Du'nlaing, M.Sharir, C.Yap : *"Retraction : A New Approach to Motion Planning"*, Courant Institute of Mathematical Sciences, 1982.

[5-PW82] E.Pervin, J.A.Webb : *"Quaternions In Computer Vision And Robotics"*, CMU Report, Pittsburg, 1982.

[5-PS85] F.P.Preparata, M.I.Shamos : *"Computational Geometry"*, Texts and Monographs in Computer Science, Springer-Verlag, New York, 1985.

[5-Sch81] J.T.Schwartz : *"Finding the Minimum Distance between Two Convex Polygons"*, Info. Proc. Lett., 13(4), 1981.

[5-Sha78] M.I.Shamos : *"Computational Geometry"*, Ph.D. Thesis, Dept. of Computer Science, Yale University, 1978.

[5-SS84] M.Sharir, A.Schorr : *"On Shortest Paths in Polyhedral Spaces"*, Proceedings of 16th ACM Symposium on Theory of Computing, New York, 1984.

[5-Sha87] M.Sharir : *"Efficient Algorithms for Planning Purely Translational Collision-free Motion in two and Three Dimensions"*, proc. IEEE, Int. Conference on Robotics Automation, Rayleigh, 1987.

[5-Sny85] J.Snygg : *"Expediting the Spinning Top Problem with a small amount of Clifford Algebra"*, Research Report, Upsalla College, New Jersey, August 1985.

[5-Yap85] C.Yap : *"Algorithmic Motion Planning"*, Advances in Robotics, 1, Lawrence Erlbaum Associates, 1985.

[5-ZA66] S.I.Zhukhovitsky, L.I.Avdeyeva : *"Linear and Convex Programming"*, Saunders, Philadelphia, 1966.

I.A. et planification

[6-BF80] A.Barr, E.A.Feigenbaum : *"Handbook of Artificial Intelligence"*, Computer Science Dpt., Stanford University, 1980.

[6-Bod77] M.A.Boden : *"Artificial Intelligence and Natural Man"*, Basic Books, New York, 1980.

[6-Bra84] M.Brady : *"Artificial Intelligence and Robotics"*, MIT Artificial Intelligence Laboratory, A.I. Memo 756, February 1984.

[6-Cha85] J.P.Changeux : *"L'Homme Neuronal"*, Gallimard, 1985.

[6-FN71] R.Fikes, N.Nilsson : *"STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving"*, AI Journal, 2-3/4, Sept. 1971.

[6-GP83] E.Grechanovsky, I.Pinsker : *"An Algorithm for Moving a Computer Controlled Manipulator While Avoiding Obstacles"*, Proceedings of 8th International Joint Conference on Artificial Intelligence, Karlsruhe, West Germany, 1983.

[6-Lau88] J.L.Laurière : *"IA : Résolution de Problèmes par l'Homme et la Machine"*, Ed. Eyrolles, 1987.

[6-Nil69] N.J.Nilsson : *"A Mobile Automaton : An Application of Artificial Intelligence Techniques"*, proceedings of the first IJCAI, May 1969.

[6-Nil80] N.J.Nilsson : *"Principles of Artificial Intelligence"*, Tioga Publishing Co, Springer-Verlag, 1980.

[6-Pea84] J.Pearl : *"Heuristics : Intelligent Search Strategies for Computer Problem Solving"*, Addison-Wesley Pub. Co., 1984.

[6-ST82] Y.Shirai, J.Tsuji : *"Artificial Intelligence : Concepts, Techniques and Applications"*, Iwanami Shoten, Tokyo, 1982; Wiley and Sons Ltd, Chichester, 1984.

[6-Win77] P.H.Winston : *"Artificial Intelligence"*, Addison Wesley Pub., 1977.

Langages de programmation

[7-Bur83] G.Burke *et al.* : *"The NIL Reference Manual"*, Laboratory of Computer Science, MIT, 1983.

[7-Maz83] E.Mazer : *"Geometric Programming of Assembly Robots (LM-GEO)"*, Int. Symposium of Advanced Software in Robotics, Liège, 1983.

[7-MM84] E.Mazer, J.F.Miribel : *"Le Langage LM - Manuel de Référence"*, Ed. Cepadues, Toulouse, 1984.

[7-PAB80] R.J.Popplestone, A.P.Ambler, I.M.Bellos : *"An Interpreter for a Language for Describing Assemblies"*, Artificial Intelligence, 14:79-107, 1980.

[7-SGS84] B.E.Shimano, C.C.Geshke, C.H.Spalding : *"VAL 2 : A New Robot Control System for Automatic Manufacturing"*, IEEE Int. Conference on Robotics and Automation, Atlanta, 1984.

[7-TSM82] R.H.Taylor, P.D.Summers, J.M.Meyer : *"AML : A Manufacturing Language"*, Int. Journal of Robotics Research, vol.1:3, Fall 1982.

