Raisonnement géométrique et méthodes de décision en robotique : application à la programmation automatique des robots

Christian Laugier

THESE

Présentée à

L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

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

par

Christian LAUGIER

◊◊◊◊

RAISONNEMENT GEOMETRIQUE ET METHODES DE DECISION EN ROBOTIQUE. APPLICATION A LA PROGRAMMATION AUTOMATIQUE DES ROBOTS

◊◊◊◊

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

G. VEILLON	Président
O.D. FAUGERAS	
G. GIRALT	Examinateurs
J.C. LATOMBE	
A. LIEGEOIS	

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Président : Daniel BLOCH

Année 1987

Vice - Présidents : René CARRE
Jean-Marie PIERRARD

Professeurs des Universités

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

Professeur Université des Sciences Sociales (Grenoble II)

BOLLIET Louis

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

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

Chercheurs du C.N.R.S

Directeurs de recherche 1ère Classe

CAILLET Marcel
CARRE René
FRUCHART Robert
JORRAND Philippe
LANDAU Ioan
MARTIN

Directeurs de recherche 2ème Classe

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

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

E.N.S.E.E.G

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

E.N.S.E.R.G

BOREL Joseph
CHOVET Alain
DOLMAZON Jean-Marc
HERAULT Jeanny

E.N.S.I.E.G

DESCHIZEAUX Pierre
GLANGEAUD François
PERARD Jacques
REINISCH Raymond

E.N.S.H.G

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

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

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

E.F.P.G

CHARUEL Robert

C.E.N.G

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

Laboratoires extérieurs

C.N.E.T

DEMOULIN Eric
DEVINE
GERBER Roland
MERCKEL Gérard
PAULEAU Yves

ECOLE NATIONALE SUPERIEURE DES MINES DE SAINT-ETIENNE

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

PROFESSEURS DE 1ère CATEGORIE

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

PROFESSEURS DE 2ème CATEGORIE

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

DIRECTEUR DE RECHERCHE

LESBATS Pierre	Métallurgie

MAITRE DE RECHERCHE

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

Personalités habilitées à diriger des travaux de recherche

DRIVER Julian	Métallurgie
GUILHOT Bernard	Chimie
THOMAS Gérard	Chimie

Professeurs à l'UER de Sciences de Saint-Etienne

VERGNAUD Jean-Maurice	Chimie des Matériaux et Chimie Industrielle

RESUME

L'un des principaux obstacles au développement de la robotique, réside dans la difficulté qu'il y a à appréhender la complexité et la diversité du monde physique dans lequel le robot est appelé à évoluer. L'objet du présent travail est de montrer que la résolution de ce problème passe par le développement de deux catégories de techniques informatiques: des techniques traditionnelles orientées vers la "programmation des robots", et des méthodes d'Intelligence Artificielle destinées à doter la machine de "capacités décisionnelles". La première partie de la thèse présente les outils informatiques nécessaires au développement des fonctions de programmation. Elle montre de quelle manière nous avons fait évoluer ces fonctions, en les couplant avec des outils dérivés de la CAO: simulation graphique, programmation graphique, programmation géométrique et apprentissage interactif. La deuxième partie traite des mécanismes de la programmation automatique. Elle développe les méthodes de modélisation et de raisonnement, sur lesquelles reposent les fonctions de planification: modélisation géométrique du robot et de son environnement, raisonnement spatial et morphologique, raisonnement sur les incertitudes. Le dernier chapitre de la thèse montre comment toutes les techniques présentées peuvent être combinées, en vue de constituer un système de programmation automatique de robots appelé SHARP.

ABSTRACT

The major obstacle to the development of robotics is the fact that the real world is too complex to process using purely algorithmic tools. The purpose of this work is to show how to solve this problem, by combining two types of computing techniques: traditionnal techniques aimed at *programming assembly robots,* and Artificial Intelligence techniques allowing to equip the machine with *decision making capabilities.* The first part of the thesis presents the computer tools which are necessary for the effective programming of assembly robots. It shows the way we have developped these tools using CAD based facilities: graphic simulation, graphic programming, geometric programming and interactive learning. The second part of the thesis deals with the automatic robot programming problem. It describes modeling and reasoning techniques we have developped for implementing planning functions: geometric modeling, spatial and morphological reasoning, and reasoning about uncertainties. The last chapter shows how it is possible to combine the above techniques, in order to obtain an automatic robot programming system named SHARP.

MOTS CLES

Robotique, Intelligence Artificielle, Modélisation géométrique, Raisonnement spatial, Raisonnement morphologique, Planification de mouvements, Programmation automatique des robots.

Remerciements

Je tiens à remercier:

- *Monsieur Gérard Veillon*, Professeur à l'Institut National Polytechnique de Grenoble, Directeur de l'Ecole Nationale Supérieure d'Informatique et de Mathématiques appliquées de Grenoble, qui me fait l'honneur de présider le jury de cette thèse, et qui a toujours fait preuve d'un grand intérêt pour les recherches menées au LIFIA.

- *Monsieur Olivier Faugeras*, Directeur de Recherche à l'INRIA, pour avoir accepté de juger ce travail.

- *Monsieur Georges Giralt*, Directeur de Recherche au CNRS, responsable du projet National "Automatique et Robotique Avancées", pour l'énergie qu'il dépense quotidiennement pour soutenir et développer la recherche Française en Robotique, pour son amitié, et pour tout le temps qu'il m'a consacré en dépit des charges multiples qu'il assume.

- *Monsieur Jean-Claude Latombe*, Professeur à l'Institut National Polytechnique de Grenoble et à l'Université de Stanford, qui a su créer à Grenoble un environnement scientifique propice au développement des recherches associant la Robotique et l'Intelligence Artificielle. Je tiens à lui exprimer ma profonde gratitude pour m'avoir donné sa confiance et son amitié, et pour m'avoir donné les moyens de mener à bien les travaux qui me tenaient à cœur. Cette confiance m'a valu, au demeurant, de vivre une expérience difficile mais combien enrichissante: succéder au *"père spirituel"* de l'équipe d'Intelligence Artificielle et de Robotique du LIFIA.

- *Monsieur Alain Liégeois*, Professeur à l'Université des Sciences et Techniques du Languedoc, pour avoir accepté de porter un œil critique sur ce mémoire.
- L'INRIA, le CNRS, l'ENSIMAG et l'Agence De l'Informatique, qui ont rendu possible le financement de ces travaux.

Je ne ferais pas preuve d'originalité en soulignant une fois de plus, que la rédaction d'une thèse est à la fois un subtil supplice, et une formidable source de satisfaction (auto-satisfaction ?) intellectuelle. Or, il est beaucoup plus facile de partager le côté désagréable de la chose avec son entourage, que de communiquer une part des satisfactions qu'elle procure. Je souhaite sincèrement que tous ceux et celles qui ont pris part à cette épreuve, trouvent au delà des remerciements traditionnels, la véritable part qui leur est due:

- Jacqueline, ma femme, pour son soutien moral constant et son aide à la réalisation de ce document.
- Les chercheurs de l'équipe de Robotique du LIFIA, pour l'enthousiasme dont ils font preuve jour après jour, et pour leurs participations actives au développement du projet SHARP. Sans eux, ce projet n'aurait pas pu aboutir. Il est donc tout à fait naturel que la plus grande part de mes remerciements leur soit adressée:
  - Bruno Dufay pour ses travaux sur l'apprentissage en robotique.
  - Jocelyne Troccaz pour sa précieuse collaboration, sa compétence, son amitié, et ses travaux de qualité sur la saisie automatique et sur la modélisation géométrique.
  - Florence Germain et Monsieur Michel Pasquier, pour leurs travaux sur la planification de trajectoires sûres.
  - Pascal Theveneau pour ses travaux sur la synthèse automatique de mouvements fins de montage.
  - Pierre Puget pour ses travaux sur la vérification/correction de programmes soumis à des contraintes d'incertitude.
  - Christian Gandon pour ses travaux sur la compliance.
  - Michel Pasquier et Pascal Theveneau, "hackeurs" de l'équipe, pour avoir supporté la majeure partie des développements de base de SHARP.

Je voudrais aussi remercier mes anciens compagnons d'armes Emmanuel Mazer et Jean-François Miribel, pour les aventures palpitantes que nous avons vécues ensemble dans les méandres encore obscurs du monde des robots, et pour toutes les discussions constructives que nous avons eues sur le thème traité dans cette thèse.

Que soient également remerciés Georges Giralt, Jean-Claude Latombe, Alain Liegeois, et Emmanuel Mazer pour les remarques et suggestions concernant ce travail, ainsi que Pierre Puget, Pascal Theveneau et Jocelyne Troccaz pour l'aide précieuse qu'ils m'ont apporté dans la réalisation de ce document.

Enfin, je tiens à exprimer ma reconnaissance à ceux et celles qui ont contribué directement ou indirectement à la préparation de ce mémoire: ma femme Jacqueline pour mille et une raisons, mon fils Yann pour avoir participé à la réalisation de quelques figures, mon fils Sébastien pour avoir accepté de ne pas jouer sur le Macintosh afin que je puisse travailler sur ma thèse, et Françoise Molines pour avoir trouvé du plaisir à relire deux fois le manuscript complet.

Sommaire

1 Introduction **15**
1.1 Contexte du travail présenté: 15
1.1.1 Les machines automatiques: 15
1.1.2 Les logiciels "intelligents": 16
1.1.3 Les robots de manipulation: 18
1.2 La thèse soutenue: 19
1.2.1 Le thème: 19
1.2.2 La thèse: 19
1.2.3 L'approche: 20
1.2.4 La démarche: 21
1.3 Le rapport: 22
1.3.1 Première partie: 22
1.3.2 Deuxième partie: 23
1.4 Principales contributions: 24

I La programmation des robots: concepts, langages et outils informatiques **27**

2 Le concept de programmation des robots et les systèmes informatiques **29**
2.1 Concepts de base: 29
2.2 Les systèmes traditionnels: 31
2.2.1 Le formalisme de base: 31
2.2.2 La modélisation de l'univers du robot: 32

2.2.3 La spécification des déplacements libres: 34
2.2.4 La spécification des conditions sensorielles liées aux mouvements: 35
2.2.5 Le contrôle des processus parallèles: 36
2.3 Situation actuelle et évolution: 37
2.3.1 Les systèmes traditionnels: 37
2.3.2 La programmation automatique: 38

3 La connexion CAO-Robot: Programmation graphique 41
3.1 Principes de base et motivations: 41
3.2 Bref état de l'art en simulation graphique de robots: 43
3.3 Principe de la connexion CAO-robot: 45
3.3.1 Principe de base: 45
3.3.2 Fonctionnement de l'interface CAO-robot: 45
3.4 Modélisation géométrique de l'univers du robot: 47
3.4.1 Modèles nécessaires: 47
3.4.2 Modèle géométrique et relationnel: 48
3.4.3 Modèle cinématique du robot: 50
3.5 Simulation du robot: 51
3.5.1 Contrôle de la cinématique: 51
3.5.2 Gestion du modèle: 52
3.5.3 Interface graphique: 53
3.5.4 Emulation du temps: 54
3.6 Simulation de l'univers physique: 56
3.6.1 Les événements physiques: 56
3.6.2 La perception sensorielle: 57
3.7 La programmation graphique: 57
3.7.1 Principe de base: 57
3.7.2 Interprétation des relations géométriques: 58
3.7.3 Modèles nécessaires: 60
3.8 Implantation et expérimentations: 60
3.8.1 Architecture logicielle: 60
3.8.2 Le système ISR: 61
3.8.3 Le système PGR: 61
3.8.4 Expérimentations: 62

4 La connexion CAO-Robot: Programmation géométrique et Apprentissage interactif 63
4.1 Problèmes posés par les incertitudes et traitements associés: 63
4.1.1 Le problème: 63
4.1.2 Les solutions traditionnelles: 64
4.1.3 L'approche par apprentissage interactif: 65
4.2 La programmation géométrique: 66

4.2.1 Principes de base: 66
4.2.2 Spécification des mouvements: 67
4.2.3 Expression des situations géométriques: 68
4.2.4 Expression des contraintes d'exécution: 69
4.2.5 Expression des conditions de garde: 70
4.2.6 Instructions de manipulation utilisées: 71
4.2.7 Structure de contrôle: 72
4.3 L'apprentissage interactif: 73
4.3.1 Principe de base et motivations: 73
4.3.2 Notion de graphe d'exécution: 75
4.3.3 La phase d'expérimentation: 76
4.3.4 La phase d'induction: 79
4.4 Implantation et expérimentations: 83
4.4.1 Le langage géométrique: 83
4.4.2 le module d'apprentissage: 85

II La programmation automatique des robots: problèmes, modèles de raisonnement et architecture de système **87**

5 La programmation automatique: problèmes et approches **89**
5.1 Présentation du problème: 89
5.1.1 Approche traditionnelle et programmation automatique: . . 89
5.1.2 Analyse d'un exemple de planification: 90
5.2 Analyse du processus de planification: 94
5.2.1 Planification de la tâche: 94
5.2.2 Planification des actions de manipulation: 96
5.2.3 Traitement des interdépendances: 98
5.3 Travaux portant sur les systèmes de niveau tâche: 99
5.3.1 Description de la tâche: 99
5.3.2 Planification de niveau tâche: 100
5.3.3 Traitement des interdépendances: 101
5.3.4 Projets de systèmes intégrés: 102
5.4 Travaux sur la planification des actions du robot: 103
5.4.1 Planification des opérations de saisie: 103
5.4.2 Planification des trajectoires de transfert: 104
5.4.3 Planification des opérations de montage: 113

6 Modélisation de l'univers du robot **117**
6.1 Modèles géométriques de base: 117
6.2 Modélisation des incertitudes de position: 120
6.2.1 Problèmes posés et solutions possibles: 120

6.2.2 Représentation des erreurs de position: 122
6.2.3 Représentation des incertitudes de position: 122
6.2.4 Structure de l'univers du robot: 124
6.2.5 Opérateurs associés au modèle: 125
6.3 Espace des configurations: 128
6.3.1 Définitions et notations: 128
6.3.2 Caractérisation analytique: 129
6.4 Modélisation des mouvements du robot: 132
6.4.1 Principaux éléments du modèle: 132
6.4.2 Les frottements: 133
6.4.3 Les erreurs de commande et de perception: 134
6.4.4 Modèle de commande: 136
6.4.5 Les conditions d'arrêt: 137
6.5 Espace de planification: 139
6.5.1 Notion d'espace de planification: 139
6.5.2 Construction de l'espace de planification: 140
6.5.3 Propriétés de la représentation: 142

7 Le raisonnement spatial **145**
7.1 Présentation et principes de base: 145
7.1.1 Problème abordé: 145
7.1.2 Raisonnements mis en jeu: 147
7.1.3 Outils informatiques nécessaires: 147
7.1.4 Notations utilisées: 148
7.2 Détermination des débattements valides: 149
7.2.1 Notion de débattement valide: 149
7.2.2 Calculs d'interférences: 150
7.2.3 Calculs de collisions: 151
7.2.4 Calcul des débattements valides: 156
7.3 Opérateurs de grossissement: 157
7.3.1 Calcul des ensembles de type $CO_A^{xyz}(B)$: 157
7.3.2 Traitement des symétries de révolution: 158
7.4 Opérateurs de réduction des d.d.l: 160
7.4.1 principe du calcul: 160
7.4.2 Méthode de calcul: 160
7.5 Raisonnement sur une structure articulée: 163
7.5.1 Principe de construction des C-obstacles: 163
7.5.2 Calcul des contraintes: 165
7.5.3 Propagation des contraintes: 167
7.5.4 Complexité algorithmique: 170
7.6 Construction de l'espace libre: 170
7.7 Recherche d'une trajectoire dans l'espace libre: 172

8 Le raisonnement morphologique **175**
8.1 Présentation et principes de base: 175
8.1.1 Problèmes abordés: 175
8.1.2 Raisonnement global mis en jeu: 176
8.1.3 Outils informatiques nécessaires: 177
8.1.4 Plan du chapitre: 178
8.2 Le concept de contact: 178
8.2.1 Notion de voisinage: 178
8.2.2 Spécification symbolique des contacts: 179
8.2.3 Raisonnement sur les contacts: 182
8.3 Les contacts potentiels: 184
8.3.1 Convexité locale et contacts potentiels: 184
8.3.2 Caractérisation analytique des propriétés: 186
8.4 Les contacts contraints: 189
8.4.1 Le concept de contact contraint: 189
8.4.2 Les contraintes topologiques et géométriques: 190
8.4.3 Les contraintes mécaniques: 192
8.5 Contraintes de mouvements associées aux contacts: 193
8.5.1 Notion de mouvement potentiel: 193
8.5.2 Représentation analytique des mouvements potentiels: . . . 195
8.5.3 Propriétés de la représentation: 199
8.5.4 Traitement des rotations: 201
8.5.5 Exploitation de la représentation précédente: 202
8.6 Construction de l'espace des solutions: 206
8.6.1 Les ensembles de prises potentielles: 206
8.6.2 Le graphe d'états associé aux mouvements fins: 209

9 Implantation et intégration: le système SHARP **215**
9.1 Architecture du système: 215
9.1.1 Structure fonctionnelle du système: 215
9.1.2 Description de la tâche: 218
9.1.3 Composants informatiques du système: 221
9.1.4 Structure de contrôle: 222
9.2 Planification des mouvements de transfert: 226
9.2.1 Approche du problème: 226
9.2.2 Algorithme général: 227
9.2.3 Fonctions de grossissement: 228
9.2.4 Fonction de discrétisation: 229
9.2.5 Traitement des rotations du poignet: 231
9.2.6 Implantation et expérimentations: 231
9.3 Planification des opérations de saisie: 234
9.3.1 Approche du problème: 234

9.3.2 Algorithme général: 235
9.3.3 Construction des prises potentielles: 235
9.3.4 Choix d'une direction d'approche: 236
9.3.5 Détermination des configurations valides: 236
9.3.6 Détermination des configurations sûres: 237
9.3.7 Implantation et expérimentations: 238
9.4 Planification des mouvements fins: 240
9.4.1 Approche du problème: 240
9.4.2 Algorithme général: 241
9.4.3 Choix des directions d'étude: 242
9.4.4 Création des nœuds et des arcs de $G(A/B)$: 242
9.4.5 Recherche d'une solution: 243
9.4.6 Synthèse du programme de mouvements fins: 245
9.4.7 Implantation et expérimentations: 246
9.5 Exemple de tâche traitée par le système: 248
9.5.1 Description graphique de la tâche: 249
9.5.2 Forme du programme de manipulation engendré: 250
9.5.3 Quelques étapes du montage: 251
9.6 Vérification/correction de plans: 253
9.6.1 Présentation du problème: 253
9.6.2 Modélisation des actions du robot: 254
9.6.3 Principe de vérification/correction: 256

Liste des Figures

3.1 Schéma synoptique de la connexion CAO-Robot du système LM. . 47
3.2 Exemple de structure relationnelle incluse dans le modèle. 49
3.3 Principe d'évaluation d'une structure cinématique. 51
3.4 Structure de type parallélogramme. 52
3.5 Représentation des paramètres d'observation. 53
3.6 Exemples de sorties graphiques produites par le système. 55
3.7 Interprétation des relations géométriques dans le système PGR. . . 59

4.1 Exemple de classes de montages. 66
4.2 Schéma synoptique du module d'apprentissage interactif. 74
4.3 Notion de graphe d'exécution. 77
4.4 Exemple de combinaison de deux traces d'exécution. 81
4.5 Règles de fusion utilisées pour combiner les traces d'exécution. . . 82
4.6 Exemple de montage d'un contacteur. 84

5.1 Exemple d'un "assemblage" simple. 91
5.2 schéma synoptique du processus de planification en programmation automatique des robots. 95
5.3 Graphe de visibilité utilisé par le robot SHAKEY. 109
5.4 Représentation de l'espace libre à l'aide de cônes généralisés. . . . 110
5.5 Illustration du concept de pré-image. 116

6.1 Exemple de hiérarchie de volumes englobants. 118
6.2 Exemple de structure topologique d'un objet. 119
6.3 Représentation de la topologie locale au voisinage d'une arête. . . . 120
6.4 Représentation des incertitudes de position associées à un contact. 124

6.5 Exemple de modification des arcs du modèle après réalisation d'un contact. 126
6.6 Configuration d'un objet bidimensionnel. 130
6.7 Cône de frottement associé à un contact ponctuel. 133
6.8 Calcul des forces en présence de frottements. 134
6.9 Exemple de situation ambigüe. 139
6.10 Notion de classe "d'états équivalents" dans l'espace libre. 141
6.11 Représentation de l'espace de planification pour un mobile ponctuel. 142
6.12 Types de contacts dans un espace bidimensionnel. 143
6.13 illustration de la propriété d'imcomlplétude. 144

7.1 Exemple de contacts potentiels pour un mobile en rotation. 152
7.2 Propriétés topologiques des contacts engendrés par un sommet. . . 155
7.3 Propriétés topologiques des contacts engendrés par deux arêtes. . . 155
7.4 Construction des configurations d'un objet en rotation. 157
7.5 Construction d'un grossissement de type $CO_A^{\pi v}(B)$. 158
7.6 Grossissement d'un obstacle polygonal par le rayon d'un mobile circulaire. 159
7.7 Calcul des orientations valides d'un mobile. 162
7.8 Exemple de construction d'une approximation de $CO_A(B)$. 165
7.9 Exemple de tranche d'orientation constituée de plusieurs domaines. 166
7.10 Etude du comportement d'un élément de la structure articulée. . . 167
7.11 Propagation des contraintes de position sur une structure articulée. 168
7.12 Exemple de représentation de l'espace libre associé à un robot à deux d.d.l. 172

8.1 Exemples de contacts. 181
8.2 Contraintes de mouvements imposées par les différents types de contacts. 183
8.3 Exemple de contact par adjacence. 184
8.4 Illustration des propriétés de convexité locale. 185
8.5 Propriété de convexité locale pour une arête rectiligne. 187
8.6 Propriété d'accessibilité locale pour un sommet. 188
8.7 Propriété de convexité locale pour une face plane polygonale. . . . 189
8.8 Illustration des propriétés d'équilibre et de stabilité. 193
8.9 Représentation sphérique des contraintes de mouvement. 196
8.10 Caractérisation des domaines de mouvements potentiels. 197
8.11 Représentation analytique des mouvements potentiels associés à un couple de contacts plans. 198
8.12 Mouvements potentiels associés à des contacts surfaciques mettant en jeu des surfaces de révolution. 199
8.13 Mouvements potentiels associés à des contacts non planaires. . . . 200

8.14 Exemples de rotations associées à un contact plan. 203
8.15 Régions susceptibles d'être atteintes par le robot. 205
8.16 Exemple de prise potentielle. 207
8.17 Exemples de configurations de P dans $\Pi(A/P)$. 208
8.18 Graphe représentant une stratégie de mouvements fins. 211
8.19 Stratégies incluses dans un graphe d'états. 213

9.1 Structure fonctionnelle du système SHARP. 217
9.2 Fonctionnement du module de planification. 223
9.3 Fonctionnement du module de vérification/correction de plans. . . 225
9.4 Exemple de trajectoire obtenue avec un découpage non uniforme de l'espace articulaire. 230
9.5 Exemple de mouvement planifié par le système. 233
9.6 Principe de grossissement des obstacles pour la recherche de prises sûres. 238
9.7 Exemple de prises engendrées par le système. 239
9.8 Exemple de stratégie de mouvements fins engendrée par le système. 247
9.9 Modifications apportées au modèle par une action de déplacement. 256
9.10 Modification des incertitudes de position à la suite d'une opération de saisie. 257

Chapitre 1

Introduction

1.1 Contexte du travail présenté:

1.1.1 Les machines automatiques:

De tout temps l'être humain a essayé de créer des machines capables de le remplacer, voire de le surpasser, dans des tâches variées telles que la conception et la production d'objets manufacturés, la conduite d'appareils divers, la compréhension de langues naturelles ou la création artistique. Cette activité l'a conduit à acquérir très tôt un savoir-faire dans des disciplines technologiques telles que la mécanique ou l'automatique. Il est par exemple remarquable de constater à quel point certains artisans des siècles derniers étaient passés maîtres dans l'art de confectionner des mécanismes sophistiqués capables de doter un automate d'un pouvoir de mimétisme gestuel lui assurant un semblant de comportement humain (parfois qualifié d'intelligent). De même, la transformation au cours des siècles du milieu industriel est significative de l'évolution technologique qui a permis de passer des *"machines ingénieuses"* (mais figées) qui marquent le début de la première révolution industrielle, aux *"machines programmables"* d'aujourd'hui allant de la machine outil à commande numérique aux robots informatisés dotés potentiellement d'une grande capacité d'adaptation à des tâches variées telles que la peinture, la soudure, la manutention ou l'assemblage mécanique. Quelques auteurs soulignent à ce propos que la seconde révolution industrielle à laquelle notre génération

est confrontée modifiera profondément la conception que nous avions de la machine et de son rôle dans notre vie quotidienne.

Sur le plan historique, la mécanique et l'électronique ont longtemps été les principaux supports de développement de toutes les machines automatiques. Ce n'est que plus tard, après l'apparition des premiers robots industriels, que l'informatique a été appelée à jouer un rôle significatif en tant qu'outil de pilotage et de programmation [Latombe 82]. Par la suite, ce rôle n'a cessé de croître sous la double pression de l'évolution socio-économique et de la maturation des techniques informatiques. Il est clair aujourd'hui que cette situation marque l'amorce d'une nouvelle technologie fondée sur l'intégration de systèmes mécaniques et électroniques avec des ensembles informatiques de plus en plus puissants.

Bien que le concept de *"comportement intelligent"* au sens strict du terme soit totalement étranger aux générations de machines précédentes, il a très souvent été assimilé dans le langage commun au fait que les mécanismes créés, étaient capables de reproduire automatiquement des séquences d'opérations parfois complexes, tout en ayant une capacité d'adaptation vis à vis de certaines variations de l'environnement. Ce pouvoir *d'auto-adaptation*, lorsqu'il existe, est très limité, et il ne permet de traiter que des modifications de l'espace opérationnel prévues et évaluées quantitativement par les concepteurs, lors de la phase de conditionnement ou de programmation de la machine. La fonction de perception est dans ce cas peu sophistiquée (elle est la plupart du temps réduite à une simple évaluation numérique de quantités physiques), et le pouvoir de décision est quasiment inexistant.

1.1.2 Les logiciels "intelligents":

Il a fallu attendre un dévelopement avancé de l'informatique pour aborder des domaines de recherche jusqu'alors peu ou pas explorés: l'Intelligence Artificielle et son impact sur les mécanismes du *comportement intelligent*. Les premières recherches en la matière il y a quelque vingt cinq ans, ont conduit à construire les bases d'une nouvelle discipline très controversée par la communauté scientifique de l'époque. Elles visaient essentiellement à reproduire certaines activités cognitives de l'être humain telles que la compréhension des langues naturelles , la résolution de problèmes ou la démonstration automatique. Ces travaux ont donné lieu au développement d'outils formels issus des mathématiques et de la logique [Winston 77]. Il sont également à l'origine de la création d'outils et de techniques informatiques propres à l'intelligence artificielle [Winston 77] [Nilsson 80]: les langages de type LISP ou PROLOG, les systèmes de production, les mécanismes d'inférence guidés par filtrage ("Pattern Directed Inference Systems")... Sur le plan pratique, ces techniques ont permis d'implanter au cours des quinze dernières années, plusieurs dizaines de Systèmes experts (systèmes spécialisés par leurs jeux

de règles d'inférence) dans des domaines d'application très divers tels que la chimie, la biologie, la médecine, les mathématiques, la géologie, la conception assistée par ordinateur, la traduction automatique, la psychologie, les jeux ou la gestion de banques de données [Farreny 80]. Certains de ces systèmes connaissent de nos jours un essor industriel.

Cette première approche de l'IA était tout à fait conforme à la conception communément admise de ce que l'on considérait être les fondements de l'intelligence, par opposition aux fonctions cérébrales dites de plus bas niveau telles que la perception ou l'instinct (Encyclopaedia Universalis [Richard 68]). Elle a de ce fait conduit les chercheurs à développer des techniques logicielles assurant un *comportement intelligent de type intellectuel*, à des programmes comportant des capacités de raisonnement et de déduction logique. Dans le contexte de la robotique, de tels systèmes déconnectés du monde réel ont parfois été qualifiés de "robots logiciels", dans le sens où l'objectif visé était de planifier automatiquement les opérations d'un robot fictif, à partir d'une spécification simple du but à atteindre. Le problème traité était alors celui de la génération de plans d'actions. Il a surtout servi de prétexte pour étudier des aspects fondamentaux de la résolution de problèmes, lorsque les raisonnements conduits mettent en jeu des notions d'actions et d'effets associés. Quelques logiciels de ce type ont été développés dans les années 70 (cf. [Farreny 80]). Parmi ces systèmes, STRIPS [Fikes, Nilsson 71] est sans aucun doute le plus connu et l'un des plus anciens logiciels de résolution de problèmes conçus dans le but de contrôler un robot.

Une autre raison à l'origine de cette approche de type "logiciel intelligent", tient au fait que le monde réel constitue un domaine d'expérimentation très complexe, nécessitant des moyens perceptifs importants. C'est pourquoi le principal terrain d'essai de l'IA a longtemps été donné par les jeux qui permettent de raisonner sur un univers bien défini, peu complexe et aisément formalisable à l'aide de la logique des prédicats [Demazeau 85]. Les "robots logiciels" possèdent également cette caractéristique dans le sens où ils opèrent dans des univers stylisés de type "monde des blocs", dans lesquels les considérations de nature physique sont ignorées. De rares tentatives ont toutefois été menées afin de prendre en compte quelques propriétés simples du monde réel lors de la phase de planification. Le système BUILD [Fahlmann 74] raisonne par exemple sur un modèle de l'environnement incluant le concept de stabilité pour engendrer des plans d'actions permettant d'empiler des blocs aux formes, tailles et poids variés. Il se distingue des autres générateurs de plans du monde des blocs par une analyse combinée des effets logiques et des conséquences physiques des actions dont il dispose. Mais ce premier pas vers la réalité physique de l'univers du robot restait encore très limité. Nous verrons dans la suite, que cette réalité ne peut pas se passer d'une modélisation des actions de "bas niveau", telles que le contrôle des déplacements ou la perception des efforts

exercés.

1.1.3 Les robots de manipulation:

A l'opposé des logiciels précédents, les recherches menées dans le domaine de la robotique de manipulation ont presque toutes portées sur la résolution des problèmes de bas niveau. Elles ont conduit au développement de systèmes de programmation plus ou moins évolués. Certains sont basés sur un mode de programmation dit "par l'exemple". Ils s'appuient pour cela sur des mécanismes réels (boitier de télécommande) ou simulés (dispositifs d'interaction graphique), permettant de déplacer manuellement le robot et de lui faire exécuter ainsi la tâche à accomplir. D'autres systèmes sont construits autour de langages symboliques comportant des constructions algorithmiques classiques combinées avec des constructions spécifiques de la robotique(cf. chapitre 2). Le principal objectif visé par tous ces systèmes est de fournir un outil permettant de décrire des *tâches répétitives, exécutées dans un environnement connu à priori, mais soumis à des variations mineures.* Le temps nécessaire à la construction et à la mise au point des programmes de commande de robots n'est pas dans ce cas un critère décisif. C'est pourquoi une description explicite des actions du robot (avec toutes les difficultés que cela comporte) peut s'avérer suffisante dans un grand nombre de cas.

Cependant, les coûts de programmation, les limitations dues à une mauvaise intégration des données sensorielles et la fiabilité médiocre de nombreux programmes, ont conduit les chercheurs à envisager une nouvelle génération de systèmes de programmation de robots. Ces systèmes s'appuient sur une structure informatique puissante, alliant un matériel à haute capacité de calcul avec des techniques d'Intelligence Artificielle et de Modélisation Géométrique. Le but recherché est alors d'automatiser le processus de programmation en décrivant au système uniquement l'objectif à atteindre, sans spécifier la manière d'y parvenir. Dans le cas du montage mécanique, une telle spécification de la tâche comporte par exemple une description géométrique des pièces et des relations de montage. On rejoint içi les premières préoccupations des chercheurs en Intelligence Artificielle. Mais l'approche suivie est plus réaliste, dans le sens où elle intègre dès le départ les modèles physiques qui faisaient défaut aux précédents systèmes. Les projets AUTOPASS [Liebermann, Wesley 77], LAMA [Lozano-Perez 76], TWAIN [Lozano-Perez, Brooks 85], et SHARP [Laugier, Troccaz 85] vont dans ce sens (cf. chapitre 5). Il conduisent à intégrer et à faire collaborer des modules de perception et de planification d'actions, spécialisés dans la préhension, la manipulation et le montage d'objets manufacturés. La principale difficulté réside dans la maîtrise de la complexité introduite par le monde réel (combinatoire des problèmes géométriques, prise en compte des incertitudes et des aléas de l'univers physique, interaction entre action et perception...), et dans la nature fortement dépendante

des sous-tâches planifiées.

1.2 La thèse soutenue:

1.2.1 Le thème:

Le travail présenté se situe dans le contexte de la réalisation d'un *"robot intelligent"*, c'est à dire d'un robot doué d'autonomie, d'intelligence, de dextérité et de capacités évoluées de perception et de communication. Cet objectif à long terme constitue le point focal des recherches menées au laboratoire LIFIA.

Les sujets traités dans cette thèse contribuent à résoudre le problème de *"l'autonomie de l'action"*. Il s'agit alors de doter le robot des fonctions logicielles, lui permettant d'agir intelligemment sur son environnement, en fonction du contexte d'intervention et du but à atteindre. Cette faculté de décision fait appel à la fois à des techniques qui relèvent de l'Intelligence Artificielle, et à un savoir-faire lié aux problèmes de bas niveau de la programmation des robots (génération de trajectoires, contrôle d'exécution, maîtrise des efforts ...).

Conscients de l'impossibilité qu'il y a à traiter le problème dans sa généralité (tout au moins dans l'état actuel de nos connaissances), nous avons décidé de l'aborder dans le contexte de la robotique d'assemblage. Ce problème peut alors être formulé comme suit: étant donné une description symbolique de la tâche à réaliser et un modèle de l'univers du robot (actions possibles, géométrie des objets, contraintes physiques ...), trouver *un plan d'actions* permettant de faire exécuter la tâche par le robot. Afin de s'affranchir des incertitudes et des aléas de l'univers physique, ce plan doit combiner des mouvements du robot et des opérations sensorielles. Nous parlerons alors de *programmation automatique des robots.*

1.2.2 La thèse:

L'objet de cette thèse est de montrer quels sont les problèmes informatiques posés par la réalisation d'un système de programmation automatique de robots, d'analyser les approches possibles, d'en formaliser les bases, et de proposer des solutions qui reposent sur des logiciels que nous avons développés.

La thèse soutenue relativement à ces sujets comporte deux volets, qui peuvent être énoncés comme suit:

1. Les incertitudes du monde physique ne peuvent être traitées de manière automatique ou semi-automatique, qu'en présence d'un *langage géométrique* exprimant les opérations du robot en termes d'objets. Ce moyen d'expression

est nécessaire pour pouvoir spécifier clairement l'objectif visé par chaque action du robot, ainsi que les relations qui existent entre le mouvement exécuté et les opérations sensorielles.

2. Le processus de planification des actions d'un robot repose sur un *raisonnement géométrique opérant à deux niveaux d'abstractions.* Ces deux niveaux permettent respectivement de proposer des solutions potentielles exhibées au moyen d'un processus de filtrage géométrique (on parle alors de "raisonnement morphologique"), et de construire des séquence de mouvements répondant au problème posé (on parle alors de "raisonnement spatial"). La raison d'être de ces deux formes de raisonnement, tient au fait que l'application d'un traitement géométrique "brutal" conduirait immédiatement à une explosion combinatoire. Elle se justifie essentiellement par la taille importante de l'espace des solutions possibles, ce qui se traduit en pratique par l'existence d'heuristiques aptes à restreindre rapidement l'espace de recherche.

1.2.3 L'approche:

L'un des principaux obstacles au développement des fonctions de programmation automatique, réside dans la difficulté qu'il y a à appréhender la complexité et la diversité du monde physique dans lequel le robot est appelé à évoluer. La résolution des problèmes induits par cette réalité physique, met en jeu deux catégories de techniques informatiques: des techniques traditionnelles orientées vers la programmation et le contrôle des robots, et des méthodes d'Intelligence Artificielle destinées à doter la machine des "capacités décisionnelles" requises.

Le premier volet de notre approche à donc porté sur la conception des outils informatiques nécessaires au développement des fonctions de programmation et de commande. Ces outils ont été conçus de manière à tirer partie des modèles géométriques offerts par la connexion CAO-robot. Ils ont surtout été développés en vue de réaliser un support symbolique pour l'expression des plans d'actions produits par le système de programmation automatique. Ils reposent pour cela sur un langage géométrique, permettant de décrire les actions du robot en termes "d'objets". Les formes de base de ce langage sont constituées de relations géométriques, de contacts et de contraintes d'exécution exprimées dans un formalisme géométrique. Cette approche a été essentiellement motivée par l'impossibilité qu'il y a à interpréter correctement les situations atteintes par le robot, lorsque les opérations réalisées sont décrites à l'aide d'un langage de manipulation traditionnel (les modèles utilisés étant trop pauvres).

Le deuxième volet de notre approche porte sur le développement des fonctions de planification. Ces fonctions reposent d'une part sur un procédé de modélisation

qui combine des représentations géométriques et physiques, et d'autre part sur un processus de raisonnement géométrique opérant en deux étapes:

- Dans un premier temps, le système recherche des *solutions potentielles* sur la base d'une analyse des propriétés locales des contacts concernés. Par exemple, l'étude des contacts potentiels entre l'outil de préhension et l'objet à saisir permettra d'exhiber un ensemble de *"prises potentielles"*, qui présentent localement toutes les caractéristiques requises. De même, l'analyse des mouvements potentiellement exécutables par un objet en situation de contact, permettra de proposer des *"directions de déplacement"* compatibles avec les contraintes topologiques locales. Ce type de traitement est appliqué pour sélectionner des directions d'approche pour la saisie. Il est surtout utilisé pour choisir des mouvements pertinents lors de la planification des opérations de montage. Le mode de raisonnement appliqué au cours de cette phase est appelé **"raisonnement morphologique"**, car il conduit à raisonner sur les indices morphologiques locaux des objets.

- La deuxième phase du traitement a pour objet d'évaluer *l'accessibilité globale* des solutions proposées. Elle opère sur la base d'une analyse des contraintes globales de la tâche, afin de rejeter ou de valider les choix réalisés antérieurement. Les calculs exécutés permettent alors d'affiner et de compléter les solutions retenues. Par exemple, la prise en compte des obstacles aux déplacements des mors conduira à restreindre l'ensemble des positions possibles de la pince. De même, le calcul du débattement associé à une direction de déplacement permettra de spécifier les paramètres du mouvement étudié. Le mode de raisonnement appliqué au cours de cette phase est appelé **"raisonnement spatial"**, car il conduit à raisonner sur les configurations spatiales du mobile.

L'approche proposée consiste donc à *dissocier l'analyse des positions et des mouvements potentiellement réalisables, de l'étude de ceux réellement exécutables.* Elle a été motivée par le fait qu'elle permet de mieux maîtriser la complexité algorithmique, en orientant progressivement les choix au travers d'une analyse de plus en plus complète des contraintes imposées par la géométrie des objets. Cette analyse est alors réalisée au moyen de *filtres géométriques simples*, appliqués par ordre de complexité croissante.

1.2.4 La démarche:

Le travail présenté dans cette thèse repose sur un travail d'équipe. Il porte sur une période de huit années, et il aborde un vaste domaine de recherche gravitant autour du concept "d'autonomie de décision en robotique". Ce domaine est situé au carrefour de disciplines voisines telles que la commande et la programmation des robots, la CAO (Conception Assitée par Ordinateur), la compréhension de

scènes, et l'Intelligence Artificielle.

Le principal objectif visé est alors de "rassembler" des techniques diverses issues de ces disciplines, d'en formaliser les interactions, et de les adapter en vue de constituer un ensemble homogène, apte à doter le robot des capacités décisionnelles qui lui font défaut. C'est pourquoi, chaque sujet traité dans ce but donnera lieu à une analyse assez complète des techniques existantes, à une formalisation des problèmes abordés, puis à une description des méthodes que nous avons développées. Nous nous appuierons pour cela sur des réalisations concrètes de l'équipe de robotique du LIFIA.

1.3 Le rapport:

1.3.1 Première partie:

La première partie de la thèse aborde le problème sous l'angle des systèmes informatiques traditionnels, c'est à dire des systèmes ne possédant aucune capacité décisionnelle.

Le chapitre 2 présente les concepts qui sont à la base de la programmation des robots. Il s'appuie pour cela sur les développements réalisés antérieurement par des chercheurs de l'équipe, en matière de langage de manipulation. Il montre également les améliorations qui ont été apportées, en vue de mieux intégrer la perception au niveau de la commande des déplacements. Ces nouvelles capacités sont essentielles, pour pouvoir exécuter les mouvements compliants engendrés par le système de planification des mouvements fins de montage.

L'objet des deux chapitres qui suivent, est de montrer comment les modèles géométriques apportés par la CAO transforment le concept de programmation de robots.

Le chapitre 3 décrit une première forme de connexion CAO-robot, permettant de mettre en oe uvre des robots sur des tâches de type "Prendre- Poser", c'est à dire des tâches insensibles aux effets des erreurs de commande et de perception. Il s'agit alors de *programmer graphiquement* le robot, en s'appuyant sur un simulateur graphique "complet". Ce simulateur comporte des fonctions permettant de simuler l'univers physique (aléas, collisions ...), et de représenter graphiquement les opérations réalisées par le robot. Il inclut également des outils graphiques pour décrire les mouvements à faire exécuter par le robot.

Le chapitre 4 aborde une deuxième forme de connexion CAO-robot, venant en complément de la précédente. L'objectif visé est alors de doter le système

de nouvelles capacités, lui permettant de traiter de véritables programmes d'assemblage. Ces programmes combinent des opérations sensorielles avec des petits mouvements du robot, afin de s'affranchir des incertitudes dues aux erreurs de commande, de perception et de modélisation. L'approche suivie consiste à combiner un langage géométrique avec des fonctions dites "d'apprentissage interactif". Le langage est conçu de manière à décrire les opérations du robot, en termes de relations simples portant sur les éléments du modèle géométrique. Une particularité importante de ce langage, est de pouvoir spécifier des mouvements qui mettent en jeu des contacts (et donc des forces de réaction). Le mécanisme d'apprentissage a été conçu comme une extension du langage. Il permet de programmer une tâche de montage, sans qu'il soit nécessaire d'évaluer exactement toutes les contraintes d'incertitudes liées à cette tâche. Il procède pour cela à une phase d'expérimentation en-ligne, suivie d'une phase d'induction permettant de faire la synthèse des informations recueillies au cours des différents essais.

1.3.2 Deuxième partie:

La deuxième partie de la thèse aborde le problème de la programmation des robots sous l'angle de l'Intelligence Artificielle. Il s'agit alors d'automatiser le processus de programmation, en dotant le robot des capacités d'analyse et de planification qui lui font défaut. Ces capacités viennent en complément des outils de base introduits dans la première partie.

Le chapitre 5 présente les différents problèmes posés par la programmation automatique des robots, en s'appuyant sur l'analyse d'un exemple simple. Il décrit ensuite, pour chaque sous-problème isolé, les différentes approches qui ont été développées au niveau international.

Le chapitre 6 aborde le problème de la modélisation de l'univers du robot. Il présente les différentes caractéristiques qu'il faut modéliser, afin de pouvoir planifier les actions du robot. Les modèles que nous décrivons combinent des représentations qui portent sur la géométrie des objets, sur leurs relations physiques et spatiales, et sur les incertitudes de position qui leurs sont associées. Ils comportent également une modélisation des structures cinématiques, et des données mécaniques qui interviennent dans l'exécution des mouvements du robot (forces, frottements et type de commande).

Le chapitre 7 traite du "raisonnement spatial". Cette forme de raisonnement géométrique a pour objet de planifier les mouvements du robot. Il repose sur des fonctions qui permettent de calculer des débattements valides, d'exécuter des calculs de grossissements d'obstacles, de construire une représentation approchée de l'espace libre, et de rechercher des solutions dans cet espace.

Le chapitre 8 aborde une deuxième forme de raisonnement géométrique, dont l'objet est de planifier les opérations qui mettent en jeu des contacts (opérations de saisie et mouvements fins de montage). Ce type de raisonnement est appelé "raisonnement morphologique". Il se place en complément du raisonnement spatial, dans le sens où il permet de réduire la complexité algorithmique inhérente à cette première forme de raisonnement. Le principe consiste à "filtrer" l'ensemble des solutions possibles (cet ensemble étant très grand en robotique), afin d'exhiber des solutions potentielles présentant de bonnes caractéristiques. Ces solutions sont alors établies sur la base d'une analyse des propriétés locales des contacts. Les méthodes mises en oe uvre sont donc orientées d'une part vers l'analyse des propriétés topologiques des contacts, et d'autre part vers la détermination des contraintes de mouvements imposées par ces contacts.

Le chapitre 9 fait la synthèse des méthodes présentées. Il montre comment ces méthodes peuvent être combinées, en vue de constituer un système de programmation automatique de robots appelé SHARP. Ce chapitre comporte une description complète des algorithmes que nous avons implantés. Il comporte également une proposition d'architecture, dont les bases reposent sur des développements partiels. Cette architecture met en évidence le rôle des contraintes d'incertitude dans la structure de contrôle.

1.4 Principales contributions:

Les principales contributions apportées par cette thèse dans le contexte de la connexion CAO-robot sont les suivantes:

1. Réalisation d'une véritable connexion entre un langage de programmation de robot et des bases de données CAO.

2. Intégration d'un mécanisme de simulation des capteurs et des événements de nature physique.

3. Développement d'un langage géométrique permettant de décrire les actions du robot en termes de relations géométriques et de contacts.

4. Développement d'un mécanisme d'expérimentation en-ligne, facilitant la détection et la correction des échecs dus aux incertitudes.

Les principales contributions apportées dans le contexte de la programmation automatique et du raisonnement géométrique, sont les suivantes:

1. Unification des approches liées à la planification des mouvements d'un robot. Nous avons formalisé pour cela les concepts de "raisonnement spatial" et de

"raisonnement morphologique", ainsi que le rôle joué par chacun dans le processus de programmation.

2. Développement des algorithmes de base nécessaires à la conduite des raisonnements précédents. Nous avons décrit le principe de ces algorithmes, ainsi que les termes de complexité qui leurs sont associés.

3. Développement d'une forme de raisonnement basée sur une représentation explicite des mouvements potentiels associés aux contacts. Cette technique a été utilisée pour planifier des séquences de mouvements fins de montage.

4. Proposition d'une architecture intégrée pour un système de programmation automatique de robots. Cette architecture repose sur un processus de filtrage réparti dans les modules de planification, un mécanisme élémentaire de backtrack, un procédé de vérification/correction de plan, un langage de description incluant des constructions pour spécifier les contraintes d'incertitudes, et un langage géométrique interprétable par le robot.

Partie I

La programmation des robots: concepts, langages et outils informatiques

Chapitre 2

Le concept de programmation des robots et les systèmes informatiques

2.1 Concepts de base:

Un programme de commande de robots représente un procédé informatique permettant de contrôler l'enchaînement d'un ensemble d'actions primitives de la machine, en vue de la réalisation d'une tâche de manipulation particulière. Toutes les actions, qu'elles soient de nature manipulatoire ou perceptive, sont transcrites de manière *quantitative au sein d'un algorithme* pouvant comporter des structures de contrôle élaborées incluant du parallélisme. l'obtention des éléments nécessaires à la construction d'un tel programme passe par la résolution de deux problèmes fondamentaux de la robotique [Latombe 82]

- *La spécification de positions "sûres":*
  Un robot manipulateur est avant tout un appareil capable de déplacer un objet d'une position à une autre. La fonction première d'un programme de manipulation est donc de définir une séquence de positions par lesquelles doit passer le robot lors de l'exécution d'une tâche. Ces positions représentent

en particulier des emplacements prévus pour la saisie des objets, des positions nécessaires à l'utilisation d'outils spéciaux, et des trajectoires diverses. Chacune des positions précédentes possède la particularité de pouvoir être atteinte par le robot à un instant t donné, sans engendrer de collisions avec les objets de l'environnement.

- *Le traitement des incertitudes:*
  L'univers physique dans lequel évolue le robot est soumis à des variations imprévisibles. Ces variations sont dues à la fois aux imperfections des structures mécaniques (imprécision, mauvaise répétabilité), aux incertitudes relatives aux objets manipulés (position, dispersion de fabrication), et aux incidents mécaniques mineurs tels que les glissements. Les incertitudes qui en découlent n'ont aucune influence sur le déroulement du programme de manipulation tant qu'elles sont inférieures à la précision requise par la tâche (jeux de montage en particulier). Une telle situation nécessite bien évidemment une préparation minutieuse de l'environnement. Cette préparation est généralement très coûteuse, ce qui est inacceptable dans le cas des petites et moyennes productions. Une autre méthode, mieux appropriée au traitement des incertitudes, consiste à utiliser des données sensorielles. Dans ce cas, le programme de manipulation comporte des stratégies qui combinent des actions du robot avec des opérations de lecture sur des capteurs. Chaque stratégie utilisée conduit alors à réduire un domaine d'incertitude donné, afin de le rendre compatible avec les impératifs du montage. Nous parlerons alors de *stratégies de mouvements fins.*

Outre la difficulté de résolution qu'ils présentent, les deux problèmes précédents possèdent de fortes inter-connexions. Le calcul de trajectoires sûres dans un espace très contraint nécessite par exemple de prendre en compte les intervalles d'incertitudes; inversement, le choix d'une trajectoire d'approche particulière lors d'une opération de montage difficile, peut faciliter l'interprétation des données sensorielles dans les situations imprévues.

Sur le plan conceptuel, un programme de commande de robot peut ainsi être considéré comme l'expression d'une *solution particulière* aux problèmes conjoints de la spécification de positions "sûres" et du traitement des incertitudes. Afin d'être exploitable industriellement, cette solution doit de plus répondre à des impératifs de fiabilité et d'efficacité souvent difficiles à obtenir avec les outils informatiques actuels.

Dans le cadre des procédés traditionnels de programmation des robots, la résolution des problèmes précédents est entièrement à la charge du programmeur. Il dispose pour cela d'outils informatiques plus ou moins évolués, variant d'un système à l'autre (cf. chapitre 3). Dans le contexte de la "programmation auto-

matique", les solutions apportées à ces problèmes sont directement calculées par le système à partir des informations géométriques et symboliques rattachées à la tâche à exécuter (cf. chapitre 5).

2.2 Les systèmes traditionnels:

2.2.1 Le formalisme de base:

Les premiers systèmes de programmation de robots étaient très pauvres sur le plan informatique. Ils étaient basés sur un mode de programmation dit "par l'exemple", permettant de reproduire automatiquement des séquences d'actions décrites auparavant par le biais de dispositifs de télécommande. Malgré toutes ses limitations [Latombe 83], ce mode de programmation est encore très répandu dans le milieu industriel. Cette situation un peu paradoxale vis à vis du savoir-faire actuel est due en grande partie à des raisons historiques et sociales. Elle est cependant en pleine mutation grâce à l'impulsion donnée conjointement par les projets de recherche nationaux (ARA en particulier) et internationaux (ESPRIT, EUREKA), amenant des laboratoires de recherche et des industriels à travailler en commun.

Les systèmes de programmation de robots qui pénètrent actuellement le milieu industriel exploitent de plus en plus les potentialités offertes par l'informatique. Ils sont basés sur un mode d'expression symbolique qui offre un formalisme général pour accéder à l'ensemble des fonctionnalités des robots. Ils s'appuient pour cela sur des *langages de manipulation* tels que AL [Finkel et al. 74], LM [Latombe, Mazer 81], et AML [Taylor et al. 82]. La plupart de ces langages sont des extensions de langages informatiques traditionnels. Ils permettent de décrire n'importe quelle tâche de manipulation en termes des *déplacements et des actions* de l'outil terminal du robot. Ils combinent pour cela des constructions algorithmiques classiques (variables symboliques, instructions conditionnelles et itératives, sous-programmes, mécanismes d'entrées/sorties), avec des constructions particulières adaptées à la robotique: modélisation de l'univers du robot, description des mouvements et des opérations de l'outil terminal, communication avec l'environnement par l'intermédiaire de capteurs, synchronisation du robot avec des processus externes tels que des mécanismes d'alimentation en pièces. Quelques unes de ces constructions sont illustrées par le programme suivant, écrit en langage LM:

```
FONCTION BOOLEEN INSERER(REPERE GOUJON,TROU; REEL FZMAX
                         ,EPS; ENTIER NMAX);
ENTIER N; VECTEUR LATF; REPERE TROU1;
DEBUT
   N := 0; TROU1 := TROU * TRANSLAT(-VZ,10);
   TANTQUE N > NMAX FAIRE
      DEPLACER GOUJON DE EXT-TRANSF(GOUJON,TROU1)
                         JUSQUA FZ > FZMAX;
      SI DISTANCE(GOUJON,TROU) < 1 ALORS RETOUR(VRAI);
      SINON
         LATF := VECT(FX,FY,0.);
         DEPLACER GOUJON DE TRANSLAT(LATF,EPS);
         EPS := EPS/2; N := N +1;
      FINSI;
   FINFAIRE;
   ECRIRE "L'INSERTION A ECHOUE"; RETOUR(FAUX);
FIN;
```

Ce programme permet d'exécuter l'insertion d'un goujon cylindrique dans un chanfrein. Il possède une syntaxe proche de celle de PASCAL, et il manipule des variables de type "vecteur" et "repère", afin de spécifier les paramètres numériques des déplacements du robot. Une particularité intéressante de ce type de programme réside dans l'utilisation des données pour stopper un mouvement en cas de contact (terme $FX > FMAX$), ou pour définir une action corrective: la direction du déplacement est donnée par le vecteur (FX, FY, 0), construit à partir des composantes du torseur de forces.

2.2.2 La modélisation de l'univers du robot:

Les langages de manipulation reposent pour la plupart sur un mode de représentation de l'univers à base de repères cartésiens. Le principe consiste à modéliser l'outil terminal du robot et les objets de l'environnement par un ensemble de repères cartésiens judicieusement choisis. Le programmeur raisonne alors en termes des positions et des orientations de ces repères, indépendamment de la forme des objets manipulés et de la structure mécanique du robot. Il dispose à cet effet de types de données appropriées tels que des repères, des vecteurs, et des transformations. Il peut également faire usage d'instructions adaptées au traitement de ces données, par exemple: le produit vectoriel ou l'extraction d'une transformation.

Dans un programme de manipulation, les repères introduits par l'utilisateur sont définis au moyen de transformations géométriques appliquées à des repères "existants", c'est à dire à des repères définis précédemment ou connus à priori

par l'interpréteur du langage, par exemple: système de référence de l'espace de travail, ou repère associé à l'extrémité du bras du robot. Certains langages, tels que AL ou LM, offrent la possibilité de lier plusieurs repères entre eux afin de les rendre temporairement ou définitivement solidaires. Ainsi, le fait de déplacer l'un des repères de la structure conduit l'interpréteur du langage à modifier automatiquement la position des autres repères dans le modèle de l'univers. Ce formalisme simplifie considérablement le travail de programmation. Il permet en particulier de faire usage d'un grand nombre de repères définis en vue d'opérations particulières telles que la localisation, la saisie, le déplacement, ou le positionnement d'un objet, sans accroître la complexité du programme.

Malgré les inconvénients qu'il comporte au niveau de la clarté des programmes (le concept de repère étant difficile à manipuler), ce mode de représentation de l'univers du robot sera probablement utilisé pendant longtemps du fait de sa nature géométrique bien appréhendée à la fois sur le plan mathématique et algorithmique. Il pourrait être avantageusement complété par des constructions permettant de représenter des incertitudes de position sur les repères, et par des algorithmes conçus de manière à propager automatiquement dans le modèle toutes les modifications qui portent sur des valeurs d'incertitudes connues. Les systèmes de programmation actuels ne permettent pas d'obtenir une représentation explicite de ces incertitudes. Les décisions concernant l'utilisation des capteurs sont de ce fait laissées à la charge du programmeur qui doit faire usage de son intuition et/ou d'expérimentation sur le robot.

L'absence de la notion d'incertitude dans les systèmes contemporains de programmation de robots, vient de deux raisons essentielles:

- Les incertitudes sont difficiles à estimer et à traiter explicitement.
- Les instructions présentes dans les langages de manipulation ne sont pas toutes interprétables en termes d'incertitudes, du fait qu'elles se situent à différents niveaux conceptuels. Par exemple, quel sera l'effet d'une instruction *"Déplacer axe i"* sur l'incertitude associée à la position de l'outil terminal? Comment prévoir l'évolution des termes d'incertitudes dans une itération contrôlée par des données sensorielles non explicitées en termes de repères?

Les solutions que nous avons apportées à ce problème sont discutées dans le chapitre 9. Elles s'appuient sur un langage géométrique que nous avons développé dans le but de répondre aux impératifs de la connexion CAO-robot et de la programmation automatique. Ce langage est décrit dans le chapitre 4.

2.2.3 La spécification des déplacements libres:

On appelle *déplacement libre*, tout déplacement du robot qui n'est soumis à aucune condition sensorielle. Ce type de déplacement est spécifié au moyen d'instructions combinant deux types d'informations: la géométrie de la trajectoire et la cinématique du mouvement.

Géométrie de la trajectoire:

La géométrie d'une trajectoire peut être décrite en termes de coordonnées articulaires. Le programme développé est dans ce cas entièrement dépendant de la structure mécanique du robot. Elle peut aussi être spécifiée en termes de coordonnées cartésiennes, et ainsi exploiter le modèle de l'univers géré par le programme de manipulation. Une manière naturelle de travailler consiste alors à utiliser des repères "déplaçables", c'est à dire des repères liés temporairement à l'extrémité mobile du robot. Le programmeur choisit dans ce cas le repère D le mieux adapté à la description de chaque déplacement. Il utilise pour cela des constructions du type:

LIER D A ROBOT; DEPLACER D A F

où D et F représentent des repères associés à des objets du modèle. Le déplacement exécuté par le robot à la suite d'une telle instruction conduit à amener le repère D en coïncidence avec le repère F. La trajectoire suivie par D peut cependant varier en fonction du mode de déplacement choisi par l'utilisateur. Cette trajectoire peut être calculée dans l'espace de la tâche. Une trajectoire dite "cartésienne" conduira par exemple à déplacer le centre du repère D suivant une ligne droite, et à exécuter des rotations autour d'une direction fixe. La trajectoire peut aussi être planifiée dans l'espace des coordonnées articulaires du robot. D décrit alors une courbe non maîtrisée par le programmeur. Une technique classique pour contourner ce problème, consiste à spécifier des points de passage pour le repère D.

Contraintes cinématiques:

L'ensemble des calculs relatifs à la génération des trajectoires, à la transformation des coordonnées cartésiennes en coordonnées articulaires, et à la production des consignes de positions adressées aux contrôleurs du robot, sont réalisés par l'interpréteur du langage [Paul 81]. L'utilisateur peut cependant contrôler les vitesses des déplacements au moyen d'instructions appropriées du langage. Il peut également faire exécuter des mouvements complexes comportant d'importantes contraintes géométriques et cinématiques. La planification de telles trajectoires met en oe uvre des calculs de courbes d'accélération qui nécessitent généralement un prétraitement "hors-ligne".

2.2.4 La spécification des conditions sensorielles liées aux mouvements:

Deux types de mouvements peuvent être exécutés en rapport avec des données sensorielles transmises par un capteur extéroceptif: les mouvements gardés et les mouvements compliants.

Les mouvements gardés:

Les mouvements gardés sont des mouvements qui peuvent être stoppés en cours d'exécution par l'évaluation à vrai d'une expression booléenne. Cette expression inclut des données sensorielles, qui sont mises à jour périodiquement par les capteurs externes du robot. Un tel mouvement peut être décrit par une instruction du type:

DEPLACER goujon DE transformation JUSQUA Fz>Fzmax

Nous verrons dans les chapitres 4 et 6 que ce type de mouvement peut être généralisé, afin de prendre en considérations des conditions d'arrêt plus complètes. Ces conditions combinent des informations qui portent sur les positions et les forces mesurées par les capteurs du robot. Elles permettent de spécifier de manière plus claire, les situations atteintes par le robot après exécution des mouvements commandés. Dans le cas où le mouvement ne met en jeu aucun contact dans l'expression de l'objectif à atteindre, la condition d'arrêt porte uniquement sur la position commandée (ce qui revient à exécuter un mouvement libre).

Les mouvements compliants:

Les mouvements compliants sont des mouvements contrôlés en position suivant certaines directions; ils sont contrôlés dans les autres directions à partir des informations transmises par des capteurs de forces ou de proximité. Les langages AL et PAL incluent par exemple des instructions de déplacement qui combinent des paramètres de position avec des contraintes portant sur les forces exercées par le robot. Le langage LM comporte également (dans une version expérimentale) des constructions du type [Gandon 86]:

DEPLACER repère VERS transformation EN-MAINTENANT contrainte JUSQUA garde

où *repère* est un repère déplaçable, *transformation* indique la direction de mouvement à suivre, *contrainte* est un ensemble de contraintes de force, et *garde* est une expression booléenne faisant référence à des données sensorielles. Les contraintes de force spécifiées sont vérifiées tout au long du mouvement; elles conduisent à engendrer automatiquement des déplacements correctifs pendant l'exécution du

mouvement.

Cette forme d'expression des mouvements compliants est basée sur un principe de séparation des termes de position et de force dans la spécification de la commande [Mason 81]. Prenons par exemple un objet A que l'on veut déplacer le long d'une surface S, en suivant une direction $D = Oy$. Le repère utilisé pour spécifier la commande est situé sur la face de contact de A, de telle sorte que cette face soit contenue le plan xOy. Les contraintes liées au mouvement sont alors les suivantes:

$$T_x = 0, \quad T_y = 1, \quad R_z = 0$$
$$F_z = 5, \quad M_x = 0, \quad M_y = 0$$

où T_x, T_y et R_z sont respectivement les directions de translation et de rotation par rapport aux axes Ox, Oy et Oz; F_z, M_x et M_y sont les composantes de force et de moment mesurées suivant les axes Oz, Ox et Oy.

Les contraintes $T_x = 0$ et $R_z = 0$ imposent qu'aucun mouvement ne doit être exécuté en translation suivant Ox et en rotation autour de Oz. Les contraintes $F_z = 5$, $M_x = 0$ et $M_y = 0$ conduisent respectivement à appliquer des translations correctives suivant Oz, et des rotations correctives suivant Ox et Oy. L'instruction LM qui décrit ce mouvement est la suivante:

DEPLACER repère-A VERS Translat(Vy,1)
EN-MAINTENANT $F_z = 5$, $M_x = 0$, $M_y = 0$ *JUSQUA temps>10*

2.2.5 Le contrôle des processus parallèles:

De nombreuses applications nécessitent de coordonner les opérations d'un robot avec d'autres robots ou avec d'autres appareils tels que des tapis roulants ou des outils spécialisés. La possibilité de synchroniser des processus concurrents n'est offerte que dans quelques langages. AL dispose par exemple d'instructions spéciales du type CODEBUT et COFIN, pour définir des sections de programmes exécutables en parallèle. Ces sections peuvent piloter différents robots opérant de manière asynchrone.

D'autres langages tels que LM ou AML comportent des constructions plus élémentaires qui permettent de lancer des opérations en parallèle, par exemple: des déplacements sur deux robots différents, ou une commande de déplacement combinée avec une action portant sur un outil spécialisé. Le formalisme utilisé conduit à rendre immédiatement le contrôle au programme de manipulation, chaque fois qu'une opération concurrente est lancée.

D'autres développements réalisés sur le langage LM, ont été orientés vers l'intégration de véritables fonctions de parallélisme. L'approche suivie consiste à synchroniser les processus parallèles à l'aide de "messages", dans un formalisme dérivé de celui utilisé dans CONKER [Muntean et al. 85]. Les constructions développées devaient permettre de spécifier des commandes du type:

$$< \textit{Condition booléenne} > \implies < \textit{Actions} >$$

Mais l'implantation réalisée n'est restée qu'à un stade expérimental.

2.3 Situation actuelle et évolution:

2.3.1 Les systèmes traditionnels:

De nombreux langages de manipulation ont été développés au cours des dix dernières années [Latombe 82]. Certains sont commercialisés: LM (ITMI), VAL (Unimation), AML (IBM), RAIL (Automatix)... La plupart de ces langages ont déjà été utilisés pour programmer une grande variété de tâches de manipulation [Paul 77] [Latombe 82]. Cependant, la programmation d'un robot à l'aide d'un tel formalisme reste une tâche délicate qui nécessite une bonne maîtrise à la fois des techniques informatiques et des phénomènes physiques: le programmeur doit stipuler quantitativement tous les déplacements du robot tout en s'assurant qu'aucune collision n'est susceptible de se produire; il doit également prévoir la gestion des données sensorielles afin de prendre en compte les incertitudes liées au monde réel. Même dans le cadre d'applications à faible degré de complexité, un tel travail ne peut être exécuté que par des programmeurs expérimentés. Cet état de choses constitue encore un frein important à l'expansion des systèmes symboliques de programmation de robots.

C'est pourquoi les concepteurs des langages ont été amenés à développer des outils informatiques permettant de faciliter la tâche du programmeur, et de rendre ainsi les systèmes accessibles à des utilisateurs non informaticiens. Par exemple, POINTY [Gini 78] et LM-EX [Bansard 83] offrent la possibilité de construire des programmes en langage AL (POINTY) et en langage LM (LM-EX) à l'aide de fonctions interactives basées sur des techniques de programmation par l'exemple. LM-GEO [Mazer 82] constitue un interface entre le langage LM et une base de modèles géométriques simples. Il permet de définir symboliquement les positions des repères en termes de relations géométriques du type:

$$\textit{entité-}E_i \ \textit{CONTRE entité-}E_j$$
ou encore
$$\textit{entité-}E_i \ \textit{PARALLELE-A entité-}E_j.$$

Ces relations sont interprétées par le système qui engendre un programme LM exécutable après compilation sur l'armoire de commande du robot. Nous verrons dans le chapitre 9 que le formalisme de base de LM-GEO peut également être utilisé pour définir les *situations objectifs* fournies en entrée d'un système de programmation automatique.

D'autres outils exploitant des bases de données CAO ont été développés dans plusieurs laboratoires de recherche. Le but recherché consiste alors à doter les systèmes précédents de facilités de programmation "hors-ligne", basées sur l'utilisation d'un poste de travail graphique. Le système comporte alors des outils de simulation et de programmation parfois relativement sophistiquées (cf. chapitre 3).

La structure informatique qui se dégage de cette évolution reste toujours centrée sur le concept de langage de manipulation. Elle conduit à construire différentes couches logicielles permettant de faciliter de plus en plus la tâche du programmeur. Les premières couches logicielles tentent de limiter la complexité par l'apport de moyens de communication homme-machine plus évolués (graphismes, symboles...). Les couches supérieures, non décrites à ce niveau, apportent des outils d'aide à la décision. Le rôle de la CAO dans cette évolution est décrit dans les trois chapitres qui suivent.

2.3.2 La programmation automatique:

Quel que soit le mode d'expression utilisé (gestuel, graphique ou symbolique), les systèmes de programmation de robots évoqués précédemment opèrent tous à un même niveau conceptuel: "le niveau manipulation".

Les futurs systèmes de programmation automatique de robots reposent sur un principe différent. Ils conduisent à décrire uniquement l'objectif à atteindre, sans spécifier la manière d'y parvenir. Cet objectif est exprimé en termes de modèles géométriques et de relations d'assemblage. Le rôle du système est alors de convertir cette description symbolique, en un programme de manipulation exécutable par un robot. Il doit pour cela *planifier* les opérations nécessaires à la réalisation de la tâche, compte tenu des contraintes imposées par l'environnement physique. La problématique de ce mode de programmation est décrite dans le chapitre 5. Les concepts sous-jacents et les méthodes de raisonnement associées sont développées dans les chapitres 6, 7 et 8.

La programmation automatique en robotique présente un double intérêt pratique et scientifique. Sur le plan pratique, elle devrait conduire à terme à accroître dans une large mesure la flexibilité des robots par l'apport d'un mode de pro-

grammation à la fois simple, puissant et efficace. En particulier, les programmes engendrés seront certainement plus fiables et mieux adaptés aux tâches à exécuter (la présence de modèles plus complets permettant au système de mieux utiliser les capteurs et de prévenir de nombreux échecs dus aux incertitudes). De plus, le niveau conceptuel auquel opèrent de tels systèmes offre un support indispensable à une véritable intégration CAO/FAO. Dans l'immédiat, l'aspect scientifique est certainement le point le plus important dans le sens où il conduit à aborder de manière réaliste des domaines de recherche fondamentaux en IA: la modélisation du raisonnement géométrique et la génération/exécution de plans d'actions en univers réel.

Bien que les techniques matérielles et logicielles ne soient pas encore réellement maîtrisées, les résultats obtenus jusqu'à ce jour sont très encourageants. Le système HANDEY [Mazer 87] et le système SHARP décrit dans le chapitre 9, sont significatifs sur ce point. Ils montrent que le problème de la réalisation d'un système de programmation automatique n'est pas insoluble, même si l'échéance est assez lointaine. Compte tenu de l'état d'avancement des recherches dans ce domaine (que ce soit au niveau des laboratoires de recherche, ou de celui du secteur industriel de pointe), il est probable que tous ces travaux déboucheront dans un proche avenir sur des résultats pratiques importants prenant la forme d'outils interactifs d'aide à la programmation. Ces outils devraient permettre de calculer des prises stables sur un objet, d'engendrer automatiquement des trajectoires "sûres" pour le robot, d'élaborer des stratégies de montage, et de vérifier la validité d'un programme vis-à-vis des contraintes d'incertitudes liées à la tâche.

Chapitre 3

La connexion CAO-Robot: Programmation graphique

3.1 Principes de base et motivations:

Un langage de programmation de haut niveau pour la robotique offre un formalisme général pour accéder aux fonctionnalités d'un robot. Il permet de décrire une tâche de manipulation en termes des déplacements et des actions de l'outil terminal du robot. Il permet aussi de "coupler" certains mouvements avec des opérations sensorielles. Ce mode de programmation présente un haut degré de généralité. Mais la puissance potentielle qu'il représente est difficilement exploitable par le programmeur, faute d'outils informatiques permettant de réellement maîtriser la complexité inhérente à l'univers physique. En particulier, la représentation des données manipulées par le programme est souvent trop abstraite pour permettre à l'utilisateur d'avoir une bonne compréhension de ce dernier. C'est pourquoi la CAO présente un intérêt particulier en tant qu'outil de modélisation des objets et de communication homme-machine. Elle offre la possibilité d'accéder plus facilement aux différentes fonctionnalités du robot. Elle autorise de surcroît l'utilisation d'un mode de programmation "hors-ligne", permettant de travailler en dehors du site d'opération par le biais d'un poste de travail graphique.

Les outils informatiques sur lesquels repose la connexion CAO-Robot peuvent présenter des degrés de sophistication divers. Ils opèrent tous sur des modèles dérivés des modèles habituellement utilisés en CAO. De telles possibilités peuvent être aisément implantées sur un calculateur de capacité moyenne connecté avec un poste de travail graphique. La qualité des outils fournis par le système varie alors considérablement en fonction de l'environnement informatique choisi [Laugier 83]. Deux facteurs importants contribuent à l'efficacité de ces outils:

- La puissance de calcul graphique (qualités des images, dynamisme, interactivité).

- La capacité du système à intégrer des données physiques, telles que les imperfections des structures mécaniques du robot, Les incertitudes liées aux objets manipulés ou les incidents mécaniques mineurs (glissements en particulier).

De par sa nature même, la CAO constitue un excellent outil pour construire et mettre au point des programmes de commande de robots. Elle permet de diminuer dans une large mesure les temps d'immobilisation des robots dans les ateliers, tout en offrant un environnement de travail plus favorable (les risques d'accidents matériels ou humains étant écartés). Elle procure de plus un mode d'expression naturel qui facilite la tâche du programmeur. En particulier, la présentation graphique des résultats permet à l'utilisateur de veiller à ce qu'aucune collision ne se produise lors de l'exécution de la tâche.

L'objectif de ce chapitre est de montrer de quelle manière il est possible de combiner un langage de programmation symbolique et un ensemble de fonctions CAO pour obtenir un système de programmation de robot à la fois général, efficace et d'utilisation relativement simple. Une telle connexion est basée sur l'utilisation d'un simulateur graphique "complet", présentant les principales caractéristiques suivantes:

1. Il peut être utilisé pour simuler l'exécution de des programmes de manipulation, afin d'en évaluer la validité. Nous parlerons alors de *simulation du robot.*

2. Il offre la possibilité de représenter les scènes obtenues sur un terminal graphique ou sur un terminal de synthèse d'images. Nous parlerons alors de *visualisation graphique.*

3. Il inclut des mécanismes informatiques permettant de simuler les capteurs et les aléas du monde physique. Nous parlerons alors de *simulation de l'univers physique.*

4. Il possède un ensemble de fonctions graphiques interactives, permettant de créer de nouveaux programmes d'application. Nous parlerons alors de *programmation graphique.*

Dans la suite, nous appelerons "programmation hors-ligne" le mode de programmation induit par un système qui possède les fonctionnalités précédentes.

Tout au long de l'exposé, nous ferons référence à l'expérience acquise lors de l'implantation du système LM [Latombe, Mazer 81] [Mazer 81] et des outils informatiques associés [Latombe et al. 84].