Evitement d'obstacles et calcul de trajectoires

Généralités

[10-BH*83] M.Brady, J.M.Hollerbach, T.J.Johnson, T.Lozano-Pérez, M.T. Mason : *"Trajectory Planning, Robot Motion : Planning and Control"*, MIT press, Cambridge, 1983.

[10-Bro83] R.A.Brooks : *"Planning Collision Free Motions for Pick and Place Operations"*, The International Journal of Robotics Research 2, 4, 1983.

[10-Don84] B.R.Donald : *"Local and Global Techniques for Motion Planning"*, S.M. Thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, 1984.

[10-Fav86] B.Faverjon : *"Object Level Programming for Industrial Robots"*, IEEE Conference on Robotics and Automation, San Francisco, April 1986.

[10-Kun82] H.Kuntze : *"Methods for Collision Avoidance in Computer Controlled Industrial Robots"*, Proceedings of 12th International Symposium on Industrial Robots, Paris, France, 1982.

[10-LP81] T.Lozano-Pérez : *"Automatic Planning of Manipulator Transfer Movements"*, IEEE Transactions on Systems, Man and Cybernetics, SMC-11, 10, 1981.

[10-MA82] J.K.Myers, G.J.Agin : *"A Supervisory Collision Avoidance System for Robot Controllers"*, Robotics Research and Advanced Applications, American Society of Mechanical Engineers, New York, 1982.

[10-NV83] Nguyen, Van-Duc : *"The Find-Path Problem in the Plane"*, MIT Artificial Intelligence Laboratory, A.I.Memo 760, 1983.

[10-Pau81] R.P.Paul : *"The Theory and Practice of Robot Manipulator Programming and Control"*, MIT Press, Cambridge, 1981.

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

[10-Udu77] S.Udupa : *"Collision Detection and Avoidance in Computer Controlled Manipulators"*, PhD thesis, Dept. of Electrical Engineering and Computer Science, California Institute of Technology, 1977.

[10-Wid74] L.C.Widdoes : *"Obstacle Avoidance"*, Internal Memo, Artificial Intelligence Laboratory, Stanford University, 1974.

Espace de planification

[11-Bro82] R.A.Brooks : *"Solving the Findpath Problem by Good Representation of Free Space"*, 2nd American Association for Artificial Intelligence Conference, Carnegie-Mellon, August 1982.

[11-BLP83] R.A.Brooks, T.Lozano-Pérez : *"A Subdivision Algorithm in Configuration Space for Findpath with Rotation"*, MIT Artificial Intelligence Laboratory, A.I. Memo 684, 1983.

[11-Fav84] B.Faverjon : *"Obstacle Avoidance Using an Octree in the Configuration Space of a Manipulator"*, International Conference on Robotics, Atlanta, March 1984.

[11-HGK83] J.A.Hansen, K.C.Gupta, S.M.K.Kazerounian : *"Generation and Evaluation of the Workspace of a Manipulator"*, The International Journal of Robotics Research 2, 3, Fall 1983.

[11-HJW82] J.Hopcroft, D.Joseph, S.Whitesides : *"On the Mouvement of Robot Arms in 2-Dimensional Regions"*, Cornell University, Technical Report 82-486, New-York, 1982.

[11-HW84] J.Hopcroft, G.Wilfong : *"Reducing Multiple Objects Motion Planning to Graph Searching"*, Technical Report 84, Ithaca, Cornell University Computer Science Dept., New York, 1984.

[11-LP83] T.Lozano-Pérez : *"Spatial Planning : a Configuration Space Approach"*, IEEE Transactions on Computers, C-32, 1983.

Robots manipulateurs

[12-Bro83] R.A.Brooks : *"Find-Path for a PUMA-Class Robot"*, AAAI, Washington DC, 1983.

[12-CZZ84] L.Chien, R.Zhang, B.Zhang : *"Planning Collision Free Paths for Robotic Arm among Obstacles"*, IEEE Transactions on Pattern Analysis and Machine Intelligence 6, 9, 1984.

[12-Don84] B.R.Donald : *"Motion Planning with Six Degrees of Freedom"*, MIT Artificial Intelligence Laboratory, A.I. Technical Report 791, 1984.

[12-Dup88] P.E.Dupont : *"Planning Collision-free Paths for Kinematically Redundant Robots by Selectively Mapping Configuration Space"*, PhD Thesis, Renssealer Polytechnic Inst., Troy, NY, May 1988.

[12-FT87] B.Faverjon, P.Tournassoud : *"A Local Based Approach for Path Planning of Manipulators With a High Number of Degrees of Freedom"*, IEEE Robotics and Automation Conference, 1987.

[12-Ger85] F.Germain : *"Planification de Trajectoires sans Collision"*, DEA Report, INP Grenoble, France, June 1985.

[12-Gil85] W.Gilles : *"Universal Computer Program for Seeking a Collision Free Path for an Industrial Robot"*, 15th ISIR, 1985.