3.2 Bref état de l'art en simulation graphique de robots:

De nombreux systèmes de simulation graphique de robots ont été implantés ou sont en cours de développement dans des laboratoires de recherche. Quelques uns opèrent dans un univers bidimensionnel. Ils permettent de simuler des robots à trois ou quatre degrés de liberté à l'aide d'un ordinateur personnel et d'un terminal graphique couleur bas de gamme. Ils utilisent pour cela un symbolisme très simple pour représenter le robot et son espace de travail. Le système développé chez IBM est par exemple conçu pour simuler un robot IBM-7535 [Meyer, Jayaraman 83]. Celui implanté à l'université de Tokyo travaille avec un robot Sankyo 4 axes commandé par un langage spécifique appelé EARLS-2 [Arai 83].

D'autres simulateurs opèrent dans l'univers tridimensionnel. Ils sont pour la plupart basés sur des systèmes de CAO existants: CATIA [Liégeois et al. 82] [Borrel et al. 83], GEOMAP [Sata et al. 81], SAMMIE [Heginbotham et al. 79] [Dooner et al. 82], GDP [Meyer 81]... Ils offrent cependant des performances qui varient beaucoup d'une implantation à l'autre. Le simulateur de l'université de Tokyo [Sata et al. 81] nécessite par exemple plusieurs secondes de CPU pour calculer une image de type "fil-de-fer", alors que la fonction "robot" du système CATIA [Liégeois et al. 84] permet d'opérer en quasi-temps réel sur des dessins simples présentés avec élimination des lignes cachées. Afin d'obtenir des temps de réponse acceptables, de nombreux systèmes utilisent des graphismes à base de parallélépipèdes et de cylindres . C'est le cas par exemple du simulateur développé au laboratoire WZL d'Aix la chapelle [Weck et al. 81]. D'autres systèmes tentent de résoudre le problème du "temps réel" en s'appuyant sur des architectures matérielles appropriées. Cette solution a été employée au laboratoire de recherche de Mitsubishi pour un système de simulation implanté par modules sur un ensemble de micro-calculateurs à base de 8086 [Tsujido et al. 83]. Les nouvelles gammes de matériel qui touchent actuellement le marché, présentent des capacités graphiques 3D qui permettent de résoudre ce problème de visualisation.

Peu de simulateurs ont été développés autour d'un véritable langage de commande de robot. Les principaux laboratoires de recherche ayant travaillé dans ce

contexte sont: le centre de recherche IBM, l'université de Stanford et le laboratoire LIFIA/IMAG de Grenoble. Le système EMULA de chez IBM [Meyer 81] est basé par exemple sur le langage AML. Il permet de simuler un robot IBM-7565 sur un calculateur IBM-370. Le simulateur développé à l'université de Stanford utilise un sous-ensemble des instructions du langage AL, pour représenter graphiquement les déplacements d'un robot PUMA-500 modélisé à l'aide du système ACRONYM [Soroka 80]. Enfin, le simulateur ISR [Laugier, Pertin 84] développé dans notre laboratoire a été construit comme une extension du système LM . Il est important de souligner que parmi tous ces systèmes, seuls les systèmes EMULA et ISR possèdent des fonctions pour simuler des capteurs. De telles fonctions ont également été développées dans un autre contexte (celui de la "téléopération assistée par ordinateur") au laboratoire IRISA de l'université de Rennes. Elles exploitent des modèles créés à l'aide du système PADL-2 de l'université de Rochester, afin de simuler les lois de commandes d'un robot asservi à des données sensorielles de nature proximétrique [André, Boulic 85].

D'autres recherches concernant la CAO de robots et la programmation graphique en robotique sont en cours dans différents laboratoires. En France, les résultats les plus significatifs ont été obtenus au laboratoire LAMM de l'université de Montpellier. Ils ont conduit à étendre les fonctionalités du système CATIA suivant deux directions: la CAO de robots (aspects cinématiques et dynamiques), et la programmation graphique de tâches simples pour un *robot simulé*. Les "programmes" engendrés par le système sont alors linéaires, et ils ne prennent pas en compte les facteurs d'imprécision et d'incertitude liés au robot réel. Les améliorations récentes portent sur la détection des collisions et sur le calcul de trajectoires mieux adaptées à la commande des robots [Liégeois et al. 86].

Bien que des résultats parfois très impressionnants aient été présentés dans des expositions par des sociétés industrielles telle que Mac Donnell Douglas, il reste un travail important à effectuer pour rendre ces résultats réellement exploitables en production. Les grands projets de recherche sur ce thème sont significatifs des problèmes non encore résolus. A titre d'exemple, le projet ESPRIT auquel participent l'université de Karlsruhe et la société Renault Automation a pour premier objectif de développer un système de programmation hors-ligne incluant un système CAO industriel et un langage de commande de robots. Ce projet est basé sur des concepts similaires à ceux décrits dans ce rapport (cf. [Dillmann, Huck 85]).

3.3 Principe de la connexion CAO-robot:

3.3.1 Principe de base:

Un système de programmation de robot comportant des outils évolués de programmation "hors-ligne" inclut deux ingrédients essentiels: un langage de manipulation et un simulateur graphique possédant les caractéristiques énoncées dans le paragraphe 3.1. Ce simulateur constitue l'élément central de la connexion CAO-robot. Il est conçu de manière à interfacer l'interpréteur du langage et l'utilisateur (cf. figure 3.1). Il exploite pour cela trois types de données:

- *Des modèles géométriques:*
  Ces modèles représentent les composants du robot et les objets présents dans son espace de travail. Ils sont complétés par des modèles cinématiques et par quelques modèles dynamiques.Ce point est développé dans le paragraphe 3.4.

- *Une description de la tâche:*
  Cette description est réalisée en termes d'instructions symboliques codées à l'aide d'un langage de manipulation. Les instructions simulées peuvent être extraites d'un programme complet entré de manière textuelle. Elles peuvent également provenir d'une séquence décrite graphiquement. Ce point est développé dans le paragraphe 3.7.

- *Une description des "événements physiques":*
  Ces événements, de nature généralement aléatoire, représentent des phénomènes physiques qui ne peuvent pas toujours être décrits formellement dans le modèle. Ils proviennent d'une part des incertitudes relatives à la tâche (dispersion de fabrication, imprécisions du robot et des capteurs...), et d'autre part d'incidents mécaniques mineurs tels que des petits glissements. Ils peuvent également être la conséquence d'actions "malheureuses" du robot, tels que des collisions ou des ruptures d'équilibres. Ce point est développé dans le paragraphe 3.6.

Les informations précédentes sont utilisées par le simulateur pour construire et maintenir à jour un modèle structuré de l'univers du robot. C'est ce modèle qui évolue au cours de la simulation, et qui est périodiquement visualisé par les fonctions graphiques du système. Le simulateur se comporte alors comme un *pseudo-robot*, capable d'exécuter des instructions de manipulation et d'acquérir certaines informations concernant son environnement simulé.

3.3.2 Fonctionnement de l'interface CAO-robot:

Le simulateur représente le cœur de la connexion CAO-Robot. Il reçoit pour fonctionner trois types de commandes en entrée:

- *Les commandes de manipulation:*
  Elles sont émises par l'interpréteur du langage en réponse au programme exécuté. Ces commandes décrivent d'une part les déplacements et les actions de l'outil terminal du robot, et d'autre part les opérations de mise en service des capteurs. Elles représentent les consignes envoyées habituellement aux contrôleurs du robot réel.

- *Les commandes de simulation:*
  Elles sont émises par l'utilisateur ou par l'interpréteur du langage. Elles concernent l'émulation du temps, le contrôle des paramètres de visualisation, la simulation de l'univers physique et des capteurs, et la gestion du processus de simulation.

- *Les commandes de téléopération:*
  Elles sont émises par l'utilisateur lors des phases de "programmation graphique". Elles permettent de guider interactivement le robot simulé afin d'entrer des positions clés nécessaires à la description de la tâche de manipulation.

Ces commandes sont utilisées par le système pour interfacer les quatre fonctions sur lesquelles repose la connexion CAO-robot (cf. paragraphe 3.1): la simulation du robot, la visualisation graphique, la simulation de l'univers physique, et la programmation graphique. Les caractéristiques de ces fonctions sont décrites dans les paragraphes qui suivent.

Deux modes opératoires sont possibles sur la base des commandes précédentes: un mode "batch" et un mode "interactif".

- Le premier mode opératoire est utilisé pour tester la validité de programmes de manipulation existants. L'utilisateur travaille alors à partir d'un programme complet incluant des commandes de simulation. Il obtient de cette manière une représentation dynamique complète de la tâche de manipulation à chaque exécution simulée. Il peut ainsi évaluer la robustesse de son programme vis-à-vis de différentes situations physiques, en réalisant plusieurs exécutions consécutives. Le programme testé inclut alors des commandes capables de modifier aléatoirement le modèle de l'environnement du robot (cf. paragraphe 3.6).

- L'autre mode opératoire est utilisé pour construire un nouveau programme, ou pour compléter un programme existant. L'opérateur travaille alors "en direct" avec le processus de simulation, afin de mettre au point et/ou de compléter son programme. Il peut alors stopper à tout moment l'exécution de la tâche de manière à modifier interactivement le contexte de la simulation, ou à entrer graphiquement de nouvelles instructions. Il peut également

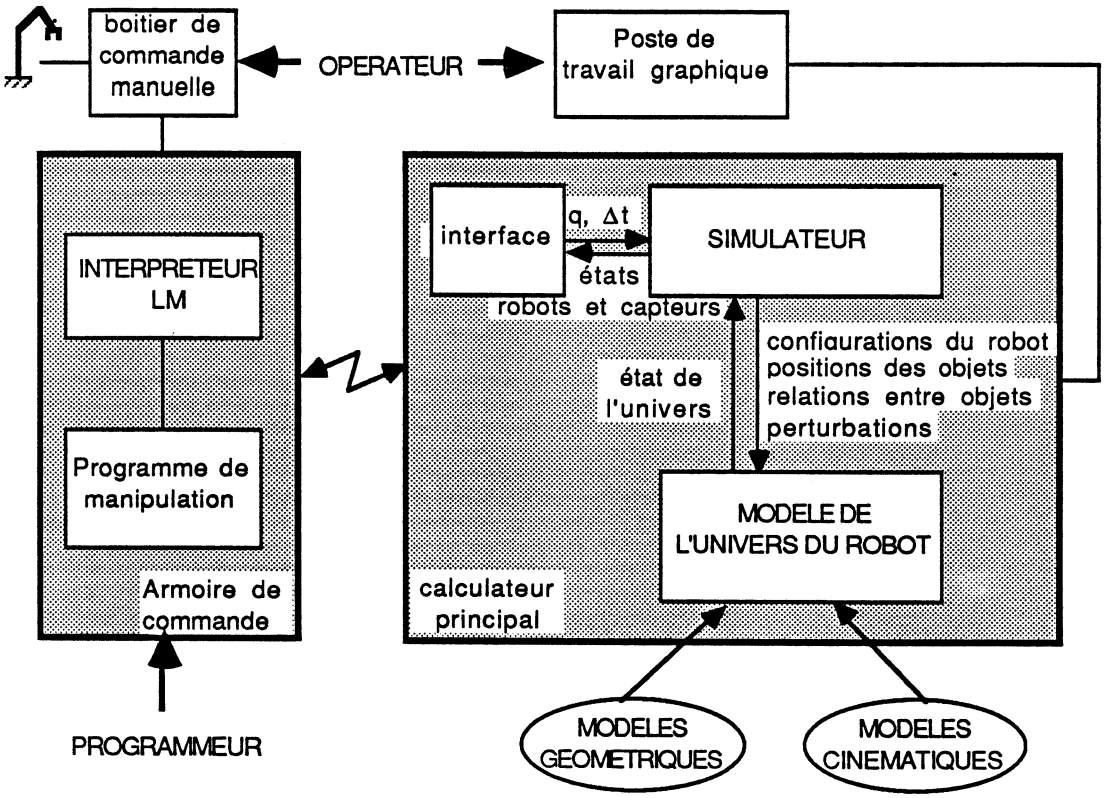


Figure 3.1: Schéma synoptique de la connexion CAO-Robot du système LM.

faire usage de fonctions géométriques à base de calculs d'interférence, pour analyser des situations de collisions ou de contacts difficiles à discerner sur les images produites par le système.

3.4 Modélisation géométrique de l'univers du robot:

3.4.1 Modèles nécessaires:

Les modèles requis par les outils de programmation hors-ligne comportent quatre composantes de base:

- *La composante géométrique:*
  Elle spécifie la géométrie des objets pour les besoins du graphique et des calculs de collisions. Les modèles utilisés sont ceux de la CAO [Requicha 80].

Ils sont constitués de représentations polyédriques qui approximent les surfaces des objets.

- *La composante relationnelle:*
  Elle modélise les relations spatiales et les contraintes mécaniques qui lient les objets de l'univers. Les représentations utilisées sont des graphes orientés et valués. Elles sont manipulées par le simulateur pour maintenir à jour un modèle cohérent du robot et de son environnement, cf. figure 3.2.

- *La composante cinématique:*
  Elle représente les propriétés cinématiques associées aux mécanismes articulés du robot. Les modèles exploités sont basés sur des lois mécaniques simples, excluant les paramètres de vitesses et d'accélérations. Ils sont utilisés par le simulateur pour calculer les différentes postures prises par le robot [Laugier et al. 84]. Ce mode de calcul est illustré par la figure 3.3.

- *La composante fonctionnelle:*
  Elle spécifie les entités géométriques créées pour les besoins de la programmation graphique. Ces entités sont indépendantes des modèles géométriques des objets. Elles représentent des points, des droites ou des plans rattachés artificiellement à ces objets. Ce point est développé dans le paragraphe 3.7.

Ces modèles peuvent être complétés par une *composante physique*, permettant de représenter les caractéristiques des objets (poids, centre de gravité, coefficients de frottements ...). Ces paramètres sont nécessaires pour simuler les effets des forces et de la dynamique. Certains d'entre eux sont utilisés par les fonctions de planification développées dans le cadre de la programmation automatique. Ils sont totalement ignorés par le processus de simulation utilisé pour la programmation "hors-ligne".

Les modèles exploités par le système que nous avons développé sont ceux du système SMGR [Troccaz 85]. Ils sont sommairement décrits dans les deux paragraphes qui suivent.

3.4.2 Modèle géométrique et relationnel:

Chaque objet de l'univers du robot est représenté par un repère cartésien associé à un liste de faces polygonales. Le repère cartésien permet au système de localiser l'objet dans l'espace de travail du robot; les faces polygonales sont utilisées pour les opérations de visualisation.

La structure relationnelle est modélisée par un graphe orienté dans lequel chaque arc dénote une relation particulière entre deux éléments du modèle, et chaque nœud symbolise un objet de l'univers. Les informations transportées par

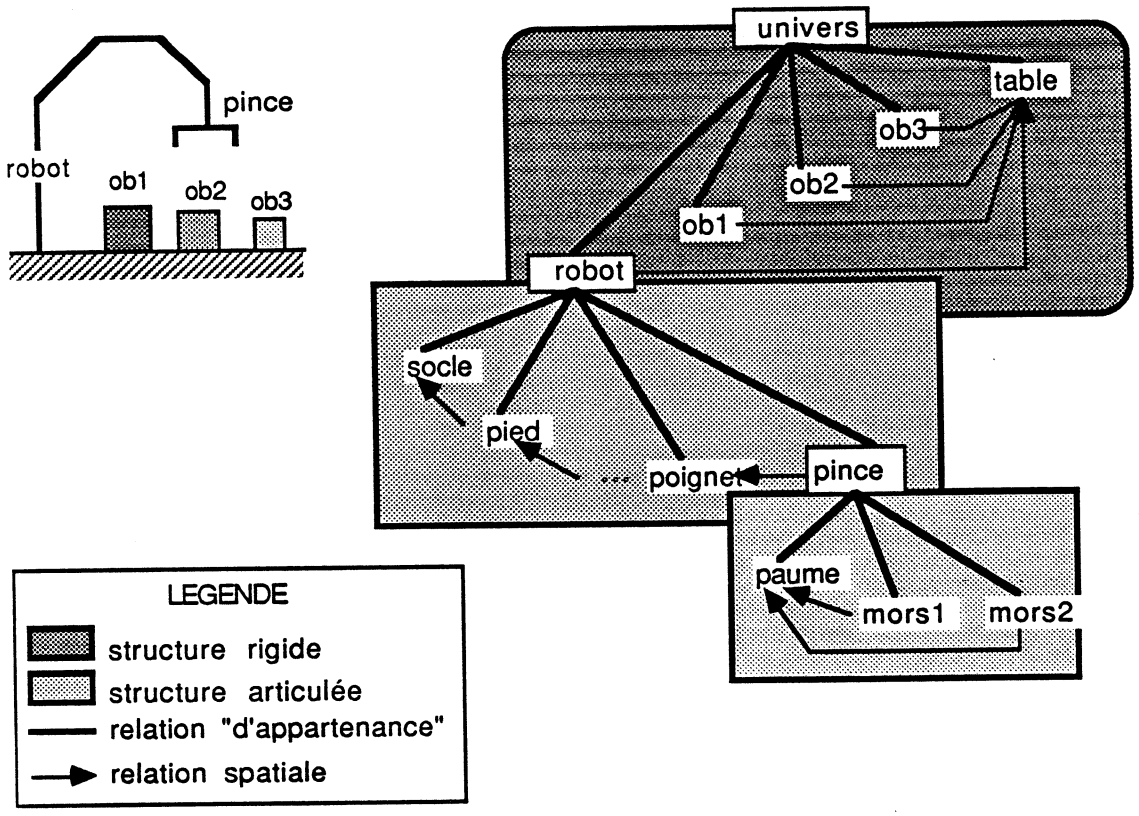


Figure 3.2: Exemple de structure relationnelle incluse dans le modèle.

les arcs sont des matrices en coordonnées homogènes caractérisant des transformations géométriques. Certaines transformations sont codées numériquement; elles ne sont dans ce cas soumises à aucune contrainte autre que les positions relatives des couples d'objets considérés. D'autres transformations sont paramétrées; elles définissent alors des contraintes mécaniques imposées par la structure.

Les données associées aux nœuds du graphe varient en fonction du type des objets représentés: dans le cas d'un objet élémentaire, l'information stockée correspond au modèle géométrique de cet objet; dans le cas des objets composés, elle se limite à la spécification des repères cartésiens associés.

Plusieurs types de structures cohabitent dans le modèle. Toutes ces structures

sont représentées par des arborescences, dont les racines constituent les points d'accès naturels. Parmi ces structures, nous distinguerons de par leurs fonctions particulières trois constructions différentes:

- Une structure hiérarchique induite par les *relations spatiales* qui lient les objets de l'univers.
- Des structures arborescentes associées aux *mécanismes articulés.*
- Des arborescences évolutives caractérisant les *structures rigides* créées par certaines opérations du robot. Ces structures représentent des groupements d'objets solidaires dans leurs déplacements. Elles peuvent avoir un caractère permanent; c'est par exemple le cas d'un assemblage par collage ou par sertissage. Elles peuvent aussi être de nature éphémère; c'est le cas pour un objet tenu par le robot ou pour un objet maintenu sur un autre par les forces de pesanteur.

La figure 3.2 illustre cette combinaison de structures. La représentation multiple qui en découle permet au système d'exécuter les opérations de mise à jour du modèle, en appliquant plusieurs méthodes de calcul choisies en fonction du contexte. Par exemple, un parcours de la structure arborescente tel que décrit dans la figure 3.3, dérive des techniques de calculs induites par les équations cinématiques.

3.4.3 Modèle cinématique du robot:

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.

D'autres articulations plus complexes peuvent également être décrites à l'aide de ces deux types de liaisons élémentaires. Une liaison de type rotule peut par exemple être représentée par une combinaison de trois liaisons rotoïdes à axes concourants. Dans ce cas, deux éléments fictifs représentés par des repères cartésiens sont introduits automatiquement par le système dans la chaîne cinématique.

Certains mécanismes possèdent des degrés de liberté qui sont fonctionnellement liés. Ils sont composés de plusieurs éléments rigides, reliés entre eux par une liaison unique. De tels mécanismes sont appelés "structures à articulations dépendantes". Ils sont décrits au moyen d'articulations spéciales du type *poly-rotoïde et poly-prismatique.* Ces nouvelles articulations représentent de simples

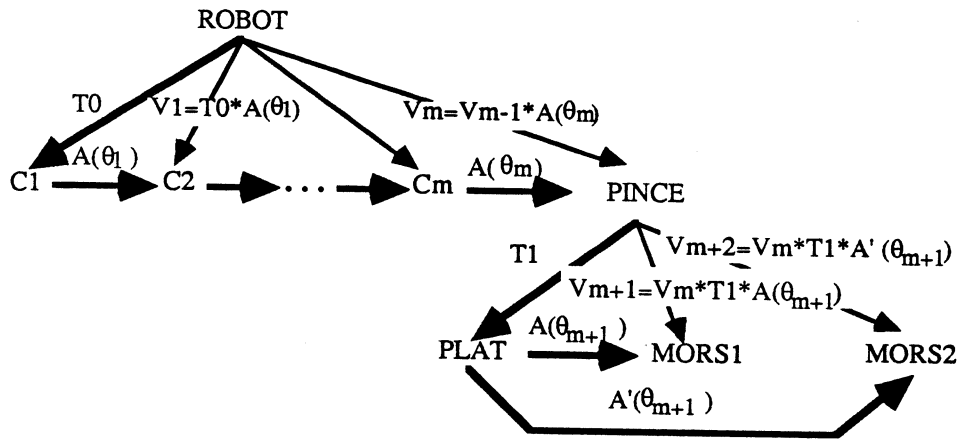


Figure 3.3: Principe d'évaluation d'une structure cinématique.

extensions de celles décrites précédemment. Elles permettent de *distribuer une commande* de mouvement sur plusieurs corps physiques. Ce type de structure correspond par exemple au mécanisme d'une pince à deux mors parallèles.

Certains robots comportent dans leurs mécanismes des structures passives asservies à des articulations commandées par des moteurs. La plus répandue de ces constructions est la boucle cinématique de type paralléllogramme, qui est illustrée par la figure 3.4). Afin de représenter de tels mécanismes, le système utilise des articulations de type *c-rotoïde* qui permettent de coupler toutes les *articulations passives* de la structure avec *l'articulation active* qui commande l'ensemble. Cette méthode, inspirée de celle utilisée dans CATIA, permet de conserver la structure arborescente adaptée au calcul de la cinématique, sans augmenter artificiellement le nombre de degrés de liberté du mécanisme.

3.5 Simulation du robot:

3.5.1 Contrôle de la cinématique:

Le module de contrôle de la cinématique de la tâche simule l'exécution des instructions incluses dans le programme de manipulation. Il travaille à partir des séquences de positions articulaires transmises par l'interpréteur du langage. Il

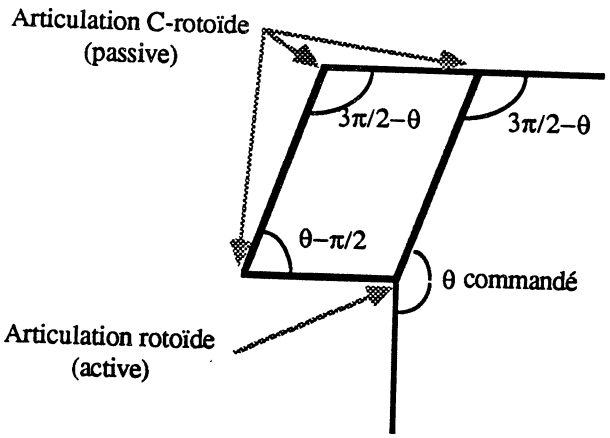


Figure 3.4: Structure de type parallélogramme.

produit en retour des *consignes de mise-à-jour* adressées au module de gestion du modèle de l'univers. Plusieurs robots peuvent être simulés simultanément de cette manière.

3.5.2 Gestion du modèle:

Ce module maintient à jour le modèle de l'univers. Il reçoit pour cela quatre types de requêtes (cf. figure 3.1):

- Des commandes de simulation permettant de positionner des objets dans l'espace de travail.
- Des commandes de simulation décrivant des liaisons rigides créées ou détruites par des instructions particulières du programme de manipulation. Ces liaisons sont temporaires ou permanentes. Elles définissent des "structures rigides", c'est à dire des groupements d'objets solidaires dans leurs déplacements. Il s'agit par exemple de simuler les effets des instructions *LIER* et *DELIER* du langage *LM*.
- Des requêtes émises par le module de contrôle de la cinématique; ces requêtes spécifient à chaque étape les nouvelles positions prises par les éléments du robot.
- Des requêtes émises par le module de simulation de l'univers physique; ces requêtes conduisent à engendrer des perturbations au niveau des paramètres

de position des objets du modèle.

3.5.3 Interface graphique:

L'interface graphique contrôle les opérations de visualisation et gère l'interaction avec l'utilisateur. Il construit les images à partir d'un modèle polyédrique des objets, et d'un modèle de visualisation comportant les informations suivantes:

- Une description de *l'éclairage* de la scène.
- Une représentation de *l'aspect visuel* des objets.
- Une caractérisation des *paramètres d'observation.*

Les données liées à l'éclairage et à l'aspect visuel sont pour la plupart traitées directement par le terminal de synthèse d'image. Ces données sont généralement stockées dans le modèle. Elles peuvent être modifiées à l'aide de certaines commandes du système.

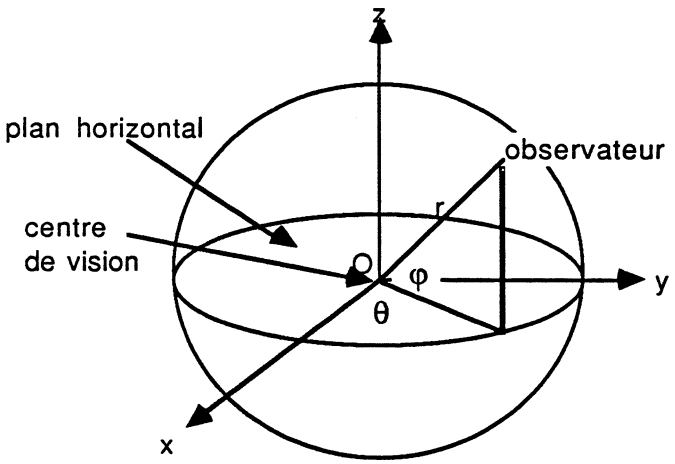


Figure 3.5: Représentation des paramètres d'observation.

Les paramètres d'observation sont utilisés par le simulateur pour déterminer les constituants de la scène à visualiser, et pour contrôler les mouvements d'une caméra virtuelle. Afin d'avoir un mode de représentation naturel et synthétique de ces paramètres, le simulateur utilise un modèle basé sur le concept de *sphère*

de vision implanté dans le système LISP-3D [Laugier 82]. Le principe consiste à caractériser l'espace observable par une sphère sur laquelle est positionnée une caméra mobile (cf. figure 3.5). La latitude, la longitude et la distance focale de cette caméra peuvent être modifiées aisément par l'opérateur à l'aide d'un ensemble de commandes appropriées. La méthode utilisée conduit à une description rapide et claire des modifications du point de vue. Elle permet également d'engendrer aisément des mouvements de caméra destinés à créer des effets spéciaux tels que des travellings, des panoramiques ou des suivis de corps mobiles. Il suffit, par exemple de fixer le "centre de vision" sur un objet en mouvement, pour contraindre la caméra à suivre automatiquement cet objet au cours de ses évolutions. De même, l'opérateur peut appliquer un zoom centré sur la pince du robot de manière à maintenir au centre de l'écran une vue détaillée de cette pince, quels que soient les mouvements exécutés par le robot.

La figure 3.6 montre le type de graphisme utilisé par le système et le type d'image produite. Une exemple de programme LM incluant des commandes de simulation est donné en annexe. Il montre différentes utilisations de ces commandes. Nous renvoyons le lecteur intéressé à [Laugier et al. 84], où il pourra trouver une description détaillée de ces commandes.

3.5.4 Emulation du temps:

Le module d'émulation du temps synchronise toutes les opérations réalisées par le simulateur. Il utilise pour cela deux bases de temps différentes associées respectivement aux opérations de mise à jour du modèle de l'univers et aux opérations de visualisation. Ces bases de temps sont synchronisées avec l'horloge interne de l'interpréteur du langage de commande de robot, par le biais des consignes de position adressées au simulateur. Si v_0 représente la vitesse d'émission de ces consignes, les vitesses v_1 et v_2 associées aux opérations de mise à jour du modèle et de visualisation sont définies comme des sous-multiples de v_0. Ainsi, l'exécution d'une instruction de déplacement du programme de manipulation conduit à répéter les opérations précédentes à des vitesses respectives de $v_1 = v_0/n$ et de $v_2 = v_1/p$; n et p représentent des paramètres modifiables par des commandes de simulation.

Cette approche permet de conserver un modèle cohérent avec la réalité, quel que soit le rythme de visualisation choisi. Elle permet également d'interrompre à tout moment le processus de simulation graphique, et de le réactiver ultérieurement sans perte d'information. Dans ce cas, le temps simulé est "suspendu", et le modèle de l'univers est mis à jour après chaque opération du robot. La durée de l'interruption est soit indéterminée (mode interactif), soit fixée par une commande de simulation.

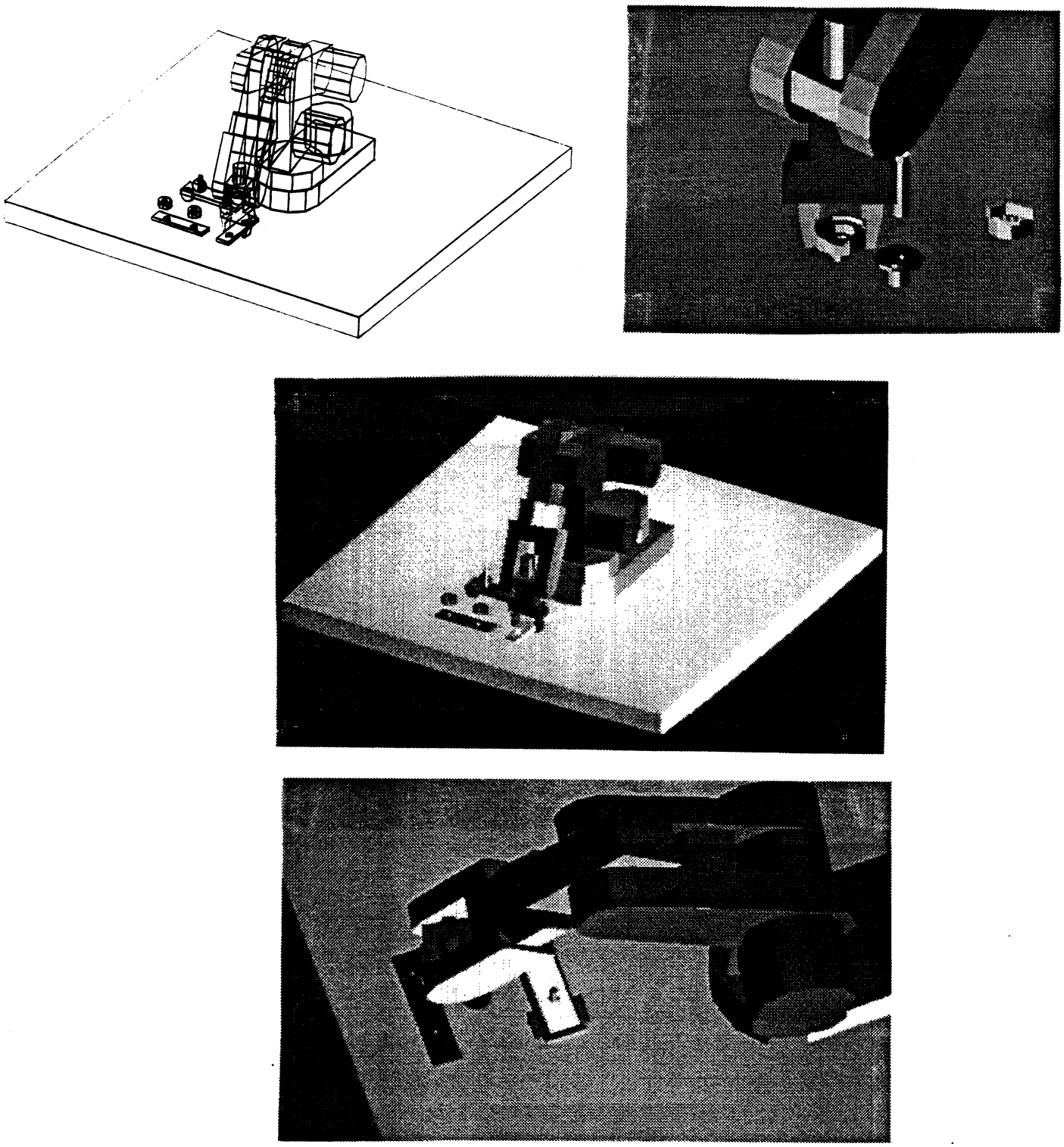


Figure 3.6: Exemples de sorties graphiques produites par le système.

3.6 Simulation de l'univers physique:

3.6.1 Les événements physiques:

Trois aspects de l'univers physique sont à considérer afin d'obtenir une simulation aussi réaliste que possible: les événements aléatoires, les collisions et les ruptures d'équilibres. Dans l'implantation actuelle du système, seuls les deux premiers aspects sont pris en compte par le simulateur. Les ruptures d'équilibres provoquées par les actions conjointes du robot et de la gravité n'ont été traitées que de manière partielle par un logiciel expérimental.

Les aléas de l'univers physique sont simulés au moyen de "perturbations" appliquées aux objets du modèle. Ces perturbations portent sur la position des objets. Elles sont engendrées par une fonction aléatoire évaluée en réponse à des commandes de simulation. Chaque commande spécifie le nom d'un objet et une valeur maximale pour la perturbation. Dans le cas d'une perturbation liée à une incertitude de position, la transformation maximale représente le "volume d'incertitude" associé à l'objet. Dans les autres cas, elle est donnée de manière arbitraire par l'opérateur. Une perturbation peut être indépendamment appliquée à un objet de l'environnement ou à un élément du robot.

La détection des collisions est réalisée par une fonction géométrique basée sur un calcul d'interférence [Boyse 79]. Cette fonction peut être associée dynamiquement à un *capteur de contact virtuel*, par une commande de simulation indiquant une liste d'objets à "surveiller". Durant toute la durée de vie de ce capteur, un calcul d'interférence portant sur ces objets est effectué après chaque opération de mise à jour du modèle. Plusieurs capteurs virtuels peuvent être actifs simultanément. Ils peuvent être supprimés séparément par des commandes de simulation.

La détection d'une interférence conduit à interrompre le processus de simulation. Dans le cas où cette interférence correspond à une véritable collision, l'opérateur peut relancer la simulation après avoir stoppé le mouvement en cours. Dans le cas d'un contact compliant, il peut relancer la simulation en annihilant le capteur (le mouvement est alors pousuivi).

Les deux fonctions précédentes permettent d'étendre dans une large mesure le domaine d'application de la simulation. Toutefois, Elles apportent un degré de réalisme insuffisant pour certaines applications. En particulier, elles ne possèdent pas de modèle de frottement ni de modèle de la gravité. Elles ne permettent pas de ce fait, de réaliser la simulation des mouvements compliants.

3.6.2 La perception sensorielle:

L'utilisation d'une donnée sensorielle dans le programme d'application conduit à adresser une commande au module de simulation des capteurs. Cette commande provoque l'évaluation d'une *fonction capteur* appropriée.

Certaines fonctions capteurs sont autonomes et ne nécessitent pas d'intervention de la part de l'opérateur. *Un capteur de vision 2D,* utilisé pour localiser des pièces, peut par exemple être simulé par une simple lecture du modèle suivie d'une légère modification aléatoire de la position lue (cette modification simule l'imprécision du capteur réel). *Des capteurs de contact et de proximité* sont aisément simulés à l'aide du mécanisme de détection des collisions décrit dans le paragraphe précédent.

D'autres fonctions capteurs nécessitent une intervention de l'opérateur. Elles utilisent des données stockées dans une zone réservée du modèle. Ces données sont initialisées au début de la simulation; elles peuvent être modifiées à tout moment par des commandes de simulation. Une fonction simulant un *capteur de force* lira par exemple des données qui représentent un torseur de forces exprimé dans un repère particulier. Ces données peuvent être entrées par l'utilisateur après une interruption du processus de simulation provoquée par le module de détection des collisions. Elles peuvent également être directement fournies par le véritable capteur lors d'une exécution conduite simultanément avec le robot réel. Ce mode opératoire est utilisé pour la construction de stratégies de manipulation basées sur l'exploitation de données sensorielles. En particulier, il permet d'acquérir des informations sur le comportement du robot, lorsque les incertitudes de positions atteignent un niveau trop élevé par rapport à la précision requise par la tâche. Ce point est développé dans le chapitre 4.

3.7 La programmation graphique:

3.7.1 Principe de base:

Les facilités de programmation par l'exemple implantées dans un langage symbolique peuvent être utilisées dans l'environnement d'un simulateur graphique. Un premier mode opératoire consiste à contrôler individuellement chaque articulation du robot à l'aide des touches de fonction du terminal graphique. Ce mode opératoire, qui est équivalent à celui utilisé sur le robot réel, conduit à des difficultés qui sont dues à la nature bidimensionnelle des scènes observées sur l'écran.

Une autre manière de procéder, consiste à remplacer le boîtier de télécommande par un *langage graphique* très simple, opérant sur une base de modèles géométriques. Cette approche conduit à simplifier dans une large mesure le raisonnement dans

l'espace tridimensionnel en permettant à l'utilisateur de spécifier les positions prises par le robot en termes de relations géométriques élémentaires. La quantification des actions du robot est ensuite réalisée automatiquement par le système qui engendre un programme de commande de robot exprimé dans un langage de manipulation existant.

Ce mode de programmation conduit à décrire symboliquement les déplacements et les actions de l'outil terminal du robot, en indiquant *graphiquement* les positions à atteindre et les contraintes de trajectoire à respecter (points de passage, type de trajectoires). Chaque position clé est alors spécifiée par une ou plusieurs relation(s) géométrique(s), mettant en jeu des éléments de l'environnement et des éléments du robot. Ces relations sont décrites par l'utilisateur au moyen des mécanismes d'interaction graphique. Elles sont toutes basées sur le concept de "mise en coïncidence" d'entités géométriques [Borrel et al. 83]. Elles sont représentées par des constructions symboliques du type:

AMENER l'entité A du robot SUR l'entité B de l'objet O

Un exemple de ce type de construction est présenté dans la figure 3.7. Le mouvement défini consiste à placer le goujon au dessus du cylindre, de manière à ce que les axes de révolution des deux objets soient alignés. La relation décrite graphiquement est alors la suivante:

$$(axe\text{-}p\ SUR\ axe\text{-}c) \wedge (P\ SUR\ C)$$

Ce mode de représentation des tâches de manipulation permet de conserver une certaine flexibilité vis-à-vis des modifications du contexte d'exécution. Cette propriété est due au fait que l'évaluation des variables de position et des transformations géométriques associées, n'est réalisée qu'au moment de la simulation ou de la production de code pour le robot. Ainsi, si des objets ont été déplacés entre deux exécutions simulées de la tâche, la description graphique demeure valide dans certaines limites (accessibilité en particulier), bien que les programmes de manipulation engendrés diffèrent sensiblement dans leurs spécifications des repères et des transformations.

3.7.2 Interprétation des relations géométriques:

En programmation graphique, toutes les actions du robot sont caractérisées par des relations géométriques complétées par des contraintes de mouvement. Chaque relation décrite est immédiatement interprétée par le système. Elle conduit à déplacer graphiquement l'entité associée à l'outil terminal du robot, de manière à l'amener en coïncidence avec l'entité associée à l'environnement. Lorsqu'aucune

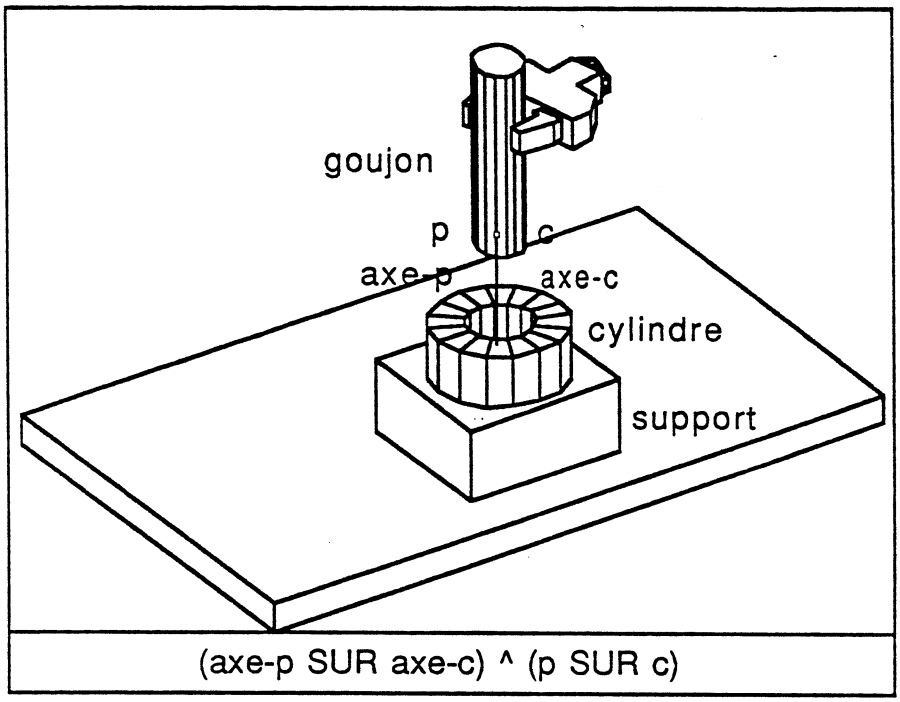


Figure 3.7: Interprétation des relations géométriques dans le système PGR:
Relations géométriques:
(axe-p SUR axe-c) $\wedge$ *(P SUR C)*
Résultat d'interprétation:
T :=< *matrice* >; DEPLACER ROBOT A ROBOT*T;

contrainte additionnelle n'est spécifiée par l'utilisateur, les paramètres du mouvement sont calculés en appliquant la règle du "déplacement minimum". Cette règle conduit à minimiser les rotations et les translations nécessaires, lors du calcul de la transformation géométrique. La trajectoire suivie par l'outil terminal est alors calculée par le générateur de trajectoire du langage, en fonction du mode de déplacement indiqué par l'utilisateur (mode libre, mode cartésien...). Une autre possibilité consiste à décrire entièrement la trajectoire à l'aide de courbes géométriques caractérisées par des ensembles de points ou par des équations paramétriques. Ce mode de spécification des trajectoires est actuellement opérationnel dans le système LM, mais la connexion avec le module de programmation graphique n'a pas encore été réalisée.

Les mécanismes d'interprétation que nous avons développés sont semblables à ceux implantés dans CATIA [Borrel et al. 83]. Cependant, les données produites se placent à des niveaux conceptuels différents: dans CATIA l'interprétation d'une sous-tâche de manipulation conduit à engendrer de *simples trajectoires* pour le robot; dans le module de programmation graphique de LM, elle conduit à ajouter de *nouvelles instructions* au programme de manipulation en cours de construction (cf. figure 3.7). Cette approche est plus générale et plus flexible que la précédente, car elle permet de combiner le pouvoir d'expression d'un langage symbolique avec la versatilité des outils CAO. Elle permet en particulier de compléter les instructions de déplacement décrites, afin d'y inclure des clauses faisant référence à des données sensorielles. Il reste cependant à résoudre les problèmes posés par les erreurs introduites par les modèles de simulation. Ce point sera abordé dans le chapitre 4.

3.7.3 Modèles nécessaires:

Les modèles requis par la programmation graphique sont constitués d'entités géométriques simples de type point, droite ou plan. Ces entités sont rattachées aux modèles CAO qui représentent l'outil terminal du robot et les objets de l'environnement. Elles peuvent être créées ou supprimées par des commandes du système. L'opérateur peut par exemple définir une nouvelle droite dans le modèle, en désignant deux points existants. Il peut aussi faire référence à des éléments caractéristiques des volumes et des surfaces représentés, en particulier: axes de révolution, centres de symétrie, plan d'une face ou support d'une arête rectiligne.

3.8 Implantation et expérimentations:

3.8.1 Architecture logicielle:

Le système décrit dans ce chapitre a été implanté en MACLISP sur un calculateur CII-HB 70. Il a ensuite été transporté en FRANZLISP sur VAX-750. L'interpréteur du langage LM avec lequel il communique par une ligne série à 9600 bauds, est localisé sur une armoire de commande ITMI. Deux robots SCEMI six axes, équipés de capteurs de forces, sont pilotés par cette armoire. La simulation graphique est réalisée sur un terminal de type TEKTRONIX ou sur une station de synthèse d'images GETRIS.

L'architecture logicielle sur laquelle repose le système est décrite dans la figure 3.2. Elle a été construite autour du système de modélisation SMGR [Troccaz 85] et d'un simulateur composé de trois modules:

- Le système ISR [Laugier, Pertin 84] [Laugier et al. 84], permettant de simuler l'exécution de programmes écrits en langage LM.

- Le système LISP-3D [Laugier 82], permettant de visualiser les scènes tridimensionnelles obtenues.
- Le système PGR [Lespinard 85], permettant de programmer graphiquement le robot.

3.8.2 Le système ISR:

L'interface entre les systèmes ISR et LM a été réalisé au niveau d'un module spécialisé de l'interpréteur LM [Miribel 84]. Les commandes qui transitent par cette interface sont sémantiquement équivalentes aux consignes envoyées aux contrôleurs du robot réel. Cette structure fonctionnelle permet d'exécuter indifféremment un même programme LM sur un robot réel ou sur un robot simulé. *Les commandes de simulation* qui sont insérées dans le programme sont envoyées par l'interpréteur lors de l'exécution d'instructions standard de LM de la forme:

ECRIRE <commande de simulation> DANS SIMUL

où *SIMUL* désigne un canal d'entrée-sortie réservé à la communication avec le simulateur. Dans le cas où ce dernier n'est pas connecté, toutes les commandes qui lui sont adressées sont enregistrées dans un fichier. Lorsque l'opérateur travaille en mode interactif, les commandes de simulation peuvent être directement entrées au moyen des fonctions graphiques (menus, clavier de fonctions, tablette graphique).

3.8.3 Le système PGR:

Le système PGR a été implanté comme une extension du simulateur ISR. Il utilise l'architecture précédente pour communiquer avec le robot. Cette architecture a cependant été complétée, afin de pouvoir transmettre des commandes de mouvement ou de perception immédiatemment exécutables par le robot réel. Deux modes opératoires ont ainsi été mis en œuvre: un mode de programmation "hors-ligne" et un mode de programmation "en-ligne".

Dans le mode de programmation hors-ligne, l'opérateur travaille uniquement avec les informations fournies par le simulateur. Les tâches décrites graphiquement sont enregistrées à l'aide du formalisme décrit dans le paragraphe 3.7. Elles donnent lieu à la production de programmes de manipulation exprimés en langage LM. Ces programmes sont ensuite compilés, afin de produire un code optimisé, directement exécutable sur l'armoire de commande des robots.

Dans le mode de programmation en-ligne, l'opérateur pilote directement le robot réel à l'aide des fonctions graphiques du système. Les commandes engendrées sont codées sous la forme de fonctions LISP. Ces fonctions sont sémantiquement équivalentes aux instructions de base de LM: mouvements du robot, actions de l'outil, lecture de données de capteurs, synchronisation. Par exemple, la séquence:

(DEP-LIBRE TRANSFORMATION) (ATTENTE)

conduit à lancer un déplacement en mode libre avec attente du signal de fin de mouvement. Cette approche a été motivée par le fait que PGR n'utilise pour la commande du robot qu'un petit sous-ensemble de LM, excluant les aspects syntaxiques, algorithmiques et structures de données. En outre, ce type de connexion LISP-LM doit être vu comme un noyau de communication, permettant d'interfacer l'armoire de commande du robot supportant l'exécutif LM, avec des outils de programmation de haut niveau opérant sur des modèles géométriques "complets". Nous verrons dans le chapitre 4, que ce mode de connexion est nécessaire pour pouvoir construire des stratégies de mouvements fins à partir d'expérimentations en ligne.

3.8.4 Expérimentations:

La connexion CAO-robot que nous avons décrite a été expérimentée sur des assemblages simples, c'est à dire sur des assemblage qui peuvent être exécutés sans l'apport de données sensorielles. Le montage d'un jouet en bois composé de cinq pièces a ainsi été réalisé. Le programme de manipulation était écrit en langage LM. Il incluait des commandes de simulation qui ont permis de produire un film d'animation de 4mn (temps nécessaire à l'assemblage du jouet par le robot). Une particularité de programme est d'être exécutable sur le robot réel ou sur le robot simulé, sans qu'il soit nécessaire d'y apporter la moindre modification. Une séquence de ce programme est donnée en annexe. Elle est complétée par des photos et des images de synthèse qui illustrent les exécutions réelles et simulées.

Quelques tâches de montage simples ont aussi été programmées à l'aide des fonctions graphiques du système. Un exemple de programme LM construit de cette manière est donné en annexe. Il permet de réaliser une opération de saisie et d'insertion non sensible aux effets des incertitudes. D'autres montages plus complexes ont été tentés, mais la difficulté de traitement des incertitudes a presque toujours conduit à des échecs. C'est pourquoi nous avons développé un langage géométrique et des fonctions "d'apprentissage", pour pouvoir traiter les opérations d'assemblage qui mettent en œuvre des mouvements fins. Cet aspect est développé dans le chapitre qui suit.

Chapitre 4

La connexion CAO-Robot: Programmation géométrique et Apprentissage interactif

4.1 Problèmes posés par les incertitudes et traitements associés:

4.1.1 Le problème:

Un robot et son environnement de travail représentent un système mécanique souvent très complexe. Il est généralement impossible d'en avoir un modèle théorique complet qui permette d'en prévoir le comportement exact. Un tel système est en particulier soumis à des imperfections et à des imprécisions, qui se traduisent par des incertitudes difficiles à appréhender. Ces incertitudes perturbent les opérations du robot. Elles peuvent ainsi faire échouer le programme construit. Afin de prendre en compte ces phénomènes, le programmeur est amené à opérer comme suit:

1. Déterminer les différentes causes d'incertitudes intervenant au cours des phases de l'assemblage.

2. Analyser les conséquences de ces incertitudes, et prévoir de quelle manière elles peuvent remettre en cause le succès des actions menées par le robot.

3. Construire un programme permettant de s'affranchir au mieux de ces incertitudes: choix de stratégies particulières, utilisation éventuelle de mouvements compliants, exploitation de données sensorielles pertinentes, tests sur les positions atteintes, ajoûts d'actions correctives, itérations.

Dans la pratique, le programmeur possède peu d'informations sur les incertitudes liées à la tâche. Il lui est donc impossible d'appliquer strictement la démarche précédente, ce qui le conduit à procéder par "essais successifs".

4.1.2 Les solutions traditionnelles:

Principe des expérimentations successives:

L'absence de modèles d'incertitudes conduit le programmeur à travailler sur la base d'une *estimation empirique* des termes d'erreurs les plus significatifs. Cette estimation reste souvent à un niveau purement qualitatif. Elle conduit le programmeur à choisir les stratégies de manipulation qui lui paraissent être les mieux adaptées au problème traité. Bien souvent, ce choix n'intègre qu'un sous-ensemble des paramètres de la tâche. Il conduit donc à des échecs divers, qui donnent lieu à des amendements successifs de la stratégie choisie. Chaque correction appliquée est réalisée sur la base d'une analyse de l'échec rencontré.

Cette technique de programmation est fondée sur l'expérience, plutôt que sur une approche scientifique. Elle reste cependant la seule possible en l'absence de modèles incluant une représentation explicite des incertitudes. Le principe de l'approche peut être résumé comme suit:

1. Proposer une stratégie simple pour le montage à réaliser.

2. Expérimenter la stratégie précédente à l'aide du robot, afin d'évaluer les effets des incertitudes sur les mouvements engendrés.

3. Amender la stratégie en cas d'échec, puis retourner en (2). Arrêter lorsque le programme construit est jugé suffisamment fiable.

C'est ce procédé que nous proposons d'appliquer dans le cadre de la connexion CAO-robot.

Utilisation de procédures spécialisées:

Une méthode parfois utilisée pour faciliter la tâche du programmeur, consiste à faire usage d'une bibliothèque de *procédures spécialisées*. Chaque procédure est

associée à une classe de montages (cf. figure 4.1). Elle est paramétrée par des valeurs qui permettent de l'adapter au cas traité. Tout le problème réside alors dans le choix de ces paramètres. Prenons par exemple le cas d'une procédure spécialisée dans la réalisation d'insertions cylindriques. Supposons que cette procédure soit paramétrée par la position initiale et la position finale du goujon, le seuil de force à ne pas dépasser, un facteur de correction latérale, un facteur de correction angulaire, un pas axial et un "centre de compliancé (point autour duquel le système doit appliquer les rotations destinées à annuler les moments). Tous ces paramètres permettent théoriquement d'adapter la procédure aux caractéristiques du montage et aux capacités du robot. Il s'avère cependant extrêmenent difficile d'estimer les valeurs à donner aux facteurs de corrections et aux seuils dans le cas d'insertions fines. En effet, si l'on donne aux facteurs de corrections des valeurs trop importantes, on provoque une divergence du procédé; si l'on abaisse de manière excessive les seuils, les simples variations des lectures du torseur de forces dues au bruit, entraînent des corrections inappropriées. De plus, il très difficile de déterminer la position du "centre de compliance", du fait que celui-ci est sensé se situer au barycentre des points de contacts; il peut suivant le cas être fixe, ou varier au cours du montage.

C'est pourquoi, même dans les cas relativement courants, le choix des valeurs à affecter aux paramètres ne peut être réalisé qu'à partir d'expérimentations relativement longues. Cette situation limite l'intérêt des procédures spécialisées. Une autre limitation est liée au fait que les stratégies implantées sont généralement très sensibles aux petites variations de la géométrie des pièces, ce qui interdit le développement de procédures suffisamment générales.

4.1.3 L'approche par apprentissage interactif:

Les modèles apportés par la connexion CAO-robot fournissent un support géométrique plus riche que celui des langages de manipulation traditionnels, mais ils ne changent pas fondamentalement le problème du traitement des incertitudes. Ce point marque toujours la limite de validité du formalisme offert par les systèmes de programmation "hors-ligne" actuels.

L'objectif de ce chapitre est de montrer de quelle manière il est possible d'étendre ce formalisme, afin de pouvoir traiter des assemblages plus complexes, c'est à dire des assemblages qui nécessitent d'appliquer des stratégies de mouvements fins (cf. paragraphe 2.1). Les outils informatiques qui sont à la base de cette extension sont les suivants:

- *Un langage géométrique* permettant de définir symboliquement les relations que le robot doit réaliser, ainsi que les contraintes d'exécution associées. La vérification de ces contraintes est alors réalisée au moyen des données

perceptives fournies par les capteurs de position et de force du robot.

- *Un système "d'apprentissage"* permettant de synthétiser un programme de montage, qui intègre toutes les corrections apportées interactivement au cours de différentes tentatives d'exécutions de la tâche.

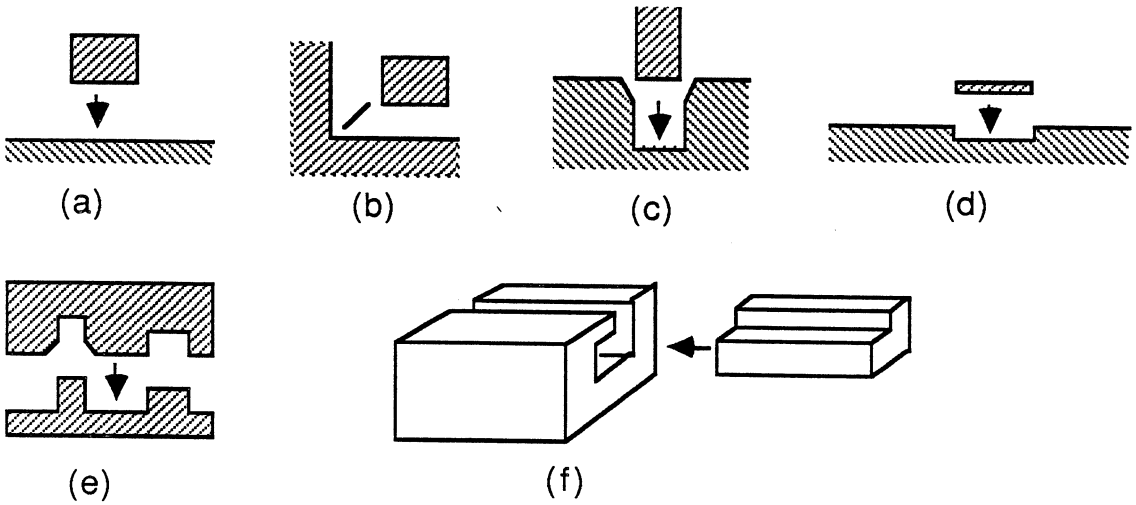


Figure 4.1: Exemple de classes de montages:
(a) Posage plan. (b) Posage d'angle. (c) Insertion cylindrique avec chanfrein.
(d) Insertion cylindrique d'une pièce de faible épaisseur. (e) Insertion double.
(f) Insertion prismatique.

4.2 La programmation géométrique:

4.2.1 Principes de base:

La programmation géométrique peut être vue comme une extension de la programmation graphique, dans le sens où elle repose sur le même concept de "mise en coïncidence" d'entités géométriques. Le langage sur lequel elle s'appuie offre cependant des constructions plus complètes, qui permettent de décrire un mouvement en termes des *conditions d'arrêt* et des *contraintes d'exécution* qui lui sont attachées. Toutes les expressions utilisées pour cette description sont exprimées

en termes de *relations géométriques* et de *contacts.*

On spécifiera par exemple que l'objectif à atteindre est une position de l'espace représentée par la relation *"axe-A SUR axe-B"*. Les degrés de liberté laissés libres par cette relation sont traités comme dans le cas de la programmation graphique (cf. paragraphe 3.7); la condition d'arrêt associée au mouvement, porte alors uniquement sur la réalisation de la position calculée. De manière similaire, on décrira un mouvement destiné à réaliser un contact plan par une relation du type: *"face-A SUR face-B"*. La condition d'arrêt associée portera alors sur les forces engendrées par ce contact. Dans le cas où ce mouvement doit être exécuté en maintenant un autre contact C, cette nouvelle contrainte sera spécifiée par une clause du type: *"maintenir C"*.

Sur le plan purement fonctionnel, cette approche n'apporte rien de plus que les langages traditionnels de programmation de robots. Elle se place par contre à un niveau conceptuel plus élevé, qui présente les principaux avantages suivants:

- Le formalisme mis en oe uvre permet d'exploiter des bases de données CAO.
- Le mode d'expression induit est à la fois plus homogène et plus facile à utiliser. Un contact est par exemple défini en termes d'objets, plutôt qu'en termes des forces qu'il engendre.
- Chaque instruction possède un objectif clair, ce qui n'est pas toujours le cas avec les langages de manipulation traditionnels. Par exemple, une instruction du type:

  DEPLACER <repère> A <position> JUSQUA <garde>

  peut définir un mouvement destiné à atteindre la position *<position>*; elle peut aussi spécifier un mouvement dont l'objectif est de réaliser un contact. Dans le premier cas, la garde indique simplement que des contacts parasites peuvent se produire par suite des incertitudes. Dans l'autre cas, elle représente la condition qui sera vérifiée lors de la réalisation de contact recherché. Ce type d'ambiguïté est sans réelle conséquence sur l'exécution de la tâche, lorsque le programme est bien construit. Il constitue par contre un obstacle à l'application des fonctions d'analyse nécessaire à l'apprentissage interactif.

4.2.2 Spécification des mouvements:

Les instructions du langage sont spécifiées à l'aide d'entités géométriques appartenant au modèle de l'univers du robot. Ces entités peuvent représenter des données purement géométriques, par exemple: repères, axes de symétries, droites; elles sont

alors appelées *entités virtuelles*. Elles peuvent aussi correspondre à des éléments de la surface des objets, par exemple: faces, arêtes ou sommets; elles sont alors appelées *entités physiques*.

Les entités virtuelles sont utilisées pour décrire des positions dans l'espace libre. Les autres permettent de spécifier des relations qui mettent en jeu des contacts. La forme générale d'une instruction de mouvement est alors la suivante:

REALISER <situation> AVEC <contrainte> JUSQUA <garde>

où *<situation>* représente une situation géométrique décrite au moyen de la relation de "mise en coïncidence", *<contrainte>* est une liste de contraintes à respecter lors de l'exécution du mouvement, et *<garde>* définit des conditions booléennes susceptibles de stopper le mouvement. Les significations de ces paramètres sont décrites dans les paragraphes qui suivent.

Les termes *<situation>* et *<garde>* définissent les états dans lesquels le robot peut achever son mouvement. Ils sont traduits en *conditions d'arrêt*, c'est à dire en conditions booléennes portant sur des données de position et/ou de force mesurées par les capteurs du robot. Dans le cas d'un mouvement visant à atteindre une position de l'espace libre, la condition évaluée teste la réalisation de cette position. Dans le cas d'un mouvement s'achevant sur un contact, la condition évaluée porte aussi sur les forces engendrées par ce contact. L'expression correspondante est alors automatiquement calculée par le système, en fonction des caractéristiques du contact et de la direction du déplacement. Ce point est discuté dans le chapitre 6. Il a donné lieu à une implantation limitée, conduisant à tester les forces qui s'opposent au déplacement (par dépassement d'un seuil dans la direction inverse du vecteur vitesse), et à s'assurer de la nullité des moments induits par un contact plan. Les problèmes dus aux glissements et aux ambiguités d'interprétation, sont alors ignorés à ce niveau.

4.2.3 Expression des situations géométriques:

Chaque situation géométrique est définie par une expression du type:

Situation S: $(R_1 \wedge R_2 \cdots \wedge R_m)$
Relation R: $(e_a \; SUR \; e_b)$

où e_a et e_b sont des entités de type point, droite, plan, repère, sommet, arête ou face.

Lorsque les entités e_a et e_b sont de nature physique, S représente une situation de contact. La réalisation de cet objectif est alors contrôlée par une condition

d'arrêt, exprimée comme indiqué précédemment.

Lorsque e_a et e_b sont des entités virtuelles, l'objectif à atteindre est purement géométrique. Sa réalisation est alors soumise à l'application d'une transformation géométrique T_{ab}, conduisant à amener en coïncidence un repère R_a du mobile (outil terminal ou objet tenu par le robot), avec un repère R_b lié à un objet de l'environnement:

- Si e_a et e_b sont des repères:
  $T_{ab} = Transformation(e_a, e_b)$.
  $R_a = e_a$ et $R_b = e_b$.
- Si e_a et e_b sont des entités de type point, droite ou plan:
  T_{ab} représente la transformation "minimale" qui permet de réaliser la mise en coïncidence des entités (cf. paragraphe 3.7). Lorsque la situation comporte plusieurs relations, la transformation globale est obtenue par composition des transformations associées aux différentes relations.
  R_a = repère de référence de l'objet A.
  R_b = repère lié à l'objet B, tel que $R_b = R_a * T_{ab}$.

4.2.4 Expression des contraintes d'exécution:

Deux types de contraintes peuvent être imposés lors de l'exécution d'un mouvement: des contraintes de trajectoire et des contraintes de contact.

Contraintes de trajectoire:

Les contraintes de trajectoire traitées par le système excluent les aspects cinématiques. Cette limitation est uniquement dictée par les objectifs que nous nous sommes fixés. Elle ne repose sur aucune difficulté réelle (l'interpréteur LM qui pilote le robot comporte tous les outils nécessaires). Les contraintes prises en compte par le langage permettent alors d'indiquer des points de passage obligatoires, de spécifier des directions de déplacement à respecter, ou d'imposer une amplitude pour le mouvement:

Contrainte de type 1:	$(VIA\ S_1, S_2 \cdots S_m)$
Contrainte de type 2:	$(SUIVANT\ d)$
Contrainte de type 3:	$(AMPLITUDE\ a)$

où S_i, $i = 1 \cdots m$, est une situation géométrique définie comme précédemment, et d est une direction de translation ou un axe de rotation.

Les contraintes de type 1 sont utilisées pour spécifier des grands mouvements. Elles peuvent aussi être exprimées en coordonnées articulaires, lorsque les trajectoires spécifiées ont été calculées par un planificateur de mouvements(cf. chapitre 9). Les contraintes de type 2 permettent d'imposer certaines directions de

déplacement, lorsque l'objectif décrit laisse plusieurs possibilités au système. C'est par exemple le cas d'une relation de contact définie par une expression du type *"face-A SUR face-B"*. La direction imposée peut alors être choisie perpendiculairement à la face *face-B*. Les contraintes de type 3 sont utilisées pour spécifier l'amplitude d'un mouvement, lorsque celle-ci ne peut être déduite des autres paramètres. Ce type de construction est employé pour définir des mouvements correctifs incrémentaux, ou pour indiquer l'écartement des mors lors d'une opération de saisie/lâcher d'objet.

Contraintes de contact:

Les contraintes de contact indiquent les contacts à maintenir tout au long de l'exécution du mouvement. Elles sont exprimées comme suit:

$$(MAINTENIR\ R_1 \wedge R_2 \cdots \wedge R_m)$$

où R_k, $k = 1 \cdots m$, est une relation de contact de type "$e_j\ SUR\ e_j$".

Le contrôle des forces et des mouvements correctifs liés à ce type de mouvement est réalisé par l'interpréteur du langage. Dans notre implantation, cet interpréteur a été écrit à l'aide des fonctions compliantes de LM [Gandon 86].

4.2.5 Expression des conditions de garde:

Les conditions de garde permettent de stopper le mouvement, lorsque l'une des conditions booléennes surveillées est évaluée à vrai. Dans la version actuelle du langage, ces conditions portent uniquement sur des contacts. Elles sont exprimées par des constructions du type:

$$(GARDE\ R_1 \vee R_2 \cdots \vee R_m)$$

où R_i, $i = 1 \cdots m$, représente une relation de contact de type "$e_i\ SUR\ e_j$".

Lorsqu'une garde est associée à un mouvement d'objectif S, la condition d'arrêt évaluée par le système est définie par la disjonction suivante:

Condition d'arrêt: $(C_0 \vee C_1 \vee C_2 \cdots \vee C_m)$

où C_0 est la condition induite par la situation S, et C_1, $C_2 \cdots C_m$ représentent les conditions associées aux contacts contenus dans la clause de garde. La présence de ces contacts hypothétiques dans l'expression du mouvement, vient de ce que les incertitudes ne permettent pas de garantir que le mouvement commandé se terminera sur l'objectif S.

D'autres clauses de garde incluant des conditions temporelles ou des conditions de force/pression, peuvent très facilement être rajoutées à l'aide des outils dont nous disposons. Par exemple, une condition du type:

$(EFFORT\ cond_1 \wedge cond_2 \cdots \wedge cond_m)$
avec:
$cond_i = (X > a)$
$X = F_x, F_y, F_z, M_x, M_y, M_z$

pourra être immédiatement traduite en une clause *JUSQUA* du langage LM.

4.2.6 Instructions de manipulation utilisées:

Les instructions de manipulation du langage géométrique sont toutes basées sur le formalisme décrit ci-dessus. Elles ont été exprimées dans une syntaxe de type LISP, pour des raisons de facilité d'implantation et d'homogénéité avec les autres fonctions du système (simulateur, interface LISP-LM et module d'apprentissage).

Quatre catégories d'instructions sont distinguées en fonction de la nature des opérations exécutées: réalisation d'une relation géométrique, création ou suppression d'un contact, correction incrémentale portant sur une relation géométrique, et réalisation d'une opération de saisie/lâcher. Ces instructions sont notées respectivement *REALISER-S*, *REALISER-C*, *CORRIGER*, *SAISIR* et *LACHER*. Elles sont paramétrées comme indiqué précédemment, par exemple:

- Une instruction permettant de réaliser un mouvement de transfert respectant une contrainte de trajectoire, aura la forme suivante:

  (REALISER-S <situation> (VIA <chemin>))

- Une instruction permettant de réaliser un mouvement fin conservant m contacts, et se terminant par la création d'un nouveau contact, sera spécifiée comme suit:

  (REALISER-C <contact> (SUIVANT d)
  (MAINTENIR <liste-contacts>))

- Une instruction permettant de corriger une relation géométrique réalisée antérieurement de manière imprécise, pourra être définie comme suit:

  (CORRIGER <situation> (SUIVANT d) (AMPLITUDE a))

- Une instruction de saisie contraignant l'ouverture de la pince, sera décrite comme suit:

  (SAISIR <situation> (AMPLITUDE a))

La clause de garde est utilisée lorsque la réalisation de l'objectif spécifié dans l'instruction, est rendue hypothétique par les incertitudes courantes. Des exemples de constructions de ce type sont donnés dans les paragraphes qui suivent.

Les opérations de correction sont destinées à réduire certaines valeurs d'incertitude. Elles font pour cela appel à des données perceptives, telles que les positions et les forces mesurées sur les capteurs du robot. Comme dans LM, ces données sont transmises au programme par l'intermédiaire de variables spécialisées. Le programmeur pourra ainsi appliquer des corrections suivant des directions définies par le torseur de force. Il pourra également en fixer l'amplitude, en fonction de la position précédemment atteinte par le robot. Des exemples de constructions de ce type sont donnés dans la suite.

4.2.7 Structure de contrôle:

La structure de contrôle d'un programme de manipulation est de nature essentiellement linéaire. Certaines parties du programme peuvent cependant comporter des constructions non linéaires, définies dans le but de s'affranchir des effets des incertitudes. Ces constructions représentent des séquences conditionnelles et des itérations. Elles sont basées sur l'utilisation de fonctions LISP classiques.

Les structures conditionnelles sont utilisées en relation avec les instructions qui comportent une clause de garde. Les conditions évaluées par le système sont alors celles qui dérivent de la description du mouvement (situation à réaliser et termes de la garde). Elles sont définies dans le programme par les mêmes constructions symboliques. Par exemple, une stratégie de correction axiale utilisée lors d'une insertion cylindrique, peut être définie comme suit:

```
(REALISER-S <insertion> (GARDE <chanfrein>))
    (COND ((<insertion>) NIL)
          ((<chanfrein>) (CORRIGER <aligne-axe>
                         (SUIVANT (Fx Fy 0))
                         (AMPLITUDE I/2))))
```

où *<insertion>*, *<chanfrein>* et *<aligne-axe>* représentent respectivement la relation à réaliser, un contact avec le chanfrein, et une relation d'alignement d'axes verticaux (ceux du goujon et du chanfrein). La direction (F_x F_y 0) est définie à partir des composantes horizontales du vecteur de force. Le paramètre I représente l'erreur maximum associée à la relation d'alignement des axes. Ce paramètre n'est pas géré automatiquement par le système actuel.

Les structures itératives sont utilisées en relation avec l'instruction *CORRIGER*. Elles permettent d'appliquer itérativement une stratégie de correction, jusqu'à ce que les valeurs d'incertitudes obtenues soient compatibles avec les impératifs de

la tâche. Dans la version actuelle du langage, il n'est pas possible d'exprimer la condition d'itération en termes d'incertitudes (celles-ci n'étant pas traitées explicitement par le système). Cette condition est alors uniquement définie en termes de relations géométriques et de contacts, ce qui ne permet pas de vérifier la convergence du procédé. Ce travail est à la charge du programmeur.
La stratégie globale associée à l'insertion cylindrique de l'exemple précédent, pourra alors être définie comme suit:

(WHILE (((not <insertion>)) (SC)))

où SC représente la stratégie corrective décrite dans l'exemple précédent.

4.3 L'apprentissage interactif:

4.3.1 Principe de base et motivations:

L'approche par apprentissage interactif que nous avons développée, est basée sur les travaux de Dufay [Dufay 83] [Dufay, Latombe 84]. Elle consiste à synthétiser un programme incluant des mouvements fins, en combinant les résultats de plusieurs tentatives d'exécution de la tâche. Elle opère pour cela en deux phases: une phase d'expérimentation et une phase d'induction (cf. figure 4.2).

Dans la phase d'expérimentation, le programmeur réalise plusieurs exécutions de la tâche en connexion avec le robot. Il utilise pour cela les outils graphiques du système, et le langage géométrique décrit précédemment. Il peut également faire usage d'une bibliothèque de stratégies élémentaires, pour corriger les erreurs d'exécution rencontrées. Il peut par exemple appliquer une stratégie de recentrage basée sur une recherche de deux contacts opposés, pour corriger la position de la pièce tenue par le robot. Tous les choix réalisés lors de cette phase d'essai, sont à la charge du programmeur. Ils reposent sur une estimation empirique des situations d'échec rencontrées. Cette estimation est alors faite sur la base des informations fournies par le système: contacts, positions et forces. Chaque expérimentation jugée significative par le programmeur est transformée en une "trace d'exécution", c'est à dire en un graphe linéaire. Ce graphe représente les séquences alternées de mouvements et de situations symboliques rencontrées.

La phase d'induction a pour objet de combiner les résultats des différents essais. Elle procède pour cela à des transformations itératives du graphe obtenu par combinaison des traces d'exécutions. L'opération de base consiste alors à fusionner tous les noe uds et tous les arcs "équivalents". Le graphe obtenu en résultat représente la structure de contrôle du programme de montage à synthétiser. Les noe uds de ce graphe définissent les instructions de manipulation du programme.