[12-Gou84] L.Gouzènes : *"Strategies for Solving Collision Free Trajectories Problems for Mobile and Manipulator Robots"*, The International Journal of Robotics Research 3, 4, Winter 1984.

[12-HT87] T.Hasegawa, H.Terasaki : *"Collision Avoidance : Divide-and-Conquer Approach by Determining Intermediate Goals"*, 3rd Int. Conference on Advanced Robotics, Versailles, 1987.

[12-Lew74] R.A.Lewis : *"Autonomous Manipulation on a Robot: Summary of Manipulator Software Functions"*, Jet Propulsion Laboratory, California Institute of Technology, 1974.

[12-Loz86] T.Lozano-Pérez : *"A Simple Motion Planning Algorithm for General Robot Manipulator"*, Technical Report, AI Memo 896, MIT Artificial Intelligence Lab., June 1986.

[12-LW79] T.Lozano-Pérez, M.A.Wesley : *"An Algorithm for Planning Collision Free Paths among Polyhedral Obstacles"*, Communication of the Association of Computing Machinery 22, 10, 1979.

[12-OMT83] H.Ozaki, A.Mohri, M.Takata : *"On the Collision Free Movement of a Manipulator"*, Proceedings in Advanced Software in Robotics, Luik University, 1983.

Robots mobiles, systèmes multi-robots

[13-Cha81] R.Chatila : *"Système de Navigation pour un Robot Mobile Autonome : Modélisation et Processus Décisionnels"*, PhD Thesis, Paul Sabatier University, Toulouse, France, 1981.

[13-Cha82] R.Chatila : *"Path Planning and Environmental Learning in a Mobile Robot System"*, in ECAI, Orsay, France, August 1982.

[13-Cro86] J.L.Crowley : *"Path Planning and Obstacle Avoidance"*, LIFIA laboratory, research report 46, January 1986.

[13-Don83] B.R.Donald : *"Hypothesising Channels Through Free Space in Solving the Findpath Problem"*, MIT Artificial Intelligence Laboratory, A.I. Memo 736, 1983.

[13-FH84] E.Freund, H.Hoyer : *"Collision Avoidance for Industrial Robots with Arbitrary Motion"*, Journal of Robotic Systems, 1984.

[13-FWY86] S.J.Fortune, G.Wilfong, C.Yap : *"Coordinated Motion of Two Robot Arms"*, proc. IEEE, Int. Symp. on Robotics and Automation, San Francisco, 1986.

[13-Mor80] H.P.Moravec : *"Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover"*, PhD Thesis, Artificial Intelligence Laboratory, Stanford University, 1980.

[13-SS84] J.T.Schwartz, M.Sharir, J.E.Hopcroft : *"On the Complexity of Motion Planning for Multiple Independent Objects"*, The International Journal of Robotics Research 3, 4, Winter 1984.

[13-Tou86] P.Tournassoud : *"A Strategy for Obstacle Avoidance and its Application to Multi-Robot Systems"*, Internal Report, INRIA, France, 1986.

[13-Yap84] C.Yap : *"Coordinating the Motion of Several Discs"*, Tech. Report 105, Computer Science Dpt, Courant Institute, NY University, 1984.

Dynamique et contrôle

[14-Kha80] O.Khatib : *"Commande Dynamique dans l'Espace Opérationnel des Robots Manipulateurs en Présence d'Obstacles"*, Thèse de Doctorat, ENSAE, Toulouse, France, Décembre 1980.

[14-Kha85] O.Khatib : *"Real Time Obstacle Avoidance for Manipulators and Mobile Robots"*, IEEE Conference on Robotics and Automation, March 1985.

[14-KM85] C.A.Klein, A.A.Maciejewski : *"Obstacle Avoidance for Kinematically Redundant Manipulators in Dynamically Varying Environments"*, The International Journal of Robotics Research 4, 3, Fall 1985.

[14-Kro84] B.H.Krogh : *"A Generalized Potential Field Approach to Obstacle Avoidance Control"*, in Robotic Research : the Next Five Years and Beyond, SME, One SME Drive, Dearborn Mich, 1984.

[14-Ram83] C.A.Ramirez : *"Stratified Levels of Risk for Collision Free Robot Guidance"*, 15th ISIR, 1985.

[14-Tho84] C.E.Thorpe : *"Path Relaxation : Path Planning for a Mobile Robot"*, in AAAI, Kaufman, August 1984.

Annexe A

Un outil de calcul géométrique : les quaternions de Hamilton

A.1 Présentation

En robotique comme en graphique ou en vision par ordinateur, les algorithmes utilisables en pratique dépendent largement de la nature des outils mathématiques disponibles. La simulation du mouvement d'un robot manipulateur et la recherche de trajectoires sans collision (détermination d'intersection de surfaces ou de volumes...) reposent essentiellement sur des calculs géométriques. Par conséquent, le formalisme utilisé pour représenter les équations du problème (morphologie des objets, position spatiale, équations du mouvement...) est un facteur capital du point de vue des performances des applications concernées, voire même de leur simple faisabilité.

Dès lors apparaît clairement le besoin d'un outil mathématique aussi puissant que la représentation vectorielle au niveau notation, mais qui autorise de manière aisée certaines opérations non directement représentables grâce aux vecteurs, comme le sont par exemple les rotations. Les quaternions de Hamilton satisfont ces besoins.
Si l'application évidente des quaternions est la géométrie, d'autres domaines sont aussi concernés, telle par exemple la physique des particules où sont utilisés des quaternions généralisés (algèbre de Clifford [5-Hes84]) afin de résoudre des problèmes de spin [5-Sny85]...

Les quaternions ont été découvert par Hamilton vers 1890, alors qu'il cherchait à résoudre le problème de savoir quel résultat donne la division d'un vecteur par un autre. La légende dit que le mathématicien, ayant trouvé la réponse alors qu'il traversait un pont, grava sur la pierre la formule qui constitue la base de

la théorie des quaternions :

$$i^2 = j^2 = k^2 = ijk = -1$$

Cette formule donne la règle de multiplication des quaternions. Hamilton a prouvé qu'il n'existait pas de système tridimensionnel possédant un nombre raisonnable des propriétés des réels et des complexes [5-Hil99]. Par contre, cela est possible en quatre dimensions. Ainsi les quaternions préservent toutes les propriétés des réels et des complexes, à l'exception de la commutativité de la multiplication; de plus ils peuvent facilement représenter toutes les opérations sur l'espace cartésien à trois dimensions : rotations, transformations affines, projections...

Bien que diverses formes soient utilisées pour décrire les quaternions, la plus simple est sans doute la forme quadrinomiale standard :

$$\mathcal{Q} = \{Q = \alpha + \beta i + \gamma j + \delta k \ : \ \alpha, \beta, \gamma, \delta \in \Re\}$$

laquelle reflète l'isomorphisme de l'espace vectoriel des quaternions engendré par la base $[i \ j \ k]$, à $\Re^4$. Nous utiliserons dans la suite la forme condensée de cette représentation :

$$Q = q_0 + \overline{q}$$

où $q_0 = \alpha$ est la "partie réelle" du quaternion et $\overline{q} = (q_x \ q_y \ q_z)$ sa "partie vectorielle". $q_x = \beta$, $q_y = \gamma$ et $q_z = \delta$ sont alors les composantes du *pseudo-vecteur* $\overline{q}$ sur la pseudo-base $(\overline{i} \ \overline{j} \ \overline{k})$ (cf. A.3.1). A l'approche "espace vectoriel réel" conduisant à la notation homogène du quadruplet, nous préférons donc l'approche "espace complexe" dont la notation dissymétrique facilite l'analogie avec l'espace affine que nous utilisons dans nos calculs géométriques.

A.2 Définitions et propriétés

A.2.1 Définition des opérations

Addition

Soient $P = p_0 + \overline{p} = (\alpha \ \beta \ \gamma \ \delta)$ et $Q = q_0 + \overline{q} = (a \ b \ c \ d)$ deux quaternions, alors l'addition est définie de la manière suivante :

$$P \oplus Q = ((\alpha + a) \ (\beta + b) \ (\gamma + c) \ (\delta + d))$$

qui s'écrit donc sous forme condensée

$$P \oplus Q = (p_0 + q_0) + (\overline{p} + \overline{q})$$

Multiplication

La multiplication des quaternions est définie par la règle de base A.1 et donne :

$$P \otimes Q = ((\alpha a - \beta b - \gamma c - \delta d)\ (\alpha b + \beta a + \gamma d - \delta c)\ (\alpha c + \gamma a + \delta b - \beta d)\ (\alpha d + \delta a + \beta c - \gamma b))$$

qui s'écrit sous forme condensée :

$$P \otimes Q = p_0 q_0 - \overline{p} \cdot \overline{q} \ + \ p_0 \overline{q} + q_0 \overline{p} + \overline{p} \times \overline{q}$$

La non-commutativité de cette multiplication apparaît nettement dans cette dernière expression du fait de la présence du produit vectoriel sur les pseudo-vecteurs $\overline{p}$ et $\overline{q}$.

Produit scalaire, norme

Le produit scalaire de quaternions est défini naturellement par :

$$P \bullet Q \ = \ p_0 q_0 + \overline{p} \cdot \overline{q}$$

et la forme quadratique de signature $(4,0)$

$$\| Q \| = \ Q \bullet Q \ = \ (q_0^2 + \overline{q} \cdot \overline{q})^{\frac{1}{2}}$$

est donc bien une norme.