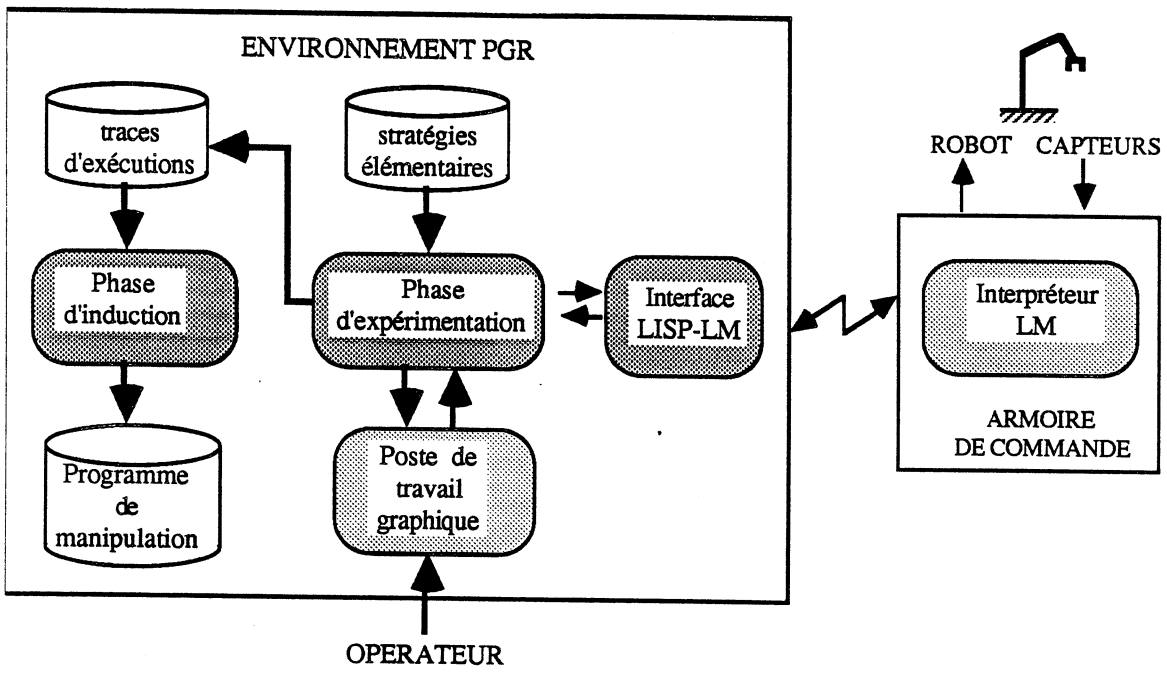


Figure 4.2: Schéma synoptique du module d'apprentissage interactif.

La méthode que nous avons développée est très proche de celle de Dufay [Dufay 83]. Elle se distingue de celle-ci sur trois points:

- Notre système n'utilise pas de connaissances expertes pour engendrer les séquences correctives. Ce travail repose sur l'opérateur, ce qui donne un plus grand pouvoir d'expression (le système n'étant pas limité par la base de connaissance). Il perd par contre l'aspect automatique.
- Notre système opère sur un modèle géométrique complet, alors que celui développé par Dufay était purement déclaratif. L'utilisation de ce modèle procure au système un degré de généralité plus élevé, dans le sens où il permet de représenter et d'analyser une infinité de situations de contacts (l'autre modèle nécessitant de donner une liste exhaustive des différentes variantes possibles).
- Notre système synthétise des programmes exprimés dans le langage géométrique que nous avons développé, ce qui permet de conserver une plus grande

indépendance vis-à-vis des variations mineures de l'environnement (cf. paragraphe 3.7). Comme dans le cas de la programmation graphique, ces programmes peuvent en dernier lieu être ré-écrits en LM, afin d'être compilés.

Le choix de l'approche par "apprentissage interactif", a été motivé par trois raisons:

- Cette approche est basée sur la technique des *expérimentations successives* décrite dans le paragraphe 4.1. Elle ne nécessite donc pas de travailler à partir d'une estimation précise des incertitudes, ni de prévoir toutes les situations qui peuvent survenir au cours du montage. Or cette situation est celle que l'on rencontre le plus couramment.

- Chaque correction apportée au programme en vue de traiter un échec d'exécution, est basée sur une analyse locale de la situation rencontrée. Elle conduit donc à appliquer une *stratégie locale*, pour tenter de se ramener à la situation initialement prévue. Cette approche est beaucoup plus facile à appliquer, qu'une approche basée sur une remise en cause globale du programme (cette dernière technique n'étant alors considérée qu'en dernier lieu).

- La *synthèse des informations* recueillies au cours des différentes expérimentations, est réalisée automatiquement par le système. Cette synthèse conduit à combiner toutes les stratégies locales qui ont été appliquées, afin de produire la solution finale.

Très peu de travaux ont été réalisés dans le domaine de l'apprentissage en robotique. Ces travaux ont presque tous été conduits dans le contexte de la robotique mobile: acquisition de schémas sensori-moteurs mettant en jeu des sensations simples et des actions élémentaires (par exemple: "si la batterie est vide, se diriger vers une source d'énergie") [Bond, Mott 81]; acquisition de méta-connaissances permettant d'améliorer le choix des actions à exécuter [Doran 70]; acquisition de spécialités par construction de macro-opérateurs [Fikes et al. 72] [Sussman 75].

Notre approche entre dans la dernière catégorie de travaux. Bien que cette approche ne relève pas réellement de l'apprentissage au sens éthymologique du terme, nous désignerons la méthode appliquée par le terme "apprentissage interactif".

4.3.2 Notion de graphe d'exécution:

L'exécution d'un programme de commande de robot peut être vue comme une succession de mouvements, conduisant chacun à un état particulier de l'ensemble

robot-environnement. Ces états représentent des *situations symboliques*, identifiables au moyen des données manipulées par le système (données géométriques et informations sensorielles). Le programme peut alors être représenté par un graphe, dans lequel chaque nœud définit une situation symbolique, et chaque arc symbolise un mouvement du robot.

Lorsque les incertitudes sont négligeables par rapport aux contraintes de précision imposées par la tâche, l'exécution du programme donne invariablement la même séquence alternée de mouvements et de situations. Ces dernières n'ont alors aucun rôle à jouer dans le séquencement des opérations, et le graphe induit est purement linéaire.

Dans le cas contraire, plusieurs schémas d'exécution sont possibles. Ceci provient de ce que les incertitudes présentes, ne permettent plus de garantir la réalisation de certains objectifs intermédiaires. Par exemple, le goujon pourra heurter le chanfrein lors de l'exécution d'un mouvement d'insertion. Les situations symboliques représentent alors des points de contrôle pour le programme. Il devient donc nécessaire de pouvoir les "identifier" à l'aide des données contenues dans le programme.

Nous appelons *graphe d'exécution* associé à un programme de mouvements fins, le graphe qui représente l'ensemble des exécutions possibles du programme. Ce graphe définit explicitement la structure de contrôle du programme. Il peut comporter des arcs qui engendrent des itérations, c'est à dire des mouvements qui ramènent le robot dans des situations antérieures. Il peut également comporter des nœuds associés à plusieurs arcs. Les arcs qui partent d'un tel nœud, représentent les différentes exécutions possibles d'un mouvement gardé. Ce type de construction est illustré par la figure 4.3.

Le rôle du processus d'apprentissage est de construire le graphe d'exécution de la stratégie recherchée, en s'appuyant pour cela sur un ensemble d'expérimentations. La démarche appliquée est décrite dans les deux paragraphes qui suivent.

4.3.3 La phase d'expérimentation:

Principe de fonctionnement:

La phase d'expérimentation commence par la donnée d'une séquence de mouvements, permettant théoriquement de passer de la situation initiale à la situation objectif (objets assemblés). Cette séquence représente une première ébauche de la stratégie qui permettra d'exécuter la tâche. Elle est exprimée à l'aide du langage géométrique décrit précédemment.

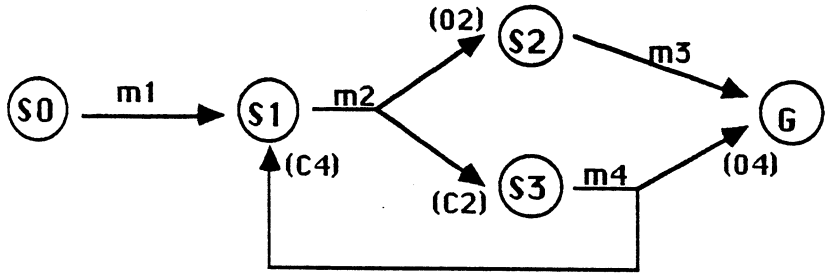


Figure 4.3: Notion de graphe d'exécution: S_0 est la situation initiale, et G est l'objectif à réaliser. Les mouvements m_2 et m_4 comportent respectivement des objectifs O_2 et O_4, et des clauses de garde C_2 et C_4.

Chaque instruction de mouvement I_i est immédiatement exécutée par le robot. Elle se traduit soit par un succès (l'objectif S_i de I_i ayant été réalisé), soit par un échec. En cas d'échec, le programmeur analyse la situation atteinte à l'issue du mouvement. Il utilise pour cela les informations fournies par le système: contacts, positions et forces. Il introduit ensuite une séquence corrective, appelée *stratégie élémentaire*, visant à ramener le robot dans la situation S_i.

Ce procédé est appliqué récursivement jusqu'à ce que la tâche soit entièrement réalisée. Il est ensuite appliqué autant de fois que nécessaire, c'est à dire jusqu'à ce que le programmeur estime que le programme construit est suffisamment fiable. Toute la difficulté réside alors dans l'interprétation des situations physiques rencontrées, et dans le choix des stratégies appropriées. Dans le contexte de l'apprentissage interactif, cette tâche repose sur le programmeur. Le rôle du système est alors de l'assister de deux manières:

- En modélisant les situations atteintes dans un formalisme facilement interprétable.
- En offrant un mode d'expression géométrique pour décrire les stratégies élémentaires.

Représentation des situations symboliques:

Les situations symboliques représentent des états caractéristiques de l'ensemble robot-environnement. Elles sont définies à l'aide des données présentes dans le

système. Dans le cas où l'on dispose d'un modèle CAO classique (c'est à dire sans représentation des incertitudes), les données utilisées sont les suivantes:

- *Des relations géométriques.* Ces relations sont celles spécifiées dans les instructions de mouvement. Elles sont exprimées dans le formalisme décrit dans le paragraphe 4.2.

- *Des relations de contact.* Ces relations sont celles spécifiées dans les situations objectifs et/ou dans les clauses de garde des instructions de mouvement. Elles sont aussi exprimées dans le formalisme du langage géométrique.

- *Des conditions sensorielles.* Ces conditions sont celles spécifiées dans certaines clauses de garde des instructions de mouvement (clause $EFFORT$). Elles portent sur les données transmises par le capteur de force. Elles ont pour objet de compléter la description précédente, lorsque celle-ci est insuffisante pour représenter les situations physiques rencontrées. Par exemple, la présence d'un couple important associé à un contact cylindrique, indiquera un cas de coincement. Cet aspect n'est pas pris en considération dans la version actuelle du système, car les clauses correspondantes n'ont pas encore été implantées dans le langage.

Interprétation des situations symboliques:

Les situations symboliques rencontrées doivent être reconnues par le système. Les données utilisées à cet effet proviennent du contexte (mouvement exécuté) et des capteurs (position/force). Lorsqu'il n'y a pas de terme de force, la relation géométrique visée est supposée avoir été atteinte. Dans le cas contraire, une analyse des contacts doit être faite en fonction du terme d'erreur maximum, et des données de position, de force et de déplacement (direction) [Laugier, Dufay 82]. Dans la version actuelle du système, cette analyse repose uniquement sur l'évaluation des conditions d'arrêt associées aux mouvements commandés. La condition d'arrêt vérifiée définit alors la relation de contact qui a été réalisée (cf. paragraphe 4.2).

Cette approche conduit à ne traiter que les contacts qui ont été explicitement prévus dans la description des mouvements. Elle repose aussi sur l'hypothèse qu'il n'y a jamais d'ambiguïté sur l'expression des conditions d'arrêt. Ce dernier point est discuté dans le chapitre 6.

Expression des stratégies élémentaires:

Une stratégie élémentaire est une courte séquence de mouvements, dont l'objectif est de réaliser une situation intermédiaire et/ou de réduire certaines composantes d'incertitude. Chaque stratégie élémentaire est exprimée dans le langage

géométrique que nous avons développé. Elle comporte une description de tous les contacts qui sont susceptibles de se produire au cours de son exécution. Cette description sert de base à l'analyse des situations symboliques rencontrées.

Les stratégies élémentaires interviennent de deux manières dans le processus d'expérimentation:

- *Définition d'une première ébauche de la stratégie de montage à construire.* La séquence correspondante représente les principales étapes à réaliser. Par exemple, un "posage d'angle" pourra être initialement défini par une séquence du type:

  (REALISER-C <contact-F1> (SUIVANT N1))
  (REALISER-C <contact-F2> (SUIVANT N2)
  (MAINTENIR <contact-F1>))

  où *<contact-F1>* et *<contact-F2>* sont les deux contacts à réaliser; $N1$ et $N2$ sont les normales extérieures aux faces supports de ces contacts. La présence de contacts susceptibles de faire échouer cette séquence, se traduira par l'introduction de clauses de garde dans l'expression des mouvements. Ces clauses définissent alors les points nécessitant d'éventuelles corrections.

- *Application d'une séquence corrective.* Les mouvements appliqués ont pour objet de réduire l'erreur portant sur la relation qui est à l'origine de l'échec. Par exemple, le fait de heurter le chanfrein lors d'une insertion cylindrique, provient d'un mauvais alignement des axes du goujon et du chanfrein. Cet échec peut alors entraîner la correction suivante:

  (REALISER-C <chanfrein> (SUIVANT (F_x F_y 0)))
  (CORRIGER <aligne-axe> (SUIVANT (F_x F_y 0))
  (AMPLITUDE $D/2$))

  où *<chanfrein>*, *<aligne-axe>*, *(F_x F_y 0)* et D représentent respectivement un contact avec le chanfrein, la relation d'alignement des axes, la direction définie par la composante horizontale du vecteur de force, et l'amplitude du mouvement précédent. Après application de cette séquence corrective, le programmeur relance le mouvement d'insertion ayant échoué. Un nouvel échec se traduira alors par l'application d'une nouvelle séquence corrective.

4.3.4 La phase d'induction:

Principe de fonctionnement:

Chaque exécution de la tâche peut être représentée par une séquence alternée de mouvements et de situations symboliques. Cette séquence est appelée *trace d'exécution.* Elle est codée sous la forme d'un graphe linéaire, comme indiqué

dans le paragraphe 4.3.2.

Le rôle de la phase d'induction est de combiner les traces d'exécution produites par la phase d'expérimentation. L'objectif visé est alors de construire un graphe d'exécution intégrant tous les essais réalisés. Ce graphe est ensuite traduit en un programme de mouvements fins, dès que le programmeur estime que le résultat obtenu est suffisamment complet.

Ces deux mécanismes de transformation de graphes et de synthèse de programmes sont décrits ci-dessous. Ils sont directement dérivés des travaux de Dufay [Dufay 83].

Représentation des traces d'exécution:

Les traces d'exécution sont représentées par des graphes linéaires, auxquels sont associées les informations suivantes:

- Un nœud représente une situation symbolique. Il inclut de ce fait les trois types de données qui permettent d'identifier algorithmiquement cette situation, cf. paragraphe 4.3.3: les relations géométriques, les contacts et les conditions sensorielles.

- Un arc symbolise une "instance de mouvement", c'est à dire une exécution particulière d'une commande de mouvement. Il comporte une description des paramètres de cette commande, cf. paragraphe 4.2: relation géométrique ou contact à réaliser, contraintes d'exécution et clauses de garde. Cette description est complétée par la spécification de la condition d'arrêt vérifiée à l'issue du mouvement.

Construction du graphe d'exécution:

Les traces d'exécution sont combinées entre elles en fusionnant les nœuds et les arcs "équivalents". La relation d'équivalence utilisée consiste à vérifier l'identité des données contenues dans les termes examinés:

- Relations géométriques, contacts et conditions sensorielles pour un couple de nœuds.

- Situations objectifs, contraintes d'exécution, clauses de garde et conditions d'arrêt vérifiées pour un couple d'arcs.

Un exemple de fusion de deux traces d'exécution est donné dans la figure 4.4. Dans cet exemple, le mouvement m_1 comporte deux instances d'exécution, représentées par les mouvements m_{1a} et m_{1b}. Ces deux instances n'ont pas été fusionnées, car elles sont caractérisées par des conditions d'arrêt différentes. Une

de ces conditions provient de la situation objectif de m_1; l'autre dérive du contact spécifié dans la clause de garde.

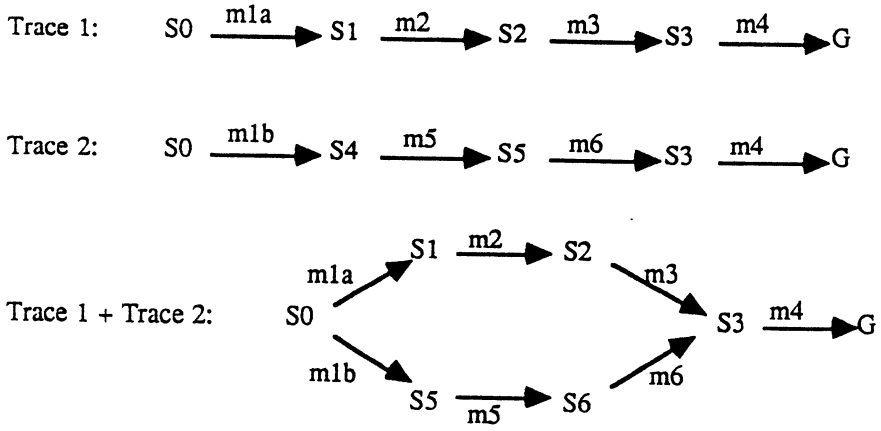


Figure 4.4: Exemple de combinaison de deux traces d'exécution.

Le processus de fusion opère par transformations successives du graphe G, obtenu par combinaison des traces d'exécutions considérées. Il applique pour cela deux règles de ré-écriture, basées sur les propriétés suivantes (cf. figure 4.5):

Propriété 4.3.1 *Si il existe un nœud s_i de G tel que les arcs a_{ij} qui partent de s_i sont équivalents, alors les arcs a_{ij} et les nœuds s_j situés aux extrémités de ces arcs peuvent être fusionnés. Nous parlerons alors de règle de "fusion descendante".*

Propriété 4.3.2 *Si il existe un nœud s_k de G tel que:*

- *Les arcs a_{ik} qui arrivent en s_k sont équivalents.*
- *Les nœuds s_i situés aux origines de ces arcs sont équivalents.*

alors les arcs a_{ik} et les nœuds s_i peuvent être fusionnés. Nous parlerons alors de règle de "fusion ascendante".

Ces règles de ré-écriture sont dérivées des techniques d'inférence de grammaires régulières. Leurs effets sur la structure du programme synthétisé sont discutés dans [Dufay 83].

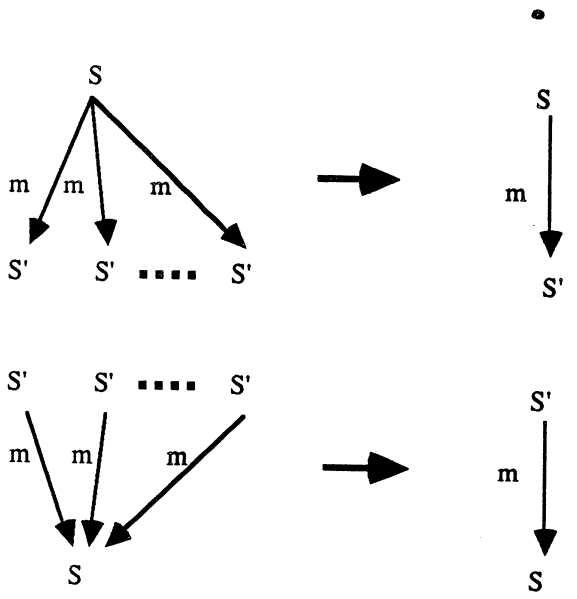


Figure 4.5: Règles de fusion utilisées pour combiner les traces d'exécution.

L'algorithme utilisé pour la construction du graphe d'exécution est alors le suivant:

1. Fusionner les nœuds extrémités (situation initiale et situation objectif) du graphe courant, avec ceux de la nouvelle trace d'exécution à intégrer. On note G_0 le graphe ainsi obtenu, et G l'état courant de ce graphe après application de plusieurs transformations.

2. Tant qu'il existe des nœuds de G qui vérifient la propriété 4.3.1, appliquer la règle de fusion descendante.

3. Si il existe un nœud de G qui vérifie la propriété 4.3.2, alors appliquer la règle de fusion ascendante et retourner en (2).
   Arrêter l'algorithme dans le cas contraire.

Synthèse du programme de montage:

Le graphe obtenu par combinaison des différentes traces d'exécution, est converti en un programme de manipulation. Ce programme est exprimé dans notre langage géométrique. Il est construit à l'aide des règles de traduction suivantes:

- Il existe un seul arc a_{ij} issu du nœud s_i. Le contrôle représenté est alors séquentiel. La règle de traduction appliquée est la suivante:

$$(s_i\ a_{ij}\ s_j)$$
$$\Longrightarrow\ (defun\ A_i()\ (m_{ij})(A_j))$$

où m_{ij} est le mouvement associé à l'arc a_{ij}, et A_j représente la construction appliquée après réalisation de la situation s_j.

- Il existe plusieurs arcs a_{ij_1}, $a_{ij_2} \cdots a_{ij_n}$ issus du nœud s_i. Le contrôle représenté est alors de nature conditionnelle. La règle de traduction appliquée est la suivante:

$$(s_i\ (a_{ij_1}\ s_{j_1})(a_{ij_2}\ s_{j_2}) \cdots (a_{ij_n}\ s_{j_n}))$$
$$\Longrightarrow\ (defun\ A_i()\ (m_{ij})(cond((b_{j_1})(A_{j_1})) \cdots ((bj_n)(A_{j_n}))))$$

où m_{ij} est le mouvement représenté par les arcs a_{ij_k} $(k = 1 \cdots n)$, b_{j_k} est la condition d'arrêt associée à la situation s_{j_k}, et A_{j_k} représente la construction appliquée après réalisation de la situation s_{j_k}.

- Il existe un arc a_{ii} dans la liste des arcs issus du nœud s_i. Le contrôle est alors de nature itérative. La règle de traduction utilisée pour traiter cet arc est la suivante:

$$(s_i\ a_{ii}\ s_i)$$
$$\Longrightarrow\ (defun\ A_{ii}()\ (while((b_i)(m_{ij}))))$$

où m_{ij} est le mouvement représenté par les arcs issus du nœud s_i, et b_i est la condition d'arrêt associée à la situation s_i. Cette règle est toujours appliquée en priorité.

Le programme engendré dans le cas du graphe représenté dans la figure 4.3, est le suivant:

$$(defun\ A_0()\ (m_1)(A_1))$$
$$(defun\ A_1()\ (m_2)\ (cond\ ((o_2)(A_2))\ ((c_2)(A_3))))$$
$$(defun\ A_2()\ (m_3)\ nil)$$
$$(defun\ A_3()\ (m_4)\ (cond\ ((o_4)nil)\ ((c_4)(A_1))))$$
$$(A_0)$$

Les termes m_i, o_j et c_k représentent alors des constructions du langage géométrique.

4.4 Implantation et expérimentations:

4.4.1 Le langage géométrique:

L'interpréteur du langage géométrique a été implanté sous la forme de deux modules, localisés respectivement sur un VAX 750 et sur le calculateur HP-1000

d'une armoire de commande ITMI. Cette armoire de commande pilote deux robots SCEMI six axes.

Le module localisé sur le VAX est écrit en KCL (Kyoto Common Lisp). Il interprète les instructions du langage, et il gère les modèles géométriques utilisés. Il adresse aussi des commandes de mouvement et de mise en service de capteurs aux contrôleurs du robot (c'est à dire au module localisé sur l'armoire de commande). Ces commandes sont codées sous la forme de fonctions LISP, sémantiquement équivalentes aux instructions de base de LM (cf. paragraphe 3.8). Elles sont transmises par l'interface LISP-LM. Ce module a été entièrement implanté.

Le module localisé sur l'armoire de commande est écrit en langage LM. Il gère à un niveau numérique tous les déplacements du robot. Chaque commande adressée par l'interpréteur du langage géométrique, conduit à faire exécuter un bloc de programme LM. Certaines fonctions liées au traitement des mouvements compliants, n'ont pas encore été implantées.

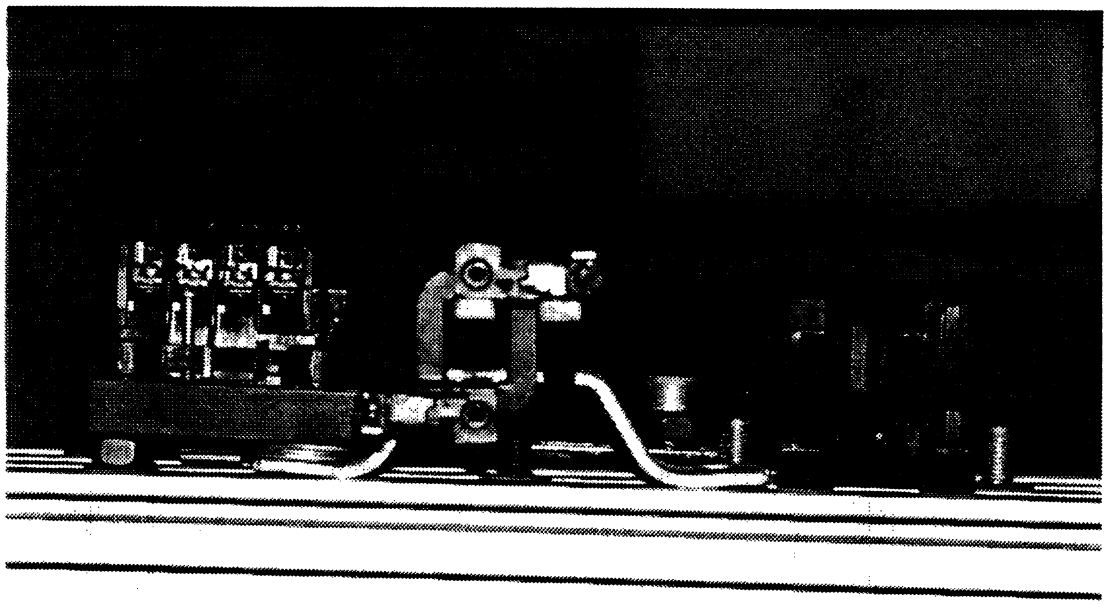


Figure 4.6: Exemple de montage d'un contacteur.

Le langage que nous avons développé a été expérimenté sur un exemple de

montage de contacteur (cf. figure 4.6). L'absence de certaines fonctions de base liées à la gestion des forces, n'a pas permis de faire des tests réellement significatifs. Il reste donc du travail à faire, avant de pouvoir valider notre approche sur le plan expérimental.

4.4.2 le module d'apprentissage:

Le module d'apprentissage interactif a été implanté en KCL sur VAX 750. Son architecture informatique est décrite dans la figure 4.2. Elle a été réalisée à partir des fonctions du système PGR, et des outils de programmation développés avec le langage géométrique.

Pour les mêmes raisons que celles évoquées précédemment (carences au niveau des outils de gestion des forces), il n'y a pas eu d'expérimentations vraiment significatives: les stratégies élémentaires proposées sont réduites à des mouvements de base du langage, et le module d'induction a été testé uniquement sur des exemples fictifs. Il reste donc un travail important pour développer de réelles fonctions d'apprentissage, et pour valider notre approche par des expérimentations sérieuses.

Partie II

La programmation automatique des robots: problèmes, modèles de raisonnement et architecture de système

Chapitre 5

La programmation automatique: problèmes et approches

5.1 Présentation du problème:

5.1.1 Approche traditionnelle et programmation automatique:

Les systèmes traditionnels de programmation de robots reposent sur des outils informatiques plus ou moins sophistiqués, conçus dans le but de servir d'interface entre le programmeur et les différentes fonctionnalités du robot (cf. première partie). Le rôle du programmeur consiste alors à décrire explicitement l'ensemble des actions et des opérations du robot, qui sont nécessaires à l'exécution de la tâche. Il doit pour cela faire face à de nombreux problèmes géométriques et physiques, afin d'éviter les collisions et les incidents mécaniques. Il doit en particulier prévoir dans quelle mesure les opérations programmées mèneront au résultat attendu, compte tenu des erreurs accumulées par les systèmes de commande et de perception. Dans la pratique, le programmeur n'a qu'une connaissance empirique et très qualitative de ces phénomènes. Il procède alors par corrections successives de son programme, afin de s'affranchir au mieux des contraintes d'incertitudes. Chaque correction est généralement provoquée par un échec détecté lors d'une tentative d'exécution du programme. Elle repose alors entièrement sur l'expérience du programmeur.

En programmation automatique, ce travail est à la charge du système. Il dispose pour cela d'une description symbolique de la tâche, réalisée par des assertions du type: *placer le cube A dans l'angle C de l'objet B.* Cette description est complétée par un modèle du robot et de son environnement, incluant une représentation des jeux de montage et des incertitudes de commande et de perception. Le rôle du système est alors de convertir la description symbolique de la tâche en un programme de commande de robot permettant d'exécuter cette tâche. Il doit pour cela résoudre trois classes de problèmes: la planification globale de la tâche, la planification des actions nominales du robot, et la modification du plan nominal afin de l'adapter à la réalité physique.

Les différents aspects de ces problèmes sont analysés dans le paragraphe 5.2. Ils sont introduits de manière intuitive dans le paragraphe qui suit, en s'appuyant sur un exemple simple.

5.1.2 Analyse d'un exemple de planification:

Description de la tâche:

L'exemple traité consiste à positionner un bloc parallélépipèdique dans l'angle d'une pièce coudée (cf. figure 5.1). Il est développé dans un langage dérivé de celui que nous avons défini dans le chapitre 4. Le formalisme géométrique utilisé pour spécifier la tâche est celui du langage LM-GEO [Mazer 82]. Le montage considéré peut alors être décrit comme suit:

$MONTER\ A\ SUR\ B\ TEL\text{-}QUE\ (R_1 \wedge R_2 \wedge R_3)$

avec:

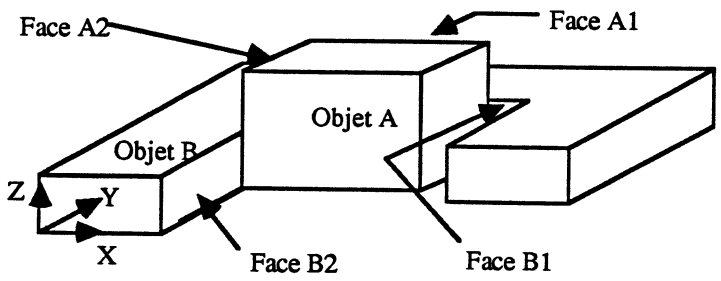
$R_1 = dessous\text{-}A\ \ CONTRE(0)\ \ dessus\text{-}table$

$R_2 = face\ A_1\ \ CONTRE(0)\ \ face\ B_1$

$R_3 = face\ A_2\ \ CONTRE(0)\ \ face\ B_2$

Planification globale de la tâche:

Cette phase constitue le point de départ du processus de planification. Elle a pour objet de définir les paramètres non explicitement spécifiés. Dans l'exemple traité, il s'agit de déterminer les emplacements initiaux des objets A et B, ainsi que les ressources nécessaires à l'exécution de la tâche (type de préhenseur, outillage de maintien pour B). Si l'assemblage traité avait nécessité plusieurs opérations de montage, le système aurait également dû déterminer l'ordre d'exécution de ces opérations.



Description symbolique: *Dessous-A CONTRE(0) Dessus-table*
Face A1 CONTRE(0) Face B1
Face A2 CONTRE(0) Face B2

Figure 5.1: Exemple d'un "assemblage" simple.

Une caractéristique de ce type de planification est d'opérer à un niveau conceptuel assez éloigné des détails de la manipulation réelle. La méconnaissance de certaines contraintes de manipulation, amène le système à faire des choix arbitraires qui peuvent conduire à des échecs. Ces choix sont alors remis en cause dès la détection de l'échec. Par exemple, l'opération de localisation de A imposée par un risque de collision entre A et B (voir plus loin), nécessite de placer initialement A dans la région située sous la caméra.

Planification des actions nominales du robot:

Cette phase de planification a pour objet de construire un plan de manipulation nominal, c'est à dire un plan bâti uniquement sur une base géométrique. Le système ignore alors la plupart des contraintes d'incertitudes qui dérivent des erreurs de modélisation, de commande et de perception. Le plan produit est alors linéaire. Il peut être représenté par la séquence suivante:

1. *INITIALISER ROBOT*
2. *REALISER pince SUR prise-A VIA <chemin>*
3. *SAISIR objet A <écart(prise-A) - ε >*
4. *REALISER* $R_1 \wedge R_2 \wedge R_3$ *VIA <chemin>* (*)

5. *POSER objet-A* $<$*écart(prise-A)* $+ \varepsilon >$

Dans ce plan, la situation symbolique *"pince SUR prise-A"* correspond à une prise à la fois stable, accessible et compatible avec le contexte de l'opération de saisie. Les paramètres de mouvement représentés par le symbole $<$ *chemin* $>$ définissent des trajectoires qui évitent les collisions. L'instruction marquée d'une étoile est une instruction nominale qui est irréalisable compte tenu des contraintes d'incertitudes liées à la tâche. Afin d'être exécutable par le robot, cette instruction devra être "éclatée" en une action d'approche, suivie d'une séquence de commandes gardées et de mouvements compliants. Cette tâche incombe à l'étape de planification suivante.

Le plan construit à ce niveau n'est donc pas directement exécutable par le robot. Seule sa validité globale vis à vis de l'objectif visé et des contraintes d'accessibilité est analysée par le système.

Amendement du plan nominal:

Cette étape de planification permet de compléter et de corriger le plan construit antérieurement. Le but recherché est de transformer ce plan en un plan exécutable par le robot. Les modifications apportées sont alors orientées vers le traitement des incertitudes. Elles se traduisent en général par l'intégration d'opérations sensorielles et/ou de stratégies de mouvements fins. Une construction possible pour l'exemple traité peut avoir la forme suivante:

1. *INITIALISER ROBOT*
2. (a) *LOCALISER objet-A PAR VISION* $<$*position-prévue*$>$
   (b) *REALISER pince SUR prise-A VIA* $<$*chemin*$>$
3. *SAISIR objet-A* $<$*écart(prise-A)* $- \varepsilon >$
4. (a) *REALISER approche*$(R_1 \wedge R_2 \wedge R_3)$ *VIA* $<$*chemin*$>$
   (b) *DEPLACER objet-A SUIVANT* $< -dz >$ *JUSQUA* $<$*contact-table*$>$
   (c) *DEPLACER objet-A SUIVANT* $< -dy >$
   EN-MAINTENANT $<$*contact-table*$>$ *JUSQUA* $<$*contact-*$B_1 >$
   (d) *DEPLACER objet-A SUIVANT* $< -dx >$
   EN-MAINTENANT $<$*contact-table* $\wedge$ *contact-*$B_1 >$
   JUSQUA $<$*contact-*$B_2 >$
5. *POSER objet-A* $<$*écart(prise-A)* $+ \varepsilon >$

avec:
$approche(R_1 \wedge R_2 \wedge R_3) = position(R_1 \wedge R_2 \wedge R_3) + dz + dx - dy$
$contact\text{-}table = (F_z > seuil_z)$
$contact\text{-}B_1 = (-F_y > seuil_y)$
$contact\text{-}B_2 = (F_x > seuil_x)$

Dans le processus de construction du plan, certaines contraintes ont un rôle actif dès la phase de planification. C'est en particulier le cas des incertitudes de montage qui sont à l'origine de la génération des stratégies de mouvements fins. Les actions *4b*, *4c* et *4d* sont de ce type. D'autres contraintes proviennent des phénomènes de propagation d'erreurs. Elles n'interviennent alors qu'à posteriori sur le plan construit. Ainsi, l'opération de localisation de l'objet A a été introduite dans le plan afin de réduire l'incertitude liée à la future position de A dans les mors de la pince. En l'absence de cette opération sensorielle, les erreurs cumulées pourraient conduire l'objet A à heurter la pièce B au cours de l'exécution de l'instruction *4a*.

Modes de traitement des incertitudes:

Il apparaît au travers de l'exemple traité, que deux catégories d'incertitudes interviennent au niveau du processus de planification: celles qui sont prises en compte localement par les fonctions de planification, et celles qui sont traitées à posteriori du fait de leurs natures fondamentalement globales.

- Le premier type de traitement a pour objet de contraindre le choix des actions, afin de réduire au maximun les risques d'échecs. Il repose sur l'utilisation de stratégies de manipulation, dont les conditions d'application sont dictées par une estimation grossière des incertitudes de position. Le système engendrera par exemple une prise stable conduisant à limiter les erreurs suivant certaines directions; il construira aussi des séquences de mouvements compliants permettant de réaliser une relation d'assemblage, en prenant appui sur des surfaces soigneusement choisies. Toutes les décisions prises à ce niveau ont un caractère strictement local.

- L'autre mode de traitement des incertitudes a pour objet de s'assurer de la validité globale du plan engendré vis-à-vis des contraintes de précision imposées par la tâche. Il s'appuie pour cela sur une étude des valeurs d'incertitudes présentes dans le plan, en propageant celles-ci au travers des actions planifiées. En cas d'incompatibilité entre les valeurs obtenues et celles imposées par la tâche, le système recherche les causes possibles de l'échec; il apporte ensuite les modifications nécessaires à l'obtention des valeurs re-

quises. L'instruction $2a$ a ainsi été insérée dans le plan, afin d'obtenir la précision requise par l'instruction $4a$.

5.2 Analyse du processus de planification:

Le processus de planification qui résulte de l'analyse intuitive réalisée dans le paragraphe précédent, est illustré par la figure 5.2. Nous allons maintenant analyser les différents problèmes de planification qui s'y rattachent. Ces problèmes ont été répartis dans trois catégories: ceux liés à la planification de la tâche, ceux engendrés par la planification des actions de manipulation, et ceux qui découlent de la présence des interdépendances.

5.2.1 Planification de la tâche:

Planification de l'environnement:

Le problème posé est celui de la détermination automatique des éléments matériels nécessaires à l'exécution de la tâche, compte tenu des contraintes physiques imposées par celle-ci. La résolution de ce problème conduit en premier lieu à sélectionner les appareils d'approvisionnement en pièces, les outillages de maintien, et les outils terminaux qui seront utilisés par le robot. Les choix correspondants sont guidés par des impératifs de précision, de fiabilité et d'efficacité.

Un autre aspect concerne la définition des zones de travail et des emplacements qui seront attribués aux objets du montage. Les choix faits à ce niveau tentent d'une part de minimiser les erreurs de positionnement et de perception, et d'autre part de réduire les temps de manipulation. Ils conditionnent de ce fait la "qualité" des plans d'actions qui seront engendrés sur cette base.

Ces problèmes ne seront pas abordés dans la suite.

Ordonnancement des opérations:

Le problème posé consiste à trouver une suite d'opérations d'assemblage permettant de réaliser la tâche considérée. Le mode de planification nécessaire est dit de *niveau tâche*. Il a pour objectif de construire une liste ordonnée de relations d'assemblage. Chaque relation représente une étape nécessaire à la réalisation de l'objectif global. Elle est associée à un ensemble de ressources propres (robot, outillage, zone de travail ...). Les connaissances qui interviennent à ce niveau sont relativement éloignées de l'aspect exécutoire des actions. Elles représente un savoir-faire global en matière d'assemblage automatisé. Les raisonnements menés portent alors essentiellement sur le rôle joué par chaque objet au sein de la tâche. Ils s'appuient pour cela sur la modélisation d'actions symboliques situées à un haut niveau d'abstraction. Par exemple, l'action *"monter l'objet A sur l'objet B"*

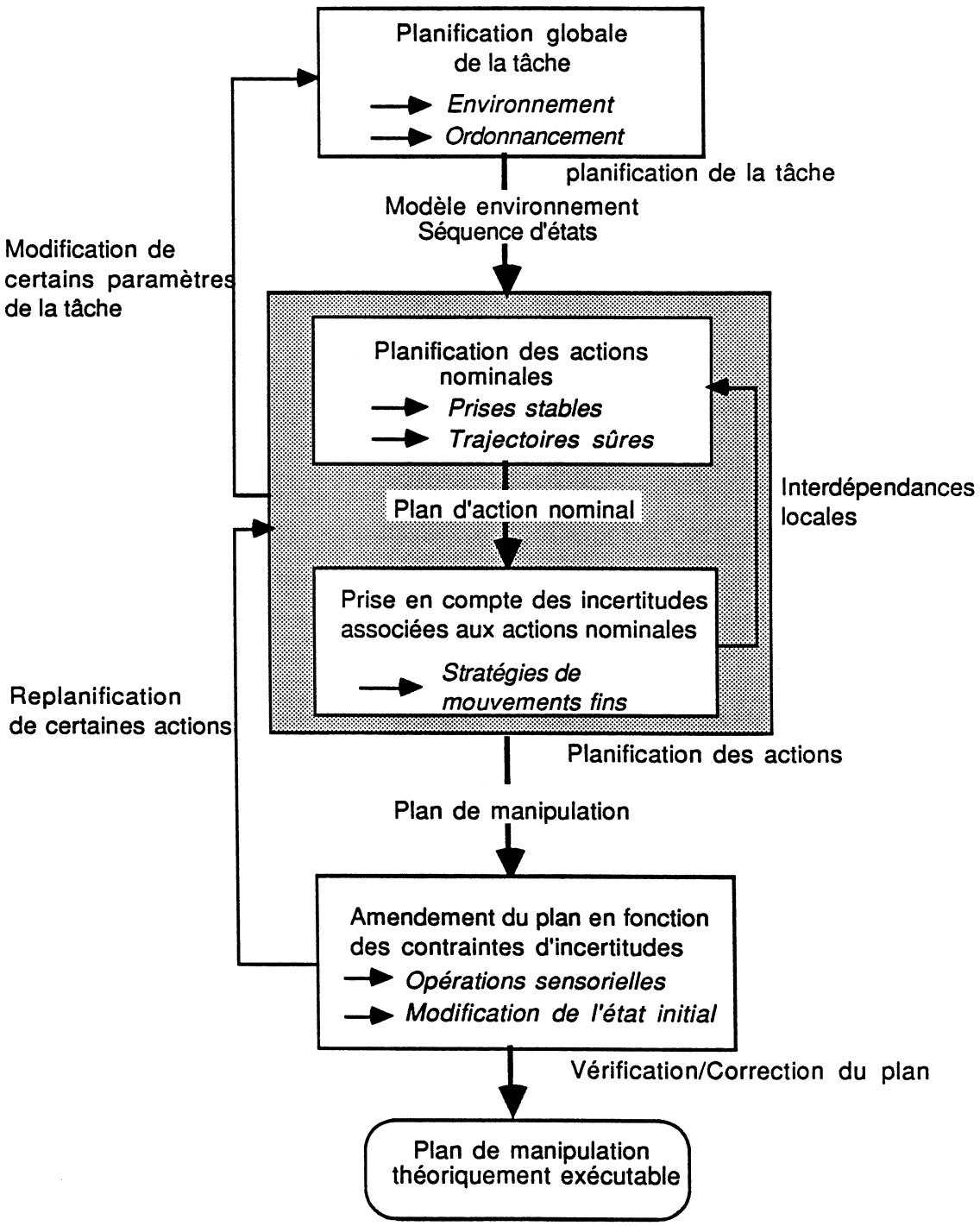


Figure 5.2: schéma synoptique du processus de planification en programmation automatique des robots.

sera représentée par une liste de préconditions et de postconditions exprimées sous forme logique. Les conditions traitées portent alors sur la présence de certaines ressources (outils, pièces ···), sur l'attitude des objets manipulés, sur leurs disponibilités dans l'espace de travail, etc.

Ce problème de planification est à rapprocher de celui de la génération automatique de gammes de fabrication, pour lequel une formulation possible est la suivante: étant donné un ensemble de ressources matérielles et un objectif décrit en termes d'objets à réaliser, trouver un procédé de fabrication permettant d'atteindre cet objectif en exploitant au mieux les ressources disponibles. Ce procédé correspond à une suite ordonnée d'opérations mettant en jeu des machines, des outillages et des contraintes d'attitudes portant sur la pièce traitée. Il est construit à partir de connaissances expertes propres au domaine de l'usinage [Descotte, Latombe 84] [Lagoude, Tsang 85].

Ce type de planification ne sera pas traité dans la suite.

5.2.2 Planification des actions de manipulation:

Le problème posé consiste à déterminer les séquences d'actions qui permettront, compte tenu des contraintes liées à la tâche, de réaliser les différents objectifs intermédiaires représentés par les relations d'assemblage. Ces actions incluent des déplacements du robot, des opérations d'outils et des consultations de données sensorielles. Les choix faits à ce niveau s'appuient essentiellement sur des considérations d'ordre géométrique et physique. Ils tentent d'adapter au mieux le comportement du robot aux impératifs de la tâche et du monde réel.

Le point le plus difficile à maîtriser dans ce problème, se situe au niveau de la prise en compte complète de l'aspect exécutoire des actions du robot. La difficulté provient de ce que toutes ces actions doivent s'exécuter dans un univers réel complexe, combinant des aspects géométriques et physiques souvent mal connus. Trois sous-problèmes doivent alors être résolus par le système: comment saisir un objet en vue d'une opération donnée? comment le transporter d'une position à une autre sans heurter les objets environnants? comment le manipuler pour réaliser un montage particulier malgré le manque de précision du robot? Ces trois sous-problèmes se distinguent les uns des autres par les connaissances qu'ils mettent en jeu, et par les raisonnements qu'il faut conduire pour les résoudre. Leurs formulations sont données ci-dessous.

Planification des opérations de saisie:

Le problème posé consiste à choisir automatiquement les paramètres de prise d'un objet en fonction du contexte de l'opération, de la morphologie de l'objet et de la géométrie de l'outil de préhension. Ces paramètres définissent la position de l'outil

par rapport à l'objet, ainsi que les opérations à exécuter (déplacements et actions de l'outil). Le choix de ces paramètres est guidé par des impératifs de stabilité, d'accessibilité et de précision. Par exemple, la prise de l'objet A du montage précédent ne doit engendrer aucune collision au cours des opérations de saisie et de manipulation de A. Elle doit aussi rester stable dans les mors de la pince tout au long du montage; elle doit pour cela être insensible aux forces engendrées par les contacts intermédiaires (*contact-table* et *contact-*B_1). Enfin, cette prise ne doit pas introduire des erreurs trop importantes dans la direction X, sous peine d'entraîner des risques de collisions entre A et B. Cette dernière contrainte suggère de saisir A par les faces A_2 et A_3. Elle est cependant incompatible avec la contrainte de "stabilité dynamique" engendrée par les forces de réaction du contact avec la face B_1. Elle est de ce fait ignorée lors du choix de la prise; elle ne sera prise en compte qu'au moment de l'analyse de cohérence globale du plan.

Planification des mouvements de transfert:

Le problème traité est celui du calcul d'une trajectoire permettant de déplacer un objet d'une position initiale donnée à une position finale peu sensible aux effets des incertitudes. Cette trajectoire ne doit engendrer aucune collision entre les éléments de l'environnement, et le robot muni de sa charge. Elle doit également vérifier des contraintes parfois contradictoires comme la fiabilité et l'efficacité. Dans l'exemple précédent, les deux trajectoires représentées par le symbole $<$ *chemin* $>$ conduisent à manœuvrer le bras du robot de manière rapide et sûre. La première trajectoire définit un chemin permettant d'accéder librement à la prise choisie; l'autre spécifie les paramètres du mouvement qu'il faut exécuter pour transporter l'objet A à proximité de la pièce B.

Planification des opérations de montage:

Le problème consiste à synthétiser un programme de commande de robot permettant de réaliser une relation d'assemblage donnée, malgré les contraintes d'incertitudes imposées par l'environnement de la tâche. Les stratégies implantées dans ce programme combinent des opérations sensorielles avec des mouvements du robot. Elles permettent ainsi de s'affranchir des erreurs de commande et de perception. Les actions utilisées à cet effet sont des commandes gardées et des mouvements compliants. Elles conduisent soit à rechercher des contacts pertinents, soit à utiliser des relations de contacts particulières pour guider le montage. Par exemple, l'instruction 4*b* permet de rechercher le contact avec la table; les instructions 4*c* et 4*d* exploitent des contacts existants pour guider les mouvements du robot.

5.2.3 Traitement des interdépendances:

Jusqu'à présent, nous avons étudié séparément les différents problèmes de planification. Nous avons également vu au travers des exemples traités, que les solutions apportées étaient très souvent dépendantes les unes des autres. Toutes ces interdépendances qui affectent les prises de décisions, proviennent de deux sources différentes: la géométrie et les erreurs.

Le premier type d'interdépendance se manifeste au niveau du choix des paramètres des actions. Par exemple, la position de saisie représentée par le symbole *prise-A* ne doit pas être trop basse, afin de ne pas entraîner des risques de collisions entre les mors de la pince et l'objet B. Elle doit aussi être orientée perpendiculairement à la face B_1, de manière à pouvoir supporter sans risques les forces créées par la commande compliante 4*d*. Ces interdépendances sont qualifiées de "locales" dans le sens où elles n'affectent que des actions voisines du plan, c'est à dire des actions appartenant à un même cycle de manipulation du type "saisie- transport-montage". Elles se manifestent essentiellement au niveau de la géométrie de la tâche. Nous parlerons alors de *contraintes géométriques.*

D'autres interdépendances affectent aussi la structure du plan. C'est par exemple le cas de l'opération de localisation de l'objet A, dont le but est d'éviter un échec éventuel dû à une collision entre les pièces A et B. Dans le plan, cette opération précède la saisie de A; elle aurait pu être placée après si les conditions de vision avaient été satisfaisantes. Les contraintes manipulées à ce niveau ont donc un caractère fondamentalement non local. Elles se propagent dans le plan d'actions sous la forme de *contraintes d'incertitudes*, provenant à la fois des erreurs de commande et de perception. Elles n'interviennent de ce fait qu'à postériori sur le programme synthétisé, par le biais d'une phase de vérification/correction. Ainsi, tout échec détecté par le système de vérification conduit à rechercher les causes possibles de cet échec en "régressant" dans le plan (cf. [Cohen, Feigenbaum]). Par exemple, la précision requise pour placer l'objet A à côté de l'objet B est obtenue en propageant la contrainte imposée en 4*a*, au travers des actions 3 et 2*b*. Les corrections apportées peuvent alors se traduire par des modifications locales, ayant pour objet de réduire les incertitudes concernées; c'est par exemple le cas de l'instruction 2*a*. Elles peuvent aussi conduire à replanifier entièrement certaines séquences d'actions.

Toutes ces interdépendances se traduisent au niveau du processus de planification par divers "retours arrières" (cf. figure 5.2). Ces retours arrières conduisent à remettre en cause certains choix antérieurs, lorsqu'un échec a été détecté dans la solution courante. Ils sont gérés par la structure de contrôle du système. Ce point est discuté dans le chapitre 9.

5.3 Travaux portant sur les systèmes de niveau tâche:

Dès l'apparition des premiers langages de programmation de robots, certains chercheurs se sont penchés sur le problème de l'automatisation de cette programmation. Les termes dans lesquels ils se sont posés les questions de fond, varient beaucoup en fonction de leurs motivations initiales: recherche en Intelligence Artificielle, développement d'un langage de haut niveau pour la robotique, étude de l'aspect exécutoire des actions de manipulation, synthèse de programmes de commande de robots. D'une manière générale, les approches qui découlent de ces travaux se placent toutes dans le contexte de la *génération automatique de plans d'actions pour un robot.* Elles abordent le sujet suivant quatre directions: la description de la tâche, la planification des opérations requises par la tâche, le traitement des interdépendances, et la planification des actions du robot.

5.3.1 Description de la tâche:

Le problème traité est celui de la spécification symbolique des objectifs intermédiaires nécessaires à l'exécution d'une tâche d'assemblage. De manière générale, ce problème a été considéré par les chercheurs comme un problème de langage et de géométrie.

Dans sa formulation initiale, le langage AL [Finkel et al. 74] comportait quelques constructions de niveau tâche. Ces constructions offraient la possibilité de décrire des modèles géométriques, afin de spécifier plus aisément certaines opérations du robot. Ce projet a motivé des recherches fondamentales sur la programmation de niveau tâche [Taylor 76] [Binford 79], mais les résultats obtenus n'ont jamais été incorporés dans le système AL. De manière similaire, ROBEX [Weck, Zuhlke 81] devait inclure des fonctions pour calculer automatiquement des trajectoires sans collision et pour introduire automatiquement des opérations sensorielles dans les programmes. Ces fonctionnalités n'ont jamais été implantées.

Les travaux les plus significatifs dans le contexte de la spécification géométrique d'une tâche d'assemblage, ont été réalisés à l'Université d'Edimbourg (langage RAPT) et à l'Institut National Polytechnique de Grenoble (langage LM-GEO).

RAPT [Popplestone et al.78] est un langage symbolique à forte composante géométrique. Dans sa forme de base, il permet d'inférer les positions spatiales de corps solides, à partir d'une description symbolique des relations géométriques qui existent entre ces corps. Une retombée intéressante du mécanisme d'inférences géométriques de RAPT, consiste à engendrer automatiquement des séquences de positions pour l'outil terminal du robot. Le système interprète pour cela des commandes de mouvements du type: *déplacer le robot parallèlement à la face Fi de l'objet O.* A ce niveau de raisonnement, seul l'aspect numérique des trajectoires

est pris en compte; les problèmes de nature physique tel que l'accessibilité, les collisions, la stabilité, et les incertitudes sont alors complètement ignorés par le système.

Le langage LM-GEO [Mazer 83] est une extension du système LM. Il a été conçu comme une interface entre LM et une base de modèles géométriques. Il permet de spécifier dans un formalisme proche de celui de RAPT, toutes les positions intermédiaires nécessaires à l'exécution d'une tâche de manipulation. après interprétation, un programme en LM-GEO est transformé en un programme LM classique.

Malgré les fonctions géométriques qu'ils possèdent, RAPT et LM-GEO restent des langages de niveau manipulation. Leur intérêt pour la programmation automatique réside essentiellement dans le formalisme géométrique qu'ils offrent pour la description des situations clés. Nous verrons plus loin que ce formalisme est relativement bien adapté à la représentation des objectifs intermédiaires de la tâche.

5.3.2 Planification de niveau tâche:

Les premiers travaux de recherche sur la programmation de niveau tâche datent du début des années 70. Ils avaient pour objectif de planifier des séquences d'opérations symboliques, en vue de la réalisation de tâches simples. Dans ce contexte, toutes les actions disponibles étaient définies en termes de leurs conditions d'applications et de leurs effets sur l'univers modélisé. Elles étaient enchaînées par le système de manière à faire passer l'univers d'un état initial donné à priori, à un état final représentatif de la tâche à exécuter.

Dans un premier temps, les chercheurs ont porté leurs efforts sur l'aspect logique du problème. C'est pourquoi, toutes les approches développées à cette époque reposent sur une modélisation logique de but et des actions de l'agent. Elle utilise pour cela un formalisme à base de logique des prédicats du premier ordre. Cette approche regroupe tous les planificateurs de la famille STRIPS [Fikes, Nilsson 71]. Elle conduit à engendrer des plans d'actions situés à un niveau d'abstraction très éloigné des réalités du monde physique. D'autres travaux plus proches de cette réalité ont été développés dans le contexte de la conception automatique de gammes de fabrication [Descotte, Latombe 84] [Matsushima et al. 82] [Lagoude, Tsang 85]. Mais les systèmes correspondants traitent uniquement du problème de l'usinage, et les méthodes employées n'ont jamais été étendues à l'assemblage automatisé.

Une approche plus réaliste (mais mal maîtrisée sur le plan de la complexité algorithmique) consiste à raisonner sur la géométrie de la tâche. Les concepts sous-

jaçents à cette approche sont présents à l'état embryonnaire dans les premières spécifications du projet "Main-œil" de l'université de Stanford [Feldman et al. 71]. Ils ont ensuite été développés de manière plus avancée par Taylor [Taylor 76] et par Lozano-Perez [Lozano-Perez 76], sans qu'aucune implantation significative n'ait été réalisée. Dans cette approche, l'idée de base consiste à analyser à priori les contraintes imposées par la géométrie de la tâche; les résultats de cette analyse sont ensuite utilisés pour déterminer les stratégies de manipulation qu'il convient d'appliquer. Chaque stratégie est alors vue comme une macro-action qu'il faudra développer plus tard. Seules ses préconditions et ses postconditions sont considérées à ce niveau de raisonnement.

Une difficulté majeure de ce type d'approche réside dans l'extrême sensibilité des stratégies employées vis-à-vis des petites variations de la géométrie des pièces; cette sensibilité rend impossible la constitution d'un ensemble raisonnable de classes identifiables. Toutes ces difficultés ont conduit les chercheurs à orienter leurs efforts sur des sous-problèmes mieux délimités tels que la planification de trajectoires sûres pour le robot, ou la prise en compte des incertitudes de position au niveau du montage. Les travaux relatifs à ces problèmes d'un autre niveau conceptuel sont décrits dans le paragraphe 5.4.

5.3.3 Traitement des interdépendances:

Le traitement des interdépendances est un problème difficile, qui a été abordé de différentes manières dans les systèmes de planification [Nilsson 80]. Dans le contexte de la programmation automatique des robots, ce problème a donné lieu à quelques travaux ponctuels, essentiellement orientés vers le calcul et la représentation des contraintes d'incertitudes [Taylor 76] [Brooks 82b]. Le mode de calcul induit opère en "chaînage avant" pour estimer les erreurs introduites par les opérations planifiées; il procède par "chaînage arrière" pour évaluer toutes les valeurs initiales qu'il est possible d'affecter à certains paramètres, sans dégrader les chances de succès du programme. Cette manière de procéder permet potentiellement de choisir des stratégies de manipulation compatibles avec les opérations antérieures. Elle permet aussi de rechercher les origines possibles des échecs détectés, afin de corriger le plan défaillant.

Les travaux plus récents réalisés dans notre équipe de recherche sont basés sur des principes analogues [Troccaz, Puget 87]. Ils sont par contre entièrement orientés vers le traitement du problème de vérification/correction de plans. Ce point sera développé dans le chapitre 9.

5.3.4 Projets de systèmes intégrés:

Quelques projets ambitieux orientés vers la conception de systèmes complets pour la programmation automatique des robots, ont vu le jour aux Etats-Unis vers le milieu des années 70: AL à l'université de Stanford [Finkel et al. 74], LAMA au Laboratoire d'Intelligence Artificielle du MIT [Lozano-Perez 76], et AUTOPASS au centre de recherche d'IBM [Liebermann, Wesley 77]. Ces projets n'ont fait l'objet que de spécifications et de réalisations incomplètes. Ils ont par contre eu le mérite de poser le problème de manière réaliste, à une époque où l'accent était mis sur les générateurs de plans d'actions logiques. Ils ont montré également que la transformation d'une description de niveau tâche en un programme de manipulation est une opération difficile, qui nécessite de résoudre un grand nombre de problèmes géométriques et physiques. Ils ont ainsi conduit les chercheurs à aborder des aspects divers tels que la modélisation géométrique de l'univers du robot, la conduite de raisonnements géométriques en vue de la planification d'actions, ou la représentation et la propagation de contraintes d'incertitudes dans un plan d'actions. La plupart des travaux mentionnés précédemment ont été réalisés dans ce contexte.

Plus tard, une architecture de système de niveau tâche a été proposée par des chercheurs du MIT [Lozano-Perez, Brooks 85]. Cette architecture, appelée TWAIN, était basée sur des techniques de propagation de contraintes géométriques. Le but recherché était alors de résoudre les interdépendances entre les modules de planification. Le rôle attribué aux contraintes était double: guider le processus de planification de l'espace de travail (positions initiales des objets essentiellement), et guider le mécanisme de sélection et de paramétrisation des stratégies de manipulation. Aucune implantation n'a cependant été réalisée.

Les systèmes HANDEY [Lozano-Perez et al. 87] [Mazer 87] et SHARP [Laugier, Troccaz 85] ont des ambitions plus modestes que celles de leurs prédécesseurs. Ils servent essentiellement de cadre à la conduite de recherches sur la programmation automatique, tout en mettant l'accent sur le problème de l'intégration. Il s'agit alors de faire coopérer des modules de planification différents, opérant sur des modèles communs. La structure de contrôle de ces systèmes reste cependant très simple (quelques retours arrières élémentaires en cas d'échecs). Les principales différences qui existent entre HANDEY et SHARP sont les suivantes:

- Handey est un système de type "Pick and Place", alors que SHARP traite des opérations d'assemblage nécessitant des mouvements fins.

- HANDEY utilise en entrée des informations fournies par un système de vision tridimensionnel, alors que SHARP opère sur un modèle complet de l'environnement.

- HANDEY planifie des opérations de saisie pouvant mettre en œuvre plusieurs prises intermédiaires, ce qui n'est pas le cas pour SHARP.
- HANDEY ne traite pas les incertitudes géométriques, alors que SHARP comporte un module de vérification/correction permettant de garantir la validité théorique des plans engendrés.
- Une version complète de HANDEY est opérationnelle, ce qui n'est pas encore le cas pour SHARP.

Les travaux du LAAS en matière de système intégré pour la robotique manufacturière, ont été orientés différemment. Ils ont conduit à développer un système (le système NNS [Alami, Chochon 85] [Chochon 86]), assurant un degré élevé d'autonomie aux robots d'une cellule flexible d'assemblage. Dans ce système, l'accent a été mis sur deux points: la modélisation de la tâche et les fonctions de décision en ligne.

5.4 Travaux sur la planification des actions du robot:

5.4.1 Planification des opérations de saisie:

Les recherches portant sur la saisie automatique adressent quatre aspects différents du processus de saisie [Troccaz 86]: le choix de prise, l'étude d'équilibre/stabilité, l'étude d'accessibilité, et l'étude de compatibilité contextuelle.

Les réalisations traitant du premier point procèdent par recherche de schémas-types plus ou moins élaborés. Cette recherche peut être réalisée sur la base d'informations visuelles [Hanafusa, Asada 77a] [Birk et al. 81] [Boissonnat 82] [Ikeuchi et al. 84] ou tactiles [Todd 81] [Romiti et al. 81]; elle peut aussi s'appuyer sur des modèles géométriques complets [Lozano-Perez 76] [Wingham 77] [Laugier 81] [Laugier 83] [Laugier, Pertin 83] [Wolter et al. 84]. Dans tous les cas, elle conduit à raisonner sur la géométrie locale de l'objet à saisir.

Le problème de l'équilibre et de la stabilité d'une prise ne peut être directement résolu par des méthodes mathématiques rigoureuses, à cause de la présence d'incertitudes variées (incertitudes de position portant sur l'objet et l'outil de préhension, mauvaise connaissance des forces appliquées et des phénomènes de friction). Malgré ces difficultés, [Wolter et al. 84] ont développé un modèle d'équilibre statique assez rigoureux, mais l'utilisation de ce modèle nécessite l'emploi de coefficients empiriques qui limitent la portée des résultats obtenus. Les autres chercheurs ont tenté de résoudre ce problème, soit de manière entièrement heuristique [Laugier 81] [Laugier 83] [Hanafusa, Asada 77b], soit par une analyse des

phénomènes de friction [Fearing 84] [Brost 85].

L'étude d'accessibilité permet de déterminer si une prise est réalisable compte tenu de l'encombrement spatial environnant. Le problème consiste à rechercher une trajectoire pour l'outil de préhension, après avoir contraint partiellement son orientation et sa direction d'approche. Les méthodes utilisées reposent parfois sur des calculs d'interférences entre objets [Wolter et al. 84]; elles s'appuient en général sur une modélisation de l'espace des configurations de l'outil [Lozano-Perez 76] [Wingham 77] [Laugier, Pertin 83]. Ces approches, qui ne sont pas spécifiques de la saisie automatique, seront discutées plus loin.

Le problème de la compatibilité contextuelle est lié au fait que l'opération de saisie est exécutée dans un but précis, autre que celui de tenir l'objet. Un premier aspect de ce problème concerne l'accessibilité de l'objet dans son environnement final, c'est à dire après exécution de l'opération de montage visée. Les techniques utilisées sont alors celles décrites précédemment; elles permettent de trouver les trajectoires d'outil nécessaires aux opérations de saisie et de dépôt de l'objet, compte tenu des obstacles présents dans l'état initial et dans l'environnement final de la tâche. Ces techniques opèrent sur un modèle géométrique obtenu par union des obstacles des deux environnements [Lozano-Perez 81] [Laugier, Pertin 83] [Wolter et al. 84]. L'autre aspect du problème concerne la compatibilté fonctionnelle avec l'opération projetée. Peu de travaux ont porté sur cet aspect. Hanafusa et Asada [Hanafusa, Asada 77b] ont développé une méthode pour adapter les prises choisies à des catégories de tâches usuelles telles que l'insertion, la mise en position d'une pièce, ou le montage à base de mouvements compliants. Lyons [Lyons 85] considère trois types de prises (englobante, latérale ou de précision), pour préconfigurer l'outil de préhension. Il exploite pour cela des critères qualitatifs portant sur la nature de la saisie (précise, ferme), et sur la morphologie générale de l'objet (gros, petit, rond, long).

5.4.2 Planification des trajectoires de transfert:

Le problème de la planification des trajectoires est un problème clé de la robotique. Il a suscité à ce titre de nombreuses recherches possédant suivant le cas une coloration fortement théorique ou un aspect plus pratique. Le premier type de recherche s'est concrétisé par des résultats théoriques importants pour la compréhension du problème général et de ses variantes. Les autres recherches ont conduit à des implantations réelles. Elles adressent des sous-problèmes volontairement mieux délimités, afin de réduire la complexité algorithmique. Elles sont réparties suivant deux classes d'approches: les approches locales et les approches globales.

Résultats théoriques:

Les recherches théoriques abordent le problème sous un angle mathématique. Elles ont pour principal objectif de développer un formalisme rigoureux susceptible de servir de base à la construction de solutions théoriques générales. Dans la pratique, ces solutions ne sont pas exploitables du fait de leurs degrés de complexité algorithmique; elles servent essentiellement de preuve à l'existence d'algorithmes potentiels. Reif [Reif 79] a ainsi montré que le problème de la planification de trajectoires sans collision est "P space complex", dans le cas d'un robot manipulateur possédant un nombre arbitraire de degrés de liberté. Les travaux de Hopcroft [Hopcroft et al. 82] ont conduit à des résultats similaires pour un robot planaire à n degrés de liberté, contraint de manœuvrer dans une région prédéterminée. Schwartz et Sharir [Schwartz, Sharir 82] ont proposé un algorithme de complexité polynomiale en $O(n^{2d+6})$, dans lequel n est polynomialement dépendant du nombre de faces des obstacles, et d représente le nombre de degrés de liberté du mobile.

D'autres auteurs ont étudié les modifications théoriques introduites par la présence de plusieurs mobiles. Schwartz et Sharir [Schwartz, Sharir 83] ont montré que la coordination de mobiles circulaires engendrait une complexité de type exponentielle, fonction du nombre de degrés de liberté du système complet. Dans le cas de rectangles, cette complexité devient de type "P space complex" [Hopcroft et al. 84]. Reif et Sharir [Reif, Sharir 85] ont proposé un algorithme de complexité polynomiale, dans le cas d'un objet bidimensionnel se déplaçant parmi des obstacles fixes ou mobiles.

Méthodes locales:

Une caractéristique essentielle des méthodes locales est d'opérer sur un modèle ne comportant qu'une représentation partielle des contraintes imposées par l'environnement. Le principe de base consiste à construire progressivement la trajectoire recherchée, en raisonnant à chaque étape sur la géométrie locale des situations rencontrées. Suivant leurs caractéristiques, ces situations déclenchent l'activation de tels ou tels autres opérateurs locaux pour déplacer le mobile.

Approche par "génération et test":
Une première manière de procéder consiste à appliquer itérativement les trois opérations suivantes [Lewis 74]:

1. Hypothèse d'une trajectoire simple.

2. Etude des risques de collision le long de cette trajectoire.

3. Modification locale de la trajectoire en cas de risque de collision.

Dans ces méthodes, la génération des hypothèses concernant les trajectoires est faite sur la base d'heuristiques simples [Lewis 74] [Powell 82]. Le système tente par exemple d'engendrer des solutions qui conduisent à passer au-dessus des obstacles, ou à longer les frontières de zones interdites. L'étude des collisions potentielles est réalisée au moyen de techniques de calcul d'interférences entre solides [Maruyama 72] [Boyse 79] [Ahuja et al. 80]. Ces techniques sont issues du domaine du graphique et de la CAO. Elles opèrent sur des modèles variés: polyèdres, approximations volumiques, octrees, projections planaires...

Sur le plan opératoire, l'approche de type "génération et test" se prête assez bien à la résolution de problèmes pour lesquels il est possible de considérer chaque obstacle de manière isolée (pas de phénomènes de renvoi d'un obstacle à l'autre). Elle devient rapidement inutilisable lorsque les contraintes imposées par l'environnement atteignent une densité trop importante. Dans la pratique, les approches de ce type sont surtout utilisées pour planifier des trajectoires courtes et simples, ne mettant en jeu qu'un nombre limité de degrés de liberté. Udupa [Udupa 77a] recherche ainsi comment exécuter les rotations requises par l'avant-bras, lorsque celui-ci se déplace sur une trajectoire calculée auparavant. Faverjon [Faverjon 86] détermine à l'aide de plans tangents aux obstacles, les séquences de mouvements élémentaires (translations pures ou mouvements de type translation-rotation) qu'il faut appliquer pour éloigner ou rapprocher l'outil terminal des obstacles voisins. Il utilise sa méthode pour calculer les portions initiales et terminales des trajectoires de transfert, ou pour résoudre le problème des orientations du poignet du robot. Laugier et Pertin [Laugier, Pertin 83] combinent une approche de type génération et test avec une approche globale pour planifier des trajectoires de saisie. Dans un contexte différent, Tournassoud [Tournassoud 86] utilise des hyperplans séparateurs pour éviter localement des collisions entre plusieurs mobiles planaires animés de mouvements simples.

Approche par création de champs attractifs et répulsifs:
Une autre approche consiste à représenter explicitement l'influence des obstacles sur le mobile, en utilisant une fonction répulsive calculable de manière discrète. Le mobile est alors supposé être soumis à un champ de forces fictif, décrivant un potentiel attractif élevé au voisinage de l'objectif à atteindre, et divers potentiels répulsifs simulant la présence d'obstacles [Khatib 80]. La valeur de la fonction répulsive est infinie au contact des obstacles; elle décroît lorsque le mobile s'éloigne des obstacles. Une autre propriété importante de cette fonction est d'être capable de définir les directions de déplacement permettant d'échapper localement à l'influence répulsive d'un groupe d'obstacles. Ces directions sont symbolisées par les minima locaux de la fonction.

La méthode des potentiels a été utilisée par Thorpe [Thorpe 84] pour planifier

les trajectoires d'un robot mobile circulaire. L'algorithme développé calcule un premier chemin sur une grille à larges mailles, comportant en chaque point une évaluation de la fonction répulsive; il lisse ensuite la solution obtenue à l'aide d'une méthode de relaxation opérant sur une grille beaucoup plus fine. Krogh [Krogh 84] a généralisé le procédé, de manière à pouvoir prendre en compte les paramètres de vitesse. Khatib [Khatib 80] a appliqué la méthode des potentiels au cas plus général de la manipulation. Son algorithme exécute les calculs de potentiel en divers points de la structure articulée du robot. Les mouvements du bras résultent alors de l'interaction entre les forces fictives engendrées et les contraintes cinématiques introduites par la commande.

Approche par recherche de "surfaces de glissement":
Faverjon et Tournassoud [Faverjon, Tournassoud 87] ont implanté une méthode dérivée de celle des potentiels. Leur méthode permet au mobile de glisser sur des hyperplans fictifs, construits tangentiellement aux obstacles.

Donald [Donald 84] a proposé une technique conduisant à construire explicitement les C-surfaces associées aux obstacles (surfaces des obstacles dans l'espace des configurations du robot), puis à glisser sur ces C-surfaces. Mais la complexité engendrée par la construction des C-surfaces limite l'intérêt de sa méthode.

Méthodes globales:

Les méthodes globales opèrent sur un modèle explicite des contraintes imposées par l'environnement physique. Le principe consiste à représenter dans un formalisme approprié, l'ensemble des configurations du robot qui n'engendrent aucune collision avec les objets environnants. La détermination d'une trajectoire sûre peut alors être assimilée à la recherche d'une séquence connexe de configurations libres, reliant la configuration initiale à la configuration terminale. Les approches développées se distinguent essentiellement par le formalisme qu'elles mettent en œuvre, et par l'aptitude de ce formalisme à traiter des variantes différentes du problème, par exemple: déplacement d'un objet rigide ou d'un mécanisme articulé, mouvements incluant ou non des rotations, possibilité de traiter raisonnablement 3, 4 ou 6 degrés de liberté.

Dans la suite, nous caractériserons les problèmes traités par les degrés de liberté qu'ils mettent en jeu. Dans le cas des robots manipulateurs, ces degrés de liberté comportent des translations et des rotations exécutées suivant des axes orthogonaux. L'espace des translations est noté $\Re^i$ $(i = 1, 2\, ou\, 3)$, et le groupe des rotations orthogonales est représenté par le symbole O^j $(j = 1, 2\, ou\, 3)$. L'espace des configurations associé à un mobile de l'espace tridimensionnel est alors caractérisé par le produit $\Re^3 \times O^3$. Dans le cas général de la planification des mouvements articulaires d'un robot comportant n liaisons rotoïdes ou primatiques, nous parle-

rons de problèmes de caractéristiques n-DDL. Le symbole n-DDL représente alors l'espace construit à partir du produit cartésien $I_1 \times I_2 \cdots I_n$, dans lequel I_i est le domaine de définition de la i-ème variable articulaire.

Les travaux de Widdoes et d'Udupa:
Les premiers résultats significatifs concernant la planification de trajectoires sûres pour un robot manipulateur ont été obtenus à l'université de Stanford par Widdoes [Widdoes 74] et par Udupa [Udupa 77a] [Udupa 77b]. Les algorithmes développés sont très dépendants des caractéristiques du bras utilisé, et ils imposent d'importantes limitations sur les mouvements du poignet. Ils reposent sur une technique originale de partitionnement de l'espace articulaire associé aux trois premiers degrés de liberté du robot. Ce découpage spatial a pour objectif de faire apparaître explicitement toutes les configurations interdites. Malgré ses limitations, le procédé de construction de l'espace libre proposé par Udupa est à l'origine d'un grand nombre de recherches [Brady et al. 82]. L'idée sur laquelle repose ce procédé consiste à considérer le mobile A comme un point se déplaçant dans un espace de dimension n, où n définit les degrés de liberté de A. Afin que cette transformation soit possible, chaque obstacle B_i est représenté dans cet espace par l'ensemble des configurations de A qui induisent une collision entre A et B_i. Cette technique de transformation du problème initial a été formalisée et généralisée par Lozano-Perez [Lozano-Perez 83].

Mobiles bidimensionnels à orientations fixes:
La technique de raisonnement dans l'espace des configurations du mobile fut initialement développée dans le cadre de problèmes de type $\Re^2$. Les réalisations correspondantes conduisent à rechercher les trajectoires possibles d'un objet circulaire ou polygonal, se déplaçant parmi des obstacles polygonaux. Cette recherche s'effectue sur un graphe symbolisant un réseau planaire de cellules libres. Le robot SHAKEY [Nilsson 69] opère sur un "graphe de visibilité" construit à partir des sommets des obstacles grossis (cf. figure 5.3). Chatila [Chatila 81] et Crowley [Crowley 85] définissent leurs graphes de connectivité sur la base d'un découpage polygonal de l'espace libre. O'Dunlaing et Yap [O'Dunlaing, Yap 82] utilisent un modèle de connectivité de l'espace libre basé sur la détermination des lignes situées à mi-distance des obstacles (diagramme du Voronoi [Drysdale 79]).

La problèmatique de la robotique mobile et la bibliographie associée est décrite dans [Giralt 83].

Appoche par construction de "couloirs de circulation":
Le travail de Brooks [Brooks 82a] apporte une dimension supplémentaire au problème, en traitant d'un mobile opérant dans un espace de type $\Re^2 \times O^1$. Sa méthode repose sur une modélisation originale de l'espace libre, formulée en termes

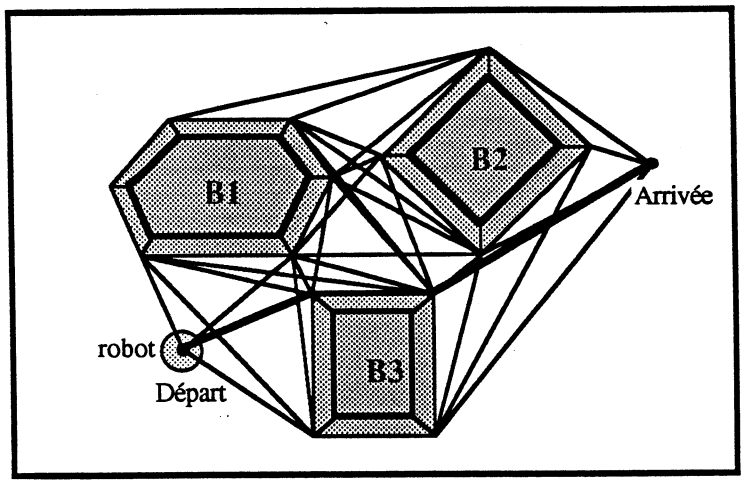


Figure 5.3: Graphe de visibilité utilisé par le robot SHAKEY.

d'un réseau de "cônes généralisés" [Binford 82]. Chaque cône symbolise un couloir de passage (freeway) entre deux obstacles. Son axe définit le lieu des déplacements possibles du point de référence du mobile, et sa section permet de calculer les intervalles d'orientations permises (cf. figure 5.4). Le passage d'un freeway à l'autre est réalisé au point d'intersection des axes; l'orientation du mobile doit alors être compatible avec les contraintes imposées par les deux freeways. La méthode des freeways a été étendue au cas d'un manipulateur PUMA 6 axes, transportant des charges en actionnant uniquement ses quatre premiers degrés de liberté (problème de caractéristique $\Re^3 \times O^1$). Les objets traités sont des prismes polygonaux droits, ce qui permet de raisonner sur des représentations bidimensionnelles associées successivement au bras, à l'avant-bras et à l'ensemble main-charge [Brooks 83]. Cette méthode donne de bons résultats lorsque le robot transporte un objet de petites dimensions dans un environnement peu encombré. Son extension à des situations plus générales (objets manipulés volumineux, nécessité d'autres degrés de liberté, contraintes plus fortes) semble difficilement réalisable.

Approche par "tranches d'orientations":
La planification de mouvements dans les espaces $\Re^2 \times O^1$ et $\Re^3 \times O^3$ a motivé d'autres recherches tendant à intégrer les rotations dans le modèle de raisonnement. La principale difficulté vient de ce que les techniques utilisées pour la construction de l'espace libre se prêtent mal à la conjonction de mouvements

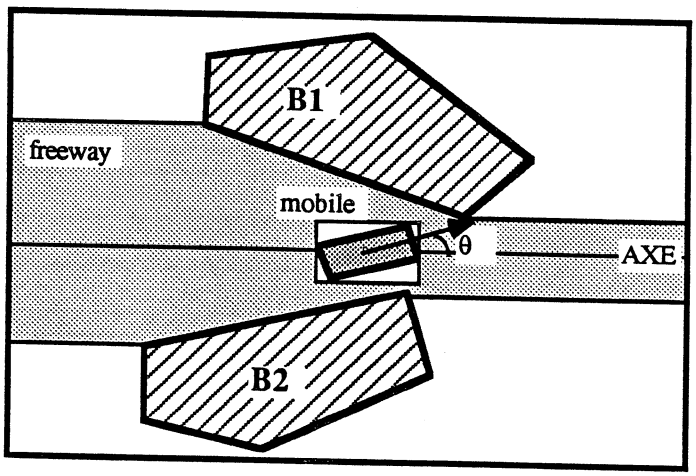


Figure 5.4: Représentation de l'espace libre à l'aide de cônes généralisés.

de translation et de rotation. Une première approche consiste à considérer les réorientations du mobile comme des opérations complémentaires qui se greffent sur certains déplacements. Cette approche a été utilisée par Lozano-Perez et Wesley [Lozano-Perez, Wesley 79], pour planifier les trajectoires d'un objet polygonal dans un espace de type $\Re^2 \times O^1$. Elle conduit à rechercher consécutivement les portions de trajectoires susceptibles de comporter des réorientations du mobile. Cette recherche est exécutée sur un ensemble de représentations isomorphes à $\Re^2$, construites sur la base d'une segmentation de l'espace des rotations. Plus tard, Lozano-Perez [Lozano-Perez 81] a généralisé cette approche afin de pouvoir traiter des problèmes réels de type $\Re^3 \times O^3$. L'algorithme implanté permet de calculer des trajectoires sûres pour un manipulateur cartésien (3 axes prismatiques et 3 axes rotoïdes). Mais ses performances se dégradent très rapidement lorsque les solutions recherchées comportent des rotations.

L'approche par "tranches d'orientations" est simple dans son principe, mais elle engendre une complexité algorithmique due essentiellement à la multiplication des représentations. Brooks et Lozano-Perez [Brooks, Lozano-Perez 83] ont tenté de supprimer ce défaut en développant une méthode exacte (à une résolution donnée) pour combiner les translations et les rotations du mobile; mais leur approche est limitée à un espace de type $\Re^2 \times O^1$. Donald [Donald 84] a implanté un algorithme complet, capable de résoudre des problèmes de caractéristique $\Re^3 \times O^3$.

Sa méthode opère sur une représentation exacte des C-surfaces (surfaces de dimension 5 dans l'espace des configurations) engendrées par les obstacles. Elle utilise des opérateurs locaux pour glisser sur ces C-surfaces et sur leurs intersections. Ce procédé permet d'engendrer des trajectoires en $(x, y, z, \varphi, \vartheta, \psi)$ pour contourner les obstacles de l'espace tridimensionnel. Mais l'algorithme implanté possède un degré de complexité qui rend le procédé impraticable dans des situations réelles.

Approche par "secteurs angulaires":
Les difficultés introduites par les rotations sont à l'origine d'un autre type d'approche, conduisant à privilégier les mouvements de réorientation du mobile. Cette approche tente d'isoler et de structurer les régions de l'espace cartésien qui présentent une homogénéité vis-à-vis des contraintes imposées par les obstacles. Chacune de ces régions est caractérisée par le fait que tout point intérieur applique les mêmes contraintes d'orientations sur le mobile (ce qui peut être vu comme une généralisation de la méthode des Freeways). Dans le cas d'un objet filiforme, le calcul des frontières peut être réalisé de manière exacte en recherchant les discontinuités engendrées par les arêtes des obstacles [Bo Zhang et al. 84]. L'extension à un mobile polygonal introduit d'autres discontinuités qui rendent la méthode exacte inapplicable. Il devient alors nécessaire de rechercher une approximation des régions frontières par le biais d'un découpage cellulaire de l'espace [Germain 84]. Dans ce cas, l'opération de segmentation est appliquée sur l'espace $\Re^2$ des translations. Ce principe de découpage de l'espace en régions homogènes, a été utilisé par Chochon [Chochon 86] pour structurer l'espace de travail d'une cellule flexible d'assemblage, en vue de la génération rapide de grands mouvements. L'algorithme implanté adresse uniquement le problème de la détermination des trajectoires de transfert permettant de passer d'un site à un autre. Il exploite un modèle structuré de l'environnement, construit par l'opérateur lors d'une phase d'initialisation du système.

Approche par discrétisation de l'espace articulaire:
Tous les travaux cités précédemment adressent des problèmes de caractéristiques $\Re^2$, $\Re^2 \times O^1$, $\Re^3 \times O^1$ ou $\Re^3 \times O^3$. Sauf quelques rares exceptions, ils ont donné lieu à des méthodes trop complexes ou pas assez générales pour être utilisables sur des manipulateurs réels [Lozano-Perez 85]. Cette situation a conduit les chercheurs à envisager d'autres solutions mieux adaptées aux problèmes de type n-DDL [Gouzenes 84] [Faverjon 84] [Laugier, Germain 85] [Germain 85] [Lozano-Perez 85]. Bien qu'elles diffèrent dans leurs mises en œuvre, ces solutions reposent toutes sur un même principe: la discrétisation hiérarchique des variables articulaires. Elles conduisent ainsi à raisonner explicitement dans l'espace articulaire du robot, alors que les approches antérieures opéraient principalement sur l'espace des configurations de l'outil terminal. Le procédé de discrétisation du problème est potentiellement applicable (de manière récursive) à un nombre quelconque de

degrés de liberté [Laugier, Germain 85]. En pratique, il ne permet pas de traiter raisonnablement plus de trois ou quatre articulations, à cause de la complexité algorithmique inhérente à tout problème général de planification de trajectoires [Schwartz, Sharir 82].

Domaines d'applications des méthodes précédentes:

Les méthodes locales sont en théorie mal adaptées au problème de la planification des trajectoires sûres. En effet, le caractère local des décisions ne permet pas de contrôler efficacement la structure globale des solutions proposées. De plus, le mode de raisonnement utilisé conduit assez souvent à des échecs. Par exemple, la présence de minima locaux de la fonction potentiel, peut bloquer le processus de recherche en l'absence d'heuristiques appropriées. Malgré tout, certains problèmes se prêtent bien à un traitement local. Un exemple typique est celui de la modification de trajectoires soumises à des contraintes locales. Cette situation se produit lorsqu'une trajectoire nominale est planifiée sur la base de considérations globales, avant d'être adaptée localement à certaines caractéristiques de l'environnement physique. Dans leurs approches respectives, Faverjon [Faverjon 86] et Donald [Donald 84] exploitent cette complémentarité, afin de réduire la complexité algorithmique des méthodes initialement proposées. L'idée de base consiste à faire collaborer une méthode globale de planification avec des "experts locaux" capables de compléter les trajectoires proposées. Les experts utilisés par Faverjon recherchent les mouvements qu'il convient d'exécuter pour sortir d'une situation difficile (concavité en particulier). Ceux de Donald permettent de suivre les C-surfaces construites antérieurement.

Les méthodes globales sont potentiellement mieux adaptées que les méthodes locales pour calculer des trajectoires sûres. Le fait qu'elles opèrent sur un modèle explicite de l'espace libre, permet de définir des algorithmes de recherche qui garantissent l'obtention d'une solution (lorsque celle-ci existe dans le sous-espace représenté). Cette solution peut de plus être choisie parmi d'autres en fonction de critères d'optimalité. Les approches globales possèdent par contre l'inconvénient de mettre en œuvre une masse importante de calculs géométriques. Afin de réduire les temps de calculs nécessaires, les algorithmes implantés utilisent différentes approximations pour représenter le robot et son environnement. Ces approximations sont compatibles avec les contraintes de la tâche, tant que les mouvements exécutés ne mettent pas en jeu des relations de contact ou de voisinage immédiat. Cette condition, qui marque généralement la limite de validité des méthodes globales, n'est pas vérifiée dans les environnements de début et de fin des trajectoires. C'est pourquoi les réalisations pratiques font très souvent appel à des heuristiques pour diminuer la complexité au voisinage de ces régions, et pour construire les portions de trajectoires correspondantes.

Compte tenu des connaissances actuelles, les méthodes locales restent les seules méthodes possibles lorsqu'il s'agit de résoudre des problèmes de complexité importante (nombreux d.d.l, espace très contraint).

5.4.3 Planification des opérations de montage:

Les problèmes posés par les incertitudes en robotique ont suscité des travaux de recherche, orientés suivant deux directions: la commande en présence d'incertitudes et la synthèse de mouvements fins. Les travaux liés à la commande ont pour objet de faire exécuter des mouvements qui exploitent au maximun la géométrie des pièces. De tels mouvements sont guidés par les informations sensorielles issues des contacts. Les autres travaux adressent le problème du développement des algorithmes qui permettront d'organiser ces mouvements en vue de la réalisation des tâches projetées. Ils reposent sur une analyse locale des incertitudes et des relations de contact.

Travaux sur la commande:
Les travaux portant sur la commande ont donné lieu au développement des notions de "compliance" [Whitney 77] et de "mouvement gardé" [Will, Grossman 75]. Les techniques qui en découlent, conduisent à utiliser implicitement les contraintes géométriques de la tâche pour guider l'exécution des opérations de montage. Les *mouvements gardés* permettent d'atteindre des positions précises caractérisées par des conditions sensorielles (situations de contact en particulier). Les *mouvements compliants* reposent sur le principe du maintien des relations de contact qui sont aptes à guider le mouvement. Ce dernier point a motivé un grand nombre de recherches orientées vers l'implantation de lois de commande intégrant un contrôle explicite des forces [Whitney 77] [Raibert, Graig 81] [Mason 82] [Merlet 86] [Reboulet, Robert 85]. Quelques uns de ces travaux ont conduit au développement d'instructions appropriées, implantées dans des langages de programmation de robots: langage LUCIFER [Giraud 83], langage LM [Gandon 86].

D'autres chercheurs ont étudié les effets des frottements, afin de mieux maîtriser les phénomènes de glissement, de blocage et de coincement. La plupart de ces études ont été réalisées dans le contexte de l'insertion cylindrique [Inoue 74] [Simunovic 75] [Whitney 82]. Mason [Mason 82] a débordé de ce cadre en considérant le problème plus général du comportement d'un objet soumis à certaines forces de manipulation. Sa méthode permet de dériver les conditions de succès de la tâche, à partir d'une analyse du but à atteindre et de l'état initial.

Synthèse de mouvements fins à partir de stratégies prédéfinies:
Les premières tentatives de synthèse automatique de mouvements fins étaient basées sur le concept de "schéma de programme" [Taylor 76] [Lozano-Perez 76].

Le principe consistait à analyser les contraintes de précision imposées par la tâche, afin de choisir puis de compléter des schémas types connus à priori. Les choix réalisés reposaient alors essentiellement sur une estimation des erreurs rencontrées. Ils conduisaient à spécifier quantitativement tous les paramètres des mouvements contenus dans le corps des procédures choisies. Plus tard, Brooks [Brooks 82b] a amélioré le procédé en propageant symboliquement les incertitudes dans le programme. Le mode de calcul induit permet d'estimer en tout point les erreurs engendrées par les actions antérieures; il permet aussi d'évaluer les valeurs initiales qu'il est possible d'affecter à certains paramètres, sans dégrader les chances de succès du programme. Les spécifications du système TWAIN [Lozano-Perez, Brooks 85] étaient basées sur ce type d'approche, mais aucune implantation significative n'a été réalisée. Une limitation importante du procédé réside dans la difficulté qu'il y a à apprécier certaines erreurs, et dans les techniques de propagation utilisées qui conduisent à surestimer très largement les incertitudes.

Les difficultés liées à l'évaluation précise des incertitudes ont motivé le développement d'une autre approche basée sur des techniques d'apprentissage [Dufay 83] [Dufay, Latombe 84]. Cette approche consiste à combiner à l'issue de plusieurs tentatives d'exécution, des stratégies partielles décrites par des règles expertes. Dans un premier temps, le système réalise plusieurs exécutions de la tâche en interaction avec les actionneurs et les capteurs du robot. Il utilise pour cela des connaissances expertes permettant d'analyser les situations rencontrées, et d'engendrer des actions correctives. La deuxième étape consiste à combiner les résultats des différents essais, afin d'obtenir un programme de mouvements fins aussi complet que possible. Le système procède pour cela à des transformations itératives du graphe obtenu par combinaison des traces d'exécution. Ce graphe représente alors la structure de contrôle du programme à synthétiser. Il peut comporter des cycles et des points de branchements multiples. Cette méthode permet de construire des programmes de mouvements fins fiables et efficaces. Elle n'est cependant pas capable d'élaborer des solutions nouvelles, ni d'analyser des situations symboliques non décrites explicitement dans la base de connaissances.

Les deux approches précédentes reposent sur l'utilisation d'un répertoire de stratégies de manipulation. Chaque stratégie représente une classe d'opérations susceptibles de résoudre un problème de montage particulier. Cette hypothèse de granularité de l'univers du montage facilite la tâche du processus de planification. Elle ne correspond cependant pas à la réalité de l'assemblage mécanique: une simple variation locale de la géométrie d'une pièce peut avoir des répercussions importantes sur la stratégie de montage utilisée. La complexité géométrique et mécanique qui en découle rend impossible la constitution d'un ensemble raisonnable de classes identifiables.

Synthèse de mouvements fins par raisonnement géométrique:
Une autre manière de procéder consiste à construire pas à pas le plan de montage, en raisonnant sur la géométrie de la tâche [Lozano-Perez et al. 83] [Valade 85] [Laugier, Theveneau 86]. Les bases formelles de cette approche ont été définies par Lozano-Perez, Mason et Taylor [Lozano-Perez et al. 83]. Elles ont été ensuite développées par Mason [Mason 84] et par Erdmann [Erdmann 84]. La méthode proposée opère dans l'espace des configurations de l'objet manipulé. Elle conduit à construire récursivement une séquence de mouvements compliants permettant de réaliser le montage. Cette séquence est obtenue en raisonnant à partir du but à atteindre. Le principe consiste à représenter l'objet manipulé par un point P, et l'objectif à atteindre par une surface G située sur le sous assemblage (cf. figure 5.5). La recherche des mouvements à exécuter repose alors sur le calcul des "pré-images" de G, c'est à dire sur le calcul des ensembles des positions de P qui permettent d'atteindre G en un seul mouvement de direction fixe. Lorsque P n'appartient à aucune des pré-images construites antérieurement, le procédé est appliqué récursivement jusqu'à l'obtention d'une séquence satisfaisante. Les effets des incertitudes et des frottements sont pris en compte dans l'algorithme en introduisant le concept de "pré-image forte", c'est à dire de pré-image permettant d'atteindre avec certitude la surface objectif.

La méthode des pré-images est potentiellement très puissante, mais elle fait surgir des problèmes théoriques qui sont loins d'être résolus. Les travaux les plus avancés dans cette voie sont ceux de Erdmann [Erdmann 84]. Ils apportent des solutions applicables à des problèmes de l'espace bidimensionnel, mais la généralisation de ces solutions à l'espace tridimensionnel semble difficilement envisageable: la complexité déjà exhibée par les algorithmes proposés laisse prévoir une explosion combinatoire dans des cas réels. C'est pourquoi nous avons développé une méthode permettant de réduire la taille du graphe de recherche en guidant heuristiquement le raisonnement géométrique, et en explorant uniquement les branches qui apparaissent comme les plus prometteuses [Laugier, Theveneau 86]. Ce procédé de planification des mouvements fins est décrit dans les chapitres 8 et 9.

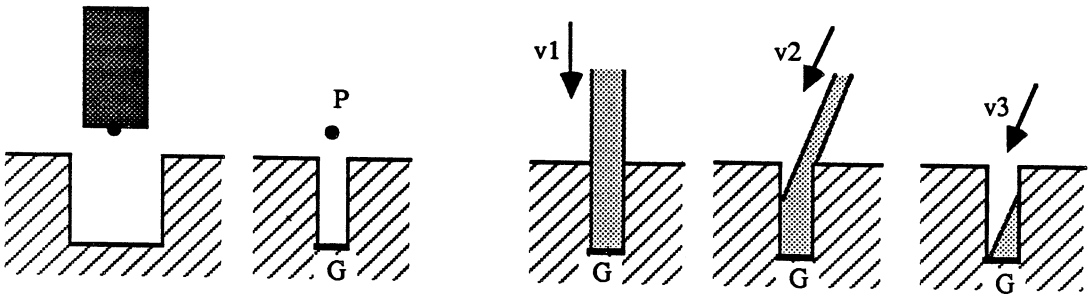


Figure 5.5: Illustration du concept de pré-image:
a- Reformulation du problème de l'insertion cylindrique.
b- Pré-images associées à un objectif G et à des directions V_i.

Chapitre 6

Modélisation de l'univers du robot

6.1 Modèles géométriques de base:

Les modèles géométriques de base utilisés pour la planification des mouvements du robot sont les mêmes que ceux utilisés pour la connexion CAO-robot (cf. paragraphe 3.4): représentation surfacique d'objets constitués de polyèdres, de parallélépipèdes, de cylindres, de cônes et de sphères; structures relationnelles modélisant les relations spatiales et les contraintes mécaniques. Les données numériques incluses dans ces modèles spécifient d'une part les caractéristiques géométriques des entités représentées (rayon de courbure des arêtes circulaires, rayon et hauteur des cylindres ...), et d'autre part les positions spatiales de ces entités (coordonnées des points dans les repères associés aux objets auxquels ils appartiennent, transformations géométriques définissant les positions et les orientations de ces objets).

Afin de faciliter la conduite des différentes formes de raisonnement, ces modèles ont été complétés par deux types de constructions permettant d'exhiber respectivement les propriétés d'encombrement spatial de la scène, et les caractéristiques morphologiques des objets. Nous parlerons alors de la *composante volumique* et

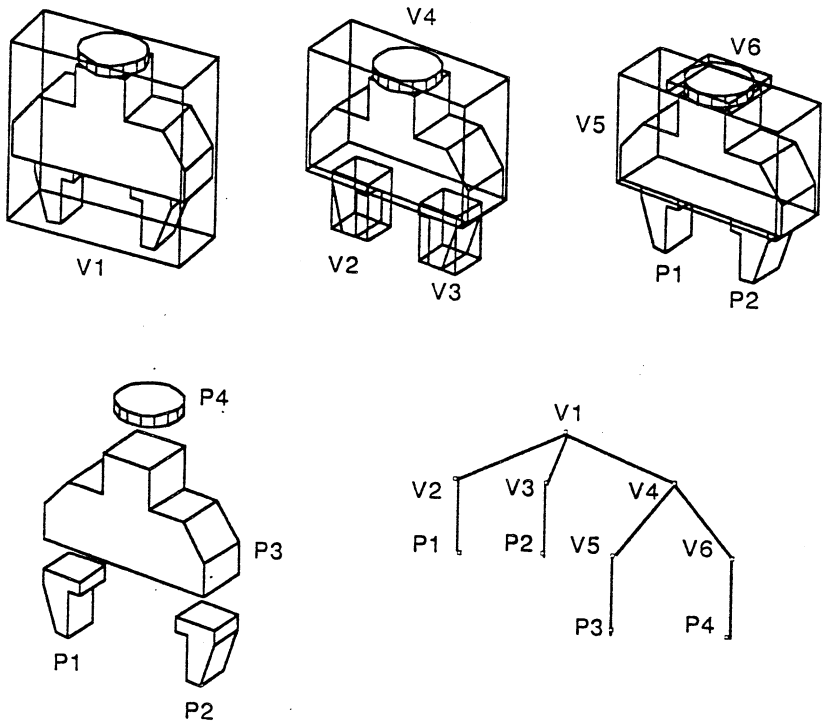


Figure 6.1: Exemple de hiérarchie de volumes englobants.

de la *composante topologique.*

La composante volumique est utilisée pour réduire la complexité du raisonnement spatial, chaque fois que les contraintes imposées par la tâche le permettent. Elle définit 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 6.1.

La composante topologique est utilisée par tous les raisonnements qui mettent en jeu des contacts. Elle combine une structuration spatiale des entités géométriques qui composent les objets, avec une caractérisation à tous les niveaux des positions relatives entité-matière. Les constructions de base sont de la forme (cf. figure 6.2):

(composant face arête sommet)

Elles sont complétées par des chaînages de type "winged- edge" [Braid 77] qui permettent de représenter explicitement les relations de voisinage dans le modèle (cf. figure 6.3). Nous verrons dans la suite que cette caractérisation de la *topologie locale* est très utile pour les raisonnements que nous avons mis en œuvre.

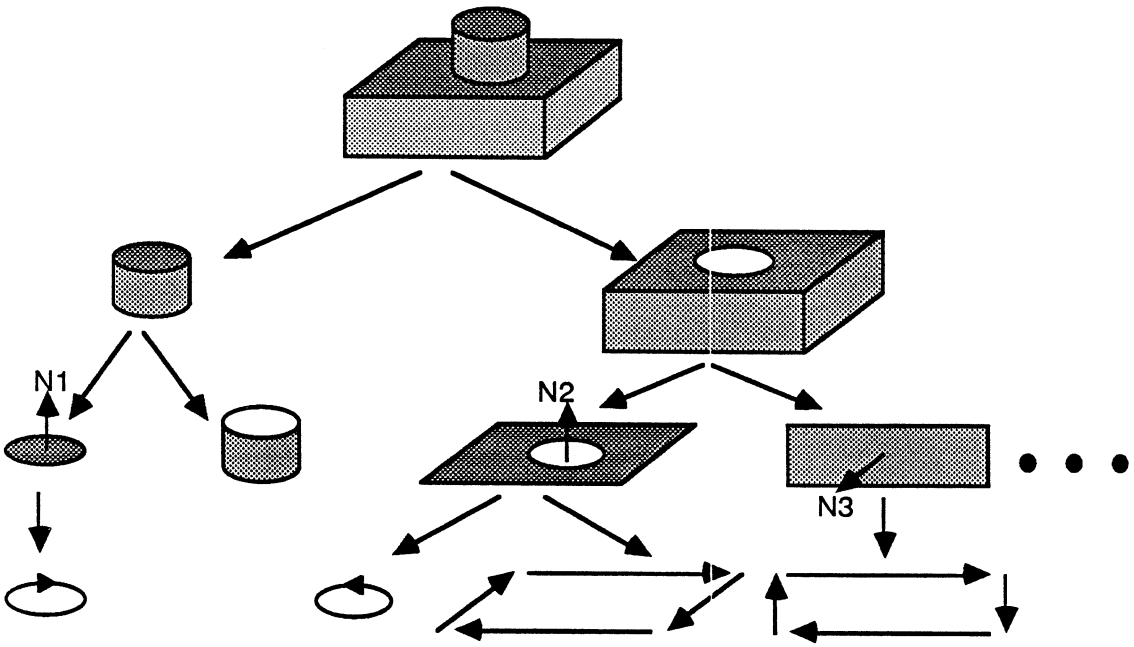


Figure 6.2: Exemple de structure topologique d'un objet.

Les informations liées à la localisation de la matière par rapport aux éléments du modèle sont codées comme suit:

- *Nature des composants.* Soit C un composant du modèle d'un objet O. Un prédicat $VIDE(C)$ indique si C est vide ou plein.
- *Normales extérieures aux faces planes.* Soit F une face plane du modèle d'un objet O. Un vecteur N_F normal à F et orienté vers l'extérieur de O, indique localement la position de la matière par rapport à F.
- *Orientation des arêtes avec la convention "matière à droite".* Soit A une arête appartenant à une face plane F du modèle d'un objet O. A est orientée dans le sens rétrograde sur le plan P_F défini par la normale extérieure N_F

de F. Cette convention permet de conserver la matière du "coté droit" de A, lorsque l'on se déplace sur A suivant l'orientation indiquée. Si A est une arête rectiligne, le vecteur V_A représente l'arête orientée avec la convention "matière à droite". Le vecteur $N_F \wedge V_A$ de P_F est alors orthogonal à A, et orienté vers l'extérieur de F. Cette propriété vectorielle sera utilisée dans la suite pour distinguer localement les situations de contact et d'inclusion (cf. chapitre 8).

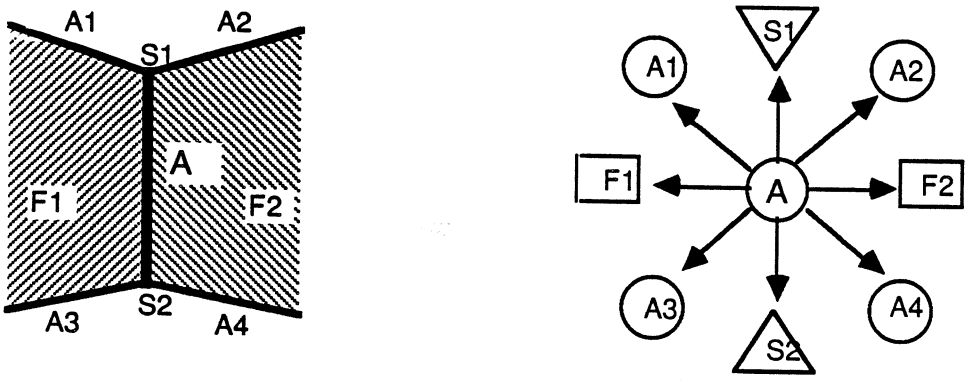


Figure 6.3: Représentation de la topologie locale au voisinage d'une arête.

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

6.2 Modélisation des incertitudes de position:

6.2.1 Problèmes posés et solutions possibles:

Le modèle que nous utilisons ne représente qu'une image imparfaite de l'univers du robot. Les principaux écarts proviennent des erreurs engendrées par les déplacements du robot, par les données transmises par les capteurs, par les systèmes d'alimentation et de positionnement des pièces mécaniques, et par les tolérances de fabrication.

Toutes ces erreurs ne sont jamais connues de manière exacte, mais il est en général possible d'obtenir une information sur leurs évolutions. Cette information

consiste très souvent en un domaine de variation associé à chaque terme d'erreur; elle peut aussi être représentée sous la forme d'une loi mathématique symbolisant un comportement statistique des erreurs. Une telle information représente *l'incertitude* associée à la grandeur considérée.

Seule *l'incertitude de position* est explicitement représentée dans notre modèle. Les incertitudes portant sur la forme des objets sont alors ignorées. Elles n'interviennent que dans la spécification des jeux de montage.

L'approche généralement utilisée en robotique d'assemblage consiste à définir l'incertitude sur un paramètre, comme l'ensemble des erreurs qui peuvent entâcher la valeur de ce paramètre. Par exemple, l'incertitude de position associée à un robot cartésien possédant trois axes, sera définie par le pavé $[-\Delta x + \Delta x] \times [-\Delta y + \Delta y] \times [-\Delta z + \Delta z]$, centré sur la position nominale P considérée. Ce pavé est alors appelé "volume d'incertitude" associé à P. Ce mode de représentation a été à la base de travaux ayant pour objectif d'évaluer les incertitudes de position en divers points d'un programme de manipulation [Taylor 76] [Brooks 82b] [Lozano-Perez, Brooks 85]. Il a également été utilisé par Brooks [Brooks 84] pour analyser les incertitudes associées aux déplacements d'un robot mobile.

Cette représentation ensembliste est attrayante dans son principe, car elle met en œuvre des procédures de calculs conceptuellement simples: toutes les opérations sur les incertitudes (composition, comparaison ...) sont traduites en termes de manipulations d'ensembles simples (union, intersection, projection ...). L'inconvénient majeur de cette approche réside dans la nature des représentations utilisées, qui conduisent à raisonner en permanence sur les erreurs maximales. Il est de ce fait impossible de prendre en compte certains phénomènes de compensations statistiques des erreurs. Cette méthode est cependant la seule possible, lorsque l'on veut s'assurer de la faisabilité d'une séquence d'opérations portant sur des objets physiques.

Une autre approche souvent utilisée pour estimer la position d'un robot mobile [Chatila, Laumond 85] ou pour résoudre des problèmes de localisation visuelle d'objets [Faugeras, Hebert 86] [Bolle, Cooper 86], consiste à associer une loi de probabilité aux grandeurs physiques sujettes à incertitude. On représentera par exemple la variation d'une erreur par une loi gaussienne normale, ayant pour valeur moyenne la valeur théorique du paramètre. Smith [Smith et al. 86] a ainsi développé un formalisme mathématique de type probabiliste, pour représenter et manipuler les incertitudes liées à un univers structuré d'objets. Ce formalisme utilise des matrices de covariance pour représenter les relations de dépendance des paramètres, afin de pouvoir combiner et propager les incertitudes de position. Il répond assez bien à la problématique de la robotique mobile. Son extension à

l'univers de la manipulation pose cependant quelques problèmes théoriques dus à la complexité des situations rencontrées, en particulier: traitement des contacts multiples et estimation des répartitions statistiques d'erreurs qu'il est possible de combiner pour satisfaire une contrainte imposée par la tâche.

Compte tenu des limitations théoriques de l'approche statistique et de la relative simplicité des opérations mises en jeu par l'autre méthode, nous avons choisi de représenter les incertitudes de position par des ensembles simples. Le modèle développé est décrit dans les paragraphes qui suivent.

6.2.2 Représentation des erreurs de position:

La position d'un objet A est classiquement définie par la transformation géométrique T_a qui permet de passer du repère de référence de l'univers au repère R_a de l'objet. Cette position représente une donnée théorique (appelée "position nominale") qui n'est jamais atteinte dans la réalité. Une *erreur* sur une position se traduit par un "écart" entre la position nominale et la position réelle. Compte tenu du formalisme utilisé pour localiser les objets dans l'univers du robot, cette erreur définit une transformation géométrique ϵ_a telle que:

$$R_a = Base \star T_a \star \varepsilon_a \qquad (1)$$

De la même manière, les positions relatives de deux objets A et B sont définies comme suit:

$$\begin{array}{ll} \text{position nominale:} & R_b = R_a \star T_{ab} \\ \text{position réelle:} & R_b = R_a \star T_{ab} \star \epsilon_{ab} \end{array} \qquad (2)$$

dans lesquelles T_{ab} représente la transformation géométrique qui permet de passer de R_a à R_b, et ϵ_{ab} est l'erreur de position du repère R_b par rapport au repère R_a.

Raisonner sur les positions des objets conduit donc à raisonner sur des repères liés par des transformations. Chacune de ces transformations comporte une composante nominale bien définie, et une composante variable qui représente le terme d'erreur. Suivant le type de raisonnement considéré, le procédé de calcul appliqué opère avec l'une ou l'autre de ces données. Dans le cas de la propagation symbolique des incertitudes, c'est le terme d'erreur qui est exploité.

6.2.3 Représentation des incertitudes de position:

Dans la représentation ensembliste que nous avons retenue, l'incertitude de position d'un repère R_b par rapport à un autre repère R_a est représentée par l'ensemble des transformations géométriques ϵ_{ab} qui permettent d'obtenir une position réelle de R_b en appliquant l'équation (2). Cette incertitude est notée $I(R_b/R_a)$. Elle

définit un sous ensemble de $\Re^3 \times O^3$, dans lequel $\Re^3$ est l'espace des translations et O^3 représente le groupe des rotations orthogonales.

Le principe de modélisation que nous avons choisi, conduit à considérer une erreur comme l'association de trois données: un vecteur de translation, un vecteur unitaire représentant l'axe de rotation, et un angle de rotation. Une incertitude de position peut alors être vue comme un sous-ensemble du produit cartésien $\Re^3 \times S(1) \times [-\pi \ +\pi]$, où $S(1)$ représente la sphère unité [Puget 85]:

$$I(R_b/R_a) = Tr \times U \times A$$

Dans la pratique, les sous-ensembles Tr, U et A ne sont pas quelconques du fait qu'ils proviennent de situations physiques mettant en jeu des relations de contact qui possèdent une certaine "isotropie" vis-à-vis des incertitudes. Ainsi, le terme Tr peut être représenté par des volumes de type sphère, disque ou segment de droite. Les rotations sont plus complexes à manipuler. Mais la réalité des contacts d'assemblage (contacts de nature essentiellement surfacique) conduit à fixer très souvent deux degrés de liberté en rotation; l'incertitude d'orientation résultante est alors localisée autour d'un axe. C'est pourquoi nous avons choisi d'approximer une incertitude par le plus petit ensemble englobant du type:

$$Tr \times U \times A$$

avec:
Tr = sphère, disque ou segment de droite
U = sphère ou vecteur unique
A = intervalle $[-\alpha \ +\alpha]$, α petit

Par exemple, un objet reposant sur un plan z=Cste présente une incertitude de position qui peut être représentée comme suit (cf. figure 6.4):

Tr = disque de rayon δ et d'axe z
U = vecteur unitaire porté par l'axe z
$A = [-\alpha \ +\alpha]$

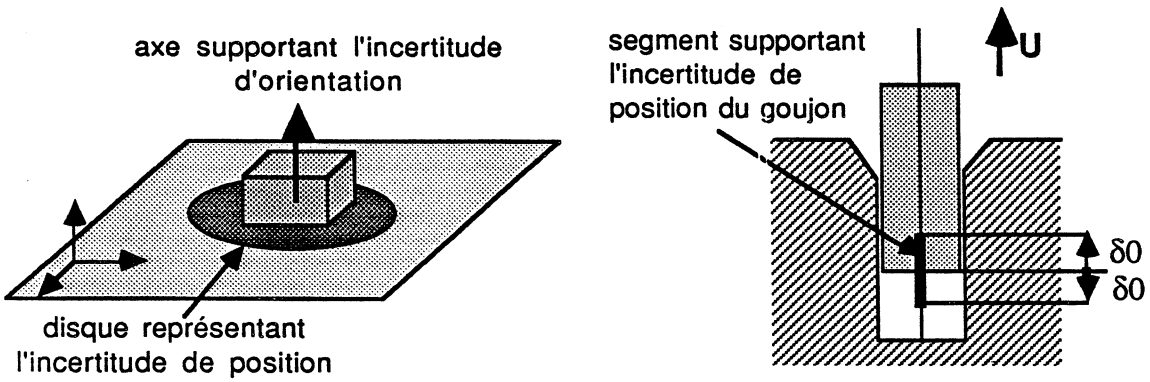
Si le contact est réalisé suivant une surface cylindrique d'axe z, l'incertitude de position peut être définie par les ensembles suivants (cf. figure 6.4):

Tr = segment porté par z, de longueur 2δ, et centré sur la position nominale de l'objet
U = vecteur unitaire porté par l'axe z
$A = [-\alpha \ +\alpha]$

Dans le cas d'un objet évoluant dans l'espace libre (outil terminal du robot en particulier), les volumes d'incertitudes sont maximum car aucun contact ne permet d'en réduire certaines composantes:

Tr = sphère de rayon δ centrée sur la position nominale de l'objet
U = sphère unitaire $S(1)$
$A = [-\alpha \; +\alpha]$

Remarque: Ce type d'approximation se justifie par l'importante simplification qu'il apporte au niveau des calculs de propagation des incertitudes. Il se traduit par contre par une perte d'information et par une surcontrainte des incertitudes dans certaines situations physiques (contacts linéiques en particulier). Dans l'assemblage mécanique, de telles situations représentent très souvent une étape intermédiaire vers la réalisation de situations plus robustes, par exemple: passage d'un contact linéique à un contact surfacique. L'approximation appliquée est alors sans réelle conséquence sur la suite, du fait de la réduction d'incertitude apportée par le contact surfacique.



contact de type plan **contact de type cylindrique**

Figure 6.4: Représentation des incertitudes de position associées à un contact de type plan et à un contact de type cylindrique.

6.2.4 Structure de l'univers du robot:

Les structures spatiales utilisées en robotique combinent toujours une représentation des positions des objets avec une représentation des relations qui lient ces objets entre-eux (cf. paragraphe 3.4). Un *état de l'univers* du robot est alors représenté par un graphe orienté, dans lequel chaque nœud symbolise un repère de localisation d'objet, et chaque arc représente une transformation géométrique

complétée par l'incertitude qui lui est associée. La structure de base de ce graphe est une arborescence ayant pour racine le repère de référence Ro de l'univers. Elle permet de définir la position de chaque objet dans un référenciel fixe connu du robot. Les autres arcs représentent des relations entre objets, créées par des opérations du robot (perception, mise en contact ou mise en correspondance d'entités géométriques). Ils permettent de raisonner sur les positions relatives des objets concernés.

Cette structure dynamique est fondamentale pour le traitement des incertitudes, car elle permet de les calculer aux endroits où elles sont significatives. Par exemple, la réalisation d'un contact entre deux objets A et B se traduit par une *réduction de l'incertitude relative* de A et de B. Il est donc naturel de représenter explicitement cette information dans la structure, et d'en propager ensuite les effets sur les autres arcs concernés, par exemple (cf. figure 6.5):

Structure avant contact:

$$Arc(R_o, R_a) = \{T_{oa}, I(R_a/R_o)\}$$
$$Arc(R_o, R_b) = \{T_{ob}, I(R_b/R_o)\}$$

Structure après contact:

$$Arc(R_o, R_a) = \text{supprimé}$$
$$Arc(R_b, R_a) = \{T_{ba}, I(R_a/R_b)\}$$
$$Arc(R_o, R_b) = \{T_{ob}, I(R_b/R_o)\}$$

Dans cette modification du modèle, le calcul de la transformation T_{ab} est exécuté en composant la transformation associée au déplacement exécuté, avec celles rencontrées sur le chemin reliant les objets A et B dans le graphe antérieur. Le calcul de l'incertitude $I(R_a/R_b)$ est exécuté de manière similaire, en utilisant des opérateurs appropriés. Ces opérateurs sont décrits dans le paragraphe qui suit.

La suppression du lien $Arc(R_o, R_a)$ est due au fait que la valeur de l'incertitude $I(R_a/R_o)$ dépend alors directement des termes $I(R_a/R_b)$ et $I(R_b/R_o)$.

Les différentes modifications apportées au modèle par les actions du robot sont discutées dans le chapitre 9.

6.2.5 Opérateurs associés au modèle:

Quatre types d'opérateurs géométriques sont nécessaires pour combiner et propager les incertitudes au travers de la structure précédente: les opérateurs d'inversion, de composition, de projection et de conjonction. Les deux premiers opérateurs sont utilisés par le système pour calculer les incertitudes qui résultent de la combinaison de données déjà présentes dans le modèle. L'opérateur de projection est employé pour réduire les composantes d'incertitudes affectées par

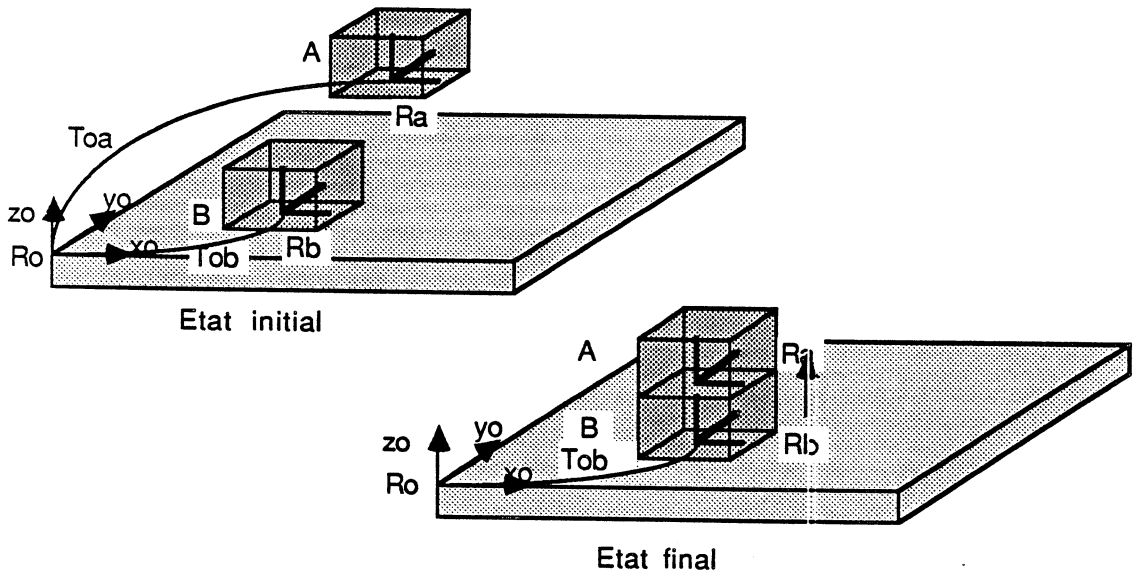


Figure 6.5: Exemple de modification des arcs du modèle après réalisation d'un contact.

des relations de contact. Celui de conjonction permet de calculer l'incertitude résultante, après application d'une opération sensorielle.

- *Inversion d'incertitudes*: On note I^{-1} l'opérateur d'inversion permettant de calculer $I(R_b/R_a)$ en fonction de $I(R_a/R_b)$:

$$I(R_b/R_a) = I^{-1}(R_a/R_b)$$

- *Composition d'incertitudes*: On note $*$ l'opérateur de composition permettant de calculer $I(R_a/R_c)$ à partir de $I(R_a/R_b)$ et de $I(R_b/R_c)$:

$$I(R_a/R_c) = I(R_a/R_b) \star I(R_b/R_c)$$

- *Projection d'incertitudes*: On note $Proj(I(R_a/R_b),(E,V,B))$ l'opérateur permettant de "projeter" l'incertitude $I(R_a/R_b)$ sur le sous ensemble $E \times V \times B$ de l'espace $\Re^3 \times S(1) \times [-\pi \ +\pi]$:

$$I^*(R_a/R_b) = Proj(I(R_a/R_b),(E,V,B))$$

Tr^*, U^*, A^* sont respectivement les projections de Tr sur E, de U sur V et de A sur B.

$I(R_a/R_b)$ est l'incertitude présente dans le modèle avant réalisation de la relation de contact.

- *Conjonction d'incertitude*: On note $\wedge$ l'opérateur permettant de calculer la conjonction de deux incertitudes portant sur un même arc $Arc(R_b, R_a)$ du modèle:

$$I^*(R_a/R_b) = I(R_a/R_b) \wedge I_{capteur}(R_a/R_b)$$

Tr^*, U^*, A^* sont les intersections des ensembles correspondants de $I(R_a/R_b)$ et de $I_{capteur}(R_a/R_b)$.
$I(R_a/R_b)$ est l'incertitude présente dans le modèle avant utilisation du capteur; $I_{capteur}(R_a/R_b)$ est l'incertitude qui résulte de la mesure faite par le capteur.

Les calculs sur lesquels reposent ces quatre opérateurs, mettent en jeu des opérations qui portent sur des ensembles de vecteurs: somme de Minkowski, intersection, application d'une rotation sur un ensemble, application d'un ensemble de rotations sur un ensemble de vecteurs... Une description détaillée de ces calculs n'étant pas nécessaire pour la compréhension de la suite, nous renvoyons le lecteur intéressé par cet aspect à [Puget 85]. Dans la pratique, tous ces calculs sont réalisés à partir des ensembles élémentaires (sphères, disques et segments de droites) utilisés pour approximer les volumes d'incertitudes.

Prenons par exemple le cas d'un cube A que le robot vient poser sur un cube B placé sur une table (cf. figure 6.5). Soient R_a et R_b les repères associés respectivement aux objets A et B; R_o est le repère de référence, choisi de manière à ce que le plan X_oOY_o coïncide avec la surface de la table. Si l'on fait abstraction du robot, les informations codées dans le modèle avant exécution de l'opération, sont les suivantes:

$$Arc(R_o, R_b) = \{T_{ob},\ I(R_b/R_o) = D(\delta, z_o) \times \{z_o\} \times [-\alpha\ \ +\alpha]\}$$
$$Arc(R_o, R_a) = \{T_{oa},\ I(R_a/R_o) = S(\delta') \times S(1) \times [-\alpha'\ \ +\alpha']\}$$

où $D(\delta, z_o)$ est un disque de rayon δ et d'axe z_o, et $S(\delta')$ est une sphère de rayon δ' centrée sur la position nominale de R_a.

Supposons pour simplifier, que le déplacement du cube A n'introduit aucune erreur supplémentaire. après exécution de l'opération, le modèle est alors transformé comme suit:

$$Arc(R_o, R_b) = \text{inchangé}$$
$$Arc(R_o, R_a) = \text{supprimé}$$
$$Arc(R_b, R_a) = \{T_{ba},\ I^*(R_a/R_b)\}$$

avec:

$$I(R_a/R_b) = I(R_a/R_o) * I^{-1}(R_b/R_o)$$
$$I(R_a/R_b) = \{S(\delta') \times S(1) \times [-\alpha' + \alpha']\} * \{D(\delta, z_o) \times \{z_o\} \times [-\alpha + \alpha]\}$$
$$I(R_a/R_b) = S(\delta + \delta') \times S(1) \times [-(\alpha' + \alpha) + (\alpha' + \alpha)]$$

$$I^*(R_a/R_b) = Proj(I(R_a/R_b), (plan\, x_o O y_o, \{z_o\}, [-\pi + \pi])) \}$$
$$I^*(R_a/R_b) = D(\delta' + \delta, z_o) \times \{z_o\} \times [-(\alpha' + \alpha) \quad (\alpha' + \alpha)]$$

6.3 Espace des configurations:

6.3.1 Définitions et notations:

Les définitions et les notations introduites dans ce paragraphe dérivent de celles utilisées dans [Lozano-Perez 81].

Définition 6.3.1 *Soit A un système mobile constitué de l ($l \geq 1$) composants rigides évoluant dans un espace $\Re^k$ ($k = 1, 2\, ou\, 3$). Une configuration c de A est un ensemble de paramètres de cardinalité minimale, permettant de spécifier sans ambiguité la position et/ou l'orientation dans $\Re^k$ de chaque composant rigide de A.*

Une configuration c de A peut ainsi être représentée par un vecteur dans un espace de dimension n. Nous dirons alors que A possède n degrés de liberté (d.d.l), et que chaque paramètre de c spécifie la valeur d'un d.d.l de A.

Nous noterons $A(c)$ la "posture" du système A dans $\Re^k$, lorsque A est en configuration c. Si chaque composant rigide A_i ($i = 1, \cdots l$) de A est représenté dans $\Re^k$ par un repère cartésien, $A(c)$ définit les l transformations géométriques qui permettent de passer du repère de base de $\Re^k$, aux repères associés aux composants A_i.

Définition 6.3.2 *On appelle* **espace des configurations** C_A *d'un système mobile A, l'ensemble des valeurs possibles du vecteur de configuration c. On appelle* **espace libre** EL_A, *l'ensemble des configurations c telles que $A(c)$ n'engendre pas de collision entre les composants rigides de A et les objets de l'environnement de A.*

EL_A est un sous-ensemble de C_A, et C_A est un ensemble de dimension n défini par le produit cartésien $I_1 \times I_2 \cdots I_n$, dans lequel I_i représente l'ensemble des valeurs possibles du i-ème paramètre de c.

Définition 6.3.3 *Soit B un objet de l'environnement susceptible de s'opposer à certains mouvements de A. On appelle C-obstacle $CO_A(B)$ le sous-ensemble de C_A défini par:*

$$CO_A(B) = \{c \in C_A : \quad A(c) \cap B \neq \emptyset\}$$

Les C-obstacles sont des hypervolumes de dimension n, dont la caractérisation exacte pose souvent des problèmes de complexité algorithmique difficiles à résoudre. Ce point est développé dans le chapitre 7.

Partant de cette dernière définition des C-obstacles, il est possible de caractériser l'espace libre EL_A comme suit:

$$EL_A = C_A - \cup_{i=1}^{m} CO_A(B_i)$$

Cette formulation de EL_A que nous utiliserons pour construire une représentation de l'espace libre (cf. chapitre 7).

Définition 6.3.4 *On appelle trajectoire T_A une courbe de C_A, et trajectoire sûre T_A^* une courbe de EL_A.*

Le problème de la planification de mouvements pour un système mobile A, sera donc ramené à la recherche de courbes particulières dans EL_A. Les techniques utilisées pour ce calcul sont décrites dans le chapitre 7.

6.3.2 Caractérisation analytique:

Objets rigides:

Soit A un objet rigide se déplaçant dans un espace E de dimension k ($k = 1, 2$ *ou* 3). Nous définissons une configuration c de A comme suit:

- Si $k = 1$, $c \in \Re$.
- Si $k = 2$, $c \in \Re^2 \times O^1$.
- Si $k = 3$, $c \in \Re^3 \times O^3$.

où $\Re^m$ est l'espace cartésien de dimension m, et O^p est le groupe des rotations orthogonales.

Soit R le repère associé à A. Une configuration c de A sera respectivement caractérisée par un vecteur (x), $(x\,y\,\gamma)$ ou $(x\,y\,z\,\alpha\,\beta\,\gamma)$, dans lequel x, y et z représentent les coordonnées du centre de R dans le repère de référence de E, et α, β et γ définissent les rotations exécutées autour des directions X, Y te Z. La figure 6.6 illustre ce mode de représentation.

Lorsque certains d.d.l ne sont pas considérés pour quelques raisons que ce soit (nature du problème traité, contraintes imposées par la tâche ...), les configurations d'un mobile tridimensionnel sont alors représentées par des éléments de l'ensemble $\Re^m \times O^p$, avec $m, p \in \{0, 1, 2, 3\}$. Si $P = 0$ (pas de rotation possible), l'espace considéré est $\Re^m$; dans le cas ou $m = 0$, il est représenté par O^p. Ce

formalisme est utilisé lors de la planification des mouvements de saisie et de montage (la structure articulaire du bras étant volontairement ignorée à ce niveau de planification). Par exemple, les contraintes imposées par une opération de saisie exécutée suivant deux faces planes parallèles conduiront à représenter les configurations possibles de la pince dans un espace de type $\Re^2 \times O^1$; $\Re^2$ définit alors les deux translations laissées libres par les contacts pince-objet, et O^1 représente les rotations que l'on peut appliquer autour d'un axe perpendiculaire aux faces saisies.

Remarque: Ce mode de représentation peut être étendu au cas des robots cartésiens qui combinent trois articulations prismatiques et trois articulations rotoïdes. Le triplet $(x\,y\,z)$ correspond alors aux trois premières valeurs articulaires, et le triplet $(\alpha\,\beta\,\gamma)$ définit les trois rotations exécutées par les derniers d.d.l du bras.

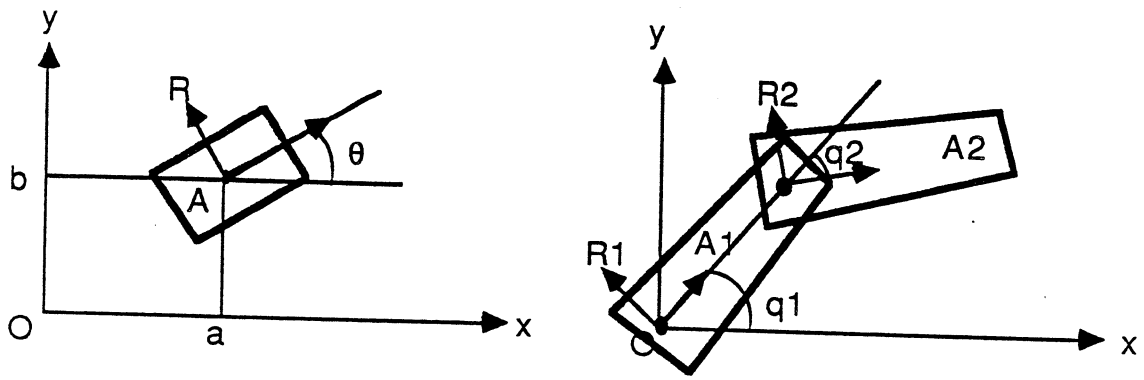


Figure 6.6: Configuration d'un objet bidimensionnel:
a- Objet rigide en translation et en rotation: $c = (a\ b\ \vartheta)$.
b- Objet articulé comportant deux d.d.l en rotation: $c = (q_1\ q_2)$.

Structures articulées:

Le système mobile A est maintenant une chaîne cinématique ouverte, composée de n corps articulés $A_1, A_2 \cdots A_n$. Ces corps sont naturellement ordonnés de la base A_1 de la structure, vers son extrémité A_n.

Une configuration de A est alors définie par un vecteur $(q_1\, q_2 \cdots q_n)$, dans lequel q_i représente une variable articulaire de type rotoïde ou prismatique:

$c = (q_1\, q_2 \cdots q_n)$
avec:
$(q_1\, q_2 \cdots q_n) \in I_1 \times I_2 \cdots I_n$.
I_i = domaine de variation de q_i.

Dans cette représentation, chaque variable est définie relativement au corps qui précède dans la structure (cf. figure 6.6). Cette définition relative des q_i permet d'appliquer plus facilement les algorithmes récursifs utilisés pour planifier les mouvements du bras du robot (cf. chapitre 7). La position dans l'espace cartésien du i-ème composant A_i est alors définie par le vecteur $(q_1\, q_2 \cdots q_i)$.

L'espace des configurations C_A de A est un espace de dimension n, qui n'est généralement pas isomorphe à un espace Euclidien de type $\Re^n$ du fait de la présence des rotations:

- dans le cas d'un robot cartésien à trois d.d.l, il est constitué d'un sous-ensemble de $\Re^3$;
- dans le cas d'un robot cartésien à six d.d.l, il est défini par un sous-ensemble de $\Re^3 \times O^3$;
- dans le cas d'un robot comportant six d.d.l de type rotoïde, il est caractérisé par un sous-ensemble du produit $C \times C \times C \times C \times C \times C$, dans lequel C représente le cercle trigonométrique (ce produit est aussi connu sous le nom de "6-tore").

La formulation générale de C_A est essentiellement utilisée pour la planification des grands mouvements qui mettent en jeu l'ensemble des éléments du bras. L'espace des configurations considéré pour la planification et la commande des autres mouvements est l'espace $\Re^3 \times O^3$. Cet espace représente l'espace des configurations de l'outil terminal du robot. Un "changeur de coordonnées" permet alors de passer de $\Re^3 \times O^3$ à l'espace articulaire C_A (cf. chapitre 2).

Erdmann [Erdmann 84] a montré que l'espace $\Re^3 \times O^3$ pouvait être approximé par un espace Euclidien, en représentant chaque groupe de rotation O^1 par un intervalle $[-\pi\ +\pi]$ de $\Re$ (ce qui conduit à éliminer les rotations qui mettent en jeu des angles plus grands que π, ou plus petits que $-\pi$). Afin d'avoir un système de coordonnées "homogène" vis à vis des translations et des rotations, les trois angles α, β et γ sont respectivement représentés par les quantités $\alpha\rho_\alpha$, $\beta\rho_\beta$ et $\gamma\rho_\gamma$; ρ_α, ρ_β et ρ_γ sont les "rayons de gyration" définis par les trois axes de rotation A_α, A_β et A_γ du poignet du robot. Une configuration du robot est alors représentée par un vecteur $(x\ y\ z\ \alpha\rho_\alpha\ \beta\rho_\beta\ \gamma\rho_\gamma)$, et la distance $d(q_1, q_2)$ peut être définie comme

la distance Euclidienne dans $\Re^6$.

Nous verrons dans le paragraphe qui suit, que cette représentation est bien adaptée à l'expression des mouvements du robot, du fait qu'elle permet de traiter de manière homogène les six paramètres de position de la pince. Nous noterons $\xi(\Re^3 \times O^3)$ cet espace.

6.4 Modélisation des mouvements du robot:

6.4.1 Principaux éléments du modèle:

Le système de commande du robot réel possède certaines caractéristiques qu'il est nécessaire de connaître, afin de pouvoir planifier des mouvements qui réalisent vraiment l'objectif visé. Ces caractéristiques sont utilisées par le système de planification pour juger de la faisabilité des actions planifiées.

Quatre types d'informations sont ainsi représentés dans le modèle:

- *Les forces de frottement.* Ces forces sont engendrées par les contacts entre le robot (ou la pièce manipulée) et les objets de l'environnement. Elles ont pour effet de provoquer un glissement sur la surface de contact, ou de stopper le mouvement en cours. Ces forces sont représentées par des ensembles appelés "cônes de frottements".

- *Les erreurs de commande et de perception.* Ces erreurs interviennent dans la spécification des objectifs à atteindre. Elles sont représentées par leurs bornes maximun.

- *La commande.* Le type de commande utilisé par le robot réel intervient dans la spécification des mouvements. Les modèles mis en œuvre sont à base d'équations différentielles qui mettent en jeu le temps, les configurations du robot et les forces exercées.

- *Les conditions d'arrêt.* Ces conditions spécifient les situations physiques dans lesquelles le robot doit se trouver à l'issue du mouvement exécuté. Elles sont définies par des expressions symboliques qui mettent en jeu des données sensorielles.

Les modèles que nous utilisons pour représenter les informations précédentes sont relativement simples. Ils ont été choisis de manière à répondre à la classe de problèmes traités: montages mécaniques présentant des contacts clairement discernables à l'aide des moyens perceptifs du robot. D'autres applications pouvant engendrer des situations ambigües, nécessiteraient de développer des modèles plus complets. Une étude théorique de ce problème est présentée dans [Erdmann 84].

6.4.2 Les frottements:

Lorsque le robot entre en contact avec un objet de l'environnement, cet objet exerce une force de réaction qui s'oppose au mouvement. Cette force comporte une composante normale issue de la force appliquée, et une composante tangentielle créée par les frottements. D'après la loi de Coulomb [Baumeister 78], les forces de réaction engendrées par un contact ponctuel sont toutes contenues dans un cône d'axe normal à la surface de contact, et d'angle $\phi = \arctan 2\mu$. Ce cône est appelé *cône de frottement*, et le paramètre μ est appelé coefficient de frottement (cf. figure 6.7). La valeur du coefficient μ dépend de la nature des matériaux en présence, et de la vitesse de déplacement du point de contact. Dans un but de simplification, nous considérerons que cette valeur reste constante quelle que soit la vitesse appliquée. Cette approximation n'est pas pénalisante dans le sens où les mouvements planifiés sont toujours choisis de manière à engendrer des forces de réactions proches des normales ou des tangentes aux surfaces de contacts (cf. chapitre 8).

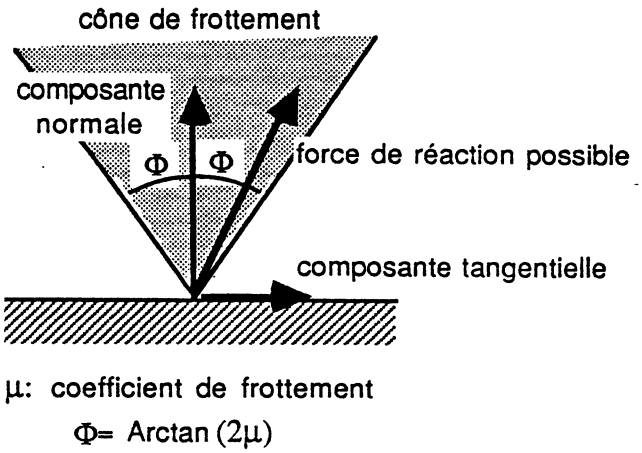


Figure 6.7: Cône de frottement associé à un contact ponctuel.

Dans le cas où la force F appliquée par le robot est contenue dans le cône de frottement, la force de réaction est égale et opposée à F; le contact créé a alors pour effet de stopper le mouvement du robot. Dans le cas contraire, la force résultante est tangente à la surface de contact; elle indique la direction suivant laquelle le robot va glisser (cf. figure 6.8). Ces propriétés seront exploitées par le système pour vérifier que les mouvements compliants sont corrects.

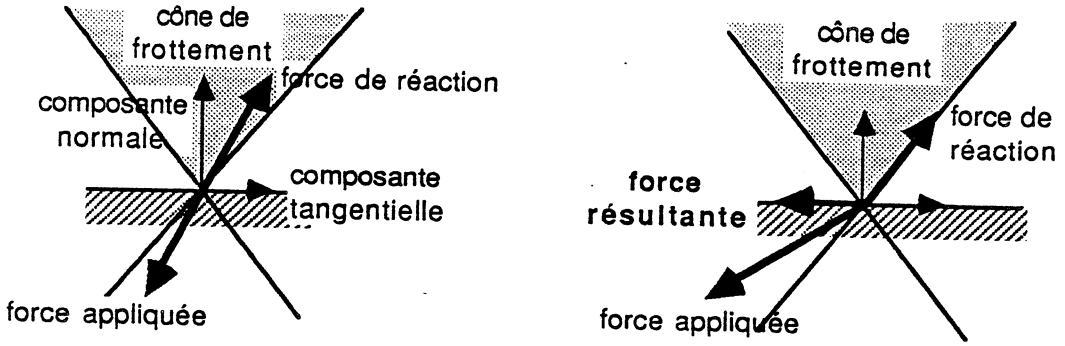


Figure 6.8: Calcul de la force résultante en présence de frottements:
a- La force appliquée est annihilée par la force de réaction. Le robot s'arrête.
b- La composante tangentielle de la force appliquée n'est pas annulée par la composante tangentielle de la force de réaction. Le robot glisse sur la surface de contact.

6.4.3 Les erreurs de commande et de perception:

Erreurs de commande:

Une erreur de commande représente une inaptitude du robot à exécuter correctement une commande de position ou de vitesse. Ces erreurs sont modélisées par leurs valeurs maximales:

- $\varepsilon_{p,t}$: distance maximum entre une position $(x\,y\,z)$ commandée, et la position réellement atteinte par le point de référence du robot.
- $\varepsilon_{p,r}$: angle maximum entre une orientation $(\alpha\,\beta\,\gamma)$ commandée, et l'orientation réellement atteinte par le repère de référence du robot.
- $\vartheta_{v,t}$: angle maximum entre le vecteur vitesse $(\dot{x}\,\dot{y}\,\dot{z})$ commandé, et la vitesse réellement exécutée par le point de référence du robot.
- $\vartheta_{v,r}$: angle maximum entre le vecteur de vitesse angulaire $(\dot{\alpha}\,\dot{\beta}\,\dot{\gamma})$ commandé, et la vitesse angulaire réellement exécutée par le repère de référence du robot.

Buckley [Buckley 87] a montré que ces erreurs pouvaient être représentées par des distances Euclidiennes dans $\xi(\Re^3 \times O^3)$:

- Erreur de position: $\varepsilon_p = \sqrt{\varepsilon_{p,t}^2 + \varepsilon_{p,r}^2}$
- Erreur de vitesse: $\varepsilon_v = \sqrt{\varepsilon_{v,t}^2 + \varepsilon_{v,r}^2}$
  avec: $\vartheta_v = \arctan \varepsilon_v$
  $\varepsilon_{v,i} = \tan \vartheta_{v,i} \quad i = t \ \ ou \ \ r$

Si p est une configuration de $\xi(\Re^3 \times \vee^3)$, l'ensemble des valeurs possibles de p après un mouvement du robot est représenté par une boule ouverte $B(p, \varepsilon_p)$, de centre p et de rayon ε_p. On dira alors que p est une ***position du robot.***

L'ensemble des orientations possibles du vecteur vitesse v est représenté par un cône d'axe v et d'angle ϑ_v.

Erreurs de perception:

Une erreur de perception est l'écart qui existe entre la valeur mesurée par le capteur et la valeur réelle. Cette valeur peut représenter une position du robot (position et orientation de la pince), ou un torseur de force. Les erreurs de perception sont modélisées par leurs valeurs maximales:

- $\varepsilon_{m,t}$: distance maximum entre une position $(x\,y\,z)$ mesurée, et la position réelle du point de référence du robot.
- $\varepsilon_{m,r}$: angle maximum entre une orientation $(\alpha\,\beta\,\gamma)$ mesurée, et l'orientation réelle du repère de référence du robot.
- $\vartheta_{f,t}$: angle maximum entre le vecteur de force $(F_x\ F_y\ F_z)$ mesuré, et la force réellement appliquée sur le point de référence du robot.
- $\vartheta_{f,r}$: angle maximum entre le vecteur de moments $(M_x\ M_y\ M_z)$ mesuré, et le moment réellement appliqué sur le point de référence du robot.

Comme précédemment, ces erreurs peuvent être représentées de la manière suivante dans l'espace $\xi(\Re^3 \times O^3)$:

- Erreur de mesure de position: $\varepsilon_m = \sqrt{\varepsilon_{m,t}^2 + (\rho \cdot \varepsilon_{m,r})^2}$
- Erreur de mesure de force: $\varepsilon_f = \sqrt{\varepsilon_{f,t}^2 + (\varepsilon_{f,r}/\rho)^2}$
  avec: $\vartheta_f = \arctan \varepsilon_f$
  $\varepsilon_{f,i} = \tan \vartheta_{f,i} \quad i = t \ \ ou \ \ r$
  $\rho = max(\rho_\alpha, \rho_\beta, \rho_\gamma)$

6.4.4 Modèle de commande:

Modèles possibles:

La commande du mouvement dans l'espace libre (pas de contacts) est réalisée dans la plupart des systèmes existant à partir de modèles du type:

commande en position:	$x = x_c$	(type 1)
commande en vitesse:	$\dot{x} = \dot{x}_c$	(type 2)
commande en acc	$\ddot{x} = \ddot{x}_c$	(type 3)

dans lesquels x_c, $\dot{x}_c$ et $\ddot{x}_c$ sont les vecteurs qui représentent les positions, vitesses et accélérations commandées dans l'espace des configurations; x, $\dot{x}$ et $\ddot{x}$ sont les paramètres réels du mouvement.

Lorsque le robot est soumis à des forces extérieures, les lois de commande combinent les termes précédents avec des termes de nature dynamique. De nombreuses solutions ont été proposées dans la littérature (voir [Buckley 87]). Les plus courantes intègrent un ou plusieurs des modèles suivants:

generalized spring [Salisbury 80]:	$F = K(x - x_c)$	(type 1)
generalized damper [Whitney 77]:	$F = B(\dot{x} - \dot{x}_c)$	(type 2)
generalized mass (loi de Newton):	$F = M(\ddot{x} - \ddot{x}_c)$	(type 3)

dans lesquelles F est la force de réaction engendrée par l'environnement; K, B et M sont les matrices contenant respectivement les coefficients de "raideur", de "damping" et "d'inertie".

Modèle utilisé:

Nous avons choisi d'utiliser un modèle de type 1 pour les raisons suivantes:

- Il est linéaire d'ordre 0, ce qui correspond au système de commande dont nous disposons actuellement sur nos robots (cf. chapitre 2).
- La commande en position qu'il représente est simple à spécifier à l'aide des éléments géométriques disponibles dans le modèle.
- Il permet, dans le domaine d'utilisation que nous nous sommes fixé, de prévoir le comportement du robot:
  - réalisation d'une position dans l'espace libre avec une incertitude ε,
  - réalisation d'un contact ne nécessitant pas une précision de commande inférieure à ε,
  - exécution d'un mouvement compliant prenant appui sur une surface de glissement.

Ce choix de modèle n'a aucune conséquence directe sur les fonctions de planification. Il intervient uniquement au niveau de la détermination des paramètres numériques du mouvement. Afin d'être compatibles avec la commande, les choix faits à ce niveau doivent vérifier certaines conditions simples:

- Un contact sera créé si la position commandée est "intérieure" à la surface de contact d'une distance au moins égale à l'erreur maximum de commande ε_p.
- Un contact C provoquera un arrêt du mouvement si l'angle α existant entre la direction de déplacement et la normale en C à la surface de contact est plus petite que l'angle d'ouverture ϕ du cône de frottement, diminué d'une valeur θ_v correspondant à l'erreur maximum pouvant être commise sur la direction commandée: $\vartheta \leq \alpha < \phi - \theta_v$.
- Un contact C provoquera un glissement sur la surface de contact si α vérifie la relation: $\pi/2 - \theta_v < \alpha < \phi + \theta_v$.

Dans la pratique, ces conditions conduisent à choisir des mouvements dont les directions sont proches des perpendiculaires ou des tangentes aux surfaces de contact, et dont les objectifs sont situés légèrement en retrait des positions souhaitées. Les cas plus complexes nécessitant une étude plus approfondie des phénomènes de frottement ne sont pas pris en compte par le système (la probabilité de rencontrer ces situations dans un assemblage mécanique étant faible).

6.4.5 Les conditions d'arrêt:

Notion de conditions d'arrêt:

Les conditions d'arrêt associées à une commande de mouvement spécifient les états du système robot-environnement qui indiquent la fin du déplacement. Elles correspondent à une généralisation de la notion de "garde" introduite antérieurement par Will et Grossman [Will, Grossman 75], et utilisée de manière primitive dans de nombreux systèmes de programmation de robots.

Dans sa forme générale, l'expression d'une condition d'arrêt peut faire appel à des informations diverses portant sur l'état instantané du robot et sur le mouvement en cours. Une étude des problèmes posés par l'interprétation de telles expressions est présentée dans [Mason 84] [Erdmann 84]. En ce qui nous concerne, nous limiterons la formulation utilisée au traitement des données de *position* et de *force*:

Définition 6.4.1 *On appelle* état sensoriel *d'un robot, tout couple* (p, f) *de données perceptives tel que* p *est une configuration du robot, et* f *est un torseur de force. On appelle* interprétation *de* (p, f)*, l'ensemble des configurations réelles*

du robot qui peuvent donner lieu à la lecture des valeurs p et f sur les capteurs de position et de force.