A.2.2 Propriétés

Addition

- ⋆ Stabilité : $\forall\, P, Q \in \mathcal{Q},\ P \oplus Q \in \mathcal{Q}$
- ⋆ Commutativité : $\forall\, P, Q \in \mathcal{Q},\ P \oplus Q = Q \oplus P$
- ⋆ Associativité : $\forall\, P, Q, R \in \mathcal{Q},\ (P \oplus Q) \oplus R = P \oplus (Q \oplus R)$
- ⋆ Identité : $\exists\, \emptyset \in \mathcal{Q} \mid \forall P \in \mathcal{Q},\ \emptyset \oplus P = P \oplus \emptyset = P$
  avec *bien sûr* $\emptyset = (0\ 0\ 0\ 0)$
- ⋆ Inverse : $\forall P \in \mathcal{Q},\ \exists Q \in \mathcal{Q} \mid P \oplus Q = Q \oplus P = \emptyset$
  avec $Q' = -Q$, *opposé de* Q
- ⋆ Conjugué : on notera $\overline{Q} = q_0 - \overline{q}$ le conjugué de Q.

Multiplication

- ⋆ Stabilité : $\forall\, P, Q \in \mathcal{Q},\ P \otimes Q \in \mathcal{Q}$
- ⋆ Associativité : $\forall\, P, Q, R \in \mathcal{Q},\ (P \otimes Q) \otimes R = P \otimes (Q \otimes R)$
- ⋆ Identité : $\exists\, \mathcal{I} \in \mathcal{Q} \mid \forall P \in \mathcal{Q},\ \mathcal{I} \otimes P = P \otimes \mathcal{I} = P$
  avec $\mathcal{I} = (1\ 0\ 0\ 0)$
- ⋆ Inverse : $\forall\, P \neq \emptyset \in \mathcal{Q},\ \exists Q \in \mathcal{Q} \mid P \otimes Q = Q \otimes P = \mathcal{I}$
  avec $Q = P^{-1} = \overline{P} \,/\, \| P \|$
- ⋆ Distributivité : $\forall\, P, Q, R \in \mathcal{Q},\ (P \oplus Q) \otimes R = (P \otimes Q) \oplus (P \otimes R)$
- ⋆ Diviseurs de zéro : $P \otimes Q = \emptyset \Rightarrow P = \emptyset \wedge Q = \emptyset$

A.3 Une nouvelle géométrie

A.3.1 Les vecteurs en tant que quaternions

On associe à tout vecteur $\underline{u} = (u_x\ u_y\ u_z)$ de l'espace cartésien le pseudo-vecteur $\overline{u} = (u_x\ u_y\ u_z)$ qui est donc le quaternion $u_x\,\overline{\imath} + u_y\,\overline{\jmath} + u_z\overline{k}$.

On voit que les quaternions et pseudo-vecteurs i, j et k peuvent être alors interprétés comme trois vecteurs formant une base orthonormale $(\overline{\imath}\ \overline{\jmath}\ \overline{k})$. Cet artifice permet donc d'assimiler points et vecteurs de l'espace cartésien à un quaternion, tout comme le sont les matrices de rotation, autorisant ainsi à effectuer les opérations géométriques usuelles dans l'espace des quaternions.

Le fait que le sous-espace vectoriel engendré par $[i\ j\ k]$ ne soit pas stable pour la multiplication des quaternions (comme le prouve assez l'équation de base $i^2 = -1$) n'est pas un obstacle puisque l'on n'aura jamais à multiplier des vecteurs au sens des quaternions...

A.3.2 Représentation des rotations

L'intérêt principal des quaternions réside dans leur capacité à représenter les matrices de rotation. En effet, la matrice associée à une rotation d'angle θ autour d'un vecteur unitaire quelconque $\overline{n} = (0\ n_x\ n_y\ n_z)$ est complexe et peu "lisible", les paramètres en question étant difficiles à extraire. Par contre, la représentation associée à la même matrice en utilisant les quaternions est très simple, à savoir :

$$\mathcal{R} = \cos\frac{\theta}{2} + \sin\frac{\theta}{2}\,\overline{n}$$

Les paramètres (axe - angle) sont cette fois immédiats à obtenir. Ainsi, pour $Q = q_0 + q_x\,\overline{i} + q_y\,\overline{j} + q_z\,\overline{k}$ on trouve :

$$\theta = 2\arctan(\frac{\| Q \|}{q_0}) \quad et \quad \overline{n} = \frac{1}{\| Q \|}(q_x\ q_y\ q_z)$$

A.3.3 Calcul vectoriel

Produit matrice - vecteur

On a dit précédemment que l'ensemble des pseudo-vecteurs

$$\mathcal{I} = \{\beta i + \gamma j + \delta k \ :\ \beta, \gamma, \delta \in \Re\}$$

n'est pas stable pour la multiplication; a fortiori le produit d'un tel élément par un quaternion n'est pas un pseudo-vecteur. On est donc amené à redéfinir le produit d'un vecteur par une matrice, ou du moins celui de leur forme quaternion : pour tout vecteur représenté par un pseudo-vecteur $\overline{u}$ et pour toute rotation représentée par un quaternion $Q = q_0 + \overline{q}$, l'image du vecteur par la rotation s'écrit :

$$\overline{u}' = Q \otimes \overline{u} \otimes Q^{-1}$$

ce qui donne sous forme condensée :

$$\overline{u}' = (q_0^2 - \overline{q} \cdot \overline{q})\,\overline{u} + 2(\overline{q} \cdot \overline{u})\,\overline{q} + 2q_0\,\overline{q} \times \overline{u}$$