Remarque: Différentes configurations du robot peuvent être représentées par un même couple (p, f) de données perceptives, du fait de la présence des erreurs de mesure. Si l'espace considéré est $\xi(\Re^3 \times O^3)$, l'interprétation de (p, f) est contenue dans la boule ouverte $B(p, \varepsilon_m)$, cf. paragraphe 6.4.3.

Définition 6.4.2 *Soit m un mouvement ayant pour objectif de placer le robot dans une région G de l'espace des configurations (G peut être dans l'espace libre ou sur la surface d'un objet). Une condition d'arrêt associée à m est définie comme l'ensemble des états sensoriels (p, f) dont les interprétations sont entièrement contenues dans G. On note $P_a \times F_a$ cet ensemble.*

Traitement des ambiguïtés:

Une difficulté importante liée à la détermination des conditions d'arrêt, réside dans le traitement des ambiguïtés produites par les erreurs de perception. Par exemple, un point p situé dans le voisinage de l'arête qui sépare les deux faces planes F_1 et F_2 de l'objet représenté dans la figure 6.9, est un objectif ambigü: la boule ouverte $B(p, \varepsilon_m)$ intersecte les deux faces, et les forces de réaction engendrées par F_1 et F_2 sont localisées dans le même cône d'incertitude.

L'approche que nous utilisons pour résoudre ce problème, consiste à éliminer à priori toutes les configurations qui peuvent conduire à une ambiguité d'interprétation [Buckley 87]:

Définition 6.4.3 *Deux points de contact x et y ne sont pas discernables en position et en force, ssi les deux conditions suivantes sont simultanément vérifiées:*

- *La distance séparant x et y est inférieure à $2\varepsilon_m$.*
- *Les cônes de frottement associés à x et à y forment entre-eux des angles inférieurs à $2\vartheta_f$.*

Cette définition est utilisée par le système pour découper l'espace en régions non ambigües du point de vue de la perception (cf. paragraphe suivant). Ces régions sont alors considérées comme des objectifs possibles pour les mouvements du robot, du fait qu'elles sont toutes identifiables par des couples (p, f) de données perceptives.

Caractérisation des conditions d'arrêt:

Soit m un mouvement ayant pour objectif de placer le robot dans une région G de l'espace des configurations. Les conditions d'arrêt $P_a \times F_a$ de m peuvent alors être caractérisées comme suit:

- Si G est réduit à une position p de $\Re^3 \times O^3$, P_a est la boule ouverte $B(p, 2\varepsilon_m)$ de centre p et de rayon $2\varepsilon_m$. Dans le cas où G est un ensemble de positions de l'espace libre ou de la surface d'un objet, P_a est défini par l'expression: $F_a = \cup_{p \in G} B(p, 2\varepsilon_m)$.
- Si P_a est inclus dans l'espace libre, F_a est l'ensemble vide. Dans le cas où P_a met en jeu un ensemble S de points de contacts, F_a est défini par l'expression: $P_a = \cup_{p \in S} Cone(n_p, \phi_p + 2\vartheta_f)$.
  n_p est la normale en p à la surface de contact, et ϕ_p est l'angle d'ouverture du cône de frottement associé.

Remarque: On prend toujours deux fois l'erreur maximum de perception ε_m, car le système peut commettre une erreur de ε_m lors de la mesure d'un point situé sur la frontière de la boule ouverte $B(p, \varepsilon_m)$. Il en est de même pour la valeur ϑ_m.

Si $P_a \times F_a$ représente les conditions d'arrêt associées à un mouvement m, l'arrêt du déplacement se produira dès que le couple (p, f) mesuré sur les capteurs du robot est inclus dans l'ensemble $P_a \times F_a$.

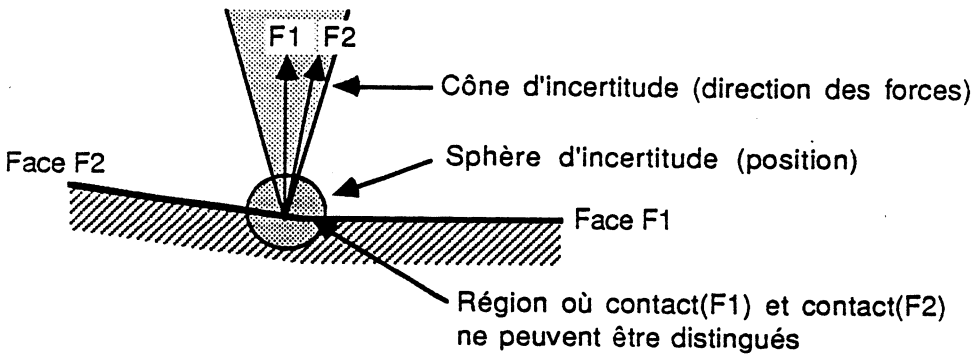


Figure 6.9: Exemple de situation ambigüe.

6.5 Espace de planification:

6.5.1 Notion d'espace de planification:

L'espace de planification est l'espace dans lequel le système raisonne en vue de planifier les mouvements du robot. 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. En pratique, les informations sensorielles utilisées permettent de mesurer des positions et des efforts. Un état du robot est alors défini comme suit:

Définition 6.5.1 *On appelle état E_p associé à une configuration p commandée, l'ensemble des états sensoriels (p^*, f^*) qui peuvent être lus sur les capteurs de position et de force après exécution de la commande. On note $E_p = P_p \times F_p$ cet ensemble.*

Si p est une position atteignable de $\xi(\Re^3 \times O^3)$, E_p est caractérisé par l'expression:

$$E_p = \{(p^*, f^*) : p^* \in B(p, \varepsilon_p + \varepsilon_m) \wedge f^* \in cone(n, \phi + \vartheta_f)\}$$

dans laquelle n est la normale en p à la surface de contact, et ϕ est l'angle d'ouverture du cône de frottement. ε_p, ε_m et ϑ_f sont les erreurs maximum de commande et de perception définies dans le paragraphe 6.4.3.

Soit C_{robot} l'espace des configurations du robot. L'espace de planification E peut alors être défini comme un ensemble de configurations p de C_{robot}, tel que:

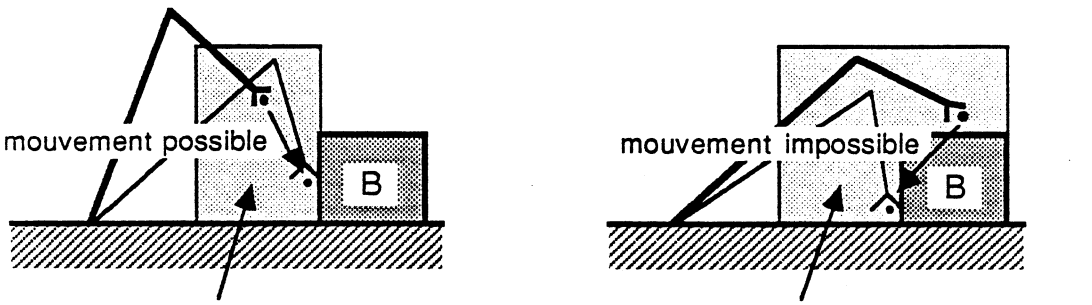
- $E_p \cap E_{p'} = \emptyset \quad \forall p, p' \in E.$
- $C_{robot} = Fermeture(\cup_{p \in E} P_p).$

6.5.2 Construction de l'espace de planification:

La construction de l'espace de planification se heurte à deux difficultés importantes: l'ensemble des états n'est pas énumérable, et la dimension de l'espace ne permet pas d'envisager une représentation exacte des contraintes imposées par les objets de l'environnement. C'est pourquoi la technique utilisée consiste à rechercher un modèle approché en regroupant les états jugés équivalents, puis en les structurant sous la forme d'un *graphe d'états* autorisant l'application d'algorithmes de complexité raisonnable. Dans cette représentation, une étape de planification se traduit au plus haut niveau par la recherche d'un chemin continu permettant de passer du nœud associé à l'état initial, au nœud définissant l'état final.

Différents critères peuvent être utilisés pour construire un graphe d'états adapté à la résolution d'un problème de planification donné. La méthode que nous avons développée, consiste à définir deux types de relations pour réaliser des regroupements d'états:

- Dans le cas où les mouvements planifiés sont exécutés dans "l'espace libre" (i.e à une distance des obstacles supérieure à $\varepsilon_p + \varepsilon_m$), une classe "d'états équivalents" est définie comme *"un ensemble connexe de configurations qui sont réalisables de manière continue, sans engendrer de collision avec les*



Ensemble de positions "équivalentes" par rapport à B

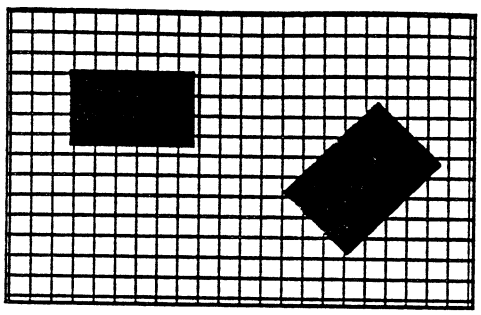
Ensemble de positions "non équivalentes" par rapport à B

Figure 6.10: Notion de classe "d'états équivalents" dans l'espace libre.

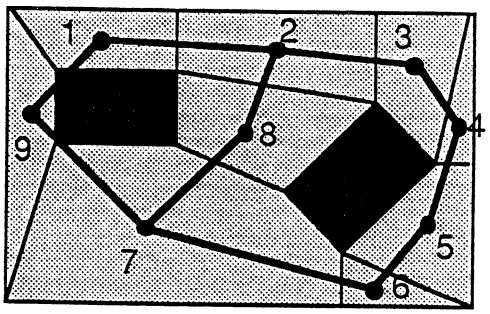
objets de l'environnement" (cf. figure 6.10). La taille du graphe d'états obtenu peut cependant beaucoup varier en fonction du procédé de découpage utilisé (cf. figure 6.11). Dans le cas d'un découpage régulier de chaque d.d.l en n intervalles, le nombre de sommets pour un robot six axes est n^6. Dans la pratique, on réduit ce nombre en augmentant la taille des intervalles chaque fois que cela est possible (cf. chapitre 7).

- Dans le cas où les mouvements planifiés mettent en jeu des relations de contact, nous avons défini une classe "d'états équivalents" comme *"un ensemble connexe de configurations qui sont susceptibles d'engendrer des forces de réaction de même nature"* [Bukley 87]. Avec cette définition, une face, une arête ou un sommet d'un objet peut servir de base à la construction d'une classe d'état. La nature des états représentés est alors définie par le type de contact réalisé (contact plan-plan, arête-plan ...). Le découpage réalisé doit cependant prendre en compte la contrainte d'unicité d'interprétation des couples (p, f) mesurés (cf. paragraphe précédent), ce qui conduit à rejeter les configurations qui ne sont pas discernables au sens de la définition 6.4.3. Le nombre de classes engendrées par un mobile et un objet fixe dépend donc directement du nombre de contacts qu'il est possible de réaliser (cf. figure 6.12). Si n est le nombre d'éléments du modèle de l'objet et m le nombre d'éléments du mobile, le produit $n.m$ représente le nombre minimum de sommets construits avec ce critère. Dans le cas où le problème traité nécessite de raisonner de manière plus fine sur les positions du robot, un redécoupage des classes précédentes doit être réalisé par le système (cf. [Buckley 87]). Le procédé que nous avons développé pour construire ce graphe d'états est

décrit dans le chapitre 8.



246 cellules libres
(grille régulière)



9 cellules libres
(polygones convexes)

Figure 6.11: Représentation de l'espace de planification associé à un mobile ponctuel se déplaçant dans un environnement $2D$.

6.5.3 Propriétés de la représentation:

Deux propriétés sont couramment utilisées pour caractériser un processus de planification: les propriétés de *"consistance" et de "complétude"*.

Un système de planification de mouvements est dit *consistant*, si tous les mouvements engendrés permettent d'atteindre les objectifs fixés, malgré les erreurs de commande et de perception. Les modules de planification que nous avons développés, vérifient cette propriété.

Un système de planification de mouvements est dit *complet*, s'il est capable de trouver une solution lorsqu'il en existe une. Cette propriété n'est pas vérifiée par les modules de planification que nous avons développés, pour deux raisons (cf. figure 6.13):

- Les approximations appliquées pour réduire la complexité algorithmique conduisent à ignorer volontairement des solutions potentielles.
- La propriété de consistance nécessite d'éliminer des solutions dont la validité n'est pas garantie par le modèle.

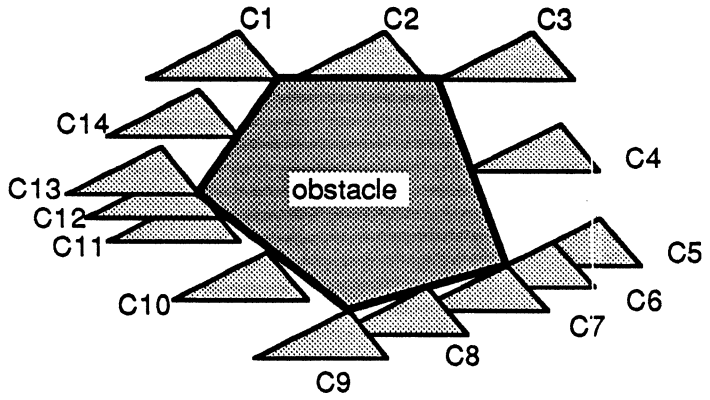


Figure 6.12: Différents types de contacts qu'il est possible de réaliser avec un mobile triangulaire se déplaçant en translation autour d'un obstacle fixe polygonal. Le nombre minimum d'ensembles "d'états équivalents" est dans ce cas 14.

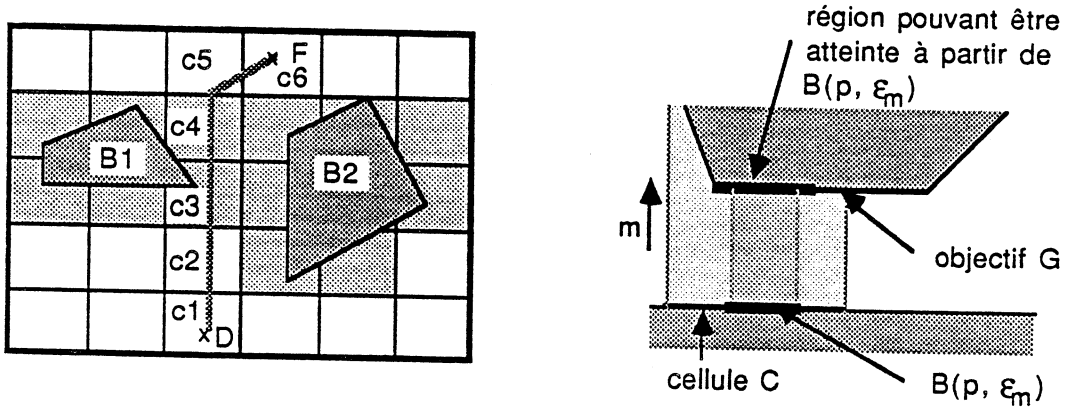


Figure 6.13: Illustration de la propriété d'incomplétude:
a- Le chemin $(C_1, C_2, C_3, C_4, C_5, C_6)$ ne peut être trouvé par le système, à cause du découpage utilisé pour représenter l'espace de planification (les cellules C_3 et C_4 sont considérées comme globalement interdites).
b- Le mouvement de direction m est rejeté par le système, car il ne garantit pas l'arrêt sur l'objectif G lorsque le robot est initialement dans C. La solution associée à la position particulière p est alors ignorée.

Chapitre 7

Le raisonnement spatial

7.1 Présentation et principes de base:

7.1.1 Problème abordé:

Le problème abordé par le raisonnement spatial est celui de la planification des mouvements du robot. Les informations prises en compte pour résoudre ce problème sont constituées de trois types de données:

- le modèle géométrique du robot et de son environnement,
- l'état initial dans lequel se trouve le robot avant exécution du mouvement,
- l'état que l'on cherche à atteindre.

La solution recherchée doit alors posséder les caractéristiques suivantes:

- la trajectoire suivie représente un *chemin continu* permettant de passer de la position initiale à la position objectif,
- le déplacement effectué n'engendre *aucune collision* entre le robot muni de sa charge et les objets de l'environnement,

- le mouvement exécuté satisfait des *contraintes diverses* liées au contexte de l'opération (contacts, précision, rapidité...).

La principale difficulté que l'on rencontre en abordant ce problème réside dans la maîtrise de la complexité algorithmique. Sur un plan théorique, il a été démontré [Schwartz, Sharir 82] que l'on pouvait exhiber des algorithmes de complexité polynomiale pour résoudre le problème de la planification de trajectoires sans collision pour n'importe quel manipulateur. Cependant, les temps d'exécution de ces algorithmes hypothétiques possèdent des termes qui les rendent inexploitables, même dans des cas relativement simples (voir chapitre 5). Heureusement, l'expérience a montré qu'il n'était pas nécessaire de résoudre ce problème dans sa forme la plus générale, et qu'il était possible d'en distinguer trois instances différentes, aptes à être traitées par des méthodes plus spécifiques:

- *Les mouvements de transfert* sont des mouvements de grandes et moyennes amplitudes, exécutés dans un espace modérément encombré. Ils mettent en jeu l'ensemble des degrés de liberté du manipulateur, ce qui contraint le système de planification à prendre en compte l'ensemble des déplacements réalisés par les articulations du bras. Une particularité essentielle de ces trajectoires est d'être "sûres", c'est à dire d'autoriser des déplacements à grandes vitesses sans risquer de provoquer de collision entre un composant du robot (bras ou charge) et un élément de l'environnement. De telles trajectoires conduisent à faire évoluer le manipulateur et sa charge à distance respective des obstacles, cette distance représentant la garde nécessaire à la prise en compte de l'imprécision du bras et des effets de la dynamique. Il est alors possible d'approximer les objets par des formes simples, afin de réduire les temps de calcul.

- *Les mouvements engendrés lors des opérations de saisie* sont des mouvements de faibles amplitudes, exécutés dans un espace très contraint mais bien localisé (seul l'environnement local de l'outil de préhension joue un rôle effectif à ce niveau de planification). Les trajectoires générées permettent d'une part d'amener l'outil de préhension en position de saisie de l'objet, et d'autre part de retirer cet outil une fois la manipulation exécutée. Ces trajectoires ont presque toujours une géométrie simple, mais elles dépendent de facteurs divers tels que la nature des contacts liant les mors de la pince à l'objet, la stabilité de cet objet une fois placé dans la pince, et l'accessibilité des éléments de prise.

- *Les mouvements fins de montage* sont des mouvements de faibles amplitudes, exécutés dans un espace très contraint. Ils possèdent la particularité d'être guidés par des informations sensorielles, afin de s'affranchir des problèmes dus aux incertitudes. La technique utilisée consiste à considérer l'environnement local du montage comme un *guide géométrique* sur lequel

s'appuie le robot pour exécuter ses mouvements. Les trajectoires engendrées sont alors conceptuellement simples; elles dépendent essentiellement de l'environnement local de l'objet manipulé, des contacts mis en œuvre, et des contraintes imposées par les incertitudes.

7.1.2 Raisonnements mis en jeu:

La résolution des trois sous-problèmes précédents met en jeu deux formes de raisonnement géométrique, conduisant respectivement à planifier des trajectoires et à analyser l'incidence des contacts sur ces trajectoires. Le premier type de raisonnement est appelé **"raisonnement spatial"**, car il conduit à raisonner sur l'organisation spatiale de l'environnement de travail du robot (positions des objets, volumes occupés et relations entre les objets). L'autre forme de raisonnement est appelée **"raisonnement morphologique"**. Ses caractéristiques, ainsi que le principe de la coopération entre les deux types de raisonnement, sont décrits dans le chapitre 8.

Dans ce chapitre, nous traiterons des modèles et des techniques algorithmiques mis en jeu par le raisonnement spatial. Certaines techniques développées sont essentiellement orientées vers la planification de trajectoires de transfert; ce sont les techniques liées aux notions de "C-obstacles" et "d'espace libre". D'autres visent plus particulièrement la planification des trajectoires qui mettent en jeu des contacts; ce sont les techniques liées au calcul des "débattements valides".

Le raisonnement appliqué dans le cas des mouvements de transfert, opère sur un modèle complet de l'espace de travail du robot. La complexité algorithmique est alors réduite en utilisant des approximations, dont la finesse varie en fonction des caractéristiques globales des solutions recherchées. Ces solutions représentent des chemins dans l'espace des configurations du robot.

Dans le cas des opérations de saisie et de montage, l'espace considéré est limité au volume occupé par la pince et par les objets concernés; mais la présence des contacts nécessite de travailler sur des modèles "exacts" (au sens de la représentation utilisée). Les solutions recherchées définissent alors des trajectoires géométriquement simples, qui se terminent généralement par l'apparition ou la disparition d'un contact.

7.1.3 Outils informatiques nécessaires:

Les outils informatiques qui sont à la base du raisonnement spatial sont les suivants:

- Des fonctions pour le calcul des *débattements valides* d'un mobile A évoluant parmi des obstacles Bi, lorsque A est animé d'un mouvement simple (translation ou rotation). Ces fonctions opèrent sur les entités géométriques qui

composent les objets (faces, arêtes, sommets). Elles réalisent essentiellement des calculs d'interférence et de collision.

- Des algorithmes pour calculer les *contraintes de position* imposées par les obstacles Bi sur le mobile A, lorsque A ne comporte que des d.d.l en translation. Ces algorithmes opèrent sur les modèles surfaciques des objets. Ils permettent de construire une représentation exacte des Bi dans l'espace des configurations de A. Sur un plan opératoire, le procédé appliqué peut être vu comme un grossissement des régions occupées par les Bi. Nous parlerons alors *d'opérateurs de grossissement.*

- Des fonctions pour le calcul approché des *contraintes de position et d'orientation* imposées par les obstacles Bi sur le mobile A, lorsque A comporte des d.d.l en rotation. Ces fonctions opèrent sur le modèle surfacique des objets. Elles conduisent à construire une représentation approchée des Bi dans l'espace des configurations de A. Le procédé employé consiste à discrétiser certains d.d.l de A. Nous parlerons alors *d'opérateurs de réduction* des d.d.l.

- Des algorithmes pour construire un graphe de *l'espace libre*, à partir d'une discrétisation de l'espace des configurations du mobile A. Ces algorithmes opèrent sur les modèles créés par les fonctions précédentes.

- Des algorithmes de type A^* pour rechercher des chemins dans le graphe qui représente l'espace libre.

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:

- Les fonctions de calcul de débattements s'appliquent sur des objets composés de polyèdres, de cylindres, de cônes et de sphères.

- Les opérateurs de grossissement et de réduction s'appliquent uniquement sur des objets polyédriques convexes. Une décomposition en composants convexes est alors nécessaire pour les objets concaves.

Les algorithmes décrits dans ce chapitre sont ceux mentionnés dans [Germain 84] et dans [Laugier, Germain 85]. Ils ont été pour la plupart développés en parallèle par Lozano-Perez [Lozano-Perez 85]. Ils reposent en grande partie sur les techniques de raisonnement proposées dans [Lozano-Perez 83].

7.1.4 Notations utilisées:

Les notations utilisées dans ce chapitre sont celles introduites dans le chapitre 6:

C_A = espace des configurations de A
$A(c)$ = posture dans $\Re^3$ de l'objet A en configuration c.
avec:
$c = (q_1\, q_2 \cdots q_n)$ si A est une structure articulée.
q_i = valeur associée au i-ème d.d.l.
I_i = domaine de variation de q_i.
$c = (x\, y\, z\, \alpha\, \beta\, \gamma)$ si A est un corps solide.
$(x\, y\, z)$ = position du point de référence de A.
$(\alpha\, \beta\, \gamma)$ = orientation du repère associé à A.
$CO_A(B) = \{c \in C_A : A(c) \cap B \neq \emptyset\}$: image de B dans C_A.
$EL_A = C_A - \cup_{i=1}^{m} CO_A(Bi)$: espace libre dans C_A.

Le principal objectif visé par les algorithmes présentés dans ce chapitre est de calculer les ensembles $CO_A(B)$, et de construire une représentation exploitable de l'espace libre EL_A. Dans le cas général, les ensembles $CO_A(B)$ représentent des hypervolumes de dimension n (les C-obstacles), bornés par des surfaces courbes de dimension $n-1$ (les C-surfaces). Une caractérisation exacte de ces ensembles fait appel à des outils mathématiques sophistiqués, dont l'exploitation informatique n'est pas réellement envisageable à cause de la complexité algorithmique induite.

C'est pourquoi nous travaillerons dans des sous-espaces de C_A, conduisant à considérer tour à tour des sous-ensembles des d.d.l de A. Par exemple, le sous-espace associé aux trois translations de A est noté C_A^{xyz}, et les obstacles correspondants sont notés $CO_A^{xyz}(B)$. De même nous noterons $C^q_{A_q, A_1(v_1)\cdots A_{q-1}(v_{q-1})}$ le sous-espace associé au q-ième d.d.l d'une structure articulée A, lorsque les d.d.l antérieurs sont fixés à des valeurs $v_1, v_2 \cdots v_{q-1}$, et les d.d.l postérieurs ne sont pas considérés (l'articulation q est alors considérée comme la dernière de la chaîne cinématique). Afin de ne pas obscurcir l'exposé par un excés de symboles, nous noterons C_A^q ce sous-espace; les obstacles correspondants sont alors notés $CO_A^q(B)$, et l'espace balayé par A_q lorsque q décrit un intervalle $[a\ b]$, est représenté par le symbole $A[a\ b]_q$.

7.2 Détermination des débattements valides:

7.2.1 Notion de débattement valide:

Le calcul des *débattements valides* est réalisé afin de déterminer tous les déplacements possibles d'un mobile animé d'un mouvement simple. Ce type de calcul est utilisé pour déterminer l'amplitude qu'il faut associer aux mouvements fins de montage. Il est également appliqué récursivement pour construire une représentation approchée de l'espace libre associé au robot.

Définition 7.2.1 *On appelle débattement valide $V_A(q)$ associé à un mobile A se déplaçant suivant une seule direction q, l'ensemble des valeurs de q telles que:* $A(q) \cap Bi = \emptyset$.

Définition 7.2.2 *Soit c_j un ensemble de valeurs articulaires* $(v_1, v_2 \cdots v_{j-1}, v_{j+1} \cdots v_n)$ *de* $I_1 \times I_2 \cdots I_{j-1} \times I_{j+1} \cdots I_n$. *On appelle débattement valide $V_{A[c_j]}(q_j)$ associé à un mobile A se déplaçant suivant une seule direction q_j de C_A, l'ensemble des valeurs q_j de I_j telles que:* $A(v_1, v_2 \cdots v_{j-1}, q_j, v_{j+1} \cdots v_n) \cap Bi = \emptyset$.

La détermination des ensembles $V_A(q)$ est basée sur des calculs d'interférences et de collisions, portant sur des modèles polyédriques. Des techniques telles que celles développées par Boyse [Boyse 79] permettent de réaliser ce type de calculs.

Afin de clarifier l'exposé, nous ne ferons référence dans ce paragraphe qu'à des mobiles dont l'espace des configurations est de dimension 1. Il est clair que les algorithmes développés s'appliquent de manière identique au cas des mobiles qui se déplacent suivant une seule direction de leur espace des configurations.

7.2.2 Calculs d'interférences:

Nous noterons $inter(A(c), B)$ la fonction qui s'évalue à vrai lorsque $A(c)$ intersecte B. Cette fonction est utilisée par le système pour tester si une configuration c de C_A appartient ou non à l'espace libre EL_A.

Définition 7.2.3 *Deux objets A et B ne sont pas en situation d'interférence ssi:*

$$(A \cap B = \emptyset) \wedge (A \not\subset B) \wedge (B \not\subset A)$$

L'opération de base pour réaliser ce calcul consiste donc à analyser l'intersection de tous les couples d'éléments des modèles de A et de B. Dans la pratique, ce test est réalisé de manière plus économique, en exploitant les propriétés topologiques des modèles traités:

- **Objets polygonaux:**
  L'algorithme que nous utilisons consiste à tester l'inclusion des sommets de A et de B (A et B étant des polygones convexes):

  $\forall s_A \in A : s_A \not\subset B$
  $\exists s_B \in B : s_B \not\subset A$

  Dans le cas général, la complexité maximum de l'algorithme est en $O(n \cdot m)$, si n est le nombre de sommets de A et m le nombre de sommets de B. Des critères simples basés sur des notions de volumes englobants et de classements d'éléments peuvent être appliqués pour réduire cette complexité [Maruyama 72]. Dans notre implantation, nous utilisons des parallélépipèdes englobants et un découpage triangulaire des faces.

- **Objets polyédriques:**
  L'algorithme que nous utilisons consiste à tester les intersections de tous les couples "face- arête" de A et de B:

  $$\forall f_A \in A, \forall a_B \in B : \; f_A \cap a_B = \emptyset$$
  $$\forall f_B \in B, \forall a_A \in A : \; f_B \cap a_A = \emptyset$$

  L'inclusion des objets est testée au moyen des volumes englobants. La complexité de l'algorithme est proportionnelle au terme: $nb(faces\ A) \cdot nb(aretes\ B) + nb(faces\ B) \cdot nb(aretes\ A)$. Elle est bornée par le terme $n \cdot m$, dans lequel n est le nombre d'arêtes de A et m le nombre d'arêtes de B (car le nombre de faces d'un polyèdre est toujours inférieur au nombre d'arêtes).

- **Composants non polyédriques:**
  L'algorithme que nous utilisons pour traiter les composants cylindriques, coniques et sphériques, consiste à tester les intersections de tous les couples "face-face" de A et de B:

  $$\forall f_A \in A, \forall f_B \in B : \; f_A \cap f_B = \emptyset$$

  L'inclusion des objets est testée comme précédemment. La complexité de l'algorithme est en $O(n.m)$, si n et m sont respectivement le nombre de faces de A et de B.

7.2.3 Calculs de collisions:

Les calculs de collisions sont utilisés pour construire les ensembles de type $V_A(q)$. Ils conduisent à déterminer les contacts qui sont susceptibles de se produire entre l'objet A et les obstacles B. Le principe consiste à analyser toutes les intersections qui existent entre les éléments de B, et les lieux géométriques décrits par les sommets et les arêtes de A [Boyse 79]. Lorsque le mouvement est une translation, le lieu d'un sommet est un segment de droite, et celui d'une arête est une surface trapézoïdale. Dans le cas d'une rotation, le lieu d'un sommet est un arc de cercle; celui d'une arête est un disque, un cylindre, un tronc de cône ou une hyperboloïde de révolution.

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, c'est à dire des configurations qui appartiennent à l'ensemble $CO^q_A(B)$. L'algorithme de construction de cet ensemble est décrit dans le paragraphe suivant.

Objets polygonaux:

Propriété 7.2.1 *Un polygone A en mouvement peut heurter un obstacle polygonal B de deux manières différentes:*

- *un sommet de A heurte une arête de B (extrémités incluses).*
- *une arête de A heurte un sommet de B.*

L'opération de base pour rechercher les contacts potentiels consiste alors à analyser les intersections suivantes:

$$\forall s_A \in A, \forall a_B \in B : \mathit{lieu}(s_A) \cap a_B \qquad \text{(type } A\text{)}$$
$$\forall s_B \in B, \forall a_A \in A : \mathit{lieu}(s_B) \cap a_A \qquad \text{(type } B\text{)}$$

dans lesquelles $\mathit{lieu}(s_A)$ est le lieu géométrique décrit par S_A, et $\mathit{lieu}(s_B)$ représente le lieu géométrique fictif que s_B décrit relativement à A (cf. figure 7.1). On appelle *sommets fictifs* de A les points engendrés par les intersections de type B. Tous les contacts susceptibles de se produire entre A et B mettent alors en jeu un sommet réel ou un sommet fictif de A.

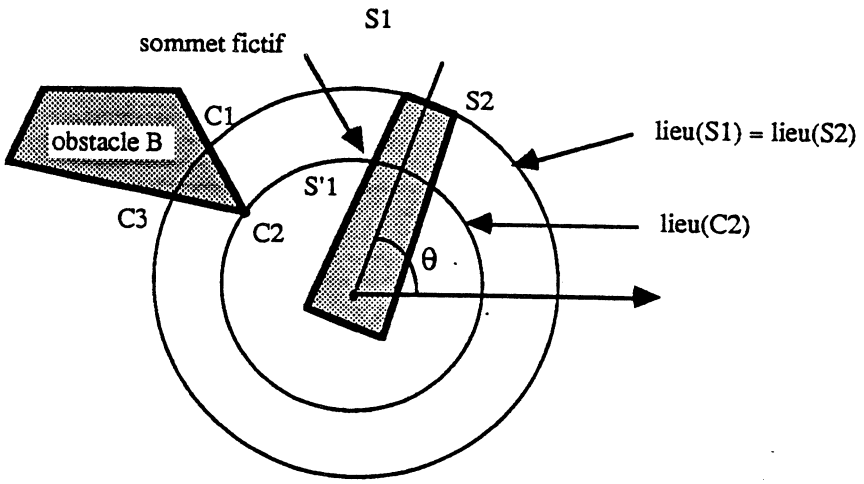


Figure 7.1: Exemple de contacts potentiels pour un mobile en rotation: (s_1, c_1), (s_2, c_3), (s'_1, c_2).

L'opération suivante consiste à analyser les solutions exhibées par l'algorithme ci-dessus, afin de déterminer celles qui sont localement réalisables eu égard à la

matière. La technique utilisée consiste à évaluer la propriété suivante pour chaque contact potentiel (s, b):

$$(-a_1) \cdot n \geq 0 \qquad et \qquad a_2 \cdot n \geq 0$$

avec:
b = arête de contact orientée dans le sens rétrograde sur $P1$.
$n = b \star rotation(\pi/2)$.
a_1, a_2 = arêtes issues de s et orientées dans le sens rétrograde sur $P2$.
$P1 = A\ ou\ B \qquad P2 = B\ ou\ A.$

Cette propriété est illustrée par la figure 7.2. elle ne garantit cependant pas la faisabilité globale des contacts pour des objets concaves ou pour des obstacles proches. Ces aspects sont alors pris en compte par l'algorithme de construction des $CO^q_A(B)$ décrit plus loin.

La complexité de l'algorithme de recherche des contacts potentiels est en $O(n \cdot m)$, si n est le nombre de sommets de A et m est le nombre de sommets de B. Cette complexité est du même ordre que celle exhibée par les algorithmes de calcul des interférences. Elle peut être améliorée de la même manière en exploitant des critères analogues (volumes englobants, classement des arêtes par orientation sur un polygone convexe ...).

Objets polyédriques:

Propriété 7.2.2 *Un polyèdre A en mouvement peut heurter un obstacle polyédrique B de trois manières différentes:*

- *un sommet de A heurte une face de B.*
- *une face de A heurte un sommet de B.*
- *une arête de A heurte une arête de B.*

L'opération de base pour rechercher les contacts potentiels consiste alors à analyser les intersections suivantes:

$$\forall s_A \in A, \forall f_B \in B : lieu(s_A) \cap f_B \qquad \text{(type } A\text{)}$$
$$\forall s_B \in B, \forall f_A \in A : lieu(s_B) \cap f_A \qquad \text{(type } B\text{)}$$
$$\forall a_A \in A, \forall a_B \in B : lieu(a_A) \cap a_B \qquad \text{(type } C\text{)}$$

Comme précédemment, les intersections de type B et C 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'analyse locale de validité des solutions exhibées par le calcul précédent est réalisée au moyen d'expressions analytiques simples, dont le domaine d'application est limité aux objets convexes:

- contact de type A ou B entre $P1$ et $P2$: (s, f)

$$a_i \cdot n \geq 0 \qquad \forall a_i \in Voisinage(s)$$

avec:
a_i = arête de $P1$ issue du sommet s et orientée à partir de s.
n = normale extérieure à la face f de $P2$.
$P1 = A\ ou\ B \qquad P2 = B\ ou\ A$.

- contact de type C entre $P1$ et $P2$: $(a1, a2)$

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

avec:
$a1$ = arête $a1$ de $P1$ orientée de manière arbitraire.
$a2$ = arête $a2$ de $P2$ orientée de manière arbitraire.
$n1, n2$ = normales extérieures aux faces $F1$ et $F2$ issues de $a1$ dans $P1$.
$n3, n4$ = normales extérieures aux faces $F3$ et $F4$ issues de $a2$ dans $P2$.
$P1 = A \qquad P2 = B$.

Ce type de calcul est illustré par les figures 7.2 et 7.3. La faisabilité réelle des solutions retenues est alors analysée implicitement par la méthode de construction des $CO_A(B)$.

La complexité de l'algorithme de recherche des contacts est proportionnelle au terme: $nb(aretes\ A) \cdot nb(faces\ B) + nb(aretes\ B) \cdot nb(faces\ A) + nb(aretes\ A) \cdot nb(aretes\ B)$. Comme dans le cas du calcul des interférences, elle est bornée par le terme $n \cdot m$ si n est le nombre d'arêtes de A et m le nombre d'arêtes de B. Elle peut être aussi améliorée par des techniques similaires.

Composants non polyédriques:

Les propriétés précédentes ne sont pas directement exploitables dans le cas des composants cylindriques, coniques et sphériques. Une technique possible consiste à étendre ces propriétés en considérant tous les nouveaux couples induits. Compte tenu de l'augmentation importante de complexité algorithmique introduite par le calcul des lieux géométriques et des intersections associées, nous avons développé une méthode qui opère uniquement sur des objets en translation. Dans les autres

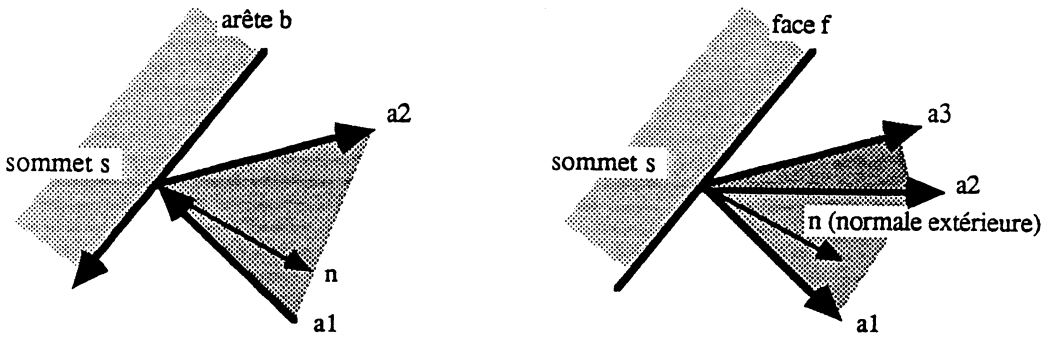


Figure 7.2: Propriétés topologiques des contacts engendrés par un sommet:
a- Contact de type (s, b): $(-a_1) \cdot n \geq 0$ *et* $a_2 \cdot n \geq 0$.
b- Contact de type (s, f): $a_1 \cdot n \geq 0$ *et* $a_2 \cdot n \geq 0$ *et* $a_3 \cdot n \geq 0$.

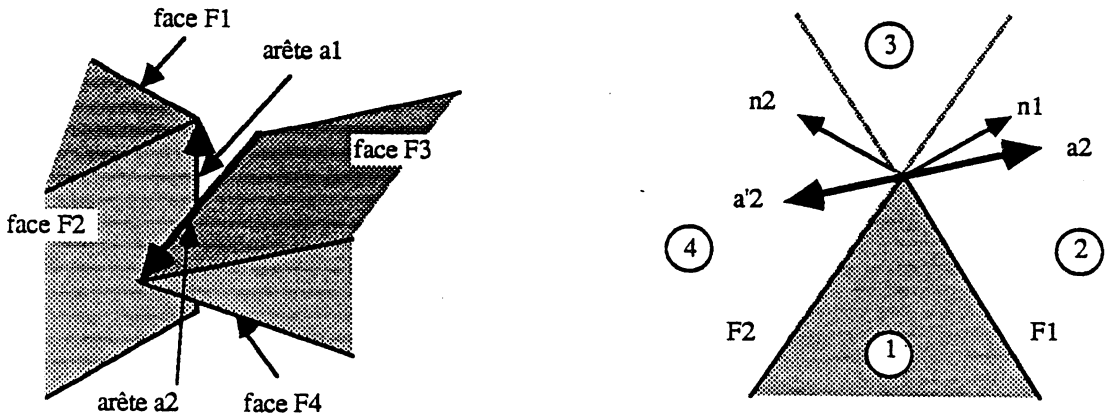


Figure 7.3: Propriétés topologiques des contacts engendrés par deux arêtes:
a- $a_2 \in$ Région (2): $a_2 \cdot n_1 \geq 0$ *et* $a_2 \cdot n_2 \leq 0$.
b- $a'_2 \in$ Région (1): $a'_2 \cdot n_1 \leq 0$ *et* $a'_2 \cdot n_2 \geq 0$.

cas, nous utilisons une approximation polyédrique des surfaces courbes. Ce point mériterait d'être approfondi dans le futur.

7.2.4 Calcul des débattements valides:

Nous noterons $VALIDE(A[D_q], B)$ la fonction qui calcule l'ensemble des débattements valides $V_A(q)$, pour $q \in D_q$.

Le calcul exécuté par cette fonction peut être vu comme l'évaluation de l'expression suivante:

$$V_A(q) = D_q - \cup_{i=1}^{n} CO_A^q(Bi)$$

dans laquelle chaque $CO_A^q(Bi)$ est un *intervalle fermé de* $\Re$ (l'existence d'un seul intervalle par obstacle provient du fait que ceux-ci sont convexes). Les bornes de cet intervalle représentent alors les configurations de A qui correspondent aux deux contacts "extrèmes" entre A et Bi. $V_A(q)$ est alors constitué d'un ou de plusieurs intervalles ouverts, 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 $CO_A^q(Bi)$ à partir d'une simple classification de ces contacts. La méthode employée consiste alors à calculer la contrainte imposée par chaque Bi sur chaque sommet s de A retenu à l'issue du premier traitement (ces sommets réels ou fictifs sont ceux impliqués dans les contacts potentiels retenus). $CO_A^q(Bi)$ représente alors la *fermeture connexe* de l'ensemble obtenu par union de ces contraintes (cf. figure 7.4):

$CO_s^q(Bi) = \{q : s(q) \cap Bi \neq \emptyset\}$
$CO_s^q(Bi) = Fermeture\{\cup_{s \in A} CO_s^q(Bi)\}$
avec:
s = sommet réel ou fictif de A.
q = configuration de A.
$s(q)$ = droite (translation) ou cercle (rotation).

La complexité de l'algorithme dans l'espace bidimensionnel est en $O((n+m) \cdot m)$, si n est le nombre de sommets de A et m le nombre de sommets de Bi. Dans l'espace tridimensionnel elle est en $O((n+m+k) \cdot l)$, si n est le nombre de sommets de A, m le nombre de sommets de B, k le nombre d'arêtes de A et l le nombre de faces de B.

Si n 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 $O(n^2)$.

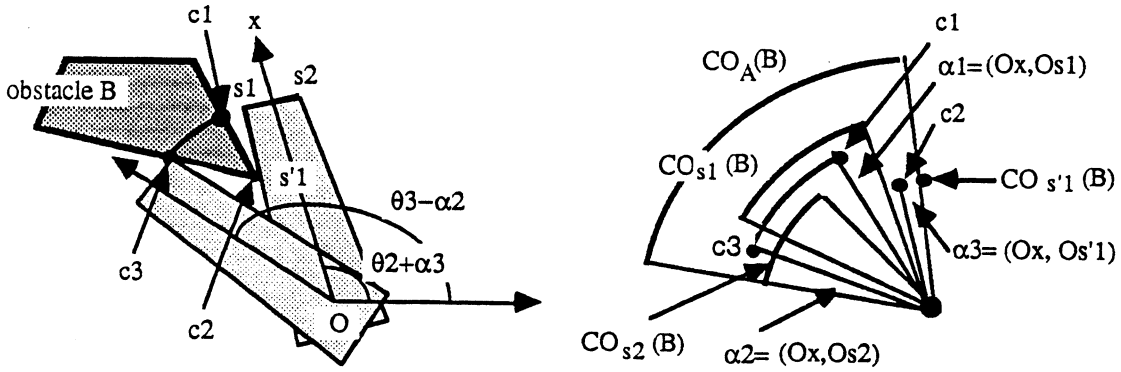


Figure 7.4: Construction des configurations de A qui appartiennent à $CO_A(B)$, lorsque A est un objet en rotation.

7.3 Opérateurs de grossissement:

7.3.1 Calcul des ensembles de type $CO_A^{xyz}(B)$:

Les opérateurs de grossissement sont utilisés pour construire une *représentation exacte* des obstacles Bi dans l'espace des configurations C_A, lorsque le mobile A conserve une orientation fixe au cours de ses déplacements. Les calculs exécutés s'appuient sur le fait que les C-surfaces associées aux obstacles ont des structures topologiques analogues à celles des surfaces réelles, lorsque le mobile se déplace en translation dans $\Re^3$ (cette propriété n'est pas vraie dans le cas des rotations). L'espace C_A^{xyz} est alors isomorphe à $\Re^3$, et les ensembles $CO_A^{xyz}(B)$ sont des polyèdres convexes, lorsque A et B sont des polyèdres convexes. Intuitivement, le procédé de construction de ces ensembles consiste à "faire glisser" le mobile A sur la surface de B: le lieu du point de référence R de A représente alors l'enveloppe de $CO_A^{xyz}(B)$.

Soit $\{s_i, i = 1, n\}$ l'ensemble des sommets de B, et $\ominus A$ l'image miroir de $A(0)$ obtenue en symétrisant A par rapport à R (R étant placé à l'origine). Un algorithme général pour calculer $CO_A^{xyz}(B)$ consiste à construire l'enveloppe convexe de l'ensemble des sommets des polyèdres obtenus en plaçant $\ominus A$ sur les sommets de B [Lozano-Perez 83]:

$$CO_A^{xyz}(B) = conv(\{s \in \ominus A(s_i),\ \forall s_i \in B\})$$

Le principe de cet algorithme est illustré par la figure 7.5. Sa complexité est en $O(n^2 \log n)$, si n représente le nombre total de sommets de A et de B.

Dans le cas où A et B sont des polygones convexes, le calcul de $CO_A^{xy}(B)$ peut être réalisé par un algorithme de complexité $O(n)$ qui exploite un classement par orientation des arêtes de $\ominus A$ et de B (cf. figure 7.5):

Enveloppe $CO_A^{xy}(B)$= Ensemble ordonné des arêtes de $\ominus A$ et de B
tel que: $angle(x, a_i) \leq angle(x, a_{i+1}),\ \forall i \in \{1, 2, \cdots n-1\}$.

dans lequel a_i est une arête de $\ominus A$ ou de B, orientée dans le sens rétrograde sur A ou sur B.

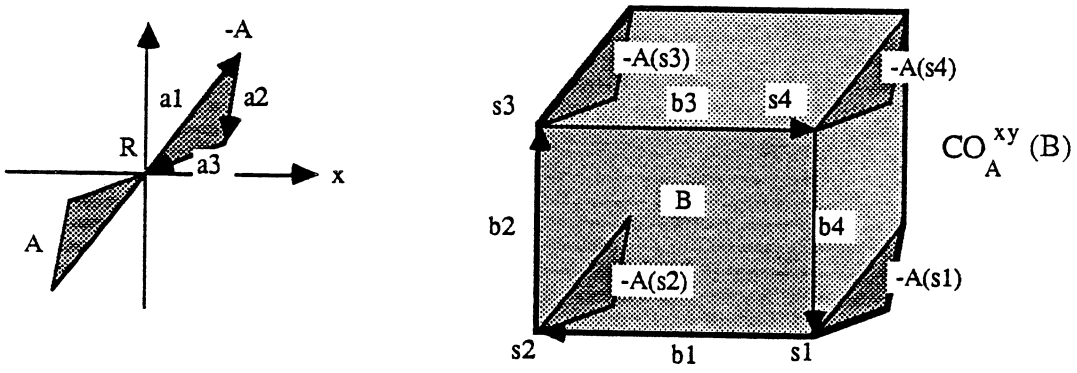


Figure 7.5: Construction d'un grossissement de type $CO_A^{xy}(B)$ dans le cas d'un objet convexe:
Cas 1: Calcul de l'enveloppe convexe de l'ensemble des sommets des polygones
$\ominus A(s_1)$, $\ominus A(s_2)$, $\ominus A(s_3)$ et $\ominus A(s_4)$.
Cas 2: Calcul de l'ensemble ordonné des arêtes de $\ominus A$ et de B:
$\{b_1, b_2, a_1, b_3, b_4, a_2, a_3\}$.

7.3.2 Traitement des symétries de révolution:

Un prétraitement des obstacles Bi peut être exécuté, lorsque le mobile A possède une symétrie de révolution qui lui donne un comportement homogène dans différen-

tes orientations. Cette situation se présente lorsque A peut être approximé par un disque, une sphère ou un cylindre se terminant par deux calottes sphériques.

Dans le cas d'un disque ou d'une sphère de rayon r, le prétraitement appliqué conduit à grossir les obstacles Bi d'une épaisseur r. Le mobile A est ainsi représenté par son point de référence situé au centre du disque ou de la sphère. Les obstacles grossis définissent alors directement les ensembles $CO_A^{xyz}(B)$.

Dans le cas du cylindre, le traitement des obstacles est identique, mais le mobile A est alors réduit à un segment de droite. Un calcul supplémentaire est dans ce cas nécessaire pour déterminer les ensembles de type $CO_A^{xyz}(B)$. Ce calcul est exécuté à partir des obstacles grossis par R et du mobile représenté par le segment de droite.

Dans la suite, nous noterons $Gros_r(Bi)$ l'obstacle Bi grossi par le rayon r, et $Squel(A)$ le mobile A squelettisé. Afin de conserver des représentations polyédriques, les nouveaux obstacles sont construits à l'aide des plans obtenus par translation "vers l'extérieur" des faces d'origine (cf. figure 7.6).

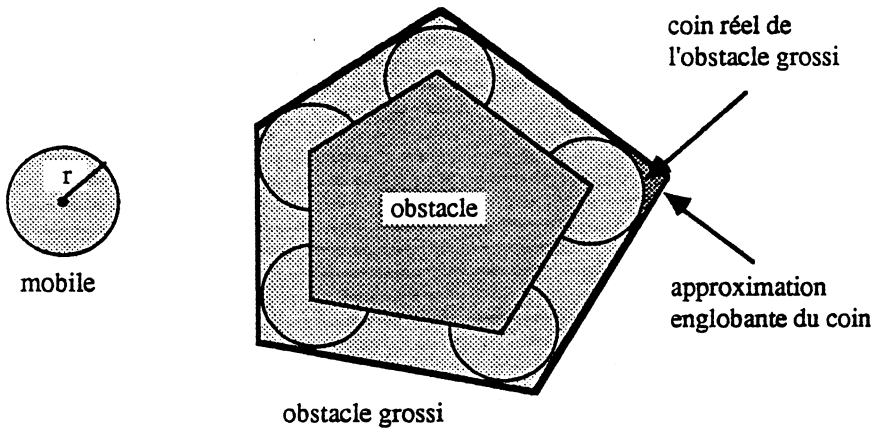


Figure 7.6: Grossissement d'un obstacle polygonal par le rayon d'un mobile circulaire.

7.4 Opérateurs de réduction des d.d.l:

7.4.1 principe du calcul:

Les opérateurs de réduction sont utilisés pour construire une *représentation approchée* des obstacles Bi dans l'espace des configurations C_A, lorsque le mobile A comporte des d.d.l qui combinent des translations et des rotations. La principale difficulté vient de ce que les C-surfaces produites par les rotations, ont des structures topologiques différentes de celles des surfaces réelles qui les ont engendrées. Les techniques de grossissement décrites dans le paragraphe précédent ne sont donc pas directement applicables.

Une méthode possible pour traiter ce problème, consiste à opérer sur des sous-espaces de C_A présentant uniquement des translations ou des rotations. Dans le premier cas, une approximation des obstacles $CO_A(B)$ est calculée pour différentes tranches d'orientation de A [Lozano-Perez 83]; dans l'autre cas, on représente explicitement les orientations valides de A pour des domaines de déplacements donnés [Germain 84]. Intuitivement, ce procédé conduit à fixer certains d.d.l de A, après leur avoir attribué des petits domaines de variation. Les C-obstacles associés à ce sous-espace correspondent alors aux "projections" des portions de C-obstacles contenues dans les tranches de mouvements considérées.

L'approche que nous avons développée, est basée sur le calcul des orientations valides. Elle conduit à construire récursivement des sous-espaces de C_A réduits à un seul d.d.l en rotation:

$$C_A^q = Reduction(C_A^{D_q})$$

avec:

D = Domaine de translation inclus dans $\Re^2$ ou $\Re^3$.

q = Secteur angulaire $[a\ b]$.

Cette opération conduit à calculer l'ensemble des orientations de A qui sont valides sur D, c'est à dire l'ensemble des orientations de A qui n'engendrent aucune collision entre A et les Bi, lorsque le point de référence R de A se déplace dans D. Ce procédé est également applicable lorsque le d.d.l représenté par q est de type prismatique.

7.4.2 Méthode de calcul:

Soit $c = (p, \alpha)$ une configuration de A dans $C_A^{D_q}$. Pour tout $p \in D$ il est possible de définir la fonction suivante:

$$F_{A,B}(p) = \{\alpha \in [a\ b] :\ A(p,\alpha) \cap B = \emptyset\}$$

avec: $B = \{B1, B2, \cdots Bn\}$.

La valeur de cette fonction pour une position p donnée est un ensemble de secteurs angulaires connexes, calculés à l'aide de la fonction $VALIDE$ décrite précédemment.

Si p décrit le domaine D, l'ensemble des orientations valides de A pour tout p peut être défini comme suit:

$$G_{A,B}(D) = \bigcap_{p \in D} F_{A,B}(p)$$

Compte tenu de sa nature (intersection sur un ensemble continu de valeurs), la fonction G ne peut être calculée directement. Une technique pour réaliser ce calcul consiste à se ramener au cas précédent (calcul de $F_{a,b}$), en réduisant le domaine D en un point, et en grossissant les obstacles en conséquence [Germain 84]:

$$G_{A,B}(D) = F_{A,B'}(r)$$

avec:
$B' = CO_D^{xyz}(B)$.
r = point de référence de D (choisi arbitrairement).

Ce procédé de calcul est illustré par la figure 7.7. Sa complexité algorithmique est en $O(p+q)$, si p et q représentent respectivement les termes provenant du calcul de grossissement et du calcul des débattements valides.

Justification:

Propriété (1): $B' = CO_D^{xyz}(B)$

Dans une représentation ensembliste, le procédé de grossissement des obstacles se traduit par l'expression d'une "différence" sur des ensembles continus de points [Lozano-Perez 83]:

$$CO_A^{xyz}(B) = B \ominus A(0) = \{b - a : b \in B \text{ et } a \in A(0)\}$$

La même formulation peut être appliquée dans le cas présent, si l'on assimile le domaine D à un mobile évoluant en translation. La position de D est alors représentée par le vecteur O_r, dans lequel O est l'origine des axes et r le point référence de D. On a alors la propriété:

$$\forall d \in D(c), \exists d \in D(0) : d = d' + c$$

Si $b \in D(c) \cap B$ alors $b = d' + c$, avec $d' \in D(0)$
d'où $c = b - d'$
soit $c \in B \ominus D(0)$.

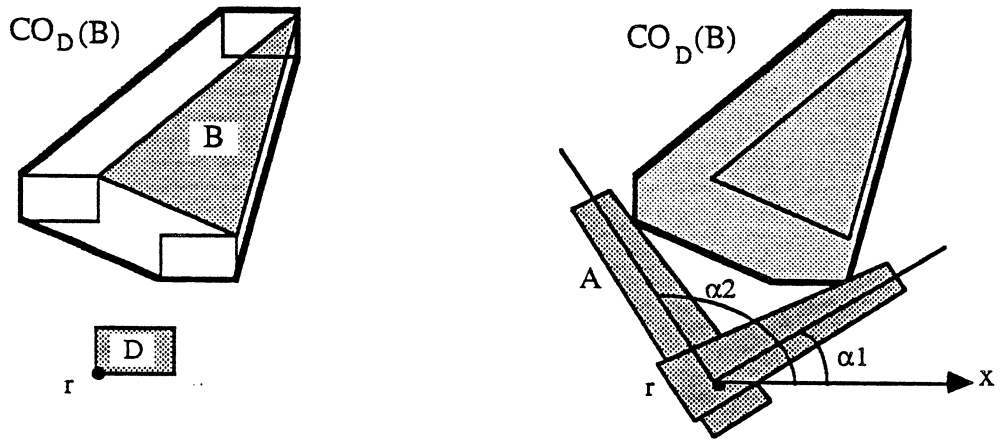


Figure 7.7: Calcul des orientations valides d'un mobile A lorsque le point de référence r de A décrit un domaine D:
a- Calcul de l'obstacle grossi $CO_D(B)$.
b- Calcul des orientations valides de A par rapport à $CO_D(B)$: $[0\ \ \alpha_1] \cup [\alpha_2\ \ 2\pi]$.

La démonstration de la propriété inverse est similaire.

Propriété (2): $G_{A,B}(D) = F_{A,B'}(r)$

(i)- $G_{A,B}(D) \subset F_{A,B'}(r)$
soit $\alpha \in G_{A,B}(D)$
par hypothèse on a: $A(p,\alpha) \cap B = \emptyset,\ \forall p \in D$
supposons qu'il existe α tel que: $A(r,\alpha) \cap B' \neq \emptyset$
alors $\exists\, a \in A(r,\alpha) :\ a \in B'$
soit $\exists b \in B$ et $d \in D :\ a = b - d$ (car $B' = B \ominus D$)
d'où $a + d = d$
mais $(a+d) \in A(p,\alpha)$ car $a \in A(r,\alpha)$ et $d \in D$
donc $A(p,\alpha) \cap B \neq \emptyset$
ce qui est en contradiction avec les hypothèses.

(ii)- $F_{A,B'}(r) \subset G_{A,B}(D)$
soit $\alpha \in F_{A,B'}(r)$

par hypothèse on a: $A(r,\alpha) \cap B' = \emptyset$
supposons qu'il existe α et p tel que: $A(p,\alpha) \cap B \neq \emptyset$
alors $\exists\, a \in A(p,\alpha) : \; a \in B$
mais $\exists\, a' \in A(r,\alpha)$ et $d \in D : \; a = a' + d$
d'où $a' = a - d$
soit $a' \in B \ominus D$ (car $a \in B$)
donc $A(r,\alpha) \cap B' \neq \emptyset$
ce qui est en contradiction avec les hypothèses.

7.5 Raisonnement sur une structure articulée:

7.5.1 Principe de construction des C-obstacles:

Le mobile est maintenant une chaîne cinématique ouverte composée de n corps articulés $A1, A2 \cdots An$. Les ensembles $CO_A(Bi)$ produits par les obstacles Bi sont alors des hypervolumes de dimension n qu'il est très difficile de calculer de manière exacte. C'est pourquoi nous ne représenterons que des approximations de ces ensembles, que nous construirons récursivement par un procédé de découpage en "tranches" de C_A. L'idée de base consiste à considérer successivement les contraintes imposées par les Bi sur les différents d.d.l de A. L'algorithme appliqué conduit à parcourir le bras de sa base vers son extrémité libre. A chaque étape, le comportement du d.d.l traité est analysé sur une *partition de son domaine de variation.* Seuls les éléments postérieurs du bras interviennent alors dans ce calcul.

Si q_1 est fixé à une valeur v_0, la contrainte appliquée par B sur le sous-système $(A2\, A3 \cdots An)$ peut être représentée comme suit:

$$CO_A^{v_0}(B) = \{(q_2 \cdots q_n), q_2 \in I_2 \cdots q_n \in I_n : \; (v_0\, q_2 \cdots q_n) \in CO_A(B)\}$$

Si q_1 décrit un petit intervalle D_{q_1} de D_1, la conjonction des contraintes appliquées par B sur le sous-système $(A2\, A3 \cdots An)$ peut être définie comme suit:

$$CO_A^{D_{q1}}(B) =$$
$$\{(q_2 \cdots q_n), q_2 \in I_2 \cdots q_n \in I_n : \; \exists q_1 \in D_{q_1}\; tq\; (q_1\, q_2 \cdots q_n) \in CO_A(B)\}$$

Une approximation de $CO_A(B)$ est alors obtenue en faisant l'union des ensembles de type $CO_A^{D_{q1}}$:

$$CO_A(B) \approx \bigcup_{i=1}^{p1} CO_A^{D_i}(B)$$

avec: $I_1 = \cup_{i=1}^{p1} D_i$ et $D_i \cap D_j = \emptyset \qquad \forall i,j.$

Ce type de calcul conduit à réduire d'une unité la dimension des hypervolumes manipulés (chaque $CO_A^{D_{q1}}$ est de dimension $n-1$). Il n'est cependant pas réalisable directement, ce qui conduit à appliquer récursivement le procédé jusqu'à obtention d'ensembles de dimension 1:

$$\forall j \in \{1, 2 \cdots n-1\}$$
$$CO_A^{Di_1 \times Di_2 \cdots Di_j}(B) =$$
$$\{(q_{j+1} \cdots q_n), q_{j+1} \in I_{j+1} \cdots q_n \in I_n : \exists q_j \in D_{i_j} \; tq \; (q_j \cdots q_n) \in$$
$$CO_A^{Di_1 \times Di_2 \cdots Di_{j-1}}(B)\}$$

avec:

$$i_1 \in \{1 \cdots p_1\}$$
$$i_2 \in \{1 \cdots p_2\}$$
$$\cdots$$
$$i_j \in \{1 \cdots p_j\}$$
$$CO_A^{Di_0}(B) = CO_A(B)$$

Les ensembles de type $CO_A^{Di_1 \times Di_2 \cdots Di_{n-1}}$ ainsi obtenus sont de dimension 1. Nous verrons plus loin qu'ils peuvent être calculés à l'aide de la fonction $VALIDE$.

Le principe de construction récursif des $CO_A(B)$ est illustré par la figure 7.8. $CO_A^{D_2}(B)$ représente la conjonction des contraintes imposées par l'obstacle B sur le manipulateur A, lorsque q_1 décrit D_2. Cet ensemble correspond à la projection de la portion de $CO_A(B)$ comprise entre les plans $P1$ et $P2$ de C_A. De même, $CO_A^{D_2 \times D'_4}(B)$ représente la conjonction des contraintes imposées par B sur A, lorsque q_1 décrit D_2 et q_2 décrit D'_4. Cet ensemble définit un intervalle Q de valeurs interdites pour q_3, lorsque $q_1 \in D_2$ et $q_2 \in D'_4$. Q est par construction plus grand que nécessaire, mais il permet d'assurer que la portion de $CO_A(B)$ représentée est entièrement incluse dans le produit cartésien $D_2 \times D'_4 \times Q$. $CO_A(B)$ est alors approximé par l'union des ensembles de type $D_i \times D'_j \times Q_k$. On peut vérifier sur l'exemple que, pour tout point a de Q, il existe un point b de D'_4 et un point c de D_2 tel que: $(c\ b\ a) \in CO_A(B)$.

Remarque: Dans le cas général, les ensembles de type $CO_A^{Di_1 \times Di_2 \cdots Di_{n-1}}$ ne sont pas réduits à un seul intervalle de I_n (cf. figure 7.9).

Le raisonnement récursif décrit ci-dessus, définit un support algorithmique pour la construction des C-obstacles. Afin d'être exploitable, ce support doit être complété par des techniques permettant:

- De *calculer localement* les contraintes appliquées par les Bi sur un élément Aj de la structure articulée.

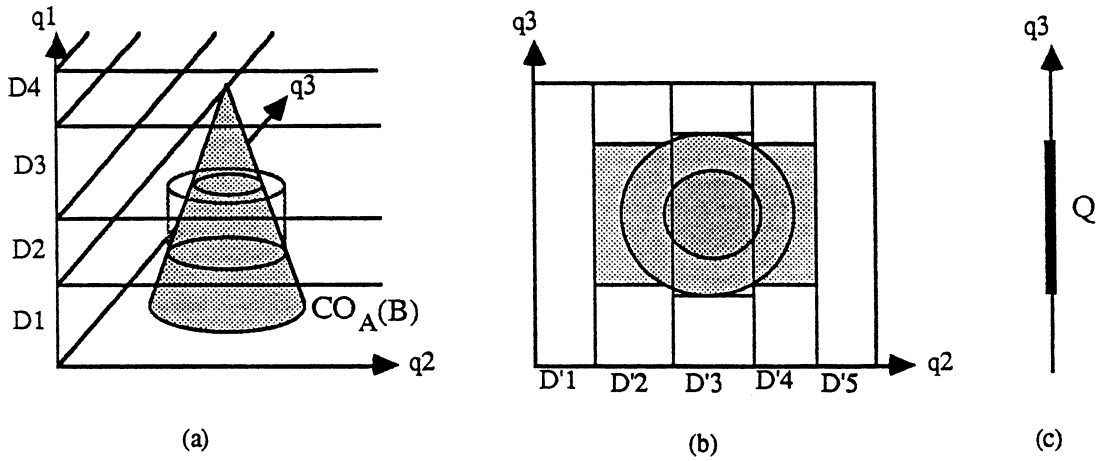


Figure 7.8: Exemple de construction d'une approximation de $CO_A(B)$:
a- Découpage de l'espace suivant q_1.
b- Approximation de l'ensemble $CO_A^{D_2}(B)$ associé à la tranche D_2.
c- Approximation de l'ensemble $CO_A^{D_2 \times D'_4}(B)$ associé aux tranches D_2 et D'_4.

- De *propager ces contraintes* vers les éléments postérieurs du bras.

7.5.2 Calcul des contraintes:

Le calcul des contraintes imposées par les obstacles est réalisé au moyen de la fonction $VALIDE$, et des opérateurs de *grossissement* et de *réduction*. Chaque élément A_j $(j = 2 \cdots n)$ du bras est alors considéré isolément, après avoir fixé la position des éléments antérieurs $(A_1 \cdots A_{j-1})$ ainsi que le domaine de variation attribué à A_{j-1}. Les éléments postérieurs $(A_{j+1} \cdots A_n)$ sont ignorés à ce niveau de raisonnement.

Soit R_j le point de référence de A_j. Afin de simplifier les calculs, nous choisirons ce point sur l'axe de l'articulation de A_j. Les hypothèses ci-dessus conduisent à associer "localement" deux d.d.l à A_j (cf. figure 7.10):

- Un déplacement en translation de R_j sur un domaine D_j créé par le débattement dq_{j-1} associé à A_{j-1}. Lorsque l'articulation de A_{j-1} est de type rotoïde, D_j est un arc de cercle; dans l'autre cas, D_j est un segment de droite. La

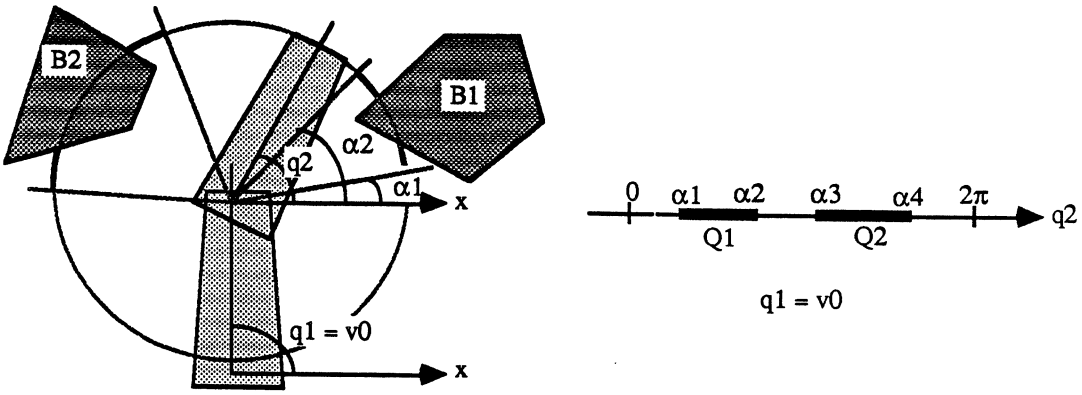


Figure 7.9: Exemple de tranche d'orientation constituée de plusieurs domaines disjoints pour le dernier d.d.l de la structure.

position de D_j dans l'espace cartésien est déterminée par la configuration $(q_1 \cdots q_{j-2})$ du sous-système $(A_1 \cdots A_{j-2})$.

- Un mouvement suivant l'articulation de A_j (rotation ou translation), réalisé dans le domaine I_j de variation de la variable q_j.

L'espace des configurations "local" de A_j est alors de type:

$C_{A_j}^{D_j \cdot q}$
avec:
D_j = domaine de translation.
q = intervalle de variation de q_j.

L'opérateur de réduction permet de se ramener à un espace de type $C_{A_j}^{q}$, en grossissant les obstacles B_i relativement à D_j (cf. figure 7.10). Le calcul des débattements possibles pour q_j peut alors être réalisé au moyen de la fonction G du paragraphe 7.4:

$G_{A,B}(D_j) = F_{A,B'}(r_j)$
avec:
$B' = CO_{D_j}^{xyz}(B)$
r_j = point de référence de D_j

Ce procédé conduit à des approximations qui restent raisonnables dans la mesure où les domaines D_j sont petits (les intervalles attribués aux dq_j sont petits).

Dans la pratique, la fonction G n'est calculée explicitement que pour le dernier d.d.l du bras. Le traitement des autres d.d.l donne uniquement lieu à l'application de l'opérateur de grossissement et à l'exécution d'un test d'interférence (voir paragraphe suivant).

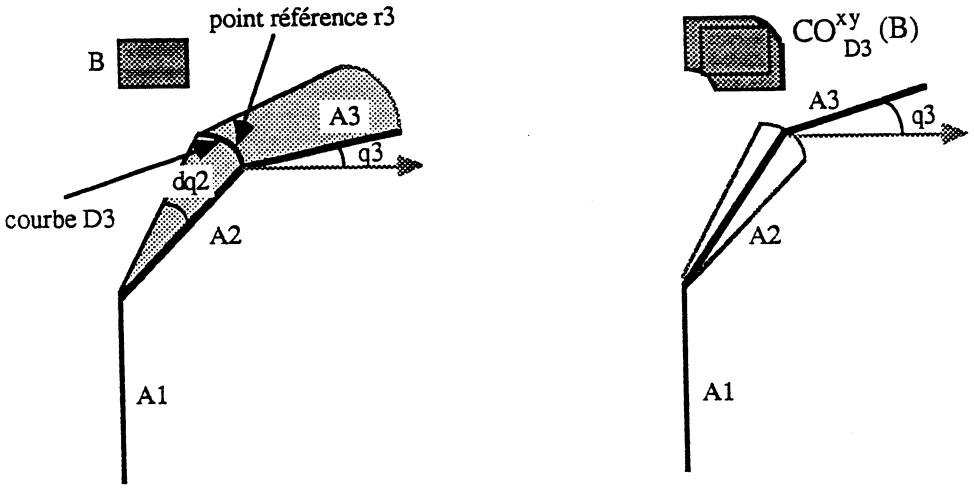


Figure 7.10: Etude du comportement d'un élément de la structure articulée, après avoir fixé les éléments antérieurs:
a- Problème initial: Calculer $\{q_3 \in I_3 : A_3(q_3) \cap B = \emptyset\}$ pour $q_1 = v_1$ et $q_2 \in dq_2$.
b- Problème étudié: Calculer $\{q_3 \in I_3 : A_3(q_3) \cap CO^{xy}_{D_3}(B) = \emptyset\}$ pour $q_1 = v_1$ et $q_2 = v_2$.

7.5.3 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 du bras (de la base vers l'extrémité). Cette opération est rendue nécessaire par le fait que les

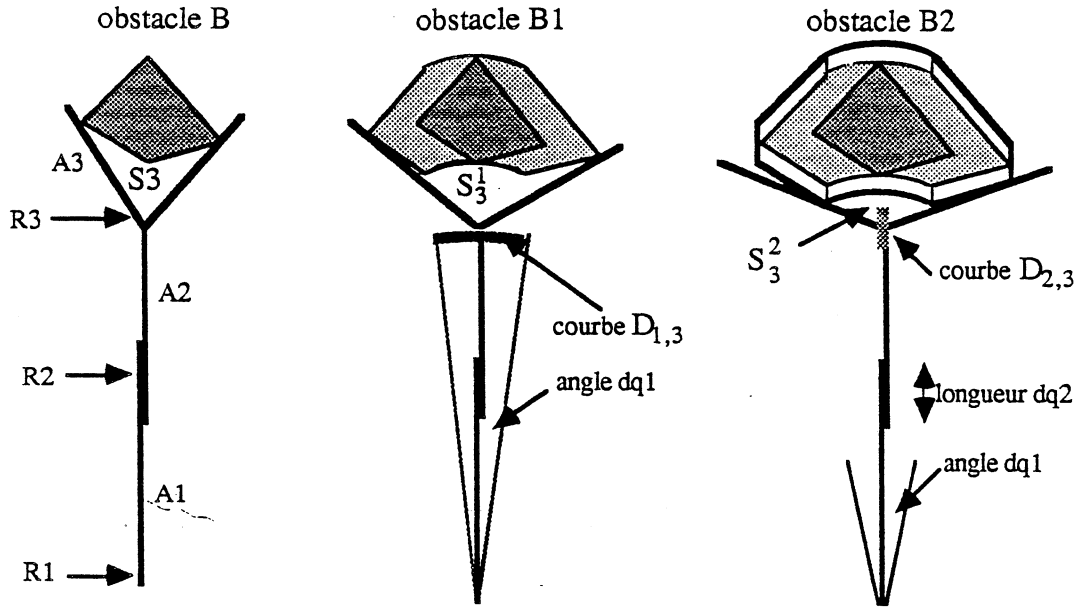


Figure 7.11: Propagation des contraintes de position sur une structure articulée.

contraintes imposées par l'environnement sur un élément A_j du bras, sont toujours dépendantes de celles imposées sur les éléments antérieurs $A1 \cdots Aj-1$. En particulier, le domaine D_j balayé par le point référence R_j de A_j, lorsque les q_i $(i = 1 \cdots j-1)$ décrivent les intervalles dq_i, est une surface complexe de $\Re^3$ qu'il est difficile de représenter avec précision. Un calcul direct de la fonction $G_{A_j,B}(D_j)$ n'est alors pas réellement envisageable.

Le processus de propagation de contraintes que nous avons développé, permet de réaliser ce calcul en plusieurs étapes. Chaque étape met en jeu des calculs simples, qui permettent de construire itérativement des approximations de l'ensemble $CO_{D_j}^{xyz}(B)$. Le principe consiste à appliquer l'opérateur de grossissement pour chaque dq_k $(k = 1 \cdots j-1)$, 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 D_j^k réduit à un segment de droite (articulation prismatique) ou à un arc de cercle (articulation rotoïde); D_j^k représente le lieu de R_j, lorsque

q_k décrit dq_k. L'algorithme appliqué pour calculer $G_{A_j,B}(D_j)$ est donc le suivant:

$B^0 = B$
Pour $k = 1, \cdots j-1$: Calculer $B^k = CO^{xyz}_{D_j^k}(B^{k-1})$
Calculer $F_{A_j,B^{j-1}}(r_j^{j-1})$

dans lequel r_j^{j-1} est le point de référence du domaine D_j^{j-1}.

Afin de pouvoir appliquer les algorithmes de grossissement précédents, les arcs de cercle sont approximés par des segments de droites. Ceci est possible du fait de la faible amplitude des dq_i. Dans le cas contraire, une approximation polynomiale peut être utilisée.

La figure 7.11 illustre le processus de propagation des contraintes. Lorsque $q_1 = v_1$ et $q_2 = v_2$, la contrainte appliquée par B sur $A3$ est représentée par le secteur angulaire S_3. Lorsque q_1 décrit dq_1, cette contrainte est représentée par le secteur S_3^1 défini par l'obstacle $B1$. Lorsque q_1 décrit dq_1 et q_2 décrit dq_2, la contrainte est représentée par le secteur S_3^2 défini par l'obstacle $B2$.

Justification de l'algorithme:

L'algorithme proposé pour calculer les ensembles $G_{A_j,B}(D_j)$ repose sur une définition récursive des ensembles $CO^{xyz}_{D_j}(B)$.

Supposons que cette propriété soit vraie à l'ordre $k-1$, pour $k < j$. L'articulation de l'élément A_k est alors fixée en un point de $\Re^3$, et les C-obstacles associés à A_k sont représentés par les ensembles B^{k-1}. Le point de référence choisi pour l'expression des B^{k-1} est alors R_j.

Si l'on associe un débattement dq_k à A_k, le point R_j décrit un domaine D_j^k de $\Re^3$. L'opérateur de réduction permet alors de fixer la position de R_j relativement à dq_k, en exprimant la conjonction des contraintes imposées par l'expression: $CO^{xyz}_{D_j^k}(B^{k-1})$.

On a donc $B^k = CO^{xyz}_{D_j^k}(B^{k-1})$.

Introduction d'approximations dans les calculs:

En toute rigueur, le processus de propagation devrait être appliqué pour chaque élément A_j et pour chaque tranche de type $dq_1 \times dq_2 \cdots dq_{j-1}$. Dans la pratique, on cherche un "majorant" des domaines D_{i_j} afin de réduire la complexité algorithmique introduite par la multitude des grossissements. Pour une tranche $dq_1 \times dq_2 \cdots dq_{n-1}$, ce majorant correspond au plus grand déplacement cartésien

δ exécuté par les points références R_i $(i = 1 \cdots n-1)$. Le domaine D considéré peut alors être "majoré" par une sphère de rayon δ. Ce procédé permet, au prix d'une perte de précision raisonnable (compte tenu de la faible amplitude des dq_i), d'exécuter un seul grossissement d'obstacles par tranche de type $dq_1 \times dq_2 \cdots dq_{n-1}$:

$$B' = Gros_r(B)$$
$$r = MAX_{i=1\cdots n}\ \{longueur(lieu\ R_i)\}$$

Dans le cas où le découpage des d.d.l est régulier, la même technique peut être utilisée pour l'ensemble des tranches étudiées.

Lozano-Perez [Lozano-Perez 85] utilise une méthode analogue en grossissant les éléments du robot d'une épaisseur égale au plus grand déplacement exécuté. La valeur de ce déplacement peut alors être calculée à l'aide de la matrice jacobienne [Gouzenes 84]:

$$\delta = MAX_{Q \in I_1 \times I_2 \cdots I_n}\ \|J(Q) \cdot dQ\|$$

7.5.4 Complexité algorithmique:

Soit p le nombre de d.d.l du robot, K le nombre d'obstacles, n le nombre moyen d'arêtes d'un objet, et N le nombre moyen de tranches associées à un d.d.l.

D'après ce qui précède on applique un grossissement, $(p-1)$ calculs de collisions et un calcul de débattement valide sur chaque tranche du type $dq_1 \times dq_2 \cdots dq_{p-1}$. La complexité de l'algorithme est donc en $O(N^{p-1})$, si l'on considère ces trois opérations comme une opération élémentaire.

Nous avons vu précédemment que les algorithmes associés à ces opérations ont les complexités maximun suivantes:

- grossissement de type $Gros_r$: $O(n)$
- collisions: $O(n^2)$
- débattements valides: $O(n^2)$

L'ordre de l'algorithme complet en fonction de p, K, n et N est donc le suivant:

$$O(N^{p-1} \star K \star n^2)$$

7.6 Construction de l'espace libre:

L'espace libre EL_A est par définition le complémentaire des C-obstacles de C_A. Le découpage en "tranches" utilisé pour le calcul des C-obstacles permet de

déterminer aisément les régions de cet espace. On obtient alors des "cellules libres" du type $dq_1 \times dq_2 \cdots dq_n$, dans lesquelles les dq_i $(i = 1 \cdots n-1)$ correspondent à des tranches de valeurs articulaires, et les dqn représentent des débattements valides de An. Ces cellules sont construites récursivement par la fonction $LIBRE$:

$$
\begin{aligned}
&LIBRE\ (j,\ D_{i_1} \times D_{i_2} \cdots D_{i_j}) \\
&\qquad \Longleftrightarrow \\
&\qquad LIBRE(j-1,\ D_{i_1} \times D_{i_2} \cdots D_{i_{j-1}}) \\
&\qquad et \\
&\qquad A_j(q_i) \cap B = \emptyset,\ \forall q_i \in D_{i_j}
\end{aligned}
$$

avec:
$LIBRE(1,\ D_{i_1}) \Longleftrightarrow A_1(q_1) \cap B = \emptyset,\ \ \forall q_1 \in D_{i_1}$

Les D_{i_j} $(j = 1 \cdots n-1)$ sont donnés par une fonction de discrétisation des domaines I_j (voir chapitre 9); les D_{i_n} sont calculés à l'aide de la fonction G.

La structure intrinsèque de l'espace libre est donc une structure arborescente dont chaque niveau correspond à un q_i $(i = 1 \cdots n)$, et chaque feuille représente une cellule libre de EL_A. Afin d'être exploitable, cette représentation est complétée par une structure de graphe conduisant à chaîner entre elles toutes les cellules "adjacentes"; on qualifie d'adjacente deux cellules qui vérifient la propriété:

$$dq_i \cap dq'_i \neq \emptyset, \quad i = 1 \cdots n$$

Ce graphe est construit récursivement à l'aide de la fonction suivante:

$$
\begin{aligned}
&ADJACENT\ (j,\ D_{i_1} \times D_{i_2} \cdots D_{i_j},\ D'_{i_1} \times D'_{i_2} \cdots D'_{i_j}) \\
&\qquad \Longleftrightarrow \\
&\qquad ADJACENT(j-1,\ D_{i_1} \times D_{i_2} \cdots D_{i_{j-1}},\ D'_{i_1} \times D'_{i_2} \cdots D'_{i_{j-1}}) \\
&\qquad et \\
&\qquad D_{i_j} \cap D'_{i_j} \neq \emptyset
\end{aligned}
$$

avec:
$ADJACENT(1,\ D_{i_1},\ D'_{i_1}) \Longleftrightarrow D_{i_1} \cap D'_{i_1} \neq \emptyset$

La figure 7.12 montre une représentation de l'espace libre dans le cas d'un robot à deux d.d.l. L'obstacle représenté par la région B_1 s'oppose au déplacement de A_1, lorsque q_1 décrit D_3, D_4 et D_5; il engendre ainsi deux composantes connexes pour l'espace libre: $(C_1\ C_2)$ et $(C_3\ C_4 \cdots C_{10})$. Le domaine D_1 donne lieu à la cellule $C_1 = D_1 \times I_2$, qui est adjacente à la cellule $C_2 = D_2 \times I_2$. Le domaine D_7 engendre deux cellules libres $C_4 = D_7 \times D'_3$ et $C_5 = D_7 \times D'_1$; ces cellules sont toutes les deux adjacentes à la cellule $C_3 = D_6 \times I_2$.

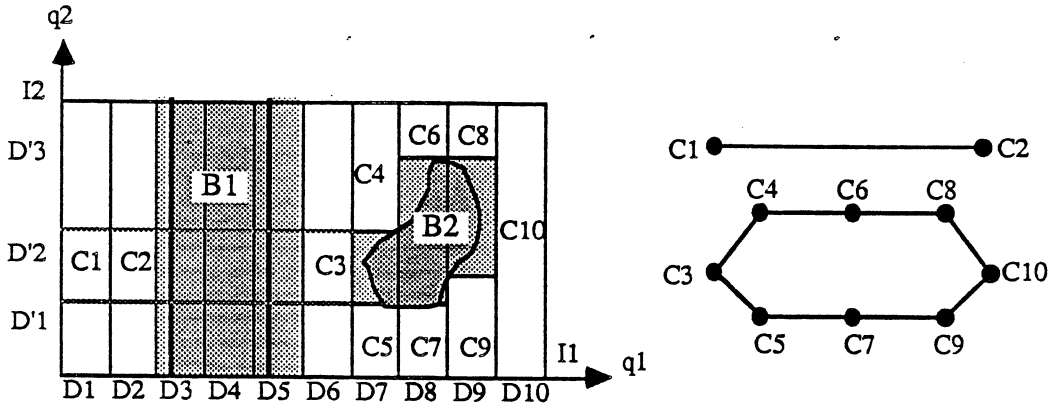


Figure 7.12: Exemple de représentation de l'espace libre associé à un robot à deux d.d.l.

Remarque: Dans le cas où le découpage des d.d.l est réalisé de manière régulière, la représentation de l'espace libre peut être rendue plus compacte en regroupant certaines régions adjacentes dans les tranches dq_i [Lozano-Perez 85]. Nous verrons plus loin qu'un résultat analogue peut être obtenu en adaptant le pas de discrétisation, en fonction des caractéristiques de l'environnement.

7.7 Recherche d'une trajectoire dans l'espace libre:

Le graphe de connectivité associé à l'espace libre représente l'ensemble des chemins possibles pour le robot (au niveau de résolution choisi). Chaque chemin dans ce graphe définit un ensemble connexe $(C_1\ C_2 \cdots C_m)$ de cellules libres du type $dq_1 \times dq_2 \cdots dq_n$. Il représente une enveloppe de trajectoires sûres, permettant au robot de se déplacer sans risque d'une configuration initiale incluse dans $C1$, à une configuration finale incluse dans Cm. Dans ces trajectoires, les configurations qui permettent de passer d'une cellule C_i à une cellule C_{i+1} $(i = 1 \cdots n-1)$, sont choisies dans l'ensemble $C_i \cap C_{i+1}$; les autres configurations définissent des ensembles continus de valeurs, choisis dans les cellules traversées.

Deux facteurs interviennent donc dans l'allure générale des trajectoires engendrées: la recherche d'un chemin dans le graphe, et le choix d'une trajectoire particulière parmi celles proposées.

La recherche d'un chemin optimal est réalisée à l'aide d'un algorithme A^* [Nilsson 80]. L'évaluation heuristique de la solution courante est alors réalisée à l'aide d'une fonction permettant de calculer la distance parcourue, et celle que l'on estime devoir encore parcourir (éloignement du but). Le choix de cette distance conditionne le type de solution que l'on désire obtenir. Dans la pratique on utilise la distance Euclidienne qui permet de minimiser le parcours de l'outil terminal du robot:

$d((q_1\, q_2 \cdots q_n)\,,\, (q'_1\, q'_2 \cdots q'_n)) = dist(p\,,\, p')$
avec:
$dist$ = distance Euclidienne.
p, p' = positions de l'extrémité du bras dans les configurations $(q_1\, q_2 \cdots q_n)$ et $(q'_1\, q'_2 \cdots q'_n)$.

Soit c_{init} la configuration initiale du robot, c_{final} sa configuration finale, et c_i la configuration médiane de la cellule C_i $(i = 2 \cdots m-1)$. Les distances évaluées après parcours de l'ensemble $(C_1\, C_2 \cdots C_j)$ sont les suivantes:

distance parcourue: $d(c_{init}, c1) + d(c1, c2) + \cdots d(c_{j-1}, c_j)$.
éloignement du but: $d(c_j, c_{final})$.

Le choix d'une trajectoire particulière parmi celles incluses dans l'ensemble $(C_1\, C_2 \cdots C_m)$, conduit à déterminer une suite continue de configurations aptes à définir un mouvement du robot. Les cellules C_i étant des hyperparallélépipèdes de C_A, il est possible de joindre en *ligne droite dans* C_A, n'importe quel point de la frontière de C_i. Le problème posé peut donc être ramené à celui du choix des configurations qui assurent la transition entre les cellules consécutives. Une solution classique consiste à choisir la configuration médiane de la zone de chevauchement des deux cellules considérées. Cette approche très simple conduit cependant à introduire des déplacements injustifiés qui nuisent à la "fluidité" du mouvement (en particulier lors de la traversée de grandes cellules). La solution que nous avons adoptée, évite en partie ce problème en recherchant à chaque étape la configuration de $C_{j-1} \cap C_j$ qui est la plus proche (au sens de la distance précédente) de la configuration courante. Des exemples de trajectoires obtenues sont donnés dans le chapitre 9.

Chapitre 8

Le raisonnement morphologique

8.1 Présentation et principes de base:

8.1.1 Problèmes abordés:

Les problèmes abordés par le raisonnement morphologique sont ceux de la planification des opérations de saisie et de montage. Ces deux problèmes se caractérisent par le fait qu'ils donnent un rôle prépondérant aux *contacts*. Trois aspects de ces contacts interviennent dans le mode de raisonnement induit:

- *L'accessibilité locale* des parties à mettre en relation. Il s'agit alors de s'assurer que les solutions étudiées sont localement réalisables, compte tenu de la matière située dans le voisinage des parties en contact. Les calculs mis en jeu sont à base de propriétés géométriques et topologiques simples.
- *Les contraintes mécaniques* engendrées par le contact physique. Il s'agit alors de vérifier la viabilité des solutions analysées, en fonction de propriétés d'équilibre et de stabilité. Les calculs exécutés reposent essentiellement sur des heuristiques, qui conduisent à raisonner qualitativement sur la statique.
- *Les contraintes de mouvement* induites par le contact. Il s'agit alors de s'assurer qu'il existe localement des mouvements qui permettent de réaliser la relation désirée, compte tenu de la géométrie des parties en contact et

des erreurs dues à la commande. Les calculs mis en jeu s'appuient sur une représentation analytique des mouvements potentiels de l'objet mobile.

8.1.2 Raisonnement global mis en jeu:

Le raisonnement mis en jeu pour résoudre les deux problèmes précédents comporte deux phases complémentaires:

- Dans un premier temps, le système recherche des *solutions potentielles* sur la base d'une analyse des propriétés locales des contacts concernés. Par exemple, l'étude des contacts potentiels entre l'outil de préhension et l'objet à saisir permettra d'exhiber un ensemble de *"prises potentielles"*, qui présentent localement toutes les caractéristiques requises. De même, l'analyse des mouvements potentiellement exécutables par un objet en situation de contact, permettra de proposer des *"directions de déplacement"* compatibles avec les contraintes topologiques locales. Ce type de traitement est appliqué pour sélectionner des directions d'approche pour la saisie. Il est surtout utilisé pour choisir des mouvements pertinents lors de la planification des opérations de montage.
  Le mode de raisonnement appliqué au cours de cette phase est appelé **"raisonnement morphologique"**, car il conduit à raisonner sur les indices morphologiques locaux des objets.

- La deuxième phase du traitement a pour objet d'évaluer *l'accessibilité globale* des solutions proposées. Elle opère sur la base d'une analyse des contraintes globales de la tâche, afin de rejeter ou de valider les choix réalisés antérieurement. Les calculs exécutés permettent alors d'affiner et de compléter les solutions retenues. Par exemple, la prise en compte des obstacles aux déplacements des mors conduira à restreindre l'ensemble des positions possibles de la pince. De même, le calcul du débattement associé à une direction de déplacement permettra de spécifier les paramètres du mouvement étudié.
  Le mode de raisonnement appliqué au cours de cette phase relève des techniques de **"raisonnement spatial"** décrites dans le chapitre précédent.

L'approche proposée pour résoudre les problèmes de planification de mouvements en présence de contacts est donc basée sur le principe suivant: *dissocier l'analyse des positions et des mouvements potentiellement réalisables, de l'étude de ceux réellement exécutables.* Elle conduit à faire coopérer deux formes de raisonnement géométrique: le raisonnement morphologique et le raisonnement spatial. Cette approche a été motivée par le fait qu'elle permet de mieux maîtriser la complexité algorithmique, en orientant progressivement les choix au travers d'une analyse de plus en plus complète des contraintes imposées par la géométrie des objets. Cette analyse est alors réalisée au moyen de *filtres géométriques simples*,

appliqués par ordre de complexité croissante. Ce point est développé dans le chapitre 9.

Dans la suite, nous traiterons des modèles et des techniques algorithmiques mis en jeu par le raisonnement morphologique. Certaines propriétés développées sont essentiellement orientées vers la recherche de prises potentielles; ce sont les propriétés liées aux concepts de "contact potentiel" et de "contact contraint". D'autres visent plus particulièrement l'étude des mouvements fins; ce sont les propriétés induites par les contraintes de mouvements.

8.1.3 Outils informatiques nécessaires:

Les outils informatiques qui sont à la base du raisonnement morphologique sont les suivants:

- Des fonctions pour le calcul de *propriétés géométriques élémentaires.* Ces fonctions opèrent sur les entités géométriques qui composent les objets (faces, arêtes, sommets). Elles réalisent des calculs de distance et d'angles, des projections, des intersections, des tests portant sur la colinéarité, le parallélisme ou l'orthogonalité.... Toutes ces opérations sont réalisées avec un coefficient de précision ε, qui permet de prendre en compte les erreurs de modélisation et de calcul.

- Des algorithmes pour l'application *d'opérateurs géométriques* sur les objets. Ces algorithmes opèrent sur les modèles surfaciques des objets. Ils permettent de réaliser des intersections, des unions, des différences, des coupes planes, des calculs d'enveloppes convexes .... Tous ces algorithmes sont très simples dans leur principe, mais ils ont la particularité de devoir traiter de facon homogène un grand nombre de cas particuliers.

- Des fonctions pour le calcul des *propriétés géométriques et topologiques associées aux contacts.* Ces fonctions opèrent sur les entités géométriques qui composent le voisinage des parties en contact. Elles mettent essentiellement en œuvre du calcul vectoriel.

- Des algorithmes pour la *détermination des d.d.l associés aux contacts.* Ces algorithmes opèrent sur un modèle symbolique des contacts. Ils manipulent une représentation analytique des mouvements contraints par les parties en contact (mouvements interdits, mouvements compliants et mouvements libres).

- Des heuristiques qui traduisent sous la forme de propriétés géométriques et topologiques, les *contraintes de stabilité et de robustesse* liées aux contacts.

Les objets traités au cours du raisonnement morphologique sont constitués de polyèdres, de cylindres, de cônes et de sphères.

8.1.4 Plan du chapitre:

Dans ce chapitre, nous ne détaillerons pas les fonctions géométriques classiques qui sont d'un usage courant dans de nombreux système CAO. Nous focaliserons par contre notre attention sur les algorithmes conçus dans le but de traiter les contacts.

Dans le premier paragraphe, nous proposons un modèle de représentation pour les contacts. Ce modèle combine des informations symboliques qui permettent de raisonner sur des familles de contacts, et des données géométriques exploitables pour la quantification des mouvements associés.

Les deux paragraphes suivants décrivent comment est utilisé ce modèle pour définir des propriétés qui caractérisent les différentes configurations de contacts. Nous introduirons pour cela les notions de "contact potentiel" et de "contact contraint".

Le quatrième paragraphe traite des contraintes de mouvement imposées par les contacts. Un formalisme apte à faciliter le raisonnement sur les d.d.l des objets en contact sera proposé et discuté.

Le dernier paragraphe présente de quelle manière il est possible de construire un "graphe des états" possibles de l'ensemble robot-environnement, à partir des informations fournies par le raisonnement morphologique.

8.2 Le concept de contact:

8.2.1 Notion de voisinage:

Le raisonnement sur les contacts étant réalisé de manière locale, il est nécessaire de pouvoir expliciter les parties du modèle qui interviennent dans ce raisonnement. Nous avons introduit pour cela le concept de voisinage, que nous définissons comme suit:

Définition 8.2.1 *Soit M(O) le modèle géométrique associé à un objet O. On appelle voisinage V(E) d'une entité géométrique E de M(O), l'ensemble des entités géométriques e_i de M(O) qui vérifient les propriétés suivantes:*

$$E \cap e_i \neq \emptyset \quad et \quad e_i \not\subset E$$

.

Si E est une face, V(E) est constitué de toutes les faces qui possèdent une arête ou un sommet commun avec E. Si E est une arête, V(E) est constitué des deux faces qui contiennent E, et des arêtes qui possèdent un sommet commun avec E. Si E est un sommet, V(E) est constitué des faces et des arêtes qui contiennent E.

8.2.2 Spécification symbolique des contacts:

Un contact entre deux objets O1 et O2 met en relation deux parties non nécessairement connexes de la surface de ces objets. Il peut être spécifié de la manière suivante:

$$CONTACT(O_1, O_2) = \{((e_1, e'_1)(e_2, e'_2) \cdots (e_m, e'_m)), T, P\}$$

où les couples (e_i, e'_i) représentent les entités géométriques en contact, T définit le type du contact (ponctuel, linéique ou surfacique), et P représente les paramètres géométriques qui caractérisent analytiquement le contact. L'information définie par T est redondante, mais sa présence est très utile pour la conduite des raisonnements qui n'exploitent pas les données quantitatives représentées par P (cf. paragraphe suivant).

Entités géométriques contribuant au contact:

Les entités géométriques présentes dans les couples (e_i, e'_i), proviennent des modèles surfaciques associés aux objets: sommets, arêtes rectilignes ou circulaires, faces planes, cylindriques, coniques ou sphériques. Les combinaisons possibles pour ces entités sont celles qui engendrent des situations physiques potentiellement réalisables malgré les incertitudes de position, c'est à dire tous les couples de type: face-face, face-arête, face-sommet et arête-arête (lorsque ces dernières ne sont pas colinéaires). Les couples de type arête-arête n'étant pas pertinents pour les opérations de saisie et de montage, nous ne prendrons en considération que les contacts qui mettent en jeu des relations de type $face-X$, avec X = face, arête ou sommet. Les couples (e_i, e'_i) d'entités (avec $e_i \in modele(O_1)$ et $e'_i \in modele(O_2)$) qui vérifient de telles relations sont appelés *éléments de contact.*

Dans la suite, nous noterons EG toute entité géométrique qui appartient au modèle d'un objet. Nous désignerons par E_1 (resp. E_2) l'ensemble des EG du modèle de O_1 (resp. de O_2) présents dans les éléments de contact. Certaines EG de E_1 ou de E_2 peuvent apparaître dans plusieurs éléments de contact (cf. figure 8.1).

Type du contact:

Le type d'un contact est défini par la topologie de l'ensemble des points en contact (point, courbes ou surfaces). Les caractéristiques de cet ensemble varient en fonction du type et de l'organisation spatiale des EG concernées. Par exemple, un couple de faces cylindriques "pleines" peut suivant l'orientation respective des deux faces, donner lieu à un contact ponctuel ou linéique. La combinaison de plusieurs éléments de contact contribue aussi à définir le type du contact. Un contact surfacique peut par exemple être construit à partir de deux éléments de contact de

type linéique (cf. figure 8.1). Intuitivement, nous dirons que ce contact multiple peut être assimilé à un contact plan, du fait qu'il possède des propriétés statiques analogues. La surface définie par l'enveloppe convexe des points de contact représente alors un contact plan fictif, dont la caractérisation dérive directement du concept de "polygone de sustentation" utilisé en statique [Joyal, Provost 66].

Définition 8.2.2 *Soit* $E1 \cap E2$ *l'ensemble des points de contact, et* $conv(E1 \cap E2)$ *l'enveloppe convexe de cet ensemble. Le contact* $(E1, E2)$ *est dit de type ponctuel, linéique, surfacique plan ou surfacique courbe, si l'ensemble* $conv(E1 \cap E2)$ *est respectivement un singleton, un ensemble de dimension un, un ensemble de dimension deux, ou un ensemble de dimension trois.*

Dans la suite nous distinguerons parmi les contacts surfaciques courbes, ceux qui sont réalisés suivant des surfaces cylindriques, coniques ou sphériques. Cette distinction est motivée par le fait que ces contacts présentent des caractéristiques différentes vis-à-vis des contraintes de mouvements induites (cf. figure 8.2).

Le nombre réduit de cas possibles permet de déterminer le type des contacts sans passer par un calcul d'enveloppe convexe en $O(n \log n)$, dans lequel n est le nombre d'EG concernées. La tèchnique employée repose alors sur une simple analyse des ensembles $e_i \cap e'_i$ produits par les éléments de contact (e_i, e'_i) [Laugier, Dufay 82]. Dans le cas des contacts multiples, ces éléments sont ensuite combinés par "fusion" des ensembles $e_i \cap e'_i$ associés [Ijel 86]. Par exemple, deux contacts ponctuels mettant en jeu une même face plane, produiront par fusion un contact linéique caractérisé par le segment $P1P2$ ($P1$ et $P2$ sont les points de contacts). L'algorithme proposé calcule également les paramètres géométriques des contacts analysés.

Paramètres géométriques du contact:

Les paramètres géométriques caractérisent analytiquement le contact par un couple $(S, \delta S)$, dans lequel S est une surface définie par les points en contact, et δS délimite la région de S qui contient les points de $conv(E1 \cap E2)$. La surface S est appelée *support du contact*; elle peut être plane, cylindrique, conique ou sphérique. La région délimitée par δS est appelée *surface de contact*; elle peut représenter des domaines plans polygonaux ou circulaires, des segments de droites, des points, ou des secteurs cylindriques, coniques ou sphériques. Les représentations utilisées sont celles présentes dans le modèle de l'univers du robot (cf. paragraphe 6.1): normale extérieure pour une face plane, axe et rayon pour une surface cylindrique, liste des sommets pour un polygone ....

Dans le cas d'un contact surfacique, S est définie sans ambiguïté par l'ensemble $E1 \cap E2$. Dans les autres cas, nous avons choisi de prendre une face de $E1$ ou de $E2$ comme génératrice de S. En cas d'ambiguïté (par exemple: contact entre une

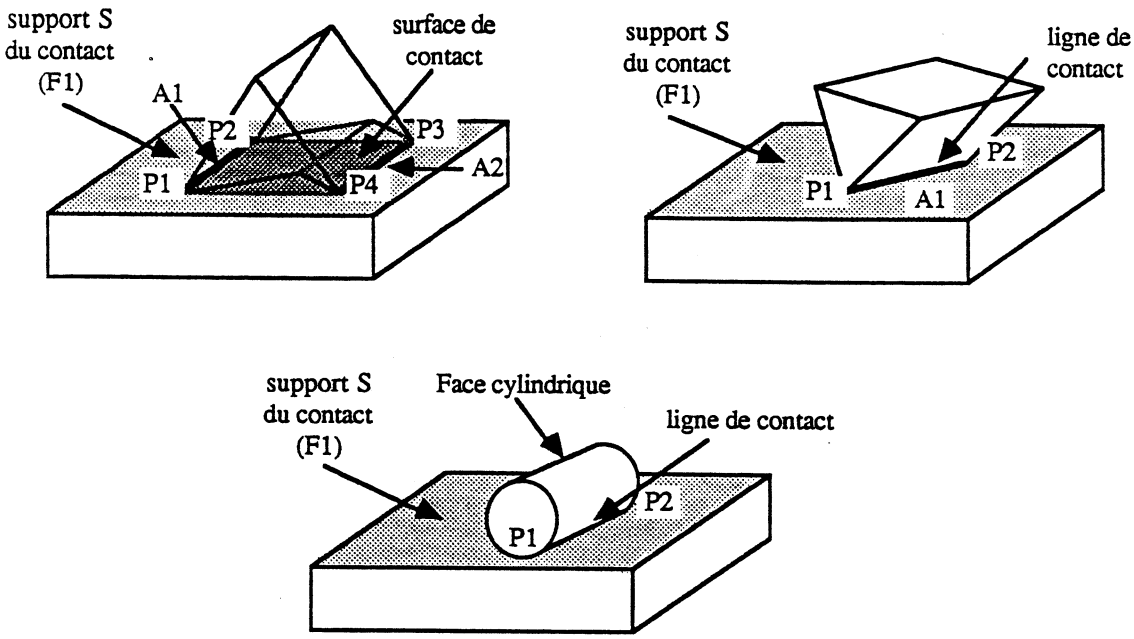


Figure 8.1: Exemples de contacts
a- Contact surfacique engendré par une face plane et deux arêtes.
b- Contact linéique engendré par une face plane et une arête.
c- Contact linéique engendré par une face plane et une face cylindrique.

face cylindrique et une face plane), nous prendrons par convention une des faces de l'objet qui subit le contact, c'est à dire de l'objet non tenu par le robot.

Les contacts illustrés par la figure 8.1 sont alors représentés de la manière suivante:

(a) (CONTACT ((FACE F1 ARETE A1) (FACE F1 ARETE A2))
(TYPE plan)
(SUPPORT plan (0 0 1))
(FRONTIERE (P1 P2 P3 P4)))

(b) (CONTACT ((FACE F1 ARETE A1))
(TYPE droite)
(SUPPORT plan (0 0 1))
(FRONTIERE (P1 P2)))

(c) (CONTACT ((FACE F1 FACE FC2))
(TYPE droite)
(SUPPORT plan (0 0 1))
(FRONTIERE (P1 P2)))

Dans le cas du contact (c), F1 est choisi comme support car cette face appartient à l'objet fixe. Le cylindre peut alors se déplacer sur cette face (dans certaines limites) sans détruire le contact.

8.2.3 Raisonnement sur les contacts:

La nature des *EG* et le type du contact permettent de raisonner symboliquement sur le contact, indépendamment de toute instance de celui-ci. Le système manipule alors des classes de contacts dont les paramètres génériques sont définis par la nature des *EG* concernées, par leur agencement spatial et par leur voisinage. Parmi ces paramètres figurent en particulier certaines caractéristiques mécaniques du contact (robustesse, contraintes portant sur les mouvements relatifs des objets), ainsi que des conditions relatives à "l'accessibilité locale" des parties d'objets concernées. Le tableau de la figure 8.2 définit par exemple, les contraintes de mouvement imposées par les différents type de contacts traités. Le raisonnement symbolique basé sur ce mode de représentation est appliqué pour rechercher des contacts potentiels susceptibles de constituer une prise sur un objet. Il est aussi utilisé pour déterminer les séquences de contacts aptes à guider les mouvements fins de montage.

La surface définie comme support du contact complète la description symbolique. Elle représente explicitement les directions de déplacement que l'on peut potentiellement appliquer à l'objet mobile, sans détruire le contact. Cette information est utilisée pour déterminer les catégories de mouvements possibles pour la pince, lors de la réalisation d'une opération de saisie. Elle est également exploitée pour définir les mouvements compliants nécessaires au montage de deux objets.

La surface de contact caractérise analytiquement une occurence du contact. Elle est utilisée pour déterminer la stabilité de ce contact, en appliquant des règles simples dérivées de la statique. On s'assurera en particulier que la force résultante de réaction passe à l'intérieur de cette surface (qui représente alors la surface de sustentation de l'objet). Le lecteur intéressé pourra trouver dans [Ijel 86] une étude de ce problème, incluant le cas des contacts multiples. En ce qui nous concerne, les tests de stabilité sont exécutés géométriquement par une propriété appropriée: la propriété de "mutuelle visibilité" (cf. paragraphe 8.4). Cette propriété explicite en termes de positions relatives des surfaces de contact, les règles de statique mentionnées ci-dessus.

type de contact	*d.d.l contraints*	
ponctuel	1 d.d.l en translation	0 d.d.l en rotation
linéique	1 d.d.l en translation	1 d.d.l en rotation
plan	1 d.d.l en translation	2 d.d.l en rotation
cylindrique	2 d.d.l en translation	2 d.d.l en rotation
conique	3 d.d.l en translation	2 d.d.l en rotation
sphèrique	3 d.d.l en translation	0 d.d.l en rotation

Figure 8.2: Contraintes de mouvements imposées par les différents types de contacts.

L'étude des variations de la surface de contact lorsque l'objet mobile se déplace, permet de déterminer les situations limites qui entraînent une rupture du contact. De telles situations se produisent en particulier lorsque la frontière δS d'un contact surfacique, dégénère en un ou plusieurs segments de droites localisés sur des arêtes des faces en contact (cf. figure 8.3). Nous parlerons alors de *contact par adjacence.* Ce type de contact est par nature instable et irréalisable matériellement à cause des incertitudes de commande et de perception. Sa détermination symbolique est cependant importante pour le processus de planification des mouvements fins de montage, car il représente une discontinuité produite par la géométrie de l'objet.

Dans un but évident d'efficacité, nous limiterons dans un premier temps les contacts traités à ceux qui ne mettent en jeu que des ensembles réduits à des singletons. Cette restriction est tout à fait raisonnable dans le sens où elle est vérifiée dans la majorité des situations rencontrées dans l'assemblage mécanique. Lorsqu'elle s'avère trop restrictive vis-à-vis d'un problème particulier, il est toujours possible d'envisager une deuxième classe de contacts plus généraux. Une telle situation se produit, par exemple, lorsqu'il est nécessaire de stabiliser une prise instable par un élément de contact supplémentaire.

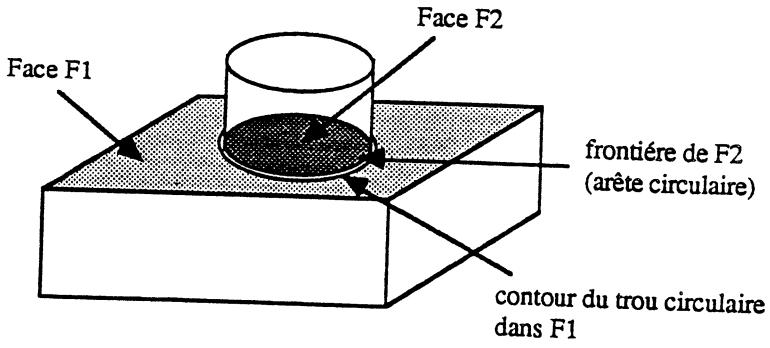


Figure 8.3: Exemple de contact par adjacence.

8.3 Les contacts potentiels:

8.3.1 Convexité locale et contacts potentiels:

La détermination des contacts potentiels constitue une étape préliminaire du processus de planification des opérations de saisie. Cette étape conduit à rechercher les parties de l'objet qui sont susceptibles de servir de support à un contact avec l'un des mors de l'outil de préhension. Ce contact devant être réalisé de manière tangentielle, les parties d'objet concernées sont celles qui vérifient certaines propriétés de "convexité locale" [Laugier 83]:

Définition 8.3.1 *Une partie E de l'enveloppe F d'un objet B est localement convexe (resp. concave), si il existe un hyperplan d'appui local de F en E, tel que la restriction de B (resp. du "vide") à un voisinage de E soit entièrement contenue dans l'un des demi-espaces définis par cet hyperplan.*

Les propriétés de convexité locale et de concavité locale (notées respectivement E-convexité et E-concavité) sont illustrées par la figure 8.4. Leurs caractéristiques topologiques sont discutées dans [Laugier 83].

Intuitivement, il est clair que ces propriétés tentent d'exprimer formellement une caractéristique topologique des contacts: toute EG localement convexe permet potentiellement de produire un contact avec un plan, et toute EG localement concave ne le permet pas. Malheureusement, les propriétés de E-convexité et de E-concavité ne sont pas duales, ce qui conduit à distinguer trois classes d'EG vis-à-vis des contacts: celles qui sont E-convexe, celles qui sont E-concave, et celles qui ne vérifient aucune de ces propriétés. Or si la convexité locale garantit la possibilité d'un contact, l'absence de cette propriété n'implique pas toujours son

impossibilité. En particulier, une face plane qui ne possède aucune des propriétés précédentes peut, sous certaines conditions, servir de support à un contact (cf. figure 8.4).

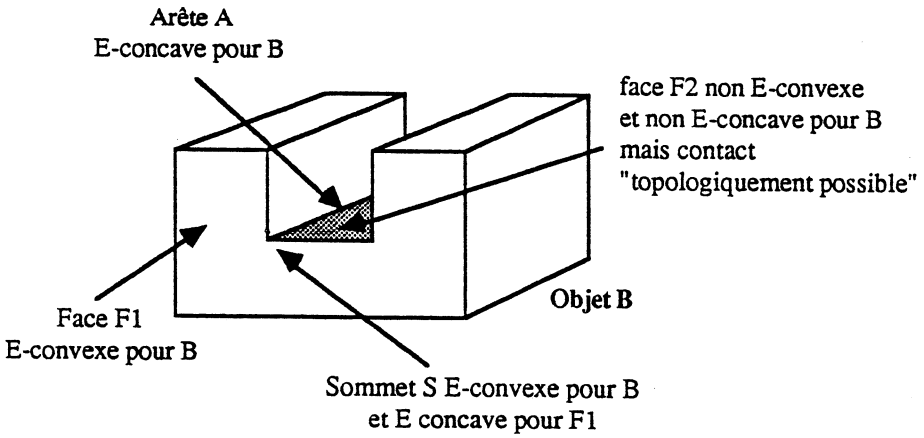


Figure 8.4: Illustration des propriétés de convexité locale.

Il est donc nécessaire, afin de pouvoir conclure formellement sur la possibilité d'un contact, d'établir pour chaque type d'*EG* la relation logique qui existe entre les propriétés de E-convexité, de E-concavité, et de contact potentiel. A cet effet, il peut être facilement démontré que la non-vérification de la propriété de E-convexité pour toute *EG* de type ponctuel ou linéique, entraîne l'impossibilité d'un contact plan (quel que soit le plan choisi, il existe toujours de la matière de part et d'autre de ce plan dans le voisinage de chaque point de cette *EG*). Par contre, l'impossibilité d'un contact plan avec une face F ne peut être établie que comme une conséquence de la E-concavité de F (toute arête convexe du contour de F constitue un support possible pour un contact). Nous caractériserons donc les contacts potentiels de la manière suivante:

Définition 8.3.2 *Un contact tangentiel avec l'objet O est topologiquement possible sur une EG de type ponctuel ou linéique, ssi O est E-convexe au voisinage de cette EG. Il est topologiquement possible sur une face de O, ssi O n'est pas E-concave au voisinage de cette face. L'objet O sera dit "localement accessible" au voisinage de cette EG.*

Remarque: Une caractérisation des contacts mettant en jeu des parties concaves (par exemple: saisie suivant une face cylindrique appartenant à une concavité de

l'objet), peut être réalisée de manière analogue à partir de la propriété de E-concavité [Laugier 83]. Afin de ne pas compliquer l'exposé, nous ne traiterons dans la suite que de contacts réalisés sur des parties convexes. Nous utiliserons également une formulation de la propriété "d'accessibilité locale" similaire à celle introduite dans [Troccaz 86].

8.3.2 Caractérisation analytique des propriétés:

En appliquant la définition 8.3.2, il est possible de caractériser analytiquement la propriété d'accessibilité locale (donc les contacts potentiels), à partir des expressions établies pour la convexité locale. Le détail et la justification de ces expressions sont donnés dans [Laugier 83]. Nous n'utiliserons dans ce qui suit que les résultats pertinents pour le problème qui nous occupe.

Propriété 8.3.1 *Un objet O est localement accessible au voisinage d'une de ses faces cylindriques, coniques ou sphériques, si cette face appartient à un composant "plein" de O.*

Propriété 8.3.2 *Un objet O est localement accessible au voisinage d'une de ses faces planes F, si il existe une arête A de F telle que O soit localement accessible au voisinage de A.*

Propriété 8.3.3 *Un objet O est localement accessible au voisinage d'une de ses arêtes A commune à deux faces planes $F1$ et $F2$, si le produit mixte $V_{A1} \cdot (N1 \wedge N2)$ est négatif. $N1$ et $N2$ sont les normales extérieures respectives des faces $F1$ et $F2$; V_{A1} est le vecteur porté par A, et orienté avec la convention "matière à droite" par rapport à $F1$ (cf. figure 8.5).*

Propriété 8.3.4 *Un objet O est localement accessible au voisinage d'un de ses sommets S commun à trois arêtes $A1$, $A2$ et $A3$, si O est localement accessible au voisinage de $A1$, de $A2$ et de $A3$.*

Les quatre propriétés énoncées ci-dessus , sont utilisées de manière courante lors de la recherche des contacts potentiels. Il existe cependant quelques situations topologiques (que l'on rencontre rarement dans l'assemblage mécanique), qui ne sont pas couvertes par les expressions qui précèdent. Ces situations sont celles où un sommet est relié à plus de trois arêtes (cf. figure 8.6). Il est dans ce cas nécessaire d'évaluer des expressions plus complètes:

Propriété 8.3.5 *Une face plane F est E-convexe au voisinage d'un de ses sommets S commun à deux arêtes $A1$ et $A2$, ssi le produit mixte $N \cdot (V_{A1} \wedge V_{A2})$ est*

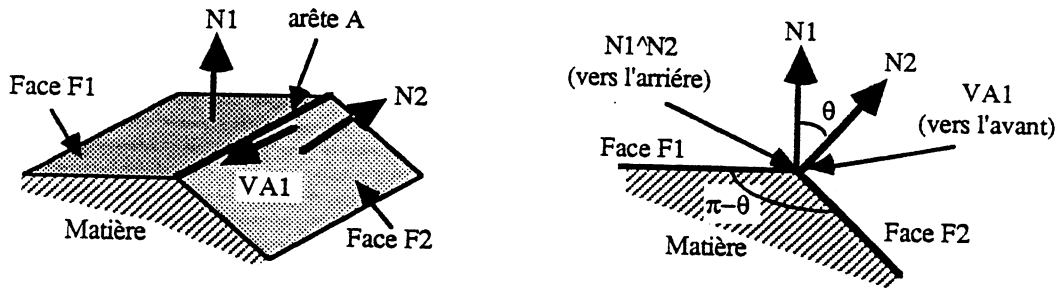


Figure 8.5: Propriété de convexité locale pour une arête rectiligne: $V_{A1} \cdot (N_1 \wedge N_2) \leq 0$.

négatif. N est la normale extérieure à la face ; V_{A1} et V_{A2} sont les vecteurs portés par $A1$ et $A2$, orientés avec la convention "matière à droite", et ordonnées de manière à ce que $A1$ précède $A2$ sur le contour orienté de F (cf. figure 8.6).

Propriété 8.3.6 *Un objet O est localement accessible au voisinage d'un de ses sommets S, si il existe un couple (A_i, A_j) d'arêtes non colinéaires de $V(S)$ tel que:*

- *O est localement accessible au voisinage de A_i et de A_j,*
- *toute face F_k de $V(S)$ est E-convexe au voisinage de S,*
- *le produit mixte $V_{Ak} \cdot (V_{Ai} \wedge V_{Aj})$ possède un signe constant pour toute arête A_k de $V(S) - \{A_i, A_j\}$. V_{Ak}, V_{Ai}, V_{Aj} sont les vecteurs portés par A_k, A_i et A_j, et orientés à partir de S (cf. figure 8.6).*

Remarque: La deuxième condition de la propriété 8.3.6 permet de s'assurer qu'il n'y a pas de concavité locale introduite par les faces de $V(S)$. La troisième condition permet ensuite de vérifier que les arêtes de $V(S)$ sont toutes localisées du même côté du plan (A_i, A_j).

Les propriétés qui précèdent permettent, au prix de calculs vectoriels simples, d'établir si il est topologiquement possible de réaliser un contact dans le voisinage d'une EG donnée de l'objet. Dans le cas d'une face plane, la propriété évaluée permet aussi d'attribuer un coefficient mesurant le "degré d'accessibilité" de la

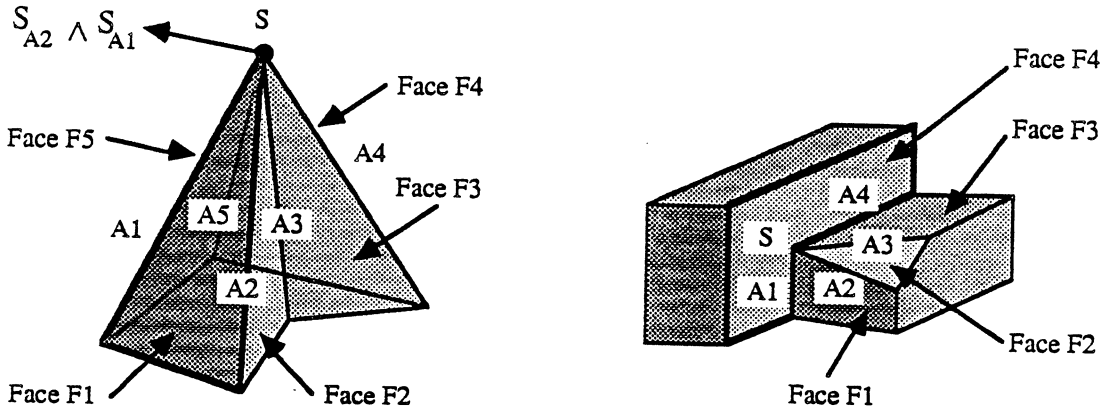


Figure 8.6: Propriété d'accessibilité locale pour un sommet relié à plus de trois arêtes:
a- O est localement accessible au voisinage de S:
- O est localement accessible au voisinage de $A1$ et $A2$.
- S est E-convexe pour les faces $F1, F2, F3, F4$ *et* $F5$.
- $X \cdot (V_{A1} \wedge V_{A2}) < 0$ pour $X = V_{A3}, V_{A4}, V_{A5}$.

b- O n'est pas localement accessible au voisinage de S:
- O est localement accessible au voisinage de $A2$ et $A3$.
- S est E-concave pour la face $F4$.
- $X \cdot (V_{A2} \wedge V_{A3}) < 0$ pour $X = V_{A1}, V_{A4}$.

face. Ce coefficient pourra ensuite être exploité heuristiquement: une face plane dont toutes les arêtes sont localement accessibles aura de meilleures chances de servir de support à un contact, qu'une autre face dont seulement quelques arêtes vérifient cette propriété.

La propriété d'accessibilité locale repose uniquement sur des considérations topologiques. Elle ne permet pas de ce fait de décider de l'accessibilité réelle des parties analysées, faute de données géométriques plus complètes. Par exemple, la face F2 de l'objet de la figure 8.4 est localement accessible; mais cette face ne pourra servir de support à un contact avec un mors de la pince, si ce dernier est plus large que le passage créé par la concavité de l'objet. Nous verrons plus loin que cet aspect dynamique des contacts est traité par des méthodes issues du

raisonnement spatial.

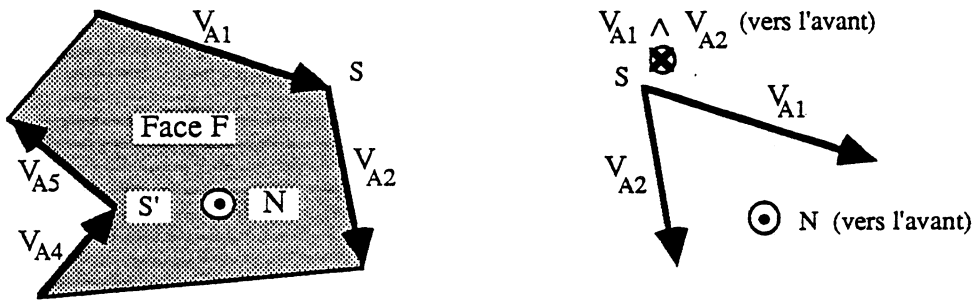


Figure 8.7: Propriété de convexité locale pour une face plane polygonale:
a- Le sommet S est E-convexe pour F: $N \cdot (V_{A1} \wedge V_{A2}) < 0$.
b- Le sommet S' est E-concave pour F: $N \cdot (V_{A4} \wedge V_{A5}) > 0$.

8.4 Les contacts contraints:

8.4.1 Le concept de contact contraint:

Le concept de "contact contraint" est utilisé pour construire des ensembles de contacts potentiels, susceptibles de se *produire simultanément* entre deux objets. Afin d'être physiquement réalisables, ces ensembles de contacts doivent vérifier certaines propriétés topologiques, géométriques et statiques. De telles propriétés contraignent l'ensemble des solutions possibles. Elles sont évaluées à l'aide de "filtres géométriques" simples, dont les caractéristiques varient en fonction de la morphologie de l'objet que l'on déplace. Ce type de raisonnement est utilisé pour contruire les *"prises potentielles"* sur lesquelles devront s'appuyer les opérations de saisie (cf. paragraphe 9.3). Lorsque l'outil de préhension considéré est une pince à deux mors parallèles, les ensembles de contacts traités sont constitués de deux éléments vérifiant des propriétés dérivées du parallélisme.

Trois types de contraintes sont à satisfaire par des contacts potentiels, pour qu'ils soient susceptibles de définir une prise sur une objet [Laugier 81] [Troccaz 86]:

- Des contraintes strictes portant sur la géométrie et la topologie des parties en contact. Ces contraintes dérivent de la morphologie de l'outil de

préhension. Elles donnent lieu à l'évaluation de propriétés géométriques simples: parallélisme ou contraintes angulaires, distance, contraintes topologiques (P-convexité).

- Des contraintes mécaniques portant sur l'équilibre du système main/objet. Ces contraintes sont explicitées sous la forme de conditions simples liées à la géométrie des parties en contact. Elles sont évaluées par la propriété de "mutuelle visibilité".

- Des contraintes mécaniques portant sur la robustesse des prises vis-à-vis des petites variations de position dues aux imprécisions de la commande. Ces contraintes visent à s'assurer que l'équilibre créé par les contacts reste stable, malgré les aléas du monde réel. Elles sont traitées heuristiquement par une simple analyse des types de contacts.

8.4.2 Les contraintes topologiques et géométriques:

Topologie locale des contacts:

Les paramètres morphologiques de l'outil de préhension imposent des contraintes sur les orientations des contacts, et sur la répartition de la matière au voisinage de ces contacts. Intuitivement, il s'agit de trouver des plans d'appuis respectant certaines contraintes d'orientation (celles imposées par la géométrie des mors), et pouvant être placés tangentiellement aux parties en contact de manière à ce que la matière au voisinage de ces contacts soit localisée dans la région délimitée par les plans. Nous utiliserons pour effectuer ce calcul la propriété de P-convexité, qui peut être énoncée comme suit [Laugier 83]:

Définition 8.4.1 *Une partie E de l'enveloppe F d'un objet B est localement P-convexe (resp. P-concave) par rapport à une direction plane P, si il existe un plan Q d'appui local de F en E, tel que Q soit parallèle à P et que la restriction de B (resp. du "vide") à un voisinage de E soit entièrement contenue dans l'un des demi-espaces définis par Q.*

Cette propriété dérive directement de la propriété de convexité locale, après avoir imposé une contrainte supplémentaire portant sur la direction du plan d'appui. Comme dans le cas des contacts potentiels, nous utiliserons dans les calculs une autre formulation de cette propriété. Nous parlerons alors de *"P-accessibilité locale"*.

Dans l'expression analytique des propriétés, nous représenterons la contrainte plane par la normale extérieure N_p au plan P donné (P représente alors la surface d'appui de l'un des mors de la pince).

Propriété 8.4.1 *Un objet O est localement P-accessible au voisinage d'une de ses faces F, si:*

- *O est localement accessible au voisinage de F,*
- *il existe une normale extérieure N de F, telle que $N = -N_p$.*

Remarque: La deuxième condition est immédiate pour des faces planes ou sphériques. Elle se traduit dans les autres cas par des conditions angulaires entre N_p et l'axe de révolution de la face (orthogonalité ou angle donné).

Propriété 8.4.2 *Un objet O est localement P-accessible au voisinage d'une de ses arêtes A commune à deux faces $F1$ et $F2$ si:*

- *O est localement accessible au voisinage de A,*
- *il existe deux réels négatifs a et b tels que $N_p = aN1 + bN2$, où $N1$ et $N2$ sont les normales extérieures respectives de $F1$ et $F2$.*

Remarque: La deuxième condition exprime le fait que A est parallèle à P; elle définit aussi les positions relatives des vecteurs N_p, $N1$ et $N2$ (donc du plan P et des faces $F1$ et $F2$), par le signe des coefficients a et b.

Propriété 8.4.3 *Un objet B est localement P-accessible au voisinage d'un de ses sommets S si:*

- *O est localement accessible au voisinage de S,*
- *le produit scalaire $V_A \cdot N_p$ est de signe négatif pour toute arête A de $V(S)$; V_A est le vecteur porté par A, et orienté à partir de S.*

Remarque: La deuxième condition exprime le fait que toutes les arêtes du voisinage de S, sont situées du côté "libre" de P (c'est à dire à l'extérieur du mors).

Géométrie des contacts:

Les paramètres morphologiques de l'outil de préhension imposent aussi des contraintes sur l'organisation spatiale des contacts: les surfaces de contact doivent être localisées simultanément sur les faces des différents mors, et la distance qui sépare ces surfaces doit rester compatible avec les débattements maximum des mors.

Le premier type de contrainte est pris en compte par la propriété de "mutuelle visibilité", en même temps que les contraintes liées à l'équilibre (voir ci-après).

L'autre type de contrainte est aisément traité par un calcul de distance portant sur les EG impliquées dans le contact. Ce calcul est exécuté par des expressions analytiques simples, paramétrées par la nature des EG concernées et par la configuration topologique des contacts. Par exemple, la distance entre une face plane P et une face cylindrique d'axe D et de rayon R, sera définie comme suit:

- $d(D,P)+R$ si les contacts sont localisés "vers l'extérieur" de la région définie par les plans de contact (cas d'un prise réalisée par serrage);
- $d(D,P) - R$ si les contacts sont localisés "vers l'extérieur" de la région précédente (cas d'une prise réalisée par écartement des mors).

8.4.3 Les contraintes mécaniques:

Equilibre:

L'équilibre du système créé par les contacts est analysé géométriquement par la propriété dite de *"mutuelle visibilité"*. Ainsi que nous l'avons déjà mentionné, cette propriété explicite en termes de positions relatives des surfaces de contact, des règles de base de la statique. L'idée de base consiste à s'assurer qu'il est possible d'associer les contacts de telle sorte que la force résultante de réaction de l'un passe par la surface de contact de l'autre. En cas d'absence de frottements, cette condition peut être aisément exprimée géométriquement du fait que les forces de réactions sont normales aux surfaces de contact (et donc aux plans des mors).

Dans le cas d'une pince à deux mors parallèles, la condition induite porte sur l'existence de points de contact localisés symétriquement par rapport au plan médian de la pince. Cette condition peut être évaluée en projetant orthogonalement les surfaces de contact sur ce plan, puis en vérifiant que l'intersection des domaines obtenus n'est pas vide (cf. figure 8.8).

La propriété de mutuelle visibilité telle que nous l'avons définie, conduit à rejeter certaines configurations de contacts maintenues en équilibre par la présence de forces de frottement. Cette situation est parfaitement cohérente avec les objectifs visés, qui consistent à ne retenir que les solutions sûres (c'est à dire celles qui sont insensibles aux incertitudes de position).

Stabilité:

Les contraintes de stabilité associées à un ensemble de contacts, conduisent à s'assurer qu'une petite modification des positions relatives de ces contacts n'entraîne pas une rupture de l'équilibre (et donc une rotation de l'objet saisi). Une technique possible pour éviter ce phénomène (appelé "twisting" dans la littérature), consiste à engendrer des configurations de contacts qui ne laissent aucun d.d.l en rotation dans les directions sujettes aux erreurs. Nous éliminerons donc les constructions qui présentent des rotations possibles suivant des axes non perpendiculaires aux

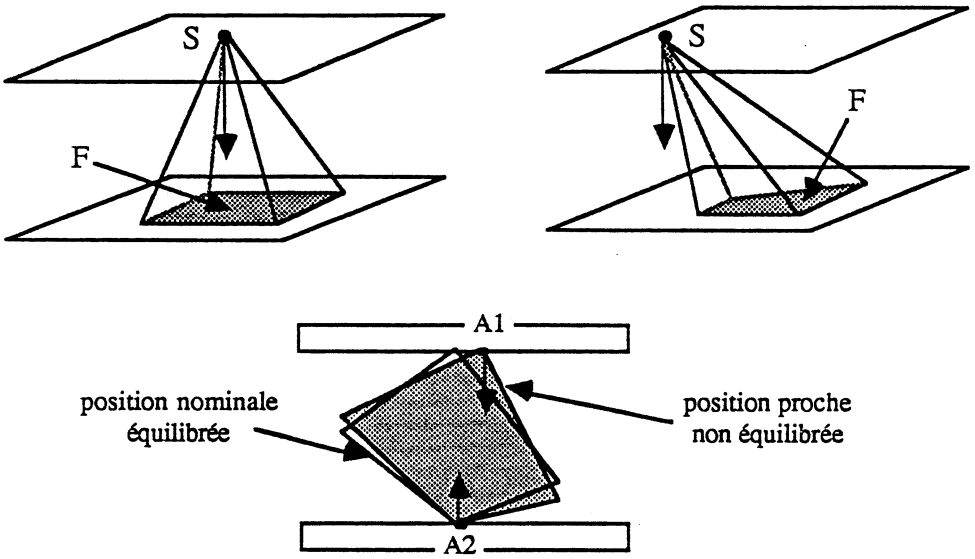


Figure 8.8: Illustration des propriétés d'équilibre et de stabilité:
a- Couple (S, F) équilibré: $proj(S) \in F$.
b- Couple (S, F) non équilibré: $proj(S) \notin F$.
c- Couple $(A1, A2)$ instable.

plans de contact (faces internes des mors), ou non parallèles à un axe de révolution de l'objet. Par exemple, un couple de contacts linéiques parallèles laisse libre une rotation susceptible de faire tourner l'objet, si ces contacts ne sont pas positionnés symétriquement sur les deux mors de la pince (car il y a alors création d'un couple dû aux forces de serrage, cf. figure 8.8). Le traitement des contraintes de stabilité est réalisé sur la base d'une simple analyse des types de contacts mis en jeu, et des d.d.l qui leurs sont associés. Le tableau de la figure 8.2 présente les données prises en compte pour ce raisonnement.

8.5 Contraintes de mouvements associées aux contacts:

8.5.1 Notion de mouvement potentiel:

Chaque contact réduit le nombre de d.d.l de l'objet mobile, c'est à dire de l'outil terminal ou de l'objet tenu par le robot. Afin de pouvoir décider des mouve-

ments à exécuter, le système de planification doit raisonner sur les directions de déplacement contraintes par les contacts. Ce type de raisonnement est appliqué pour construire les séquences de mouvements fins. Il est aussi utilisé pour choisir les déplacements de l'outil de préhension qui permettent de réaliser les prises désirées.

Le problème à résoudre est alors celui de la détermination des mouvements qui sont *potentiellement exécutables* par l'objet mobile, à partir d'une situation mettant en jeu une ou plusieurs relations de contact. Ces mouvements définissent soit des translations pures, soit des rotations pures. Ils sont considérés sur un plan *strictement local*, indépendamment des amplitudes de déplacement qu'il est possible de leur associer. Si A est un objet mobile en contact avec un autre objet B, nous définirons un mouvement potentiel pour A de la manière suivante:

Définition 8.5.1 *Soit m un mouvement simple de type translation ou rotation ne comportant aucune condition d'arrêt. m est appelé mouvement potentiel pour A relativement aux contacts $C1, C2, \cdots Cn$, ssi il existe une amplitude ds supérieure à l'erreur de commande ε, telle que le mouvement de direction m et d'amplitude $2ds$ n'engendre aucune collision entre les parties en contact.*

Définition 8.5.2 *Un mouvement potentiel sera dit d'ordre 0 si il conduit à relâcher tous les contacts initiaux. Il sera respectivement d'ordre $1, 2, \ldots n$, si il permet de conserver $1, 2, \ldots n$ contacts.*

Il est clair que les mouvements d'ordre n ($n > 0$) représentent des mouvements compliants qui prennent appui sur des surfaces de contact, et que ceux d'ordre 0 conduisent à placer l'objet A dans l'espace libre. Les conséquences globales du mouvement (création d'un nouveau contact en particulier) ne sont par contre pas analysées à ce niveau, du fait du caractère local de la représentation. Par exemple, un mouvement d'ordre 1 lié à un couple de contacts $(C1, C2)$ conduira à relâcher le contact $C1$ et à glisser sur la surface associée à $C2$; le fait que ce mouvement puisse se terminer après apparition d'un nouveau contact $C3$, relève d'un raisonnement plus global qui n'est pas réalisé à ce niveau. Cet aspect est traité dans un deuxième temps au moyen d'un calcul de "débattement". Ce n'est qu'à l'issue de ce calcul que le mouvement potentiel m est instancié en un mouvement complet, permettant de passer sans ambiguïté de l'état caractérisé par le couple $(C1, C2)$ à celui défini par le couple $(C3, C2)$. Ce point est développé dans le paragraphe 9.4.

La définition 8.5.1 donne une caractérisation formelle de la notion de mouvement potentiel. Elle n'est cependant pas directement utilisable pour la détermination de ces mouvements. C'est pourquoi nous avons développé un support analytique pour représenter explicitement les mouvements interdits, ceux qui conservent les

contacts et ceux qui les détruisent. Ce formalisme mathématique est décrit dans le paragraphe qui suit.

8.5.2 Représentation analytique des mouvements potentiels:

Le modèle de représentation proposé dans ce paragraphe est développé dans le cadre de mouvements exécutés en translation. Son extension au cas des rotations est discutée à la fin du paragraphe.

Une translation de l'espace tridimensionnel est communément représentée par un vecteur de $\Re^3$. Dans notre modèle, nous avons choisi de la représenter par un couple (v, a), dans lequel v est un vecteur unitaire de $\Re^3$ et a est un réel. v et a définissent respectivement la direction de la translation et son amplitude. Un groupe de translations peut alors être vu comme un sous-ensemble de $S(1)X\Re$, dans lequel $S(1)$ est la sphère unité. Ce mode de représentation permet de raisonner séparément sur les directions de déplacements (ensembles de vecteurs v), et sur les mouvements complets (ensembles de couples (v, a)). Dans ce paragraphe, nous nous intéressons uniquement aux ensembles de vecteurs de $S(1)$.

Un point de la sphère unité définit sans ambiguïté une direction de translation, et une calotte sphérique représente un ensemble connexe de déplacements possibles. Par exemple, une face plane intervenant dans un contact engendre un domaine en forme de demi-sphère, et un couple de contacts détermine un ensemble de déplacements valides délimité par l'intersection des domaines associés à chacun des contacts (cf. figure 8.9). Le calcul des d.d.l résultants peut alors être exécuté dans un espace bidimensionnel, en projetant les domaines obtenus sur un plan (φ, ϑ).

Contacts à supports plans: Dans le cas des contacts ayant pour support une surface plane (contacts de type "face plane - X", avec X quelconque), ce calcul met en jeu des fonctions du type [Theveneau 85]:

$$\vartheta = \arctan((N_x \cdot \cos\varphi + N_y \cdot \sin\varphi)/ - N_z) \tag{8.1}$$

où $(N_x\, N_y\, N_z)$ représente la normale extérieure au plan de contact, c'est à dire la normale orientée vers l'extérieur de l'objet fixe. La calotte sphérique représentée est alors une demi-sphère délimitée par le plan de contact d'équation $N_x.x + N_y.y + N_z.z = 0$. Les principales caractéristiques de cette fonction sont illustrées par la figure 8.10. Il peut être aisément montré que les fonctions de cette famille possèdent de "bonnes propriétés", qui permettent de calculer très facilement les domaines résultants. Ce calcul est illustré graphiquement par la figure 8.11. Son étude détaillée n'étant pas utile pour la compréhension de la suite, nous renvoyons le lecteur intéressé à [Theveneau 85].

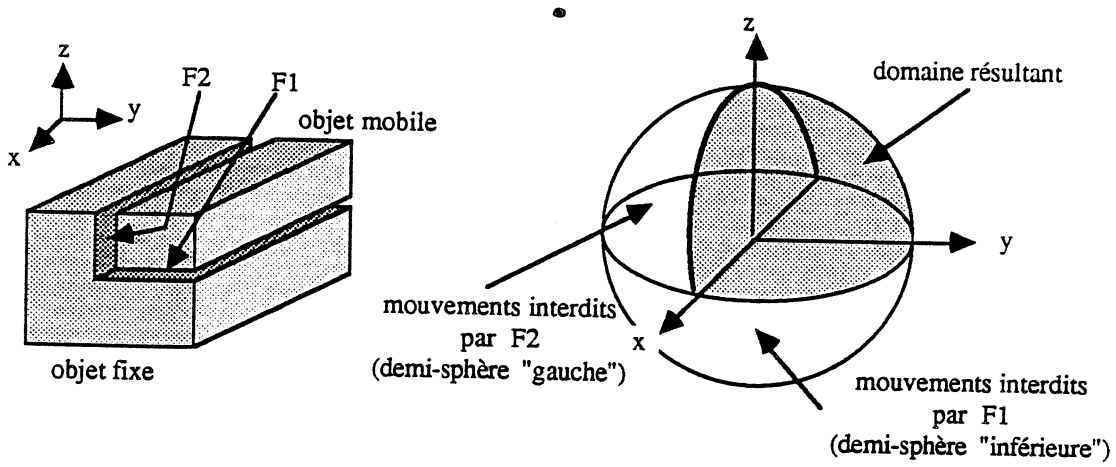


Figure 8.9: Représentation sphérique des contraintes de mouvement.

Contacts à supports non planaires: Un calcul similaire peut être appliqué lorsque les parties en contact ne sont pas planes. Les fonctions utilisées pour ce calcul varient alors avec le type des surfaces traitées (cf. figure 8.12):

- Un contact de type sphérique ne laisse aucun d.d.l à l'objet mobile.
- Un contact de type cylindrique détermine deux directions de mouvements portées par l'axe du cylindre.
- Un contact de type conique définit un ensemble de mouvements potentiels délimité par la fonction:

$$\cos\vartheta(a\cos\varphi + b\sin\varphi) + c\cos\vartheta - \cos\alpha = 0 \qquad (8.2)$$

où $(a\,b\,c)$ et α représentent respectivement l'axe du cône et son angle d'ouverture.

D'autres types de contacts peuvent être engendrés par un couple d'EG non planaires. Une technique possible pour se ramener aux cas connus consiste à introduire des plans de contacts fictifs, définis tangentiellement aux parties en contact. Il est alors possible de traiter ces contacts en combinant simplement les fonctions précédentes (cf. figure 8.13):

- Un contact "tangentiel" de type ponctuel ou linéique est assimilé à un contact plan. Les mouvements potentiels associés sont alors définis par

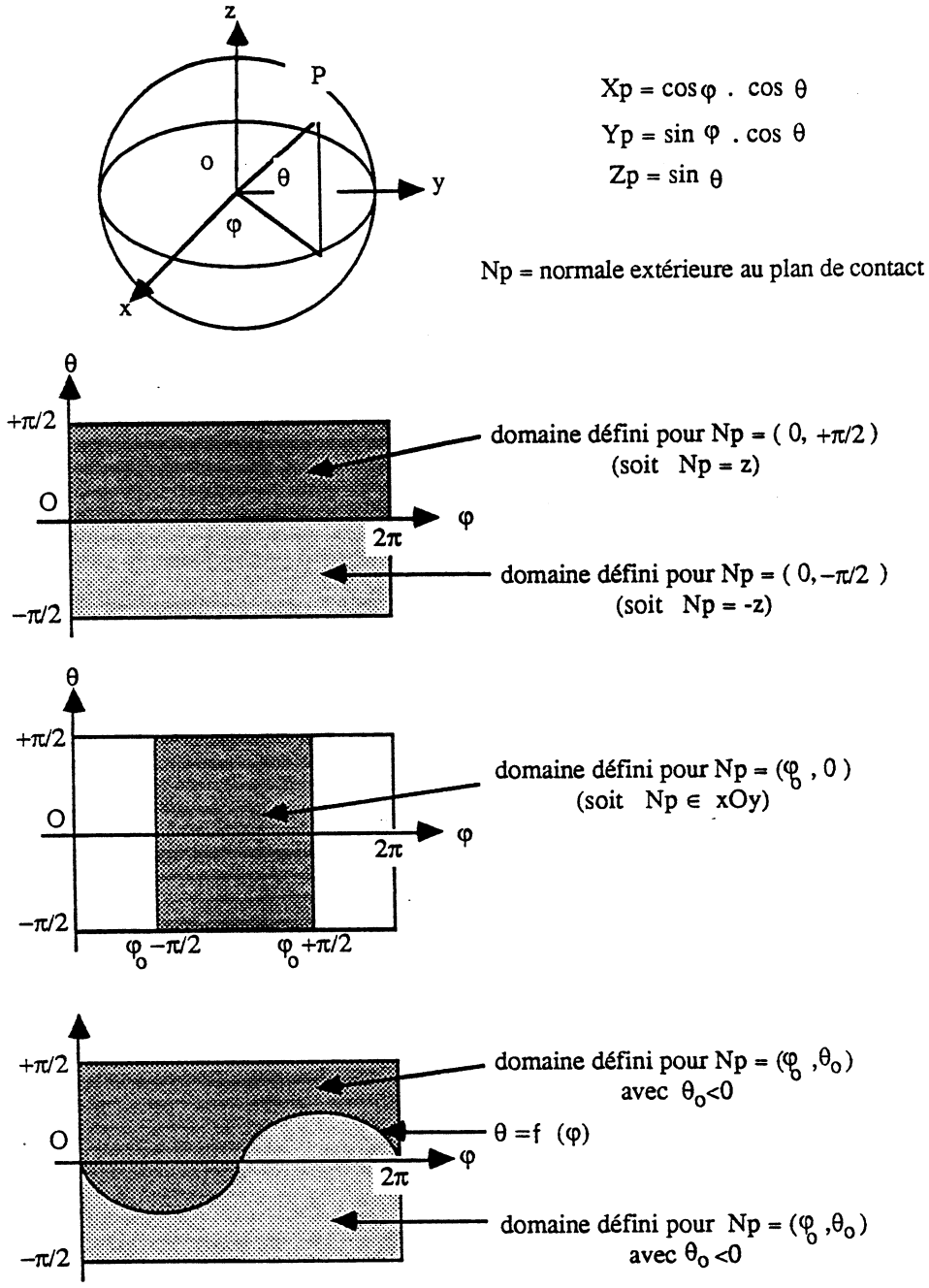


Figure 8.10: Caractérisation des domaines délimités par la fonction: $\vartheta = \arctan((N_x \cdot \cos\varphi + N_y \cdot \sin\varphi)/ - N_z)$.

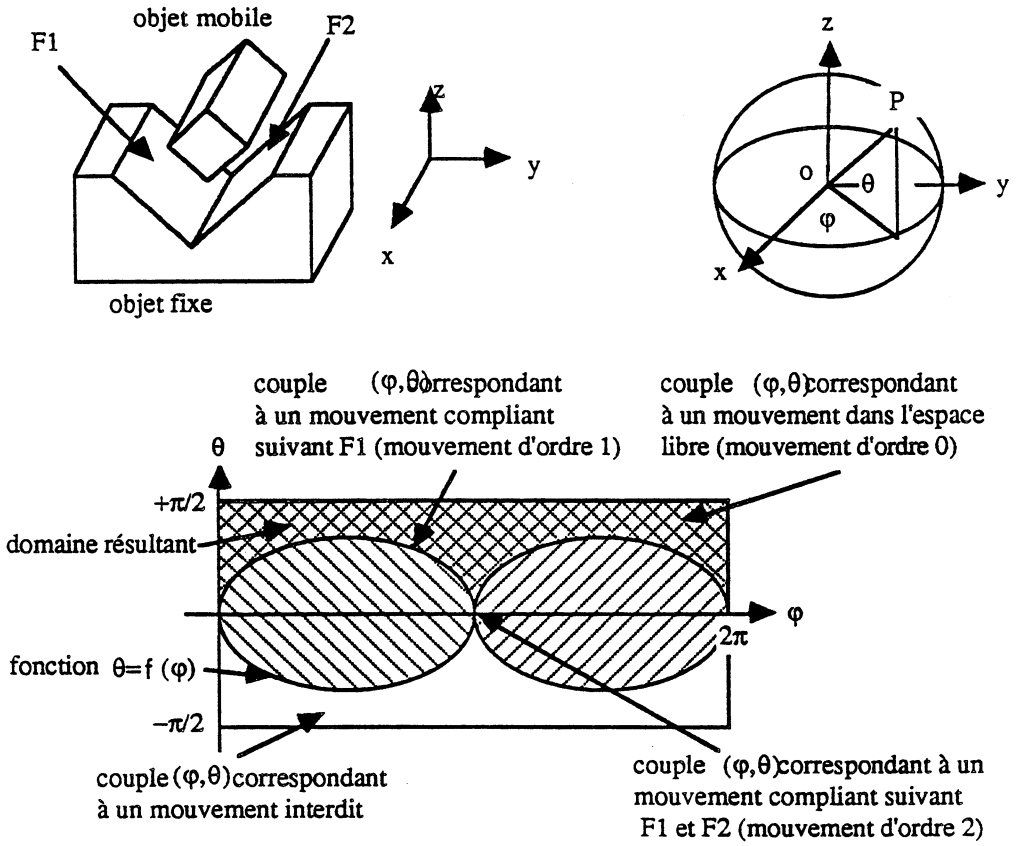


Figure 8.11: Représentation analytique des mouvements potentiels associés à un couple de contacts plans.

la fonction (8.1), dans laquelle $(N_x\, N_y\, N_z)$ représente la normale au plan tangent.