Si θ est l'angle de la rotation et $\overline{n}$ son axe, on a $q_0 = \cos\frac{\theta}{2}$ et $\overline{q} = \sin\frac{\theta}{2}\,\overline{n}$, ce qui nous donne l'expression simplifiée suivante :

$$\overline{u}' = \cos\theta\,\overline{u} + (1 - \cos\theta)(\overline{q} \cdot \overline{u})\,\overline{q} + \sin\theta\,\overline{q} \times \overline{u}$$

Produits matriciels

Le produit de deux matrices de rotation de quaternions associés P et Q est simplement $P \otimes Q$, et l'inverse d'une rotation a pour quaternion associé l'inverse Q^{-1}. Si l'on n'utilise que des "vraies" rotations, soit de norme unitaire, on a $Q^{-1} = \overline{Q}$, et inverser une matrice revient à changer trois signes! D'autre part, le résultat intuitif qui veut que seules commutent des rotations dont les axes sont parallèles se retrouve dans l'expression de ce produit (A.2.1) où l'on voit bien que l'identité $P \otimes Q = Q \otimes P$ n'est vérifiée que pour un produit vectoriel nul, donc des vecteurs axiaux colinéaires.

D'autre part, la généralisation du théorème de De Moivres :

$$e^{\theta n} = \cos(\theta) + \sin(\theta)n$$

permet de représenter une rotation d'angle θ et d'axe $\overline{n}$ sous la forme $e^{\theta\overline{n}/2}$, permettant ainsi la définition de fonctions hyperboliques et donnant surtout d'intéressantes relations sur leurs produits, comme $Q^{\alpha+\beta} = Q^{\alpha} \otimes Q^{\beta}$ ou $Q^{\alpha\,\beta} = (Q^{\alpha})^{\beta}\ldots$

Autres relations intéressantes

Signalons la relation entre le produit de vecteurs au sens des quaternions, le produit scalaire et le produit vectoriel :

$$\overline{u} \otimes \overline{v} = \overline{u} \times \overline{v} - \overline{u} \cdot \overline{v}$$

On en déduit, vu l'antisymmétrie du produit en question :

$$\overline{u} \cdot \overline{v} = -\frac{1}{2}(\overline{u} \otimes \overline{v} + \overline{v} \otimes \overline{u}) \quad et \quad \overline{u} \times \overline{v} = \frac{1}{2}(\overline{u} \otimes \overline{v} - \overline{v} \otimes \overline{u})$$

D'autre part le produit mixte de trois vecteurs s'écrit assez simplement :

$$[\overline{u}\,\overline{v}\,\overline{w}] = \frac{1}{2}(\overline{w} \otimes \overline{v} \otimes \overline{u} - \overline{u} \otimes \overline{v} \otimes \overline{w})$$

A.3.4 Les transformations affines

Définition

Une transformation affine est la composition d'une rotation par une translation. Comme elle ne peut être directement représentée par un quaternion, on est amené à utiliser une structure *vecteur + quaternion* :

$$\mathcal{T} = (T, Q)$$

où T est le vecteur de translation (en fait le pseudo-vecteur associé) et Q le quaternion associé à la rotation.
Selon cette représentation, une transformation affine nécessite 7 paramètres (contre 12 pour les matrices homogènes par exemple), et même seulement 6 si l'on utilise exclusivement des rotations normées, éliminant ainsi toute redondance.

Composition et inverse de transformations

La *composition* est définie simplement par :

$$(T_1, Q_1) * (T_2, Q_2) = (T_1 \oplus (Q_1 \otimes T_2 \otimes Q_1^{-1}), Q_1 \otimes Q_2)$$

de manière analogue au produit par blocs des matrices homogènes :

$$\left(\begin{array}{c|c} R_1 & T_1 \\ \hline 0 & 1 \end{array}\right) \times \left(\begin{array}{c|c} R_2 & T_2 \\ \hline 0 & 1 \end{array}\right) = \left(\begin{array}{c|c} R_1\,R_2 & T_1 + R_1 T_2 \\ \hline 0 & 1 \end{array}\right)$$

L'*inverse* pour la composition est de la même façon :

$$(T, Q)^{-1} = (-Q^{-1} \otimes T \otimes Q, Q^{-1})$$

A.3.5 Comparaison d'efficacité dans les calculs

A l'avantage d'offrir une représentation plus claire et plus facile à manipuler, dont les paramètres sont aisés à extraire, l'utilisation des quaternions ajoute de meilleures performances au niveau implantation.