- Un contact surfacique ne mettant en jeu qu'une portion de la surface d'une EG, détermine un domaine délimité à la fois par le contact surfacique et par les plans tangents situés aux bornes de cette surface. Le calcul est alors réalisé à l'aide des fonctions (8.1) et (8.2). Par exemple, le contact cylindrique de la figure 8.13 définit un domaine caractérisé par l'expression "$D(P1) \cap D(P2) \cap D(P3) \cup D(cylindre)$", dans laquelle $D(P1)$, $D(P2)$ et $D(P3)$ représentent respectivement les domaines associés aux plans tangents $P1$, $P2$ et $P3$ (ces trois domaines sont calculés à l'aide de la fonction (8.1)), et $D(cylindre)$ correspond aux deux directions axiales. De même, un contact

conique partiel engendrera un domaine du type "$D(P1) \cap D(P2) \cap D(P3) \cup D(cone)$", dans lequel $D(cone)$ est un domaine calculé à l'aide de la fonction (8.2).

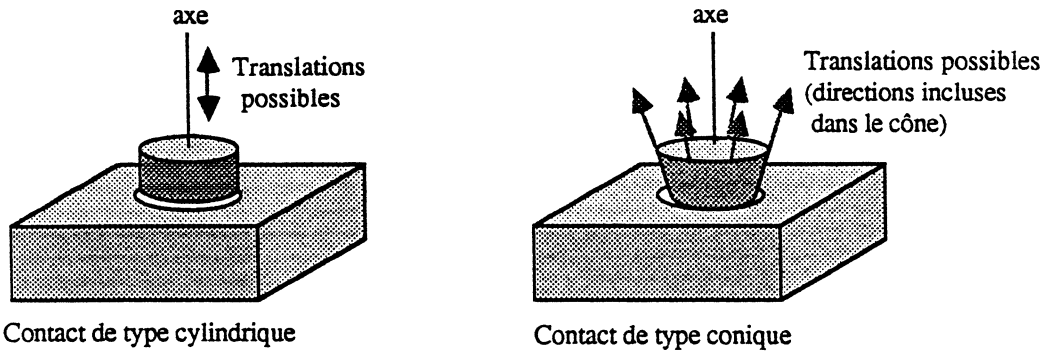


Figure 8.12: Mouvements potentiels associés à des contacts surfaciques mettant en jeu des surfaces de révolution.

8.5.3 Propriétés de la représentation:

Soit $S = \{C1, C2, \ldots Cn\}$ une situation mettant en jeu n contacts $C1, C2, \ldots Cn$. Les contraintes de mouvements associées à S sont représentées par un domaine D, défini comme l'intersection des domaines $D1, D2, \ldots Dn$ associés aux contacts $C1, C2, \ldots Cn$. Chaque domaine D_i $(i = 1 \ldots n)$ est construit à l'aide des fonctions (8.1) ou (8.2). On note D_i^* son intérieur, et δD_i sa frontière $(D_i = D_i^* \cup \delta D_i^*)$. On note de la même manière l'intérieur et la frontière de D.

Une caractéristique importante de la représentation utilisée est de permettre de distinguer les différentes catégories de mouvements qu'il est possible d'associer à S (cf. figure 8.11):

- tout couple (φ, ϑ) n'appartenant pas à D représente une direction de déplacement interdite;
- tout couple (φ, ϑ) inclus dans D^* définit un mouvement d'ordre 0 qui conduit à relâcher tous les contacts;

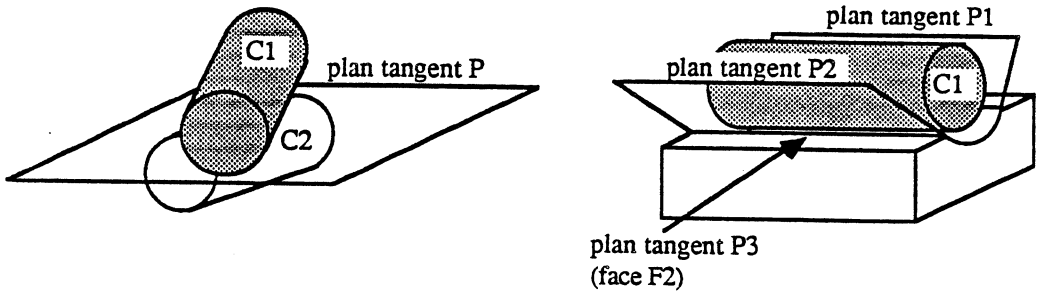


Figure 8.13: Mouvements potentiels associés à des contacts créés par des EG non planaires:
a- Contact ponctuel créé par deux surfaces cylindriques: $D(P)$.
b- Contact cylindrique partiel: $D(P1) \cap D(P2) \cap D(P3) \cup D(cylindre)$.

- tout couple (φ, ϑ) de δD représente un mouvement compliant qui conduit à glisser sur certaines surfaces de contact:
  si $(\varphi, \vartheta) \in \delta D_{i_1} \cap \delta D_{i_2} \cap \cdots \delta D_{i_j}$ et $(\varphi, \vartheta) \notin \delta D_k \, (k \neq i_1, i_2 \cdots i_j)$, alors (φ, ϑ) définit un mouvement d'ordre j qui conserve les contacts $C_{i_1}, C_{i_2} \ldots C_{i_j}$, et qui relâche les autres contacts.

Cette caractéristique est essentielle pour le processus de planification, car elle permet de raisonner à chaque étape sur les modifications de contacts engendrées par les mouvements sélectionnés.

Une autre propriété de ce mode de représentation est d'être "complet", c'est à dire d'être apte à représenter tous les mouvements qui sont possibles. Ainsi, n'importe quelle solution potentielle au problème de la planification d'un mouvement à partir de S, peut théoriquement être trouvée dans le domaine D associé. La difficulté réside alors dans l'impossibilité qu'il y a à analyser toutes les solutions représentées. Ce point est discuté dans le paragraphe suivant.

La propriété de complétude n'est cependant pas conservée dans le cas des contacts courbes qui mettent en jeu des plans tangents fictifs. En effet, ces plans conduisent à approximer les domaines représentés, en ne considérant aux limites que les mouvements engendrés tangentiellement aux surfaces de contact. Cette approximation est cependant raisonnable dans le sens où elle tente de caractériser

localement les mouvements possibles de l'objet mobile. Ceci est parfaitement cohérent avec le fait que les mouvements destinés à conserver le contact ont dans ce cas des directions tangentielles qui varient constamment avec la courbure de la surface. Il est cependant probable que l'extension de ce mode de représentation au cas de surfaces courbes plus générales, introduise des difficultés de calcul qui en limitent l'intérêt.

8.5.4 Traitement des rotations:

Les rotations sont intrinsèquement plus difficiles à manipuler que les translations. Elles pourraient théoriquement être représentées à l'aide d'une sphère unitaire, dans laquelle chaque vecteur définit la direction d'un axe de rotation. Mais les domaines obtenus dépendent presque toujours de la position spatiale des axes considérés. Considérons par exemple le contact de la figure 8.14:

- Tout point $M \in S^*$ ($S^* = S - \delta S$) ne permet de définir qu'un seul axe de rotation perpendiculaire au plan P. Cette solution unique est commune à l'ensemble des point de S^*. Elle possède la propriété d'engendrer des rotations qui conservent le contact.

- Chaque point $M \in \delta S$ permet de définir un couple d'axes respectivement perpendiculaire à P et tangent à δS. Il existe donc une solution différente pour chaque point de δS, du fait de la variation d'orientation des tangentes. Les rotations engendrées par ces solutions possèdent par contre des propriétés communes: celles exécutées autour de l'axe perpendiculaire à P conservent le contact; celles liées aux axes tangents transforment le contact en un contact ponctuel.

- Chaque point $M \in \Re^3 - S$ permet de définir un ensemble d'axes dont les caractéristiques varient en fonction de la position de M. Les rotations engendrées par les axes non perpendiculaires à P et non tangents à δS conduisent à rompre le contact.

Cette particularité des rotations rend inutilisable le modèle précédent, du fait de la multiplicité des représentations qu'il est possible d'associer à un contact. Elle suggère par contre une autre approche consistant à regrouper dans une même représentation, tous les axes qui ont un comportement "homogène" vis-à-vis des contacts:

Définition 8.5.3 *Un contact C est respectivement d'ordre 1, 2 ou 3 si C est de type ponctuel, linéique ou surfacique. Il sera dit d'ordre 0 si le contact entre les éléments de C n'est pas réalisé.*

Définition 8.5.4 *Un ensemble $\mathcal{A}$ d'axes de rotation sera dit "homogène de degré $\mid n - m \mid$" par rapport à un contact C d'ordre n, $n \in \{0,1,2,3\}$, si tout axe a*

de $\mathcal{A}$ définit des rotations qui permettent potentiellement de passer de l'ordre n à l'ordre m, $m \in \{0,1,2,3\}$.

Il est clair que les ensembles de degré 0 représentent les rotations qui conservent potentiellement le contact (si $n \neq 0$), et que ceux de degré i ($i \neq 0$) correspondent aux rotations qui en modifient le type. Dans le premier cas, les mouvements engendrés ont généralement pour objet de réorienter l'objet mobile en prenant appui sur la surface de contact. Ce type de mouvement est par exemple appliqué lors de la réalisation d'un montage de type baïonnette. Dans l'autre cas, les déplacements exécutés s'insèrent dans le cadre de stratégies destinées à réaliser progressivement des contacts surfaciques, en passant par des contacts ponctuels et linéiques.

Dans la pratique, les objets manipulés définissent des ensembles d'axes aisément représentables pour les cas courants. Par exemple, les ensembles de degré 0 associés aux contacts surfaciques sont constitués:

- De l'ensemble des droites perpendiculaires au plan de contact dans le cas d'un contact plan.
- De l'axe de révolution de la surface dans le cas d'un contact cylindrique ou conique.
- De l'ensemble des droites qui passent par le centre de la sphère dans le cas d'un contact sphérique.

Cette approche permet de traiter simplement un grand nombre de cas courants. Elle mériterait cependant d'être développée de manière plus complète dans le futur.

8.5.5 Exploitation de la représentation précédente:

Les opérations de saisie:

Les contacts mis en jeu par les opérations de saisie sont ceux qui lient les éléments de prise de l'objet avec les mors de l'outil de préhension. Les contraintes de mouvement imposées dépendent alors essentiellement de la géométrie de cet outil. Par exemple, une pince à deux mors parallèles limitera les mouvements main-objet suivant deux composantes:

- Une translation de direction parallèle au plan des mors.
- Une rotation d'axe perpendiculaire au plan des mors.

Cette propriété permet de connaître *a priori* les contraintes de mouvement induites, ce qui limite l'intérêt des techniques de calcul précédentes. Nous avons

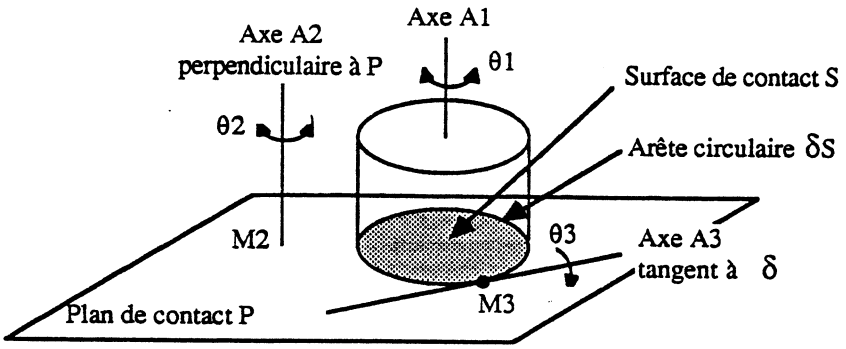


Figure 8.14: Exemples de rotations associées à un contact plan:
a- $Rot(A1, \vartheta 1) \Rightarrow$ contact non modifié.
b- $Rot(A2, \vartheta 2) \Rightarrow$ contact potentiellement conservé $\forall M2 \in P$.
c- $Rot(A3, \vartheta 3) \Rightarrow$ passage à un contact ponctuel $\forall M3 \in \delta S$.

donc choisi de représenter ces contraintes au moyen d'un formalisme mieux adapté au processus de construction des prises. Dans le cas d'une pince à deux mors parallèles, ces contraintes sont représentées par un plan parallèle aux plans des mors. Ce plan est appelé *"plan de préhension"*.

Si la pince considérée est équipée d'un troisième mors, ce plan est réduit à une droite du fait de la suppression d'un d.d.l en translation. Inversement, certaines prises exécutées sur des objets possédant une symétrie de révolution, conduisent à introduire une rotation supplémentaire autour de l'axe de symétrie. Par convention, nous considérerons alors que le plan de préhension peut tourner autour de cet axe.

L'exploitation de ce formalisme lors du choix d'une direction d'approche pour la pince est décrit dans le paragraphe 8.6.1.

Les mouvements fins:

La représentation analytique que nous avons développée, permet de caractériser l'ensemble des mouvements qui sont potentiellement exécutables par le robot, à

partir d'une situation de contact donnée. Elle n'est cependant pas directement exploitable par le système de planification du fait que les domaines représentés définissent chacun une infinité de solutions possibles.

Une technique très souvent utilisée en robotique pour résoudre ce problème, consiste à *discrétiser les ensembles de solutions étudiés.* Cette technique a déjà été présentée dans le contexte du raisonnement spatial. Dans le cas présent, elle conduit à découper le domaine analysé en calottes sphériques du type $\Delta\varphi X\Delta\vartheta$. Chaque calotte ΔS représente alors un ensemble de mouvements qui seront étudiés "globalement" par le système. Cette approche impose que toutes les directions m de ΔS conduisent théoriquement (compte non tenu des erreurs de commande) à une même situation de contact ou de non contact, après un déplacement issu du point courant P, et exécuté suivant m. Dans le cas où les objets sont polyédriques et où les mouvements sont des translations, cette contrainte peut être vérifiée au moyen de techniques d'analyse de "visibilité" communément employées en techniques graphiques [Laugier 79]. Cette méthode a déjà été utilisée par Buckley [Buckley 87], pour simuler le comportement d'un robot soumis à des incertitudes de commande. Elle permet de calculer récursivement les régions atteintes par le robot, après plusieurs mouvements exécutés suivant des directions comprises dans des cônes d'incertitudes.

Dans notre cas, les domaines de variation associés aux mouvements analysés sont ceux définis par le procédé de discrétisation. Le calcul est alors réalisé à l'aide du point référence R de l'objet mobile A, et des obstacles grossis $CO_A(B)$. La figure 8.15 illustre ce calcul à l'aide d'un exemple simple. L'ensemble $R1$ des points atteints par R pour $m \in \Delta S1$ est alors représenté par l'ensemble des points de $CO_A(B)$ qui sont "visibles" de R suivant les directions de $\Delta S1$. De même, l'ensemble $R2$ des points atteints par R après deux mouvements $m1$ et $m2$ de $\Delta S1$ et de $\Delta S2$, est défini par l'expression: $R2 = \cup_{M \in R1} Visible(M, \Delta S2)$. Le cas représenté en (b) correspond à un mauvais découpage de l'ensemble des solutions potentielles, car un mouvement de ΔS peut alors conduire à deux contacts différents $R1$ et $R2$. Dans ce cas, le domaine élémentaire ΔS doit être soit rejeté par le système, soit redécoupé en vue d'une nouvelle analyse.

Une autre technique permettant de réduire l'arbre de recherche, consiste à utiliser une *fonction choix*, dont le rôle est de sélectionner les directions à étudier. Ces directions sont alors choisies parmi celles contenues dans les domaines calculés. L'approche que nous avons implantée repose sur ce principe. Trois raisons esssentielles ont motivé notre choix pour cette méthode:

- Sa simplicité et son efficacité dans les cas courants, par opposition à la complexité algorithmique induite par l'autre approche.

- Son aptitude à traiter des surfaces courbes et des rotations, alors que les techniques de calculs précédentes ne le permettent pas (dans l'état actuel de nos connaissances).

- Le fait que la plupart des mouvements nécessaires à la réalisation des montages mécaniques sont exécutés suivant des *directions privilégiées* définies par les surfaces de contacts. En particulier, de nombreux mouvements fins sont appliqués perpendiculairement ou parallèlement aux contacts voisins.

Les directions d'étude sélectionnées par la fonction choix, correspondent alors à des points caractéristiques du modèle analytique: points d'intersections entre des courbes de type (8.1) ou (8.2), points répartis périodiquement sur les limites des domaines. La fonction choix que nous avons implantée est décrite dans le paragraphe 9.4.

Dans le futur, nous envisageons de développer une méthode qui combine les deux approches mentionnées ci-dessus. Il sera pour cela nécessaire de faire un important travail de recherche et de formalisation des outils mathématiques requis.

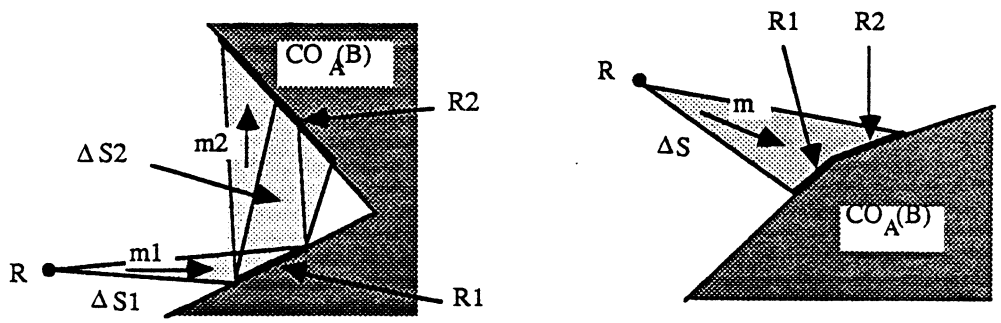


Figure 8.15: Régions susceptibles d'être atteintes par le robot après exécution d'un mouvement m inclus dans un ensemble ΔS:
a- Application récursive du procédé pour deux mouvements $m1$ et $m2$ consécutifs.
b- Cas de domaine ΔS pouvant conduire à deux contacts différents $R1$ et $R2$.

8.6 Construction de l'espace des solutions:

8.6.1 Les ensembles de prises potentielles:

Les ensembles de contacts retenus à l'issue du raisonnement morphologique sont organisés en *prises potentielles*. Chacune de ces prises est basée sur une configuration particulière de contacts main-objet. Elle représente toute une classe de prises potentiellement réalisables par le robot.

Afin de guider le mécanisme de sélection des prises, un classement partiel est établi sur la base d'une évaluation heuristique de la "qualité" de chacune. Ce procédé permet d'analyser en premier lieu les prises qui ont le plus de chances d'être retenues.

Cette relation d'ordre sur les prises potentielles est orientée vers l'application d'un backtrack simple en cas d'échec: rejeter la prise courante et prendre la suivante dans la liste. Il n'est par contre pas possible de mettre en œuvre plusieurs prises successives (technique dite de "regrasping"). Une solution consiste alors à chaîner entre elles toutes les prises qui sont potentiellement réalisables dans les différentes positions d'équilibre de l'objet. Cette structure complémentaire permet alors de construire des séquences de prises successives, en recherchant un "chemin" conduisant d'une prise associée à la position d'équilibre initiale, vers la position d'équilibre souhaitée. Une telle approche a été implantée dans le système HANDEY [Lozano-Perez et al. 87].

Représentation d'une prise potentielle:

Soit A l'objet à saisir, et P l'outil de préhension utilisé. Une prise potentielle $\Pi(A/P)$ est représentée de la manière suivante:

$$\Pi(A/P) = (\pi, P_c, P_m, h)$$

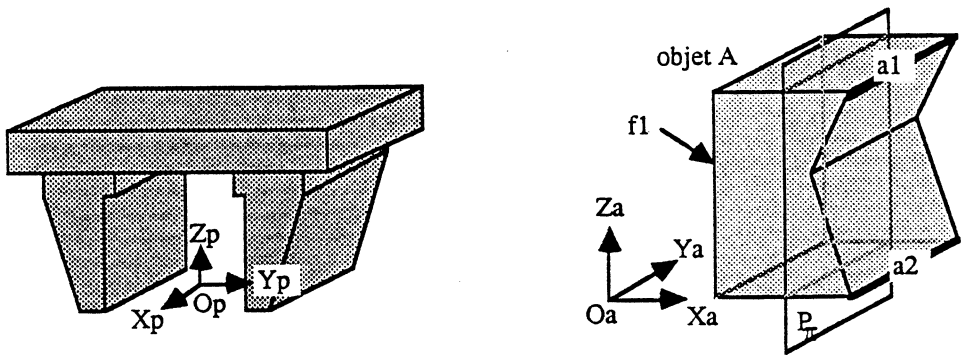
où

- $\pi = (C_1, C_2 \cdots C_m)$.
  dans lequel chaque C_i définit un contact qui met en jeu la face d'appui du i-ème mors de P, et un groupe d'EG de A. Les paramètres géométriques de ce contact ne sont alors spécifiés que de manière partielle, car la position exacte de P par rapport à A n'a pas encore été déterminée.

- P_c = Paramètres de préconfiguration de P.
  Ces paramètres définissent d'une part la configuration de P avant l'opération,et d'autre part l'action que doit exécuter la pince pour saisir A (ouverture/fermeture des mors).

- P_m = Contraintes d'orientation et de mouvements imposées à P.
  L'outil de préhension que nous utilisons étant constitué de deux mors parallèles, P_m peut être représenté par un plan particulier: le *plan de préhension* P_π (cf. paragraphe précédent). Soit $R_A = (O_A\, X_A\, Y_A\, Z_A)$ le repère associé à l'objet A, et $R_P = (O_P\, X_P\, Y_P\, Z_P)$ le repère associé à P comme indiqué dans la figure 8.16. Les contraintes de P_m s'expriment alors de la manière suivante:
  $O_P \in P_\pi$ et $X_P O_P Z_P \,//\, P_\pi$.
  P_π est alors défini comme étant le plan médiateur des deux surfaces planes qui supportent les contacts C_1 et C_2.

- h = Evaluation heuristique de la prise.
  Ce paramètre tente d'évaluer d'une part la *qualité* de la prise (stabilité, robustesse), et d'autre part son *degré d'accessibilité*. Il est utilisé pour guider le processus de sélection des prises.



Π (A/P) = ((((a1 a2) plan) ((f1) plan)
(fermer 1.5)
((0.75 0.0 0.0) (1.0 0.0 0.0)) h)

Figure 8.16: Exemple de prise potentielle.

Expression d'une solution dans $\Pi(A/P)$:

Une prise potentielle $\Pi(A/P)$ représente une classe de solutions potentielles pour l'opération de saisie de A par P. La recherche d'une solution particulière dans cet ensemble, conduit à rechercher une configuration de P qui vérifie les contraintes

imposées par π et par P_π, et qui soit réalisable dans les environnement d'exécution des opérations de saisie et de lâcher de A.

On note $P(X, \Delta)$ la configuration de P obtenue lorsque $O_p = X$ et $O_pZ_p = \Delta$, W l'ensemble des EG de A et des obstacles présents dans les environnements de saisie et de lâcher de A, et E l'ensemble des EG de A concernées par les contacts de π. La configuration $P(X, \Delta)$ est entièrement définie par le couple (X, Δ), car on sait par hypothèse que le plan $X_pO_pZ_p$ est parallèle à P_π.

Propriété 8.6.1 *Soit p un point de P_π et Δ un vecteur de P_π. Un couple (p, Δ) sera considéré comme une solution particulière de $\Pi(A/P)$, ssi:*

- $P(p, \Delta) \cap e \neq \emptyset, \quad \forall e \in E$
- $P(x, \Delta) \cap \{W - E\} = \emptyset, \quad \forall x \in Demi-droite(p, \Delta).$

La première expression vérifie que tous les contacts de π sont réalisés simultanément dans la configuration $P(p, \Delta)$ de la pince (cf. figure 8.17). On parlera alors de configuration **valide.**

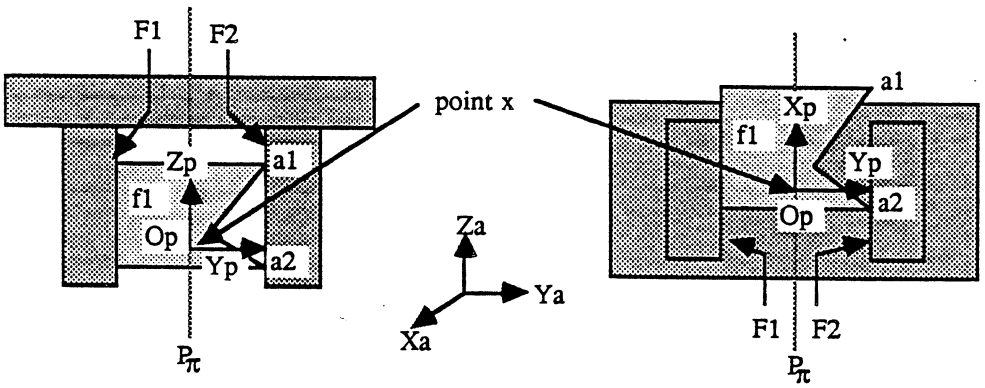


Figure 8.17: Exemples de configurations de P dans $\Pi(A/P)$:
a- Configuration $P(x, Z_A)$ valide.
b- Configuration $P(x, -X_A)$ non valide, car $F_2 \cap a_1 = \emptyset$.

L'autre expression vérifie que les parties à saisir sont accessibles dans la direction Δ. On dira alors que la configuration analysée est **sûre.** On fait dans ce cas

l'hypothèse que les mouvements d'approche et de retrait de l'outil sont exécutés suivant Δ. Dans le cas contraire, il est toujours possible de définir deux directions différentes Δ_1 et Δ_2.

Recherche d'une solution dans $\Pi(A/P)$:

La recherche d'une solution dans $\Pi(A/P)$ est avant tout un problème de planification de trajectoires, car il s'agit de déterminer les paramètres dynamiques de l'opération de saisie. L'approche que nous utilisons pour résoudre ce problème, opère en trois phases:

1. Choisir heuristiquement une direction Δ de P_π.
2. Construire l'ensemble S_π des points p de P_π, tel que le couple (p, Δ) vérifie la propriété 8.6.1. On utilise pour cela les fonctions développées pour le raisonnement spatial. Si $S_\pi = \emptyset$, retourner en (1).
3. Choisir un point p dans S_π, tel que la configuration $P(p, \Delta)$ minimise les risques de glissement.

Cette méthode de recherche d'une prise réalisable est décrite plus en détail dans le paragraphe 9.3. Le choix a priori d'une direction Δ est motivé par le fait que les techniques de raisonnement spatial dont nous disposons, ne permettent pas de traiter simultanément les rotations et les translations sans discrétiser l'espace des configurations. L'accroissement de complexité qui en découle s'avère alors trop important en regard des résultats attendus.

L'approche utilisée dans HANDEY [Mazer 87] pour résoudre ce problème, est basée sur une philosophie complètement différente. L'objectif visé est alors de trouver *une trajectoire*, qui permette de faire passer la pince d'une position initiale prédéfinie, à une position de saisie caractérisée par un couple de faces parallèles de l'objet. Le fait que le système ne tente pas de construire un ensemble de prises potentielles, permet de ne faire aucune hypothèse a priori sur la solution recherchée (orientation de la pince et type de trajectoire). La trajectoire de l'outil est alors construite au moyen d'une fonction de planification de type "potentiel" (cf. chapitre 5). Aucun critère de stabilité ne permet de guider le choix de la solution dans ce cas.

8.6.2 Le graphe d'états associé aux mouvements fins:

Les ensembles de contacts et de mouvements potentiels exhibés par le raisonnement morphologique sont organisés sous la forme d'un graphe. Ce graphe est appelé *graphe d'états associé aux mouvements fins.* Il représente les séquences de

mouvements et d'états du robot, qui ont été retenues comme des *solutions potentielles* pour le problème posé. Nous verrons dans le paragraphe 9.4 que ce graphe n'est pas figé, et qu'il peut être localement affiné en cas d'échec. Un tel échec est détecté lors de la recherche d'une solution particulière. Il indique qu'aucune stratégie exécutable n'est contenue dans le graphe d'états courant.

Représentation du graphe d'états:

Soient p une position de $\xi(\Re^3 \times O^3)$ et P un ensemble de positions. Dans le paragraphe 6.5, nous avons défini un état E_p du robot comme un ensemble de couples (p^*, f^*) de données sensorielles. Par extension, nous noterons E_P l'ensemble des états associés à P. Afin de ne pas compliquer inutilement l'exposé, nous parlerons dans tous les cas de "l'état E_P".

Soient A et B les deux objets à assembler. Le graphe d'états $G(A/B)$ associé aux mouvements fins de montage de A sur B, est modélisé par un graphe orienté $(\mathcal{S}, \mathcal{A})$. Les nœuds de $\mathcal{S}$ représentent des états E_P qui vérifient les propriétés énoncées dans le paragraphe 6.5. Les arcs de $\mathcal{A}$ symbolisent les mouvements qui permettent de passer d'un état à l'autre. Une *stratégie de mouvements fins* est alors représentée par un sous-graphe particulier de $G(A/B)$. La figure 8.18 illustre cette définition intuitive qui sera précisée plus loin.

Un nœud s_i de $\mathcal{S}$ est caractérisé par les informations suivantes:

$$s_i = (P, C, D, I, Q)$$

où P est un ensemble de positions de $\xi(\Re^3 \times o^3)$, C est un ensemble de contacts, D est un ensemble de mouvements potentiels, I est une information qualitative portant sur la situation physique représentée (insertion cylindrique par exemple),et Q est une évaluation heuristique de la "qualité" de cette situation (robustesse vis-à-vis des incertitudes et des aléas mécaniques). Les paramètres P et C définissent symboliquement l'état E_P. Le paramètre D indique les mouvements qui peuvent être théoriquement appliqués à partir de l'état E_P; cette information est utilisée pour affiner localement le graphe en cas de nécessité. Les paramètres I et Q ont pour objet de guider la recherche d'une solution.

Un arc a_{ij} de $\mathcal{A}$ est caractérisé par les informations suivantes:

$$a_{ij} = (T, C, A, Q)$$

où T est une transformation géométrique, C et A sont des ensembles de faces sur lesquelles l'objet mobile doit respectivement glisser et s'arrêter, et Q est une pondération heuristique traduisant la "qualité" du mouvement (risques de collisions ou de blocages dus à de petites variations de l'environnement, sensibilité aux

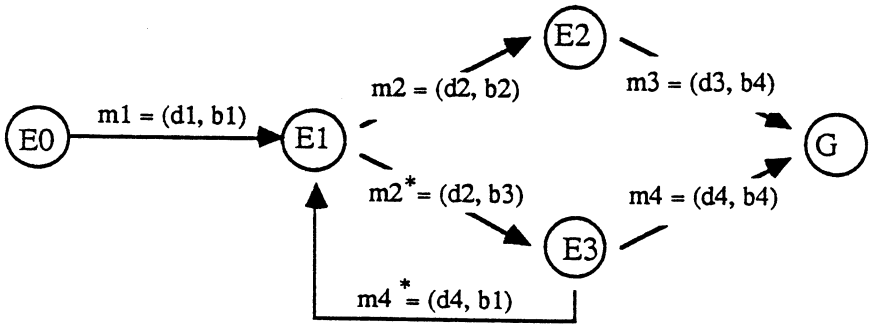


Figure 8.18: Graphe représentant une stratégie de mouvements fins:
Le mouvement m_1 permet de passer de l'état initial E_0 à l'état E_1. Le mouvement défini par les paramètres de déplacement d_2, permet ensuite d'atteindre l'état E_2 ou l'état E_3; Ces états sont identifiés à l'exécution par les conditions d'arrêt b_2 et b_3. Si le robot est dans l'état E_2, il exécute le mouvement m_3. Dans l'autre cas, il exécute le mouvement défini par d_4; ce mouvement peut se terminer en G ou ramener le robot dans l'état E_1.

effets de la pesanteur ...). Les paramètres T, C et A définissent le mouvement qui permet de passer de l'état E_{P_i} à l'état E_{P_j}. Le paramètre Q est utilisé pour guider la recherche d'une solution.

Le procédé de construction et d'interprétation de ce graphe est décrit dans le paragraphe 9.4.

Expression d'une solution dans $G(A/B)$:

On note $EI = \{EI_1, EI_2 \cdots EI_m\}$ l'ensemble des nœuds de $G(A/B)$ qui possèdent un ensemble vide de contacts, et EF le nœud symbolisant l'état où A et B sont assemblés. Les nœuds de EI sont considérés comme des points de départ possibles pour la construction d'une stratégie de mouvement fins, du fait qu'ils représentent des états directement réalisables par le robot. Le nœud EF définit l'objectif à atteindre.

Intuitivement, nous dirons que tout chemin de $G(A/B)$ permettant de passer

d'un nœud de EI au nœud EF, représente une stratégie de mouvements fins apte à réaliser le montage de A sur B. La présence dans $G(A/B)$ de mouvements susceptibles d'amener le robot dans des états différents par suite des erreurs de commande, conduit à considérer des sous-graphes plutôt que de simples chemins (cf. figure 8.18). Il est en effet nécessaire d'inclure dans chaque solution, tous les états pouvant être atteints après exécution des mouvements retenus. Afin de formaliser cette notion intuitive, nous utilisons les définitions suivantes:

Définition 8.6.1 *Soient d et b les paramètres de déplacement et les conditions d'arrêt associés à un mouvement m. Deux mouvements $m_1 = (d_1, b_1)$ et $m_2 = (d_2, b_2)$ issus d'un même état E_P du robot sont dits semblables, ssi $d_1 = d_2$. Ces deux mouvements peuvent alors être considérés comme un seul mouvement défini par le couple: $(d_1, b_1 \vee b_2)$.*

Par extension de cette définition, nous dirons que deux arcs a_{ij} et a_{ik} sont semblables lorsque les mouvements représentés le sont.

Propriété 8.6.2 *Soient SG un sous-graphe représentant une stratégie de mouvements fins, s_i un nœud quelconque de SG, et $Succ(s_i)$ l'ensemble des successeurs de s_i dans $G(A/B)$. Un nœud s_j de $Succ(s_i)$ appartient à SG, ssi il n'existe pas d'autre nœud s_k de $Succ(s_i)$ tel que:*

- $s_k \in SG$.
- a_{ik} *"non semblable"* a_{ij}.

Justification: Soient s_j et s_k deux successeurs de s_i, tels que les arcs a_{ij} et a_{ik} correspondent à des mouvements non semblables. Ces deux nœuds ne peuvent pas appartenir à un même SG, car il y aurait alors une ambiguïté sur le mouvement à appliquer à partir de l'état E_i.

Définition 8.6.2 *Un sous-graphe SG de $G(A/B)$ représente une stratégie de mouvements fins, si il vérifie les propriétés suivantes:*

- *Il existe un et un seul nœud x de SG, tel que $x \in EI$.*
- *Le nœud EF appartient à SG.*
- *Tout nœud de SG vérifie la propriété 8.6.2.*

La figure 8.19 illustre cette définition. Dans l'exemple représenté, les arcs a_{12} et a_{13} définissent un seul mouvement m_2 caractérisé par le couple $(d_2, b_2 \vee b_3)$; a_{12} et a_{13} sont de ce fait inclus dans le même sous-graphe. Il en est de même pour les arcs a_{3g} et a_{31}. Inversement, les arcs a_{01} et a_{04} représentent des mouvements différents; ils sont donc répartis dans deux sous-graphes différents.

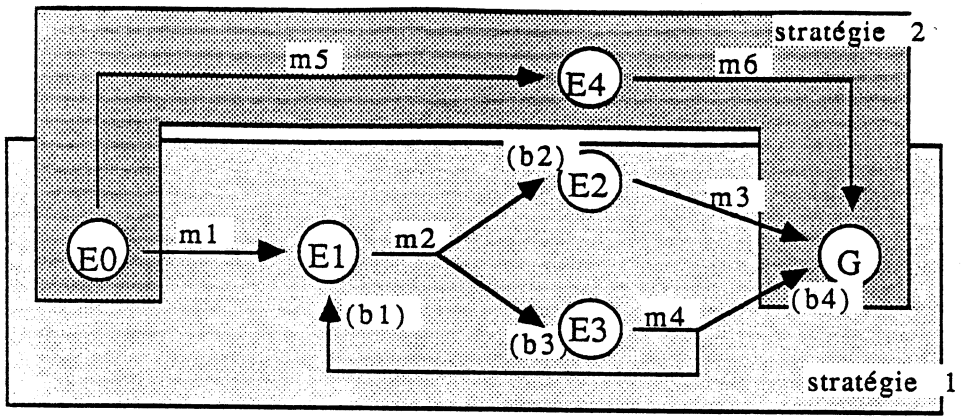


Figure 8.19: Stratégies incluses dans un graphe d'états:
Les arcs a_{12} et a_{13} représentent un même mouvement m_2, comportant deux objectifs possibles caractérisés par les conditions b_2 et b_3; a_{12} et a_{13} appartiennent au sous-graphe associé à la stratégie 1. Il en est de même pour les arcs a_{3g} et a_{31}. Inversement, les arcs a_{01} et a_{04} représentent des mouvements différents; ils appartiennent de ce fait à deux sous-graphes différents.

Recherche d'une solution dans $G(A/B)$:

Rechercher une solution dans $G(A/B)$ consiste à rechercher un sous-graphe SG qui vérifie les conditions énoncées dans la définition 8.6.2.

Si la taille de $G(A/B)$ n'est pas trop grande, une technique possible consiste à construire tous les sous-graphes SG possibles, puis à choisir celui qui comporte la meilleure évaluation heuristique. Dans l'autre cas, le parcours de $G(A/B)$ est guidé par une fonction qui permet de choisir à chaque étape les nœuds qui semblent être les mieux adaptés. Cette fonction exploite les informations heuristiques associées aux nœuds et aux arcs du graphe. L'algorithme de recherche d'une solution dans $G(A/B)$ est alors le suivant:

1. *CHOISIR* un nœud x dans EI, et initialiser SG avec le couple $(\emptyset, x)$.
2. *CHOISIR* un nœud y dans $Succ(x)$. Si $EF \in Succ(x)$, prendre $y = EF$.
   Ajouter le couple $(arc[x,y], y)$ ou l'arc $arc[x,y]$ à SG.
   Si y était déjà présent dans SG ou si $y = EF$, alors *retour*.
   Sinon, appliquer récursivement l'étape (2) pour le nœud y.

3. Inclure dans SG tous les couples $(arc[x,z],z)$ ou les arcs $arc[x,z]$ qui vérifient la propriété: $z \in Succ(x) \ \wedge \ Semblable(arc[x,y], arc[x,z])$.
   Appliquer récursivement l'étape (2) pour chaque nœud z ajouté à SG, puis retourner en (3).
   S'il n'y a plus de nœuds à traiter, retourner SG si $EF \in SG$ et *échec* sinon.

L'implantation que nous avons réalisée introduit quelques limitations qui sont discutées dans le paragraphe 9.4.

Chapitre 9

Implantation et intégration: le système SHARP

9.1 Architecture du système:

9.1.1 Structure fonctionnelle du système:

SHARP est un système de programmation automatique orienté vers la programmation des robots à postes fixes. Son architecture informatique a été conçue de manière à répondre aux impératifs de la robotique manufacturière classique: les tâches programmées sont répétitives, et elles sont exécutées dans un environnement connu à priori; les programmes construits doivent respecter des contraintes d'optimalité et de fiabilité en regard de certaines variations de l'univers de la tâche. Ces caractéristiques nous ont conduits à choisir une structure décisionnelle calquée sur le processus de planification décrit dans le paragraphe 5.2. Nous nous sommes cependant limités aux fonctions de planification de niveau manipulation. Ceci implique que le modèle de la tâche fourni au système comporte les données suivantes:

- Une description complète de l'environnement (outillage utilisé, positions initiales des pièces mécaniques).
- Une liste ordonnée des opérations d'assemblage à exécuter.

- Une description des relations d'assemblage et des jeux de montage associés.
- Un modèle des objets et du robot incluant une représentation des incertitudes de position et des erreurs de commande et de perception.

Le rôle du système est alors de produire automatiquement un programme de manipulation exécutable par le robot. Trois étapes sont appliquées pour résoudre ce problème (cf. figure 9.1):

1. Planification des actions nécessaires à l'exécution des différentes opérations d'assemblage. Seules les interdépendances locales sont traitées à ce niveau, cf. paragraphe 5.2.
2. Amendement itératif du plan de manière à le rendre compatible avec les contraintes d'incertitude imposées par la tâche. Cette étape a pour objet de traiter les interdépendances qui existent entre différentes parties du plan, cf. paragraphe 5.2.
3. Adaptation du plan à l'environnement matériel. Cette étape permet d'intégrer les aléas de l'univers physique non représentés dans le modèle. Elle permet aussi d'optimiser les programmes produits, en supprimant des opérations de vérification ou de correction inutiles.

Les motivations qui sont derrière le découpage des étapes (1) et (2) ont été données dans le paragraphe 5.2. La raison d'être de la dernière étape provient de ce que les connaissances du système sont imcomplètes, notamment en ce qui concerne la dynamique et les incertitudes. Elles conduisent de ce fait à ignorer certains risques d'échecs tels que des glissements, des blocages mécaniques ou des retournements de pièces. Elles conduisent aussi à raisonner à partir des hypothèses les plus défavorables, ce qui se traduit par des solutions surcontraintes (et donc mal adaptées à la tâche). Cette étape *d'adaptation/optimisation* du plan de manipulation est exécutée en connexion avec le robot réel. Elle correspond à une phase "d'expérimentation en-ligne" similaire à celle présentée dans le chapitre 4. Il reste encore un travail important (probablement des années de recherche), avant de pouvoir automatiser le procédé "d'apprentissage" utilisé.

La distinction qui est faite entre les niveaux conceptuels de planification, de vérification/correction et d'adaptation/optimisation, constitue une caractéristique originale de SHARP. Elle offre l'avantage d'associer à chaque niveau des raisonnements appropriés concernant respectivement la manipulation, les incertitudes liées à l'environnement de la tâche, et les caractéristiques réelles du contexte d'exécution. Cette approche permet théoriquement de produire des programmes dont le degré de spécificité peut varier: au plus haut niveau, le programme résout

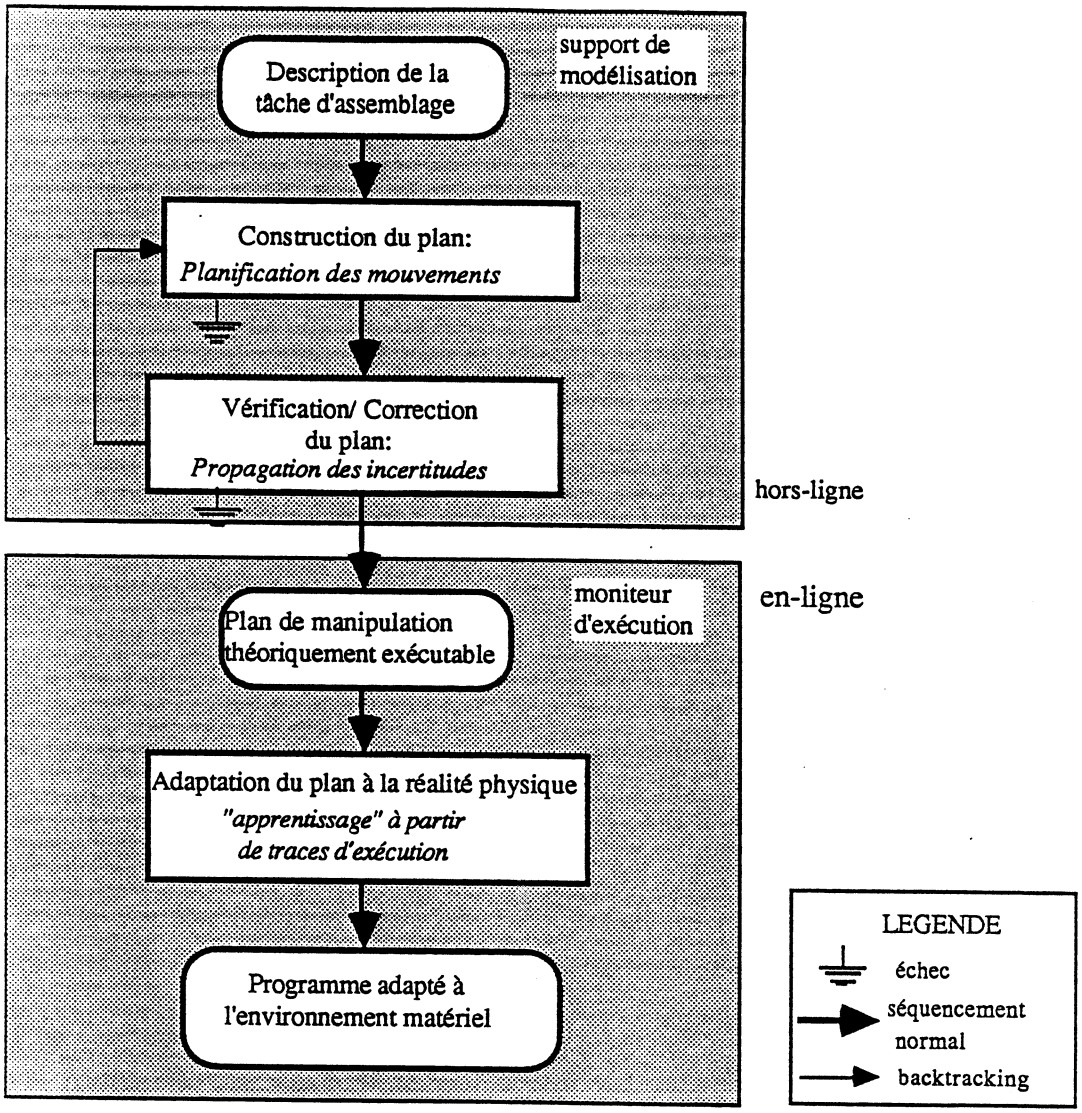


Figure 9.1: Structure fonctionnelle du système SHARP.

le problème de manipulation posé tout en restant relativement éloigné du contexte d'exécution; au dernier niveau, il représente un procédé particulier dont la fiabilité et les performances dépendent directement de l'environnement matériel utilisé. On peut ainsi envisager de construire des programmes nominaux, qui pourront

ensuite être adaptés à des contextes d'exécution différents: robots, mécanismes d'alimentation en pièces, disponibilité des capteurs...

Dans la suite, nous désignerons par SHARP la partie "hors-ligne" du système, c'est à dire la partie qui réalise les fonctions de planification et de vérification/ correction.

9.1.2 Description de la tâche:

Principe de la description:

La tâche d'assemblage à réaliser est décrite au système sous la forme d'une séquence ordonnée *d'opérations d'assemblage*. Chaque opération définit un objectif intermédiaire à atteindre, ainsi qu'un ensemble de contraintes à respecter lors de la réalisation de cet objectif. Ces objectifs correspondent tous à des relations d'assemblages, exécutables sans application d'opérations technologiques (vissage, soudage $\cdots$). Ils ne s'adressent qu'à des couples d'objets rigides.

Le mode de description utilisé repose sur un formalisme géométrique dérivé de celui développé dans le langage LM-GEO [Mazer 83]. L'exclusion actuelle des opérations technologiques conduit à limiter les contraintes à l'expression des jeux de montage autorisés. La forme d'une opération d'assemblage est alors la suivante:

PLACER objet-A SUR objet-B TEL-QUE <situation> AVEC <contraintes>

dans laquelle *<situation>* est une expression géométrique qui définit la relation d'assemblage à réaliser, et *<contraintes>* représente les contraintes d'incertitude à respecter lors de l'exécution du montage.

Expression des situations géométriques:

Une situation géométrique est décrite par une conjonction de *relations géométriques* élémentaires, mettant chacune en jeu un couple d'entités du modèle des deux objets:

Situation S: $(R_1 \wedge R_2 \cdots \wedge R_m)$
Relation R: $(<op>\ e_a\ e_b)$
avec:
e_a, e_b: points, droites ou plans.
<op>: CONTRE, COPLANAIRE, ALIGNE, PARALLELE, SUR.

Les entités e_a et e_b représentent les éléments de la géométrie des objets qui interviennent directement ou indirectement dans les relations d'assemblage: faces planes, arêtes, axes de parties cylindriques ou coniques, sommets, centres de parties sphériques.

Les relations géométriques décrivent la manière dont sont agencées les entités précédentes dans le montage. Elles traduisent essentiellement des propriétés géométriques simples telles que le parallélisme, l'alignement ou la coplanarité. Elles sont aussi assorties de conditions portant sur la répartition de la matière au voisinage des parties d'objets concernées. Par exemple, la relation *"CONTRE(0) F_1 F_2"* indique que les faces F_1 et F_2 sont coplanaires, et que les normales extérieures à ces faces sont opposées. Dans le cas de la relation *COPLANAIRE*, les deux faces sont positionnées de sorte que la matière reste localisée du même côté par rapport au plan des faces.

La plupart du temps, les relations géométriques utilisées représentent des contacts de l'assemblage: la relation *CONTRE* exprime généralement un contact plan entre deux faces; la relation *ALIGNE* définit très souvent l'alignement des axes associés à une cavité cylindrique ou conique, et à une partie pleine de même nature. Les autres relations ont alors pour objet de contraindre les positions et les orientations relatives des deux objets, dans les directions non soumises à des contacts.

Interprétation des situations géométriques:

Soit S une situation géométrique impliquant plusieurs objets O_1, $O_2 \cdots O_n$. Le problème à résoudre consiste à déduire les positions relatives des objets O_1, $O_2 \cdots O_n$, à partir de la description symbolique de S.

Plusieurs méthodes ont été proposées dans la littérature pour résoudre ce problème [Brady et al. 82]. Quelle que soit la technique utilisée, les méthodes proposées opèrent toutes suivant l'algorithme général suivant:

1. Définition d'un système de référence pour les objets et pour les entités géométriques qui leur sont associées.
2. Transformation des relations symboliques en équations portant sur les paramètres de configuration (positions et orientations) des objets.
3. Combinaison des équations liées à chaque objet.
4. Résolution du système d'équations afin de trouver les paramètres de configuration de chaque objet.

Le programme d'inférence géométrique que nous nous proposons d'utiliser dans SHARP, est celui qui a été développé dans LM-GEO [Mazer 82]. Ce programme a été conçu de manière à rechercher numériquement toutes les solutions possibles, même lorsque le problème est sous-contraint. Cette propriété permet de travailler

avec des spécifications imcomplètes, lorsque les objets manipulés comportent des symétries de révolution. Par exemple, on ne précisera pas l'orientation axiale d'un goujon cylindrique dans la description de la relation d'insertion.

Expression des contraintes:

Les contraintes de montage sont exprimées comme des tolérances qui portent sur les positions nominales des objets. Elles sont donc associées aux relations géométriques qui spécifient ces positions:

Situation S: $(R_1 \wedge R_2 \cdots \wedge R_m)$
Contraintes sur S: $(C_1 \wedge C_2 \cdots \wedge C_m)$

Dans cette expression, chaque contrainte C_i définit l'erreur maximum qui est tolérée sur la réalisation de la relation R_i. Elle est spécifiée par un couple (x_i, α_i), dans lequel x_i est l'erreur maximum en translation, et α_i est l'erreur maximum en rotation. Les termes d'erreurs concernés par x_i et par α_i sont ceux liés aux d.d.l contraints par R_i. Par exemple, une relation de coplanarité entre deux faces planes F_a et F_b conduira à contraindre la direction de translation orthogonale à F_b (B est alors l'objet fixe); la tolérance représentée par x_i portera uniquement sur cette direction.

L'interprétation par le système des expressions C_i, conduit à construire des contraintes du type:

$$Proj(I(R_a/R_b), (E, V, B)) < I_0$$

où $Proj$ est l'opérateur de projection d'incertitude défini dans le chapitre 6, $I(R_a/R_b)$ est l'incertitude de position de l'objet A par rapport à l'objet B, (E, V, B) et I_0 sont des sous-ensembles de $\Re^3 \times S(1) \times [-\pi \ +\pi]$, et le symbole "$<$" représente la relation d'ordre définie dans le paragraphe 9.5.

Les ensembles E, V et B sont définis par le type de la relation R_i considérée. Par exemple, les ensembles associés à la relation de coplanarité précédente sont les suivants: $E = droite \perp F_b$, $V = S(1)$ et $B = [-\pi \ +\pi]$.
Les ensembles V et B associés à une relation de colinéarité sont les mêmes, mais l'ensemble E correspondant est représenté par un plan perpendiculaire à la droite support D_b.

Les volumes d'incertitude représentés par I_0 sont aussi exprimés comme des sous-ensembles de $E \times V \times B$. Par exemple, la relation de coplanarité contrainte par le couple (x_i, α_i), engendrera le volume d'incertitude suivant:

$$I_0 = Segment[\perp F_b, 2x_i] \times S(1) \times [-\pi \ +\pi]$$

Remarque: Lorsque la relation R_i représente un contact sans jeux (par exemple: un objet posé sur la table), la contrainte C_i est exprimée par le couple $(0,0)$. Inversement, le couple (x_i, α_i) est omis lorsque la relation n'est soumise à aucune contrainte particulière (par exemple: positionnement du robot dans l'espace libre). L'erreur commise à l'exécution est alors celle introduite par la tâche (opérations antérieures, commande $\cdots$).

Prenons par exemple une insertion cylindrique devant être exécutée sans corrections d'orientation, et nécessitant une précision d'exécution inférieure à 0.1 mm. Cette opération pourra être spécifiée comme suit:

PLACER piston SUR cylindre TEL-QUE <montage> AVEC <jeux>
avec:
<montage>: (ALIGNE axe-piston axe-cylindre) $\wedge$
(CONTRE(0) fond-piston fond-cylindre)
<jeux>: $(0.1,0) \wedge (0,0)$

La plupart des outils nécessaires à l'implantation de ce langage de description ont été développés dans le cadre du projet SHARP (interprétation des relations géométriques, représentation et calculs d'incertitudes). Ce mode de description n'a cependant pas encore été utilisé, à cause de l'absence actuelle de connexion avec les fonctions de vérification/correction.

9.1.3 Composants informatiques du système:

SHARP est construit autour d'un système de modélisation géométrique, d'un module de planification, et d'un module de vérification/correction de plans. Ces trois composants coopèrent à travers une structure de contrôle qui gère les appels et les opérations de backtrack. Nous verrons dans le paragraphe qui suit, que cette structure de contrôle est actuellement réduite à sa plus simple expression.

Le système de modélisation est dérivé du système SMGR décrit dans [Troccaz 86]. Il a été initialement implanté en MACLISP sur un calculateur CII-HB 70. La version actuelle est opérationnelle sur machine SUN. Elle inclut les structures et les fonctions qui ont été décrites dans les chapitres 3 et 6. Elles inclut également un mécanisme de "démons", permettant de mettre à jour automatiquement certaines propriétés géométriques des objets manipulés [Theveneau 88].

Le module de planification traite respectivement des problèmes de saisie d'objets, de transfert de pièces et de montages mécaniques. Il opère sur des modèles qui combinent des informations de nature géométrique et physique. Ces modèles sont

ceux mentionnés précédemment. Une caractéristique importante de l'approche utilisée, est d'être basée sur un procédé génératif (par opposition aux techniques de planification conduisant à combiner et à compléter des "squelettes de programmes" [Lozano-Perez, Brooks 85]). Ce procédé consiste à engendrer des solutions potentielles sur la base d'une analyse des propriétés locales des parties d'objets concernées; la validité de ces solutions est ensuite évaluée au moyen d'une analyse d'accessibilité globale. Cette approche a été développée dans les chapitres 7 et 8. L'organisation des fonctions de planification est décrite par la figure 9.2. Les détails d'implantation sont donnés dans les paragraphes 9.2, 9.3 et 9.4.

Le module de vérification/correction a pour objet d'évaluer la compatibilité du plan courant avec les jeux de montage autorisés. Il s'appuie pour cela sur un modèle formel de propagation de contraintes géométriques, dont le principe est décrit dans le paragraphe 9.5.

9.1.4 Structure de contrôle:

Organisation générale du contrôle:

L'architecture logicielle de SHARP est organisée autour des fonctions de planification et de vérification/correction. Elle repose sur une structure de contrôle à deux niveaux, permettant de traiter les interdépendances qui se manifestent au niveau des actions, et celles qui affectent la structure même du plan (cf. paragraphe 5.2). Le premier niveau est intégré dans le module de planification; il gère essentiellement les contraintes d'accessibilité qui s'opposent à l'exécution des séquences d'actions du type "saisie - transport - montage". L'autre niveau a pour objet de traiter les interdépendances qui se manifestent au travers des contraintes d'incertitude; il conduit à amender itérativement le plan construit, chaque fois qu'un échec est détecté.

Seules les interdépendances dues aux contraintes d'incertitude sont traitées rigoureusement par une méthode de propagation assez proche des techniques de régression (cf. paragraphe 9.5). Les autres interdépendances sont par nature plus difficiles à formaliser et à quantifier. Elles sont prises en compte dans le processus de planification par une structure de contrôle non linéaire, conduisant à résoudre en priorité les problèmes jugés les plus contraints, et à appliquer des techniques de backtracking en cas d'échec.

Traitement des interdépendances produites par la géométrie:

Les interdépendances dues aux contraintes géométriques sont traitées dans le module de planification. Une manière idéale de procéder consisterait à ne rechercher les solutions à chacun des sous-problèmes (saisie, transport et montage),

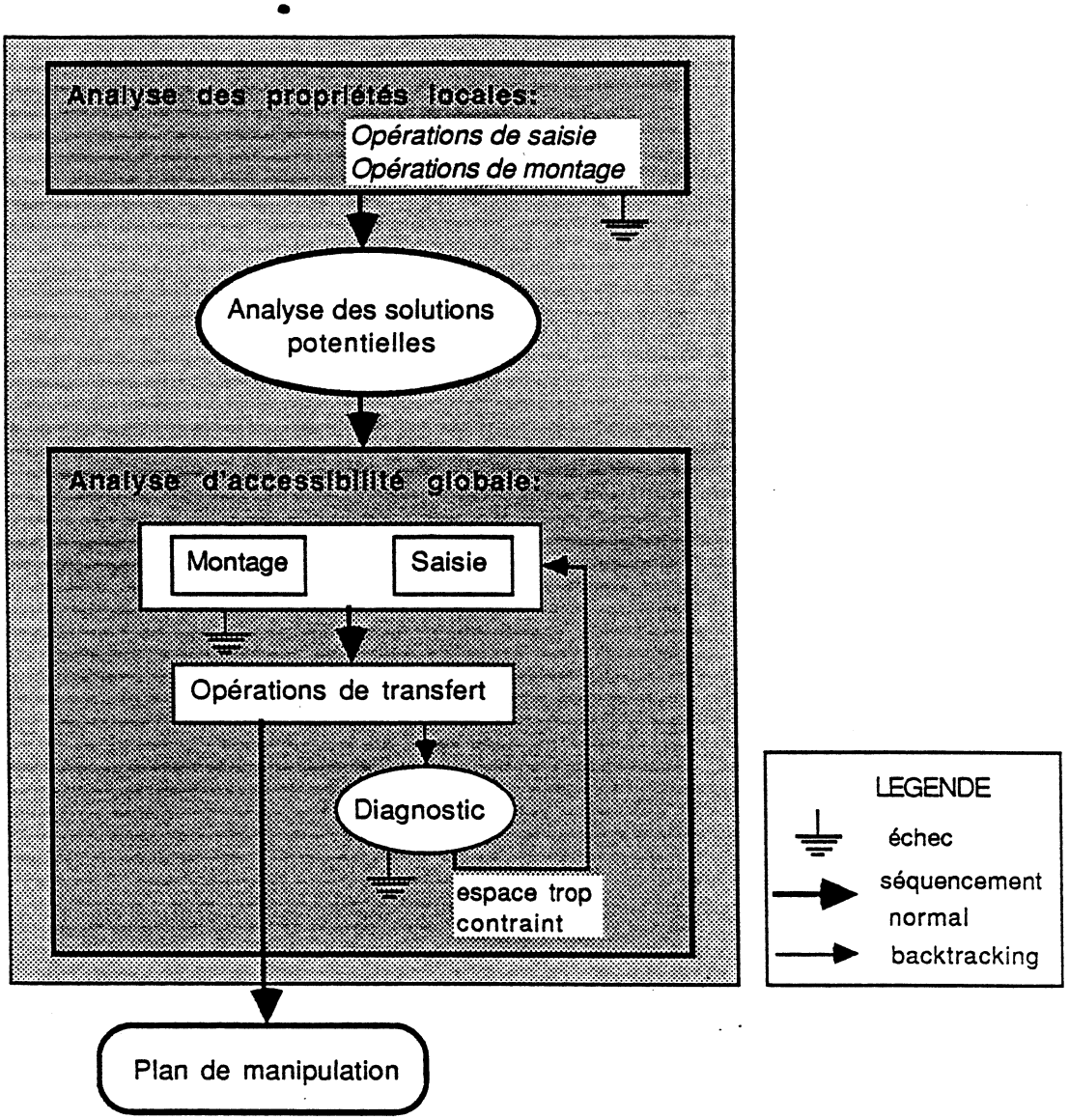


Figure 9.2: Fonctionnement du module de planification.

qu'après avoir obtenu une parfaite connaissance des contraintes imposées par les autres sous-problèmes. Par exemple, on aimerait pouvoir tenir compte d'une stratégie d'assemblage particulière pour guider le choix des paramètres de saisie; inversement, la détermination préalable d'une prise robuste compatible avec le contexte de saisie, imposerait des contraintes qui réduiraient l'ensemble des solu-

tions possibles pour le montage.

Dans la pratique, il n'est pas possible de connaître à priori l'ensemble des contraintes qui sont liées à l'exécution de la tâche. Une manière de procéder consiste alors à traiter indépendemment chaque sous-problème, puis à effectuer une vérification à postériori de la compatibilité des solutions proposées. En cas d'échec, des techniques de backtracking sont appliquées afin de rechercher et de modifier les choix qui sont à l'origine de cet échec.

La structure de contrôle qui découle de cette approche est décrite par la figure 9.2. Elle a donné lieu à une implantation très limitée, conduisant à rechercher séquentiellement des solutions pour les problèmes de montage, de saisie et de transfert. En cas d'échec, ces solutions sont rejetées afin d'être remplacées par d'autres. Aucune analyse de l'échec n'est alors réalisée, ce qui conduit à appliquer systématiquement la même procédure de retour arrière. Bien que l'espace des solutions admissibles soit très grand, et que les heuristiques utilisées permettent très souvent de trouver l'une d'elles, il serait souhaitable d'étudier ce problème de manière plus appronfondie.

Traitement des interdépendances produites par les incertitudes:

Les interdépendances dues aux incertitudes sont traitées au niveau de la structure de contrôle qui gère les interactions entre les modules de planification et de vérification/correction [Laugier et al. 87].

Le principe consiste à engendrer un plan par juxtaposition des solutions partielles produites par les fonctions de planification du système. La validité globale de ce plan vis-à-vis des contraintes d'incertitude imposées par la tâche, est ensuite évaluée par le module de vérification/correction. Chaque contrainte non satisfaite conduit alors à amender le plan. Une description fonctionnelle de ce mécanisme est donnée dans la figure 9.3. Le procédé appliqué repose sur un modèle formel de propagation de contraintes, permettant de calculer les valeurs d'incertitudes en n'importe quel point du plan. Le processus itératif de vérification/correction appliqué, opère alors suivant deux modes:

- Un mode de propagation "descendant", qui permet de vérifier que les incertitudes obtenues en n'importe quel point du plan sont "compatibles" avec les contraintes imposées par la tâche. Le mécanisme correspondant calcule à chaque pas les incertitudes résultantes, à partir des incertitudes présentes avant l'exécution de l'action.

- Un mode de propagation "ascendant", qui est utilisé en cas d'échec pour rechercher les causes possibles de cet échec. Le procédé appliqué opère par

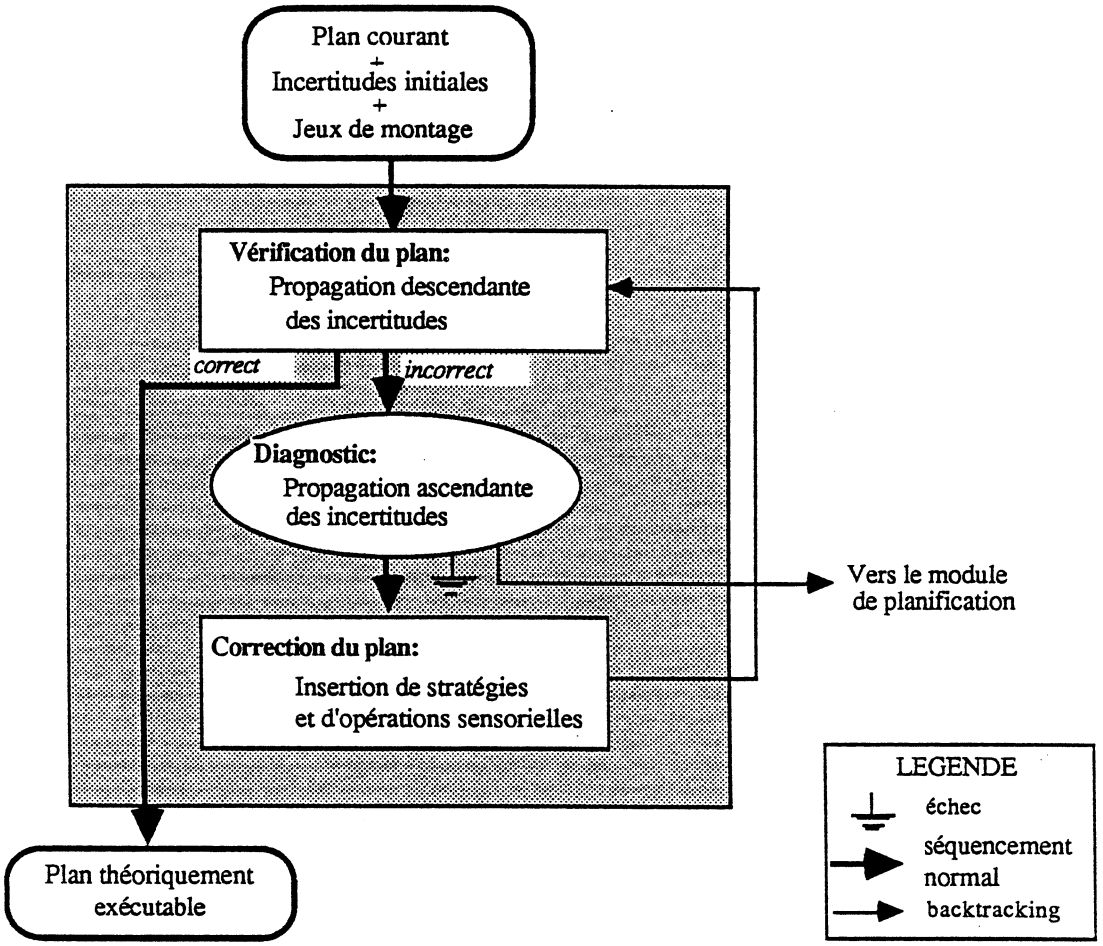


Figure 9.3: Fonctionnement du module de vérification/correction de plans.

régression, afin de calculer les contraintes d'incertitude qui auraient dû être vérifiées dans les étapes antérieures du plan.

Le mécanisme de propagation de contraintes sur lequel repose cette approche est décrit dans le paragraphe 9.5. Il n'a pour l'instant été implanté que de manière partielle.

9.2 Planification des mouvements de transfert:

9.2.1 Approche du problème:

Le problème traité par ce module de planification est celui du calcul de trajectoires permettant au robot de transporter des objets dans un espace parsemé d'obstacles. Les trajectoires engendrées portent sur l'ensemble des éléments constitutifs du bras articulé. Elles doivent satisfaire des contraintes physiques strictes: accessibilité, absence de collisions, lois cinématiques, insensibilité aux incertitudes et aux perturbations introduites par la dynamique. Elles doivent aussi présenter un degré d'optimalité satisfaisant.

La résolution de ce problème de planification nécessite de raisonner sur l'ensemble des états possibles du système articulé. Un tel raisonnement s'appuie sur une représentation explicite et concise de l'espace des configurations du mobile. Il conduit à construire un graphe de l'espace libre, dont l'aptitude à représenter la réalité physique dépend d'un compromis entre deux critères contradictoires: la complexité du modèle et la finesse des approximations appliquées. Cette approche conduit à ignorer certaines solutions potentielles localisées près des obstacles, au profit de trajectoires entièrement sûres.

La méthode de planification de trajectoires de transfert que nous avons implantée, opère suivant un schèma devenu classique:

1. Construction d'un graphe représentant une approximation de l'espace libre.
2. Recherche d'une trajectoire sûre par application d'un algorithme de parcours de graphe de type A*.

L'originalité de la méthode réside 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 de grossissement d'obstacles. Il présente la particularité d'être paramétrable par des contraintes diverses imposées par la tâche (allure générale de la trajectoire), par son environnement physique (taux d'encombrement), et par la structure mécanique du bras (chaîne cinématique). Dans l'implantation actuelle du système, les paramètres liés à la tâche et à l'environnement ont été fixés de manière empirique. Ceux liés à la structure cinématique sont directement dérivés du modèle du robot.

9.2.2 Algorithme général:

Le module de planification des mouvements de transfert de SHARP a été implanté autour des fonctions géométriques décrites dans le paragraphe 7.3. L'algorithme utilisé pour construire l'espace libre est le suivant:

Fonction $LIBRE(j, D)$
debut
si $j = n$ alors
debut
$D_1 := VALIDE(A_n[I_n], B)$;
*** D_1 est une liste du type $(D_1^n\ D_2^n \cdots D_{p_n}^n)$ ***
pour $i = 1, p_n$ faire
$EL_A = EL_A \cup (D \times D_i^n)$;
retour(EL_A);
fin-si;

$D_1 := PARTITION(I_j)$;
** D_1 est une liste du type $(D_1^j\ D_2^j \cdots D_{p_j}^j)$ ***
pour $i = 1, p_j$ faire
debut
$B := GROSSIR(B, D \times D_i^j, j)$;
$q_j := milieu(D_i^j)$;
si $\neg INTER(A_j(q_j), B)$ alors $LIBRE(j + 1, D \times D_i^j)$;
fin-faire;
fin;

Les fonctions $VALIDE$ et $INTER$ utilisées dans cet algorithme, sont celles définies dans le chapitre 7. Elles permettent respectivement de calculer les débattements valides et de déterminer si il y a interférence entre un élément du robot et un objet de l'environnement. Les fonctions $GROSSIR$ et $PARTITION$ sont décrites et discutées ci-après.

Les notations utilisées sont les suivantes:

A_j: j-ème élément du bras du robot.
B: ensemble $(B_1, B_2 \cdots B_m)$ d'obstacles.
I_j: domaine de variation de q_j.
D_i^j: i-ème intervalle de variation associé à q_j.
EL_A: espace libre.

La fonction $LIBRE$ est exécutée avec les paramètres suivants:

$EL_A = \emptyset$;
$LIBRE(1, \emptyset)$;

L'ensemble EL_A obtenu par application de l'algorithme précédent est ensuite structuré à l'aide de la fonction $ADJACENT$ (cf.paragraphe 7.6). Cette structure permet de chaîner entre elles toutes les cellules de EL_A qui possèdent une frontière commune.

La recherche d'une trajectoire particulière dans le graphe ainsi construit, est alors réalisée en deux temps:

1. Recherche d'un chemin dans le graphe à l'aide d'une fonction de type A^*.
2. Choix d'une trajectoire particulière parmi celles incluses dans la liste de cellules délivrée par la fonction précédente.

Les algorithmes utilisés pour réaliser ce travail sont ceux décrits dans le paragraphe 7.7. Les heuristiques incluses dans ces algorithmes tentent de minimiser les distances Euclidiennes mises en jeu.

9.2.3 Fonctions de grossissement:

La fonction $GROSSIR$ est utilisée pour calculer une approximation des C-obstacles. Elle est basée sur les algorithmes décrits dans les paragraphes 7.4 et 7.5. Sa forme générale est la suivante:

Fonction $GROSSIR(B, D, j)$
début
** D est une liste du type $(D_1, D_2 \cdots D_j)$ ***
pour $i = 1, j$ faire
debut
$s := LIEU(R_{i+1}, D_i)$;
$retour(CO_s(B))$;
fin-faire;
fin;

La fonction $LIEU$ calcule le lieu géométrique du point de référence R_{i+1} de A_{i+1}, lorsque q_i décrit l'intervalle D_i. L'ensemble symbolisé par $CO_s(B)$ est calculé comme indiqué dans le paragraphe 7.5.