Opérations nécessaires

A peu près équivalents aux matrices homogènes dans les calculs de composition (lignes 1 à 3 du tableau ci-dessous), les quaternions s'avèrent bien meilleurs au niveau du passage de la représentation algébrique vers la représentation numérique et réciproquement (lignes 4 et 5). Or c'est là une opération très fréquente en robotique, en particulier dans l'utilisation du changeur de coordonnées.

Opération	Quaternions	Coord. Homogènes
redondance	non	oui
nombre de paramètres	6	12
composition de rotations	24 +, 8 ×	15 +, 25 ×
rotation d'un vecteur	12 +, 22 ×	6 +, 9 ×
composition de transformations	21 +, 38 ×	21 +, 34 ×
codage d'une rotation	4 ×, 1 sin cos	10 +, 15 ×, 1 sin cos
extraction de ses paramètres	4 ×, 1 $\sqrt{}$, 1 arctan	10 +, 15 ×, 1 sin cos

Précision des calculs

L'utilisation des quaternions offre une meilleure "robustesse" dans la mesure où la précision des calculs reste toujours assez bonne alors qu'elle se dégrade très vite avec les matrices homogènes.

Ainsi, par exemple, le produit d'une rotation d'angle $\frac{2\,\pi}{n}$ par elle-même peut être effectué de l'ordre de $10 \cdot k \cdot n$ fois avec les quaternions et seulement $k \cdot n$ fois avec les matrices homogènes pour obtenir la même précision. Et pour des cas limites, comme par exemple celui d'un angle de rotation très petit, ce rapport augmente encore considérablement...

D'autre part, des formules comme celle donnant l'image d'un vecteur par une rotation (A.3.3) présentent l'avantage de pouvoir être approximées de manière intéressante. En l'occurence, pour un angle petit, il vient $\overline{u}' = \overline{u} + \theta\,\overline{q} \times \overline{u}$ et l'on voit que la variation du vecteur se fait selon une direction normale au plan défini par lui-même et l'axe de rotation, et on obtient un résultat assez correct même si celui-ci est connu avec une très forte imprécision.

A.4 Applications à la robotique

A.4.1 Représentation géométrique d'une chaîne cinématique

Chaque élément de la chaîne cinématique d'un manipulateur est référencé par rapport à un repère propre, lequel est lié par une transformation à son prédécesseur et à son successeur dans la hiérachie des repères. Déterminer la position d'un élément du manipulateur demande de pouvoir extraire facilement les paramètres de la liaison à partir de la donnée de la transformation et, inversement, de calculer celle-ci de manière performante à partir des premiers (changeur de coordonnée).

Intérêt général

L'utilisation des matrices homogènes et des angles d'Euler est donc trop coûteuse, et d'autres méthodes ont été proposés pour pallier cet inconvénient. Ainsi les paramètres de Denavit-Hartenberg [5-DH55] permettent de représenter une liaison entre deux éléments de la structure articulée par quatre paramètres seulement, dont l'un est d'ailleurs la valeur articulaire elle-même. Toutefois ce formalisme souffre de quelques restrictions notoires interdisant son emploi généralisé.

Si la représentation des liaisons prismatiques est à peu près équivalente dans tous les formalismes, il n'en est pas de même des liaisons rotoïdes. La facilité des quaternions à représenter une rotation quelconque est donc sur ce point un avantage prépondérant, qui plus est dans le cas particulier des transformations

effectuées dans un changeur de coordonnées, les rotations étant alors très souvent coaxiales ou orthogonales.

Ainsi, si l'on considère une suite d'éléments E_i liés entre eux par des articulations rotoïdes d'axe n_i et d'angle θ_i, celles-ci s'expriment par un quaternion $Q_i = e^{\theta_i \overline{n}_i/2}$ et la composition de deux rotations consécutives s'écrit

$$e^{\theta_i \overline{n}_i/2}\ e^{\theta_{i+1} \overline{n}_{i+1}/2}$$

Dans le cas coaxial, la composition se ramène trivialement à l'expression

$$e^{\frac{1}{2}(\theta_i \pm \theta_{i+1})\overline{n}_i}$$

Dans le cas orthogonal, la relation

$$e^P e^Q = e^{P+Q} \iff P \text{ et } Q \text{ commutent}$$

permet de simplifier l'expression en

$$e^{\frac{1}{2}(\theta_i \overline{n}_i + \theta_{i+1} \overline{n}_{i+1})}$$

Exemple du calcul d'un changeur de coordonnées direct

Dans l'exemple du robot SCEMI que nous avons utilisé (cf 10.1.2), les six axes rotoïdes correspondent parfaitement à ces critères de parallélisme ou d'orthogonalité, et seuls trois translations interviennent dans la transformation globale entre le repère station et le repère de la pince. Cette transformation vaut :

$$(T_1, Q_1) * (T_2, Q_2) * (T_3, Q_3) * (T_4, Q_4) * (T_5, Q_5) * (T_6, Q_6)$$

Les trois derniers axes étant rotoïdes purs et concourants, la rotation associée s'écrit :

$$Q_{4-6} = (T_4, Q_4) * (T_5, Q_5) * (T_6, Q_6) = e^{\frac{1}{2}\sum_{i=4}^{6} \theta_i n_i}$$

De plus T_{4-6} est nulle et multiplier à gauche par $\mathcal{T}_3$ donne donc :

$$(T_3, Q_3 \otimes Q_{4-6})$$

Multiplier alors à gauche par $\mathcal{T}_1 \otimes \mathcal{T}_2$ donne la transformation globale :

$$(T_1 \oplus Q_1 \otimes T_2 \otimes Q_1^{-1} \oplus (Q_1 \otimes Q_2) \otimes T_3 \otimes (Q_1 \otimes Q_2)^{-1}, Q_1 \otimes Q_2 \otimes Q_3 \otimes Q_{4-6})$$

Cette expression peut encore être simplifiée par le fait que Q_1 et Q_2 sont orthogonales et que Q_2 et Q_3 sont coaxiales, d'où :

$$Q_1 \otimes Q_2 = e^{\frac{1}{2}(\theta_1 \overline{n}_1 + \theta_2 \overline{n}_2)} \quad et \quad Q_1 \otimes Q_2 \otimes Q_3 = e^{\frac{1}{2}(\theta_1 \overline{n}_1 + (\theta_2 \pm \theta_3)\overline{n}_2)}$$

On peut alternativement utiliser l'orthogonalité de Q_2 et T_3 pour calculer aisément $Q_2 \otimes T_3 \otimes Q_2^{-1}$.

Enfin, il se trouve que le vecteur de la translation T_2 est orthogonal à l'axe de Q_1, donc leur produit commute (illustrant au passage l'intérêt d'avoir une même représentation pour vecteurs et rotations) et par conséquent :

$$Q_1 \otimes T_2 \otimes Q_1^{-1} = T_2$$

Finalement, la transformation globale s'écrit sous la forme très simplifiée :

$$(T_1 \oplus T_2 \oplus e^{\frac{1}{2}(\theta_1 \overline{n}_1 + \theta_2 \overline{n}_2)} \otimes T_3 \otimes e^{-\frac{1}{2}(\theta_1 \overline{n}_1 + \theta_2 \overline{n}_2)},\ e^{\frac{1}{2}\sum_{i=1}^{6} \theta_i n_i})$$

On s'aperçoit qu'il reste assez peu de multiplications "standards", la plupart ayant été réduites en utilisant la morphologie du manipulateur et les propriétés des quaternions. Les performances du changeur de coordonnées s'en trouvent donc nettement améliorées.

A.4.2 Calcul de débattements et de collision

Les algorithmes détaillés précédemment reposent sur un mécanisme d'approximation permettant de construire un modèle surcontraint mais simplifié des C-surfaces, c'est-à-dire des surfaces d'intersection entre les éléments du robot et les obstacles. Une méthode analytique n'est pas en effet envisageable, la nature des contraintes induisant en particulier dans les calculs l'apparition de fonctions transcendentales...

Les propriétés géométriques mises en évidence par l'utilisation des quaternions permettent, en particulier dans le cas bidimensionnel, l'utilisation d'algorithmes très performants, tels ceux développés par J.F.Cany [5-Can84] concernant le calcul d'interférence entre objets.

INPG département des études doctorales

AUTORISATION de SOUTENANCE

VU les dispositions de l'article 15 Titre III de l'arrêté du 5 juillet 1984 relatif aux études doctorales

VU les rapports de présentation de Messieurs

. Ph. COIFFET , Directeur de Recherche CNRS

. A. LIEGEOIS , Professeur

Monsieur PASQUIER Michel

est autorisé(e) à présenter une thèse en soutenance en vue de l'obtention du diplôme de DOCTEUR de L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE, spécialité " Informatique "

Fait à Grenoble, le 13 janvier 1989

Pour le Président de l'I.N.P.-G.
et par délégation,
le Vice-Président
P. VENNEREAU

Résumé

Cette thèse traite du problème fondamental que constitue la planification de trajectoires de robots manipulateurs. Une première partie précise le contexte robotique de notre travail et présente le système général de programmation automatique développé au LIFIA. L'importance de la représentation des connaissances nécessaires aux raisonnements géométriques particuliers à la planification de déplacements est ensuite analysée. Les méthodes de modélisation et les concepts de représentation que nous avons mis en œuvre sont présentés. Une deuxième partie traite de la résolution du problème de planification de trajectoires pour une structure articulée. Une méthode de planification globale par construction de l'espace des configurations est détaillée, ainsi qu'une méthode de replanification locale par application de champs de potentiels et, enfin, une méthode hybride réalisant la synthèse de ces approches complémentaires, pour lesquelles sont décrits algorithmes, résultats d'expérimentation et futurs développements.

Mots-clés : Robotique, Programmation Automatique des Robots, Modélisation Géométrique, Représentation, Géométrie Algorithmique, Raisonnement Géométrique, Planification de Trajectoires, Evitement d'obstacles.