Ainsi que nous l'avons déjà mentionné, nous utilisons une forme simplifiée de cette fonction dans le but de réduire la complexité algorithmique:

Fonction $GROSSIR(B, D, j)$
debut
$r := LONGUEUR(LIEU(R_n, D_j))$;
$retour(Gros_r(B))$;
fin;

Lorsque la taille des domaines engendrés par la fonction $PARTITION$ le permet, la fonction $GROSSIR$ n'est appliquée qu'une seule fois par tranche de type $dq_1 \times dq_2 \cdots dq_n$. La valeur calculée pour r est alors la suivante:

$$r = MAX_{i=2\cdots n}\ (LONGUEUR(LIEU(R_n, D_i)))$$

Ce choix n'est pas réalisé de manière automatique dans l'implantation actuelle du système.

9.2.4 Fonction de discrétisation:

La fonction $PARTITION$ permet de discrétiser l'espace articulaire du robot. Dans sa forme de base, cette fonction réalise un découpage régulier des domaines I_j pour $j = 1 \cdots n-1$.

Nous avons vu précédemment que l'efficacité de l'algorithme de planification de trajectoires dépend fortement de la qualité du découpage réalisé (terme en $O(N^{p-1})$), dans lequel N représente le nombre moyen d'intervalles par d.d.l. Il est donc très important de pouvoir minimiser ce facteur en adaptant le procédé de discrétisation en fonction des caractéristiques de l'environnement. Malheureusement, la paramétrisation de la fonction $PARTITION$ est difficile à réaliser, car elle dépend de facteurs divers tels que le taux d'occupation de l'espace ou les relations spatiales qui lient les obstacles aux éléments du robot. La formalisation de ces paramètres constitue une voie de recherche ouverte.

Dans une première version du module de planification de trajectoires, nous avons expérimenté une fonction de discrétisation à ***précision variable*** [Germain 84] [Laugier, Germain 85]. Cette fonction réalisait en premier lieu un découpage grossier de l'espace articulaire, aprés avoir squelétisé le robot et grossi les obstacles en conséquence. Dans le cas des trois premiers d.d.l d'un robot à articulations rotoïdes, ce découpage était réalisé à partir des sommets des obstacles. L'opération suivante consistait alors à affiner la représentation obtenue, en appliquant itérativement des opérations de "fusion" et de "partage". Les conditions d'application de ces opérateurs étaient définies par un triplet $(dq_{min}, d_{min}, d_{max})$, dans lequel dq_{min} représente la taille minimum autorisée pour le découpage, et d_{min} (resp. d_{max}) est la "distance" minimum (resp. maximun) qui peut exister entre un obstacle et son approximation dans une tranche dq_i de C_A:

- Si $distance(B, approximation(B)) < d_{min}$, alors fusionner les tranches dq_i et dq_{i+1}.

- Si $distance(B, approximation(B)) > d_{max}$, alors partager la tranche dq_i en deux.

Considérons par exemple le cas du robot planaire à trois d.d.l de la figure 9.4. La scène représentée a mis en jeu un graphe de 146 sommets, dans lequel chaque sommet définit une cellule libre de type $dq_1 \times dq_2 \times dq_3$, avec $dq_i = 0.1, 0.2ou0.4$ radians. La même scène modélisée à partir d'un découpage régulier de 0.15 radians, conduit à un graphe composé de 1200 sommets répartis sur deux composantes connexes. Malgré la taille importante du graphe, il n'y a alors pas de solution pour la trajectoire recherchée.

Ce procédé de découpage adaptatif de l'espace articulaire n'a pas été intégré dans la version actuelle de SHARP, car il reposait sur une estimation empirique des caractéristiques de la scène (seule l'expérience de l'opérateur permettait de choisir de bonnes valeurs pour les paramètres). D'importantes recherches sont encore nécessaires avant de pouvoir automatiser entièrement ce type de fonction.

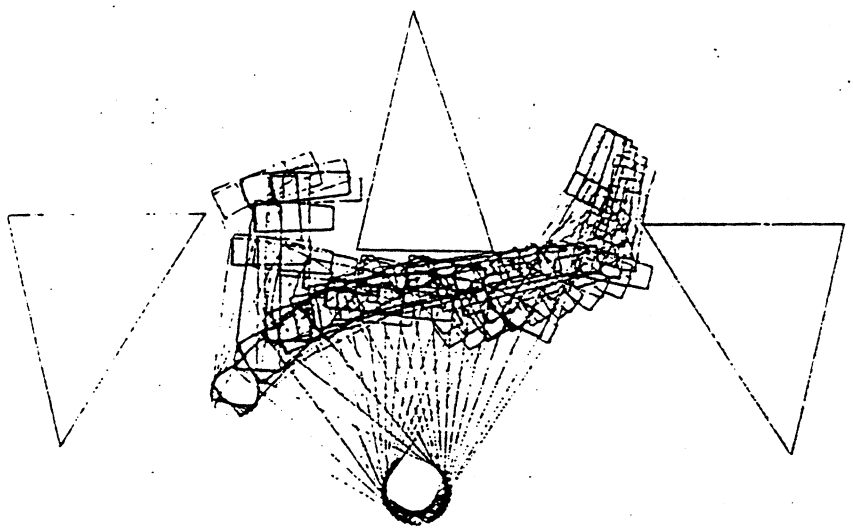


Figure 9.4: Exemple de trajectoire obtenue avec un découpage non uniforme de l'espace articulaire.

9.2.5 Traitement des rotations du poignet:

L'algorithme que nous avons implanté permet en théorie de planifier des mouvements pour l'ensemble des éléments du bras du robot. Dans la pratique, cet algorithme n'est utilisé que pour planifier les mouvements des trois premiers d.d.l du bras. Les rotations du poignet sont alors prises en compte au moyen de stratégies appropriées, permettant de programmer les opérations de réorientations sur certaines sections de la trajectoire planifiée. Les sections situées à proximité des obstacles (environnements de départ et d'arrivée en particulier), sont exécutées à orientation constante; celles localisées dans un espace bien dégagé (i.e un espace contenant le volume balayé par la pince et sa charge lors de l'exécution des rotations), servent de support à la réalisation des mouvements de réorientation du poignet.

Cette approche permet de mieux maîtriser la complexité algorithmique. Elle est raisonnable dans le sens où la plupart des trajectoires de transfert ne mettent en jeu que de petites rotations du poignet. Dans le cas contraire, il est presque toujours possible d'exécuter les opérations de réorientation dans des régions de l'espace peu encombrées (ce qui justifie les approximations utilisées).

Le traitement des d.d.l associés au poignet conduit donc à appliquer deux étapes de planification complémentaires:

- Détermination des mouvements à orientation constante dans le voisinage des positions de départ et d'arrivée.
- Détermination des mouvements de réorientation le long de la trajectoire décrite par le centre du poignet.

Ces deux étapes reposent sur l'utilisation d'heuristiques très simples.

9.2.6 Implantation et expérimentations:

Le module de planification des mouvements de transfert a été initialement implanté en MACLISP sur un calculateur CII-HB70. Il a ensuite été redéveloppé en LUCID-LISP sur machine SUN 260. Des expérimentations ont été faites en simulation sur des robots planaires à deux ou trois d.d.l, puis sur un robot SCEMI six axes modélisé comme indiqué dans le chapitre 3.

La figure 9.5 donne un exemple de trajectoire planifiée par le système. Le traitement de cet exemple a nécessité 28mn de CPU, pour construire un modèle de l'espace libre. Ce temps d'exécution relativement long a été obtenu sur une version provisoire du système, comportant des fonctions géométriques écrites en KCL non compilé. Le graphe construit est constitué de 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 d.d.l du bras). Une fois le modèle construit, le système peut engendrer une trajectoire telle que celle présentée dans l'exemple, en moins de 4s.

Comme nous l'avons déjà indiqué, seuls les trois premiers d.d.l du bras sont traités par l'algorithme général de planification. Les rotations du poignet sont alors programmées à l'aide de deux heuristiques très simples:

- Pas de rotations au voisinage des positions de départ et d'arrivée. L'absence de collisions le long des translations engendrées est alors testée au moyen d'un calcul d'interférence. Ce test est appliqué lorsque les positions considérées mettent en jeu des contacts ou des obstacles trop proches vis-à-vis des approximations faites par le procédé de grossissement.

- Recherche d'une trajectoire insensible aux mouvements de réorientation du poignet. Le système considère alors que la pince et l'objet transportés sont inclus dans une sphère, ce qui permet de répartir librement les rotations le long de la trajectoire planifiée. En cas d'échec, nous avons fait l'hypothèse qu'il existe toujours une région de l'espace de travail qui peut être utilisée pour exécuter les opérations de réorientation.

Les heuristiques que nous avons implantées marchent bien dans les cas courants. Aucune évaluation sérieuse n'a cependant été faite, faute de connexion avec le robot réel (cette connexion est en cours de réalisation). Malgré le manque d'expérimentation en grandeur nature, il est clair que ce point constitue une faiblesse du système. En particulier, les heuristiques utilisées sont mal adaptées au traitement des objets longs. C'est pourquoi, nous nous proposons de développer prochainement des fonctions permettant de planifier des morceaux de trajectoires sur la base d'informations locales (cf. [Faverjon, Tournassoud 87]). Ces fonctions pourront être utilisées pour engendrer des trajectoires dans les régions trop contraintes pour l'algorithme général.

Le principal avantage de la méthode que nous avons implantée réside dans sa relative simplicité algorithmique, et dans son aptitude à traiter dans des temps "raisonnables" des problèmes mettant en jeu moins de quatre d.d.l (les autres d.d.l étant traités comme indiqué précédemment). Bien que relativement longs, ces temps sont comparables à ceux que mettrait un opérateur humain pour planifier des trajectoires similaires à l'aide d'un système interactif de programmation de robots. Par contre, il est clair qu'une telle approche serait incompatible avec les impératifs d'un système contraint de prendre des décisions en ligne (robot opérant

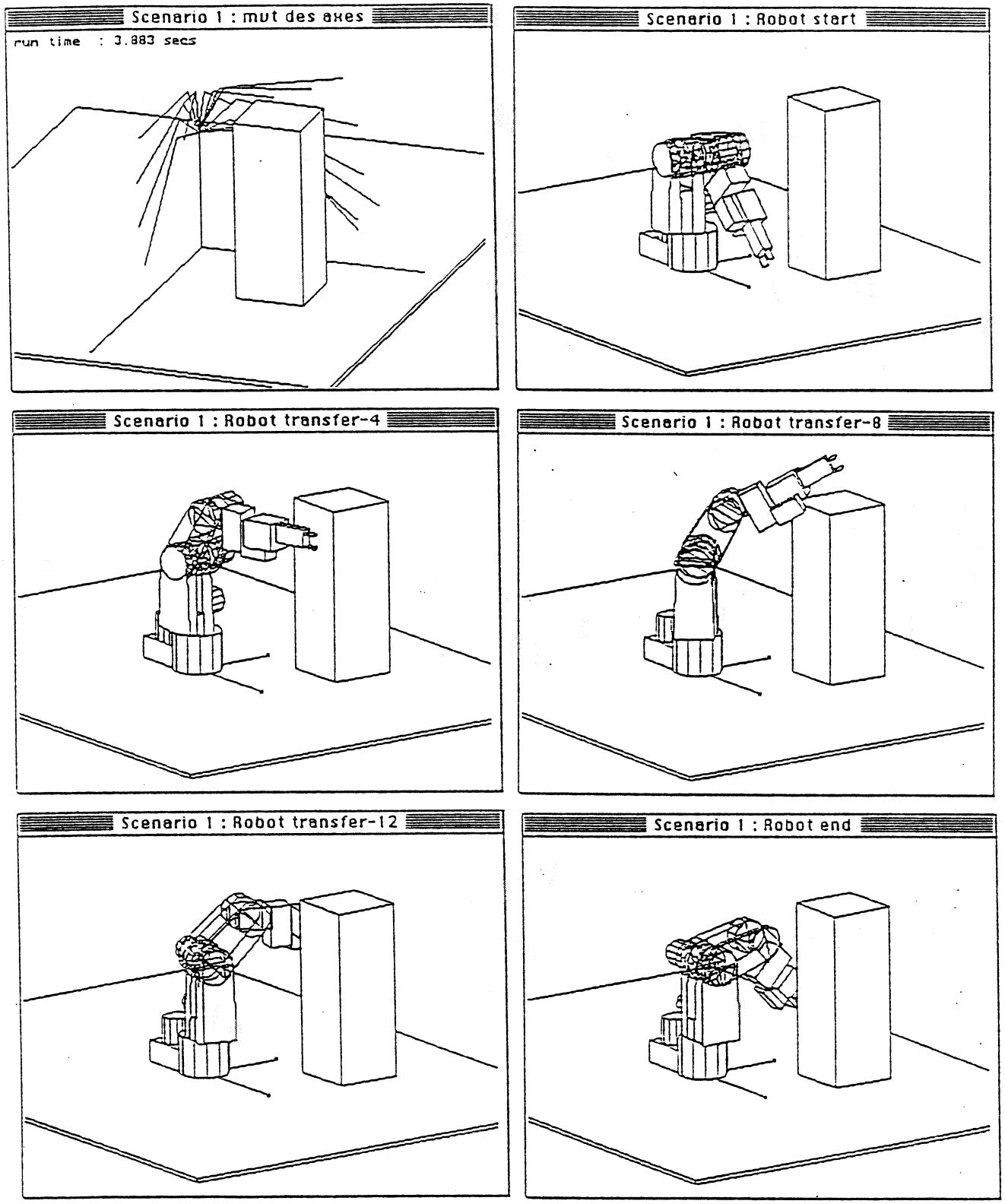


Figure 9.5: Exemple de mouvement planifié par le système.

en univers non structuré en particulier).

Les principaux inconvénients de la méthode sont liés au procédé de discrétisation qui peut conduire d'une part à masquer certaines solutions très contraintes, et d'autre part à faire croître rapidement les temps d'exécution lorsque le nombre de d.d.l considérés augmente (la complexité de l'algorithme comportant un terme en N^{p-1}). Ce dernier point est cependant inhérent au problème traité.

9.3 Planification des opérations de saisie:

9.3.1 Approche du problème:

Le problème abordé par ce module de planification est celui de la construction de prises qui sont à la fois stables, accessibles et compatibles avec l'opération de manipulation projetée. La résolution de ce problème met en jeu deux formes de raisonnement géométrique visant respectivement à rechercher les éléments de prise sur l'objet, et à spécifier les paramètres dynamiques de l'opération. C'est pourquoi le système que nous avons implanté opère en deux phases organisées suivant un schéma de prédiction-vérification classique en Intelligence Artificielle [Laugier 83] [Troccaz 86]:

- La première phase a pour objet de rechercher des configurations de contact main-objet, qui sont localement réalisables et stables. Cette recherche repose sur une analyse des indices morphologiques locaux de l'objet. Elle conduit à construire des ensembles de prises potentielles incomplètement spécifiées (cf. paragraphe 8.6).

- La deuxième phase vérifie que les solutions exhibées sont accessibles dans les environnements de saisie et de lâcher de l'objet. Le raisonnement appliqué repose sur une étude des configurations de l'outil qui sont à la fois compatibles avec la prise choisie, et réalisables dans l'espace de travail du robot. Cette étude conduit d'une part à rejeter les solutions impossibles, et d'autre part à spécifier les paramètres dynamiques de l'opération de saisie (orientation et position finale de l'outil, directions d'approche et de dégagement).

La première phase repose sur les techniques de raisonnement morphologique décrites dans le chapitre 8. Elle évalue successivement les propriétés qui caractérisent les contacts potentiels, et celles qui permettent de construire des ensembles de contacts contraints.

La deuxième phase utilise les techniques de raisonnement spatial décrites dans le chapitre 7. Elle construit une représentation explicite de l'espace des configurations de la pince, lorsque celle-ci exécute les mouvements autorisés par la

prise étudiée. Les paramètres manquants sont alors choisis à l'aide de cette représentation.

9.3.2 Algorithme général:

Si l'on utilise les notations introduites dans le paragraphe 8.6, l'algorithme général de construction d'une prise est le suivant:

- **Phase 1**

  1. Recherche des contacts potentiels.
  2. Recherche des couples de contacts. Construction des prises potentielles de type $\Pi(A/P)$.

- **Phase 2**

  1. Choix d'une prise potentielle $\Pi(A/P)$ compatible avec la position de A dans les environnements de saisie et de lâcher.
  2. Choix d'une direction Δ du plan de préhension P_π.
     Si il n'y a plus de direction non examinée, retourner en (1).
  3. Détermination des configurations valides de $\Pi(A/P)$ pour l'orientation Δ:
     $$V_\pi(\Delta) = \{p \in P_\pi : \ P(p,\Delta) \cap e \neq \emptyset, \forall e \in E\}$$
     Si $V_\pi(\Delta) = \emptyset$, retourner en (2).
  4. Détermination des configurations sûres de $\Pi(A/P)$ pour l'orientation Δ:
     $$S_\pi(\Delta) = \{p \in V_\pi(\Delta) : \ P(p,\Delta) \cap \{W - E\} = \emptyset\}$$
     Si $S_\pi(\Delta) = \emptyset$, retourner en (2).
  5. Choix d'un point P de $S_\pi(\Delta)$, tel que p maximise la surface de contact, et p minimise le couple créé par la pesanteur.

Dans le cas d'une prise définie sur une surface de révolution, l'algorithme est complété par une étape supplémentaire qui permet de choisir un plan de préhension. Le choix de ce plan repose sur une heuristique semblable à celle utilisée pour le choix de Δ.

9.3.3 Construction des prises potentielles:

Le procédé employé pour rechercher les éléments qui constituent les prises potentielles, opère par application de filtres géométriques simples. L'objectif visé est alors de réduire au maximum l'arbre de recherche, en utilisant des filtres aussi discréminants que possible. Ces filtres implantent les propriétés décrites dans les

paragraphes 8.3 et 8.4.

Trois critères interviennent dans le classement des filtres géométriques:

- *Les contraintes d'antériorité* provenant de la définition même des propriétés analysées.
- *Le pouvoir discréminant* attribué aux expressions évaluées.
- *Le coût algorithmique* associé.

D'une manière générale, l'ordre choisi est obtenu par respect des contraintes d'antériorité, en essayant de minimiser les coûts algorithmiques et d'utiliser en priorité les filtres les plus discréminants. Par exemple, la propriété de P-accessibilité locale possède un fort pouvoir discréminant, mais elle doit par construction être appliquée après celle d'accessibilité locale; inversement, la propriété de Mutuelle Visibilité n'a pas de réelle contrainte de précédence, mais son évaluation est assez coûteuse.

Il n'y a donc pas de méthode formelle pour optimiser l'ordonnancement des filtres. Dans SHARP, nous avons choisi d'appliquer l'ordre suivant: (1) Accessibilité locale, (2) Parallélisme, (3) P-accessibilité locale, (4) Distance, et (5) Mutuelle Visibilité.

9.3.4 Choix d'une direction d'approche:

Le choix de Δ est basé sur une étude grossière des obstacles potentiels à la pince. Cette étude est réalisée dans le plan de préhension P_π, après avoir projeté tous les obstacles présents dans le volume délimité par les plans tangents aux extrémités de la pince. Les solutions potentielles pour Δ sont alors représentées par des secteurs angulaires libres, construits à partir d la projection sur P_π du centre de gravité de l'objet. La direction Δ est choisi verticalement lorsque cela est possible; Elle est choisie au centre du plus grand secteur dans le cas contraire.

9.3.5 Détermination des configurations valides:

La détermination des configurations valides de P est réalisée par application de l'opérateur de grossissement défini dans le paragraphe 7.3. Ce calcul est exécuté dans le plan de préhension P_π [Troccaz 87]:

$$V_\pi(\Delta) = \cap_{e_{ij} \in E} \, Proj_{P_\pi}(CO^{xz}_{F_i[\Delta]}(e_{ij}))$$

où $Proj_{P_\pi}(X)$ est la projection orthogonale de X sur P_π, et $F_i[\Delta]$ est la face de contact du i-ème mors ($F_i[\Delta]$ est orienté par rapport à Δ).

9.3.6 Détermination des configurations sûres:

La détermination des configurations sûres de P est aussi réalisée par application de l'opérateur de grossissement. Le calcul exécuté peut alors être représenté par l'expression:

$$S_\pi(\Delta) = V_\pi(\Delta) - Proj_{P_\pi}(CO^{zz}_{P[\Delta]}(B))$$

où $P[\Delta]$ est la pince orientée suivant Δ, et B définit l'ensemble des obstacles (objet A compris).

Dans notre implantation, ce calcul est réalisé dans un espace bidimensionnel. Cet espace est construit par projection de tranches découpées parallèlement au plan de préhension. Le détail de ce procédé est décrit dans [Laugier, Pertin 83] et dans [Troccaz 86]. Son application au cas de la saisie guidée par des informations sensorielles $3D$ est détaillée dans [Troccaz 87]. Le principe du calcul est le suivant:

$$S^i_\pi(\Delta) = S^{i-1}_\pi(\Delta) - Proj_{P_\pi}(CO^*_{P_i[\Delta]}(B_i))$$

où

- $S^i_\pi(\Delta)$ est l'ensemble des configurations sûres associées aux tranches $1, 2 \cdots i$.
- $S^0_\pi(\Delta) = V_\pi(\Delta)$.
- $P_i[\Delta]$ et B_i sont respectivement les portions de P et de B comprises dans la tranche i.
- CO^* est une opération spéciale de grossissement dans P_π.

L'opération notée CO^* est différente du grossissement CO^{yz}, car la pince ne peut être considérée indépendamment du bras qui la supporte. La solution que nous avons choisie pour tenir implicitement compte du bras, consiste à considérer que la pince se prolonge dans la direction Z_P. Ceci se traduit par la construction d'une "zone d'ombre" inaccessible derrière chaque obstacle. Cette zone d'ombre est alors obtenue en prolongeant les obstacles grossis dans la direction $-\Delta$. L'algorithme de construction est le suivant:

1. Calcul de $CO^{zz}_{P_i[\Delta]}(B_i)$.
2. Soit s_1 et s_2 les deux sommets extrémums de $CO^{zz}_{P_i[\Delta]}(B_i)$ dans la direction perpendiculaire à Δ, s_3 le sommet le plus éloigné de $S^{i-1}_\pi(\Delta)$ dans la direction $-\Delta$. On note respectivement l_1, l_2 et l_3 les droites (s_1, Δ), (s_2, Δ) et $(s_3, \perp \Delta)$.
$$Ombre(B_i, \Delta) = Polygone(s_1, l_1 \cap l_3, l_2 \cap l_3, s_2)$$
3. $CO^*_{P_i[\Delta]}(B_i) = CO^{zz}_{P_i[\Delta]}(B_i) \cup Ombre(B_i, \Delta)$.

La ligne l_3 marque la limite de la région à étudier, du fait que les points à atteindre (ceux de $S^{i-1}_\pi(\Delta)$) sont tous situés au-dessus de cette ligne. Ce procédé de calcul est illustré par la figure 9.6.

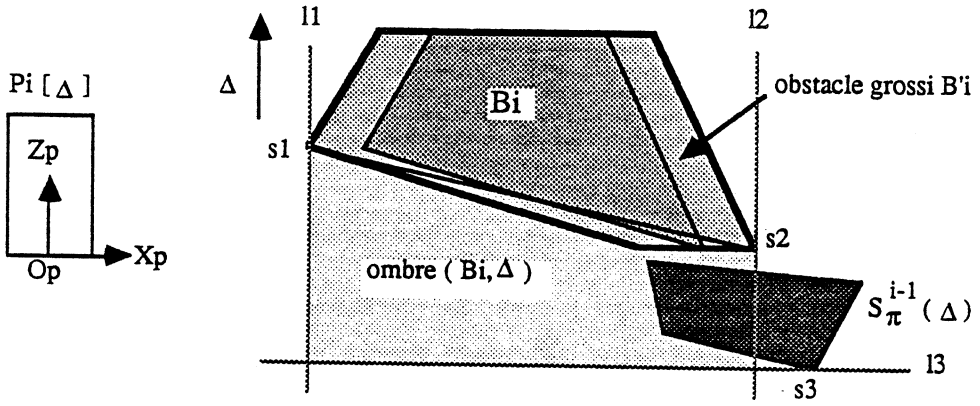


Figure 9.6: Principe de grossissement des obstacles pour la recherche de prises sûres. $P_i[\Delta]$ et B_i sont respectivement les portions de P (avec l'orientation Δ) et de B. B_i' représente l'obstacle grossi $CO^{zz}_{P_i[\Delta]}(B_i)$. S représente l'ensemble des points valides $S_\pi^{i-1}(\Delta)$, obtenu à l'issue de l'étape précédente.

9.3.7 Implantation et expérimentations:

Le module de planification des opérations de saisie a été implanté en MACLISP sur un calculateur CII-HB70. Son transfert en LUCID-LISP sur machine SUN 260 est en cours de réalisation.

Des expérimentations ont été faites en simulation sur des objets assez simples (typiquement: objets comportant moins de 20 faces), saisis à l'aide d'une pince à deux mors parallèles. Les temps d'exécution peuvent alors être de plusieurs minutes. Aucune évaluation sérieuse n'a par contre été faite en ce qui concerne les heuristiques de choix possibles (celles utilisées étant très simples).

Prenons par exemple l'objet de la figure 9.7. Cet objet comporte 8 faces planes, 2 faces cylindriques, 12 arêtes rectilignes, 4 arêtes circulaires et 8 sommets. Seulement 12 prises potentielles (sur 32! possibilités) ont été retenues par le système à l'issue de la première phase. Le temps de calcul nécessaire est alors de l'ordre de la seconde. Les deux prises présentées dans la figure, ont été obtenues en choisissant des directions d'approches perpendiculaires aux bords des entités concernées. Dans cet exemple, aucune contrainte d'environnement n'a été prise en compte. Le

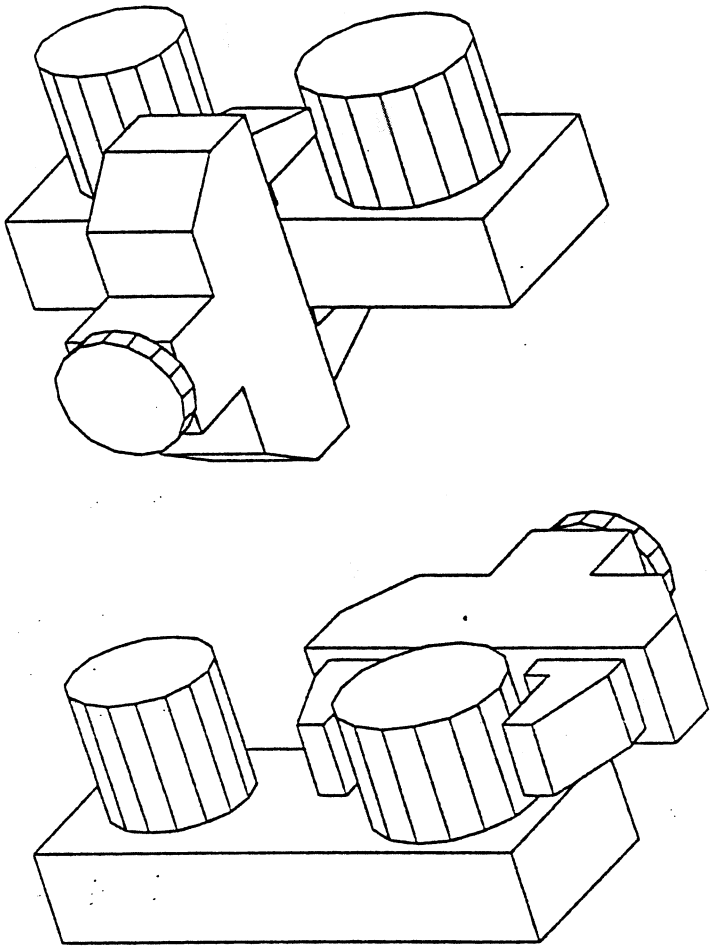


Figure 9.7: Exemple de prises engendrées par le système.

temps total nécessaire à la génération de ces prises est alors réduit à quelques dizaines de secondes.

Les principales limitations introduites par l'implantation actuelle sont les suivantes:

- Utilisation d'une pince à deux mors parallèles. L'extension à d'autres types

d'outils à mors plans ne pose pas de problèmes. Le cas d'outils plus généraux nécessiterait par contre de développer d'autres techniques.

- Pas de changement de prise. L'analyse des solutions qui mettent en œuvre plusieurs prises successives n'est pas envisageable dans l'état actuel du système. Comme nous l'avons déjà indiqué, une technique possible pour autoriser ce type de solution consiste à chaîner entre elles toutes les prises potentiellement réalisables dans les différentes positions d'équilibre de l'objet.

- Pas d'analyse de compatibilité avec les mouvements fins. Cet aspect est en principe pris en charge par la structure de contrôle du système (cf. paragraphe 9.1). Il serait cependant nécessaire de prévoir des changements de prises au cours de l'exécution de certaines stratégies de mouvements fins, par exemple: posage d'un bloc parallélépipédique dans un coin.

9.4 Planification des mouvements fins:

9.4.1 Approche du problème:

Le problème résolu par ce module de planification est celui de la synthèse automatique de programmes de mouvements fins. L'objectif visé est alors d'engendrer des mouvements aptes à réaliser une relation d'assemblage donnée, malgré les contraintes d'incertitudes imposées par la tâche. Les stratégies implantées dans ce programme combinent des opérations sensorielles avec des petits mouvements du robot. Elles permettent ainsi de s'affranchir des erreurs de commande et de perception.

Planifier une opération de montage peut ainsi être vue comme la recherche d'une séquence ordonnée de relations de contact, aptes à guider le robot vers l'objectif à atteindre. Chaque relation réalisée permet de se rapprocher du but, en réduisant certaines composantes d'incertitude. Elle met en jeu des mouvements destinés à rechercher un contact, et/ou à en maintenir d'autres créés par des opérations antérieures. Le processus de planification tente d'abord de définir symboliquement les contacts qui semblent présenter un intérêt pour le montage. Il choisit ensuite parmi les solutions possibles, celles qui lui paraissent être les mieux adaptées au contexte de la tâche. Les paramètres des mouvements engendrés (direction, amplitude, conditions d'arrêt, forces) sont alors déterminés en fonction de critères d'accessibilité et de fiabilité.

La méthode que nous avons développée, procède par chaînage arrière afin de *déduire un plan de montage à partir d'une étude des démontages possibles de l'objet.* Cette approche se justifie par le fait que les contacts existants contraignent les mouvements relatifs des deux objets, ce qui réduit le nombre des hypothèses à

considérer. L'algorithme implanté opère en deux phases d'analyse et de recherche [Laugier, Theveneau 86]:

- *La phase d'analyse* a pour objet de construire un graphe qui représente les différentes manières théoriques de démonter l'assemblage. Le système fait alors abstraction des contraintes réelles de la manipulation (risques d'échecs dus à l'imprécision du robot en particulier). Il analyse uniquement les contraintes de mouvement imposées par les contacts. Les transitions du graphe représentent alors des mouvements potentiels, dont le rôle est de relâcher un à un les contacts mis en jeu par l'assemblage.
- *La phase de recherche* conduit à choisir un "chemin" inverse dans le graphe de démontage, afin de construire une stratégie de mouvements fins particulière. Cette recherche peut conduire en cas d'échec à raffiner certaines parties du graphe. Par exemple, une relation de contact n'est pas réalisable à cause des erreurs de commande. La recherche est alors guidée par des heuristiques et des connaissances expertes qui tentent d'optimiser la solution en termes d'efficacité et surtout de fiabilité des opérations.

La principale originalité de cette approche réside dans le fait que le graphe d'états construit peut être localement affiné lors de la recherche d'une solution. Les motivations qu'il y a derrière cette approche sont doubles:

- Réduire l'arbre de recherche.
- Exécuter les calculs géométriques coûteux uniquement lorsque cela s'avère nécessaire.

Les modes de raisonnement utilisés sont ceux décrits dans le chapitre 8. Ils sont complétés par des calculs de déplacements exécutés au moyen de la fonction $VALIDE$ définie dans le paragraphe 7.2.

9.4.2 Algorithme général:

Les notations utilisées dans cet algorithme sont celles introduites dans le paragraphe 8.6.2. Nous désignerons par $OUVERT$ la liste des nœuds non encore traités, et par D_x l'ensemble des mouvements potentiels associés au nœud x.

- **Phase d'analyse**

  1. Créer le nœud EF, et le mettre dans la liste $OUVERT$.

2. Si $OUVERT = \emptyset$ retourner $G(A/B)$, sinon traiter le nœud x situé en tête de liste $OUVERT$.
   Choisir n directions d'étude $d_1, d_2 \cdots d_n$ dans D_x.
   Créer un arc a_{xi} pour chaque direction d_i précédente.
3. Créer un nœud y_i pour chaque arc a_{xi}.
   Si l'état E_i associé à y_i est déjà représenté par un nœud z de $G(A/B)$, fusionner y_i et z.
4. Mettre tous les nouveaux nœuds qui possèdent un ensemble de contacts vide dans EI; mettre les autres dans la liste $OUVERT$.
   Retourner en (2).

- **Phase de recherche**

1. Rechercher un sous-graphe SG suivant la procédure indiquée dans le paragraphe 8.6.2.
2. Vérifier que chaque arc de SG représente un mouvement exécutable par le robot (pas de collision, faisabilité de l'objectif).
   En cas d'échec affiner $G(A/B)$, puis retourner en (1).
3. Synthétiser le programme de mouvements fins représenté par SG.

9.4.3 Choix des directions d'étude:

Comme nous l'avons indiqué dans le paragraphe 8.5.5, le système n'étudie pas toutes les directions incluses dans le domaine D_x considéré. Seuls quelques mouvements pertinents sont alors sélectionnés. Le mécanisme de sélection utilisé repose sur des "conseils", codés sous la forme de règles expertes.

Les règles de sélection sont activées par les données contenues dans le champ I des nœuds du graphe (cf. paragraphe 8.6.2). Par exemple, quatre directions de déplacement seront initialement engendrées pour un couple de contacts supportés par deux faces non parallèles F_1 et F_2: $d_1 = N_1 \wedge N_2$, $d_2 = -d_1$, $d_3 = d_1 \wedge N_1$ et $d_4 = d_2 \wedge N_2$, où N_1 et N_2 représentent respectivement les normales extérieures de F_1 et de F_2. Les directions d_1 et d_2 définissent des mouvements compliants qui conservent les deux contacts; d_3 et d_4 conduisent à rompre respectivement les contacts suivant F_2 et F_1. Les hypothèses de mouvement ne sont retenues par le système que dans la mesure où elles vérifient les contraintes représentées par D_x.

9.4.4 Création des nœuds et des arcs de $G(A/B)$:

Les nœuds de $G(A/B)$ sont créés à l'issue d'un calcul géométrique permettant de:

- Déterminer les contacts mis en jeu (paramètre C).
- Calculer l'ensemble des mouvements potentiels (paramètre D).

- Vérifier que l'état E_i considéré n'est pas déjà représenté dans $G(A/B)$, et qu'il peut être discerné des états voisins.

Les calculs utilisés pour résoudre les deux premiers points sont ceux décrits dans les paragraphes 8.2 et 8.5; ceux appliqués pour traiter le dernier point ont été développés dans les paragraphes 6.4 et 6.5. Le paramètre P de position est alors directement défini par le mouvement qui a permis d'atteindre l'état représenté, et le paramètre I est un sous-produit des calculs précédents.

Les arcs de $G(A/B)$ sont créés à l'issue d'un calcul géométrique permettant de:

- Déterminer l'amplitude maximum du déplacement (paramètre T).
- Déterminer les contacts impliqués dans le mouvement (paramètres C et A).

Le premier point est traité à l'aide de la fonction $VALIDE$ du paragraphe 7.2; l'autre point met en jeu des calculs tels que ceux développés dans le paragraphe 8.2.

Les heuristiques que nous utilisons pour évaluer la qualité Q des nœuds et des arcs, sont codées sous la forme de "conseils" facilement modifiables. Les plus significatives de ces heuristiques sont les suivantes:

- Si la face d'appui pour le mouvement fait un angle de plus de 45^o avec l'horizontale, alors $poids(x) = faible$.
- Si l'angle entre la face d'appui et la face d'arrêt est plus grand que 45^o, alors $poids(x) = faible$.
- Si la surface du contact est faible, alors $poids(x) = faible$.

9.4.5 Recherche d'une solution:

Principe de la recherche:

La fonction de recherche d'un sous-graphe assimilable à une stratégie de mouvements fins est basée sur l'algorithme du paragraphe 8.6.2. Elle présente cependant deux particularités liées à notre implantation:

- La fonction $CHOIX$ peut conduire à affiner le graphe, lorsque la transition sélectionnée n'est pas directement exécutable par le robot.
- La recherche des arcs "semblables" (étape (3) de l'algorithme) n'est pas réalisée, car le graphe $G(A/B)$ initial ne comporte pas de mouvements susceptibles d'amener le robot dans des états différents. Seules les procédures

d'expansion du graphe sont habilitées à introduire ce type de mouvement. Le sous-graphe ainsi créé est alors inclus entièrement dans la solution en cours de construction. Il est clair que ce procédé ne reste valide, que dans la mesure où les modifications du graphe sont de nature locale (ce qui est généralement le cas). Les mécanismes de sélection que nous avons implantés respectent cette règle.

Heuristiques utilisées:

Deux fonctions sont simultanément utilisées pour guider la recherche d'une solution:

- *La fonction coût* combine les pondérations heuristiques attribuées aux nœuds et aux arcs du graphe. Elle conduit d'une part à minimiser le nombre de mouvements, et d'autre part à maximiser la qualité des solutions choisies. Un autre effet de cette fonction est de contraindre le système à toujours tenter d'accroître le nombre de degrés de liberté bloqués par les contacts.
- *Les conseils dynamiques* sont activés chaque fois qu'une situation singulière est détectée dans le graphe. Ils permettent essentiellement de traiter des situations intermédiaires irréalisables par le robot, à cause des erreurs de commande et de perception (contacts par adjacence, obstacles proches $\cdots$). Par exemple, si un contact ne peut être directement relâché à cause de la présence d'un obstacle dans son environnement immédiat, il est conseillé de glisser auparavant sur la surface de contact; cette opération a pour effet de créer de nouveaux nœuds et de nouveaux arcs dans le graphe. De même, la rencontre d'une relation d'adjacence conduira à rechercher des contacts voisins permettant de guider le robot.

Cette approche permet de réduire la complexité algorithmique, en ne développant que les parties significatives du graphe de recherche. Elle permet aussi d'introduire des stratégies locales connues, afin de résoudre certains problèmes de montage caractéristiques (utilisation de stratégies d'insertions par exemple). Ce dernier aspect n'a pas encore été implanté.

Analyse de validité de la solution:

La procédure d'analyse de validité des mouvements retenus, réalise deux types de vérifications:

- Aucune collision n'est susceptible de se produire, compte tenu de l'incertitude initiale de position ε_i et de l'erreur maximum de commande ε_p. Un test de collision est alors réalisé, après avoir grossi les obstacles potentiels d'une épaisseur égale à $\varepsilon = 2(\varepsilon_p + \varepsilon_i)$:

$$Balayage(A, d) \cap Gros_\varepsilon(B) = \emptyset$$

où $Balayage(A, d)$ est le volume balayé par A au cours du déplacement d, et $Gros_\varepsilon(B)$ représente les obstacles grossis suivant le procédé décrit dans le paragraphe 7.3.

- Le contact objectif est toujours réalisé, compte tenu des paramètres ε_p et ε_i précédents. Un test d'interférence est alors réalisé, après avoir réduit les obstacles potentiels d'une épaisseur égale à $\varepsilon = 2(\varepsilon_p + \varepsilon_i)$:

$$A(p) \cap Gros_\varepsilon^{-1}(B) \neq \emptyset$$

où p est la position commandée, $A(p)$ est l'objet A en position p, et $Gros_\varepsilon^{-1}(B)$ représente les obstacles réduits par ε. Le procédé de réduction est alors similaire à celui de grossissement.

Dans le cas où la position commandée est incluse dans une boule ouverte $B(p, r)$, les opérations de grossissement sont exécutées avec le paramètre $(\varepsilon + r$. Une telle situation se produit lorsque plusieurs directions d'étude sont analysées simultanément; elle peut également se produire lorsque la position initiale du robot à partir de laquelle est défini p, est localisée dans une boule ouverte $B(p^*, r)$. Compte tenu des choix que nous avons faits (cf. paragraphe 8.5.5), ce cas ne se présente pas dans l'implantation actuelle du système.

9.4.6 Synthèse du programme de mouvements fins:

Soit SG le sous-graphe qui représente la stratégie de mouvements fins choisie. On note s_i le nœud courant, et $a_{ij_1}, a_{ij_2} \cdots a_{ij_n}$ les arcs issus de s_i. Par construction, ces arcs représentent un même mouvement m_i pouvant conduire à n états différents. Les conditions d'arrêts associées sont alors notées $b_{ij_1}, b_{ij_2} \cdots b_{ij_n}$.

Le sous-garphe SG est transformé en un programme exécutable par le robot, en appliquant les règles de conversion suivantes pour chaque nœud s_i:

1. Engendrer une instruction de manipulation du type:

DEPLACER <objet-A> SUIVANT <T> EN-MAINTENANT <C>
JUSQUA <A>

où T, C et A sont les paramètres de mouvement associés aux arcs $a_{ij_1}, a_{ij_2} \cdots a_{ij_n}$. Les paramètres T et C définissent le déplacement (transformation géométrique et liste de contacts à maintenir); ils sont par construction communs aux arcs $a_{ij_1}, a_{ij_2} \cdots a_{ij_n}$. Le paramètre A indique tous les contacts qui sont susceptibles de stopper le mouvement; il est représenté par une disjonction logique, dont les termes sont les conditions b_{ij_k}, $k = 1 \cdots n$.

2. Si il existe plusieurs arcs issus de s_i, alors engendrer une instruction conditionnelle portant sur l'exécution des mouvements associés aux nœuds $s_{j_1}, s_{j_2} \cdots s_{j_n}$. Les conditions évaluées sont alors celles définies par les termes $b_{ij_1}, b_{ij_2} \cdots b_{ij_n}$.

3. Si il existe une boucle élémentaire (c'est à dire un arc a_{ii} revenant sur le sommet s_i), alors engendrer une instruction itérative portant sur le mouvement m_i. La condition évaluée est alors celle définie par le terme b_{ii}. Les autres circuits de SG donnent lieu à la génération d'instructions de type "allera", permettant de revenir sur des mouvements antérieurs.

Les instructions de manipulation utilisées dans le programme synthétisé, sont celles que nous avons définies dans le chapitre 4. Les valeurs attribuées aux paramètres d'exécution sont calculées à partir du modèle géométrique et des termes T, C et A: la position commandée est directement dérivée de la transformation T; les conditions sur les forces sont calculées à partir du vecteur vitesse et des caractéristiques des faces de contact. Une face d'arrêt engendrera par exemple une condition de type "$F_v > seuil$", pour stopper un déplacement de direction $-v$ (F_v est la composante suivant v de la force de réaction, et v est supposé être contenu dans le cône de frottement). De même, la direction de la force nominale à conserver au cours d'un mouvement compliant simple (c'est à dire d'un mouvement ne mettant pas en jeu des forces qui s'opposent), pourra être obtenue en additionnant les normales extérieures aux faces de glissement. Dans la pratique, cette force devra rester dans un petit domaine défini par un cône d'erreurs et par un intervalle de valeurs. Cet aspect n'a été implanté que de manière partielle. Il mériterait d'être approfondi dans le futur.

9.4.7 Implantation et expérimentations:

Le module de planification de mouvements fins a été implanté en LUCID-LISP sur machine SUN 260. Des expérimentations ont été faites en simulation sur des objets polyédriques comportant moins de 20 faces. Les temps d'exécutions sont relativement longs à cause du volume des calculs et des choix d'implantation. En particulier, nous avons basé la détermination des contacts et de leurs propriétés sur un mécanisme de "démons" [Winston 77]. Ce procédé a l'avantage d'automatiser la gestion des contacts dans le modèle (toute modification de celui-ci entraîne une mise à jour automatique des relations de contact). Il facilite ainsi l'introduction de nouvelles stratégies ou de nouvelles fonctions dans le système. Il possède par contre l'inconvénient d'augmenter de manière importante les temps de traitement.

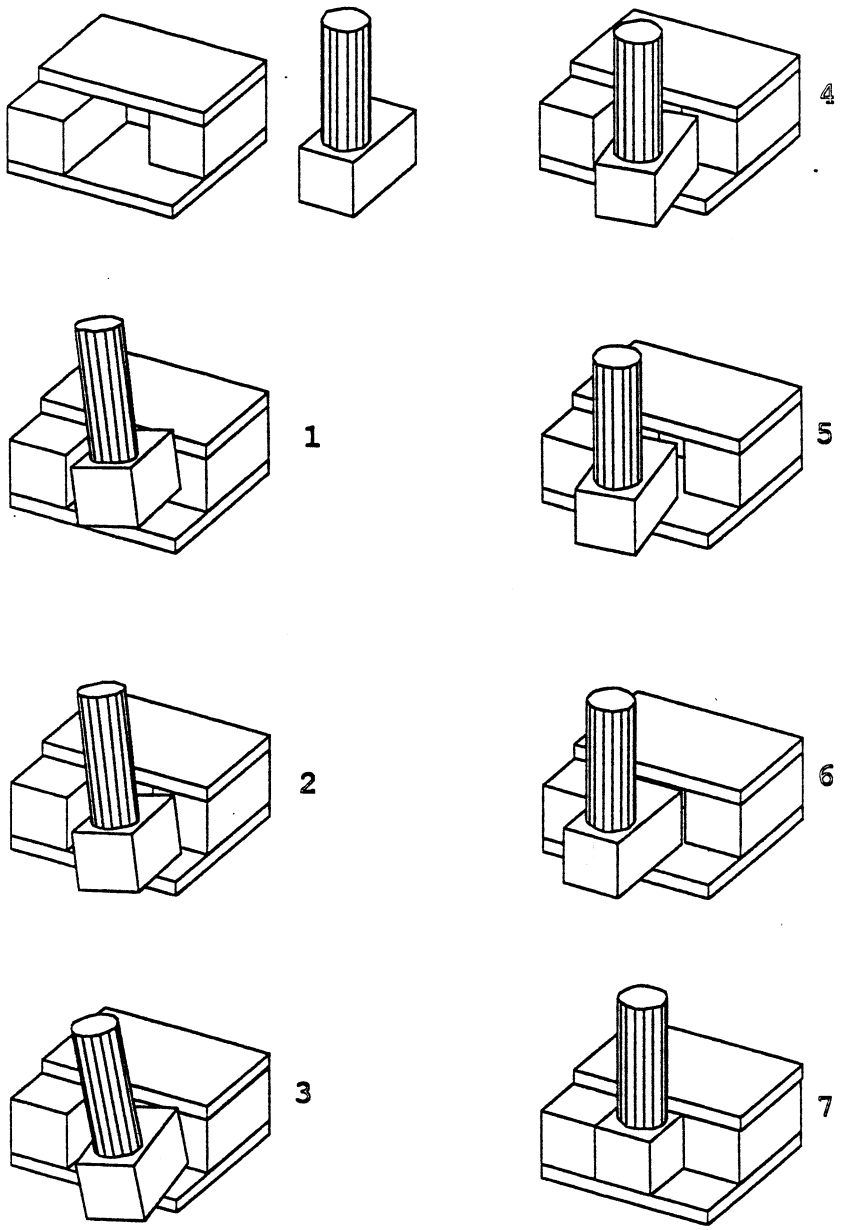


Figure 9.8: Exemple de stratégie de mouvements fins engendrée par le système.

Considérons par exemple le montage présenté dans la figure 9.8. Les objets concernés possèdent respectivement 8 et 12 faces. Le traitement de ce montage a nécessité 9 minutes de temps CPU. Il a donné lieu à un graphe d'états comportant 50 nœuds et 80 arcs. Le programme synthétisé est alors constitué de cinq

instructions de manipulation.

Les principales limitations introduites par l'implantation actuelle sont les suivantes:

- Le système analyse les directions d'étude une à une. Ce procédé a l'avantage de la simplicité et de l'efficacité dans les cas courants. Il permet de plus de faciliter l'analyse de validité des mouvements engendrés, du fait que les ensembles de positions nominales associées aux nœuds du graphe sont réduits à des singletons. Il peut par contre conduire à des échecs, faute de pouvoir analyser toutes les directions possibles. Bien que le mécanisme d'expansion dynamique du graphe permette de pallier partiellement à ce défaut, il serait souhaitable d'étendre l'algorithme en vue de pouvoir traiter simultanément des ensembles de mouvements potentiels.

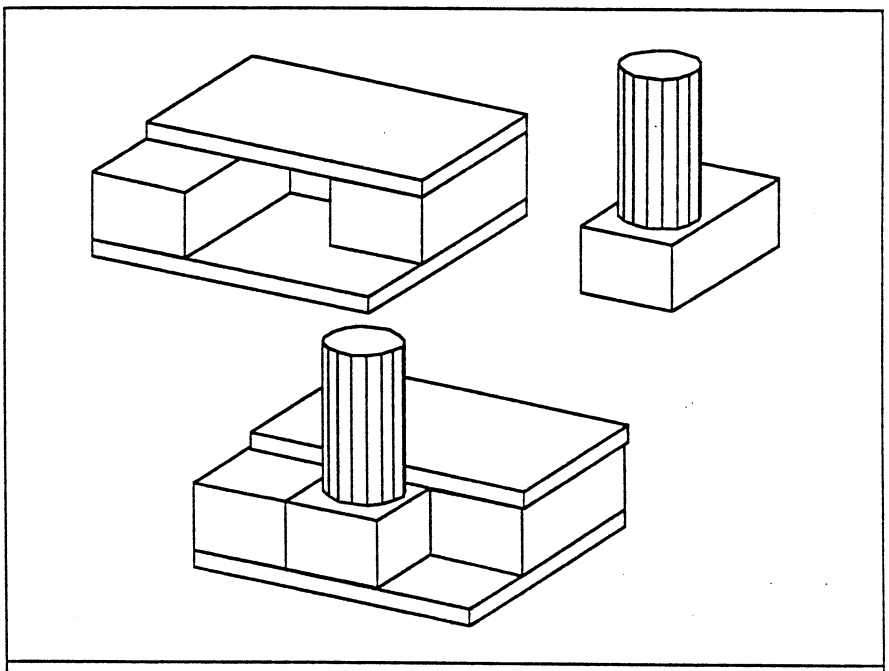
- Le mécanisme de sélection des directions d'étude ne choisit que les directions qui garantissent la réalisation d'un objectif du type: création ou destruction d'un contact. Ce procédé exclut tous les mouvements qui sont susceptibles d'amener le robot dans des états différents, par suite des erreurs de commande et de perception. Par exemple, le système n'engendrera pas de stratégie d'insertion conduisant à "tenter" l'opération et à corriger itérativement jusqu'à l'obtention de la relation souhaitée. Nous avons prévu d'intégrer ce type de stratégie spécialisée, au moyen des conseils dynamiques utilisés pour affiner le graphe.

- Seules les rotations exécutées autour de l'axe d'une partie cylindrique ou conique, et celles exécutées en vue de modifier la nature d'un contact (ponctuel à linéique et linéique à plan), sont traitées par le système. Ce type de limitation interdit de traiter les montages qui mettent en jeu des corrections d'orientations autres que celles engendrées par les axes de révolution ou par la réalisation progressive des contacts plans (insertions prismatiques en particulier). Ce point nécessite de résoudre des problèmes fondamentaux entièrement ouverts.

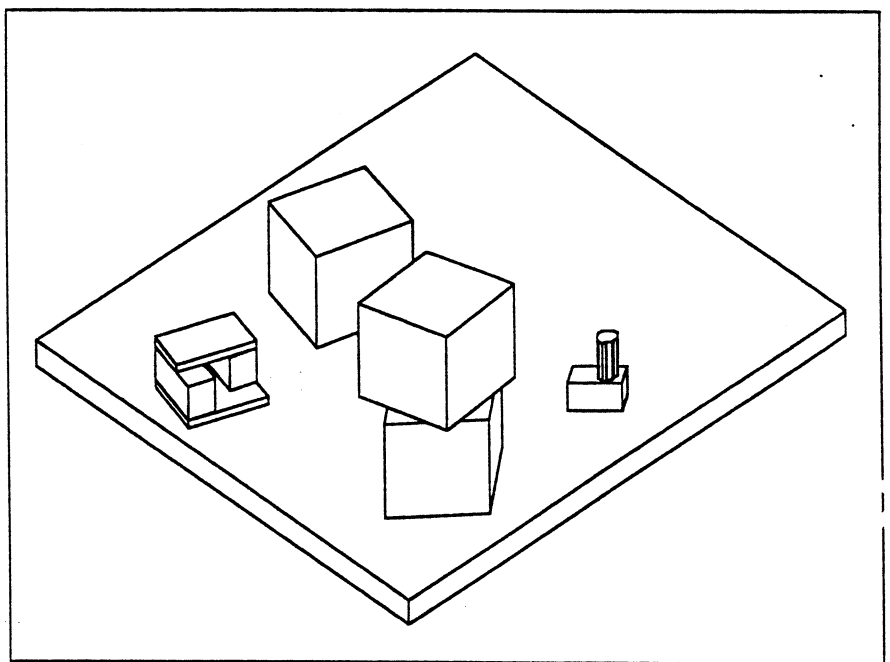
9.5 Exemple de tâche traitée par le système:

Dans cet exemple, les données initiales ont été entrées numériquement (le langage de description n'étant pas encore développé), et l'exécution des mouvements compliants est réalisée à l'aide d'une bibliothéque de fonctions écrites en langage LM (le langage géométrique décrit dans le paragraphe 4.2 n'étant pas encore entièrement implanté).

9.5.1 Description graphique de la tâche:



MONTAGE A REALISER

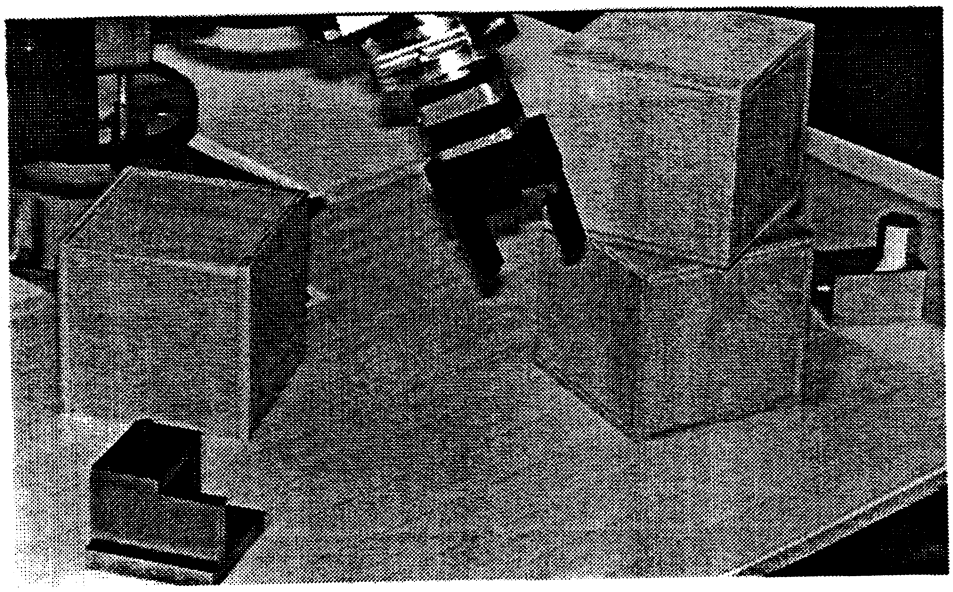


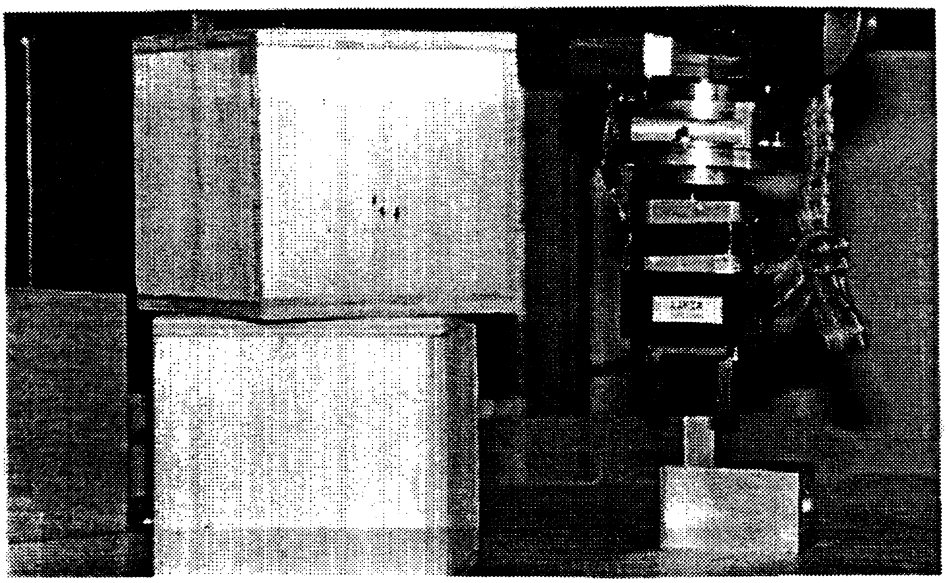
ETAT INITIAL

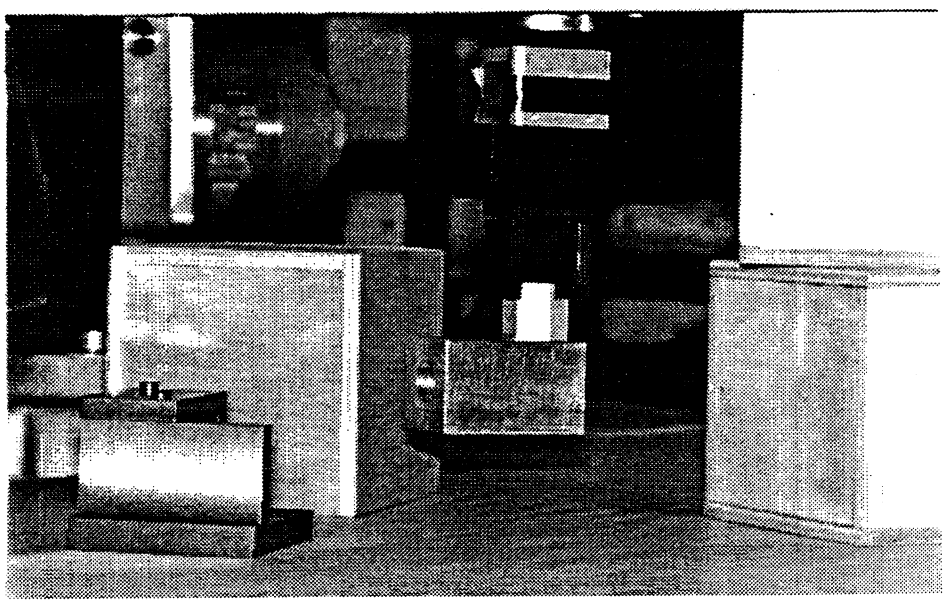
9.5.2 Forme du programme de manipulation engendré:

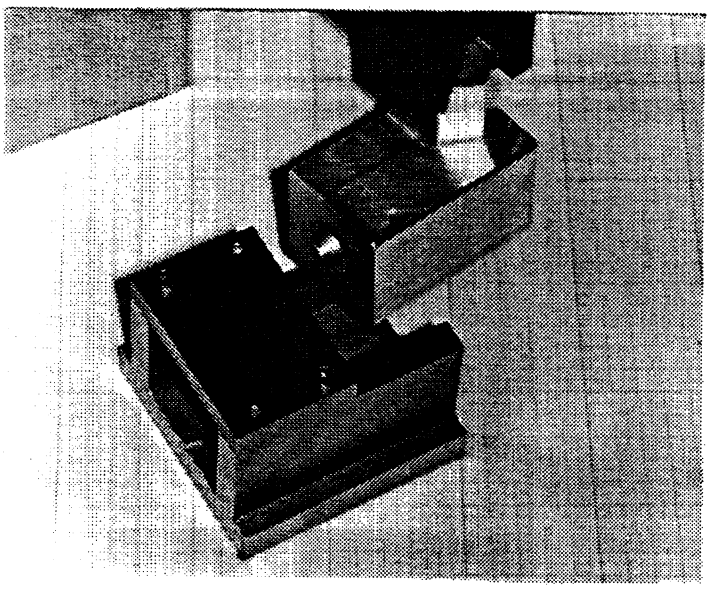
```
(REALISER-S
      (R-robot SUR R0)
      (VIA (C0 C1 C2 . . . . )))

(SAISIR
      (R-robot SUR R-prise0))

(REALISER-S
      (R-robot SUR R1))

(REALISER-S
      (R-robot SUR R2)
      (VIA (C00 C01 C02 . . . . )))

(REALISER-C
      (point-P1 SUR face-B1)
      (SUIVANT (TRANSLATION :VECT vecteur-V1)))

(REALISER-C
      (arete-A1 SUR face-B1)
      (SUIVANT (ROTATION :AXE(make-axe    :PT point-P1
                                          :VECT vecteur-V2)))
      (MAINTENIR (point-P1 SUR face-B1)))

(REALISER-C
      (face-F1 SUR face-B1)
      (SUIVANT (ROTATION :AXE(make-axe    :PT point-P1
                                          :VECT arete-A1)))
      (MAINTENIR (arete-A1 SUR face-B1)))

(REALISER-C
      (arete-A2 SUR face-B2)
      (SUIVANT (TRANSLATION :VECT vecteur-V3))
      (MAINTENIR (face-F1 SUR face-B1)))

(REALISER-C
      (face-F2 SUR face-B2)
      (SUIVANT (ROTATION :AXE(make-axe    :PT point-P2
                                          :VECT arete-A2)))
      (MAINTENIR (face-F1 SUR face-B1)(arete-A2 SUR face-B2)))

(REALISER-C
      (point-P3 SUR face-B3)
      (SUIVANT (TRANSLATION :VECT arete-A1))
      (MAINTENIR (face-F1 SUR face-B1)(face-F2 SUR face-B2)))

(LACHER
      (R-robot SUR R3))
```

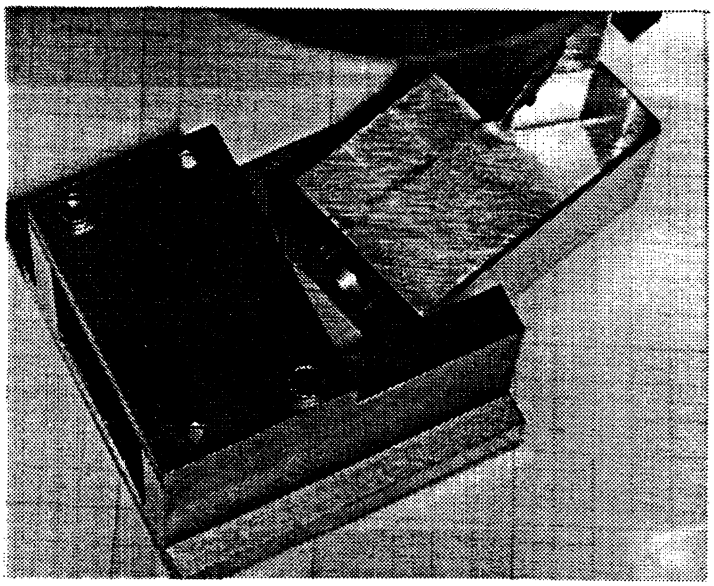
9.5.3 Quelques étapes du montage:

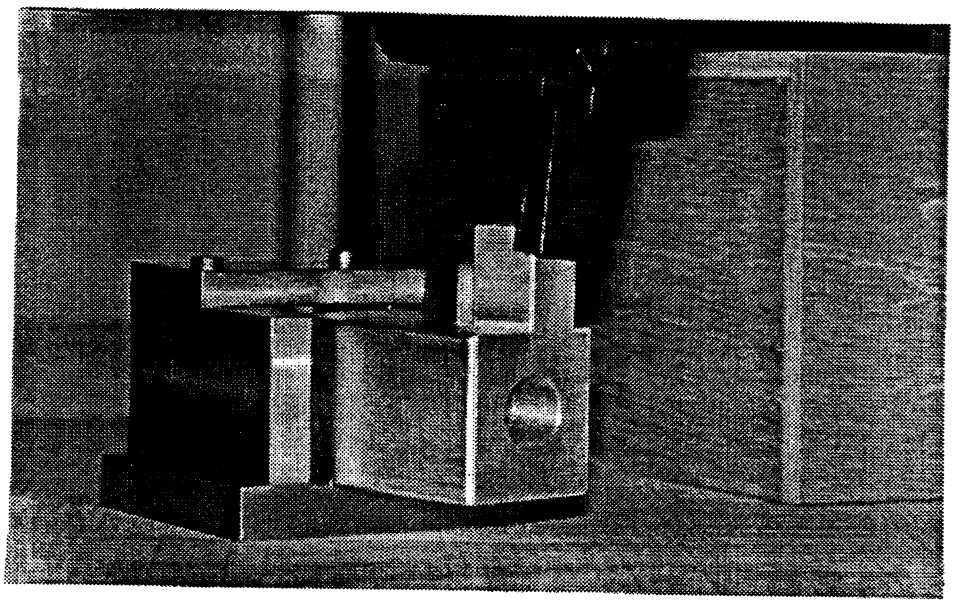


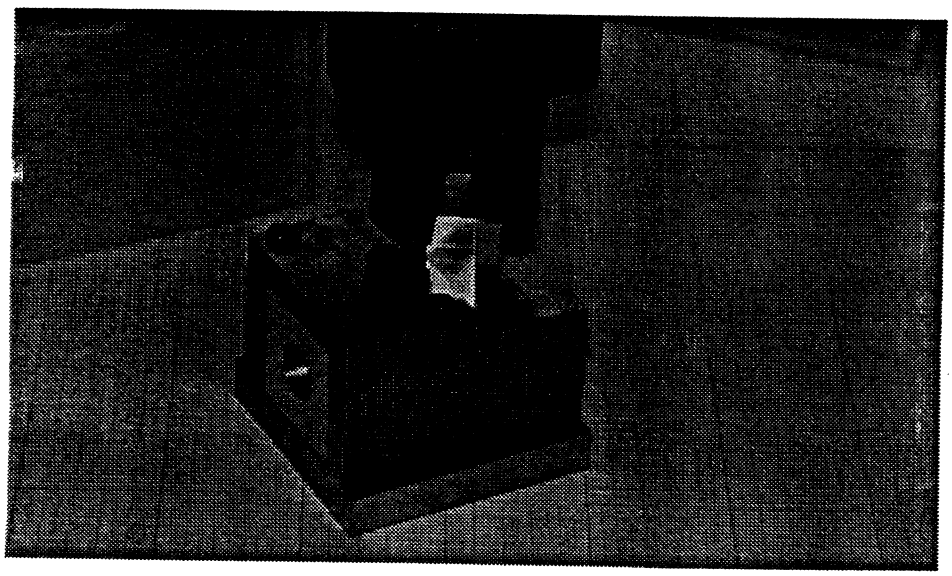












9.6 Vérification/correction de plans:

9.6.1 Présentation du problème:

Le problème posé consiste à s'assurer que le plan d'actions engendré par le module de planification, vérifie toutes les contraintes d'incertitudes imposées par la tâche. Ce problème vient de ce que les variables de position spécifiées dans le plan, ne sont jamais connues de manière exacte lors de la planification. Le système est donc amené à raisonner sur des valeurs nominales, auxquelles sont associées des termes d'erreurs. Une variable X est alors représentée par un couple (X^*, ε), dans lequel X^* est la position nominale de l'objet considéré, et ε est le terme d'erreur. Seule la valeur X^* est utilisée par le module de planification. La fonction de base du système de vérification/correction est donc de déterminer toutes les valeurs possibles des variables X (c'est à dire des couples (X^*, ε)), afin de vérifier que ces valeurs sont "compatibles" en tout point du plan avec les contraintes imposées par la tâche.

Dans un formalisme ensembliste, nous dirons que X prend ses valeurs dans un ensemble $Poss_i(X^*)$, après que la i-ème action du plan ait été exécutée. Une contrainte $C_i(X^*)$ portant sur la variable X, spécifie alors l'erreur maximum qui peut être commise relativement à X^*, sans risquer de mettre en échec l'opération projetée. Par exemple, l'erreur sur la position de la pièce à assembler doit être inférieure aux jeux de montage. Nous dirons alors que X *doit* appartenir à un ensemble $Adm_i(X^*)$. Une formulation intuitive du problème de vérification peut ainsi être énoncée comme suit: *l'action A_i est "compatible" avec les contraintes imposées par la tâche, ssi l'ensemble $Poss_i(X^*)$ est inclus dans l'ensemble $Adm_i(X^*)$, pour chaque variable X modifiée par A_i.*

Tous le problème consiste alors à calculer les ensembles $Poss_i(X^*)$. La principale difficulté provient de ce que ces ensembles dépendent de la nature des actions exécutées antérieurement. Par exemple, l'erreur portant sur la position X d'un objet saisi et transporté par le robot, comportera un terme lié à l'incertitude initiale de position de la pince par rapport à l'objet, et un terme lié à l'imprécision de la commande de mouvement. Le mécanisme de calcul associé devra donc procéder par *propagation* de ces données au travers des actions du plan.

Les bases formelles de ce problème sont développées dans [Troccaz, Puget 87] et dans [Puget 88]. Les paragraphes qui suivent décrivent le principe des calculs mis en œuvre.

9.6.2 Modélisation des actions du robot:

Caractéristiques des actions traitées par le système:

Les actions de manipulation utilisées dans la spécification du plan sont celles définies dans le chapitre 4. Elles sont toutes exprimées dans un formalisme géométrique qui présente un niveau sémantique plus riche et plus homogène que celui offert par les langages traditionnels de programmation de robots. Deux objectifs ont guidé le développement de ce formalisme: éviter les ambiguïtés d'interprétation des situations rencontrées par le robot, et décrire des opérations dont les effets sont prédictibles en termes de positions et d'incertitudes. La première propriété est nécessaire à l'exécution des fonctions "d'apprentissage" décrites dans le chapitre 4; l'autre est nécessaire à la conduite des raisonnements mis en œuvre par le processus de vérification/correction.

Compte tenu des modèles incomplets que nous possédons, la propriété de "prédictibilité" ne peut être vérifiée que par des actions simples ayant les caractéristiques suivantes:

- L'action possède un *objectif unique*, exprimé sous la forme d'une relation géométrique ou d'une relation de contact. Dans le premier cas, il s'agit de déplacer le robot vers une position de l'espace libre; dans l'autre cas, le système doit engendrer un mouvement ayant pour objet de créer ou de détruire le (les) contact(s) spécifié(s).

- L'action ne met en œuvre *aucune opération compliante*, conduisant à glisser sur des surfaces d'appuis.

Les actions actuellement prises en compte par le mécanisme de vérification/correction, se limitent donc aux opérations suivantes: déplacement dans l'espace libre, saisie et lâcher d'objets, localisation d'objets. La structure des plans traités est alors globalement linéaire. Les constructions plus complexes associées aux stratégies de mouvements fins, sont dans ce cas considérées comme "globalement correctes". Cette approche est cohérente avec le fait que ces stratégies ont toutes été engendrées sur la base d'une analyse locale des incertitudes liées aux opérations concernées. De telles constructions sont considérées par le système comme des *macro-actions*, permettant de réaliser des relations d'assemblage avec une incertitude finale inférieure aux jeux de montage.

Principe de modélisation:

Le formalisme géométrique utilisé pour décrire les actions de manipulation s'appuie sur des éléments simples du modèle de l'environnement (points, droites, plans,

faces, repères). Nous définirons la sémantique d'une action par les modifications que cette action apporte sur deux composantes du modèle (cf. paragraphe 6.2):

- Les positions (positions nominales et incertitudes associées) des éléments géométriques impliqués dans la description.
- Les relations spatiales qui lient ces éléments dans le modèle.

Plusieurs facteurs interviennent dans la spécification de ces modifications: la précision avec laquelle le robot exécute ses déplacements, la précision de mesure des capteurs utilisés, et la transformation des relations de contact (création ou suppression).

Prenons par exemple une action de déplacement de type "*REALISER A SUR B*". On note $T(R_i/R_j)$ la position du repère R_i par rapport au repère R_j, et $I(R_i/R_j)$ l'incertitude de position associée. Cette action a pour effet de modifier le modèle comme suit (cf. figure 9.9):

$$T(Robot/Base) = T(R_b/Base) * T(R_a/R_b) * T^{-1}(R_a/Robot)$$
$$I(Robot/Base) = I_{dep}$$

où *base*, *Robot*, R_a et R_b sont respectivement les repères de référence de l'univers, de la pince du robo , de l'objet contenant l'entité A, et de l'objet contenant l'entité B; I_{dep} est l'incertitude de position liée à la commande de mouvement. Ces expressions sont calculées à l'aide des fonctions décrites dans le paragraphe 5.2.

Dans le cas où l'action met en jeu des contacts, les modifications induites se traduisent par une projection de volume d'incertitude, et par une modification des structures spatiales associées (cf. paragraphe 5.2). Par exemple, l'action "*SAISIR <prise>*" conduira à appliquer les opérations suivantes sur le modèle (cf. figure 9.10):

- Suppression de l'arc $ARC(Base, R_a)$ et création de l'arc $ARC(Robot, R_a)$.
- Calcul des paramètres de $ARC(Robot, R_a)$:
  $T(R_a/Robot) = T^{-1}(Robot/Base) * T(R_a/Base)$
  $I(R_a/Robot) = Proj(I_{dep}^{-1} * I(Ra/Base),\ (E, V, B))$
  avec:
  E= plan des mors, V= vecteur normal aux mors, $B = [-\pi \quad +\pi]$

Une action perceptive destinée à localiser une pièce A dans le champ de la caméra, conduira à modifier les paramètres de position et d'incertitude associés à A. Cette modification est réalisée à l'aide de l'opérateur de "conjonction d'incertitude" décrit dans le paragraphe 6.2. Elle pose cependant quelques problèmes théoriques qui seront développés dans [Puget 88].

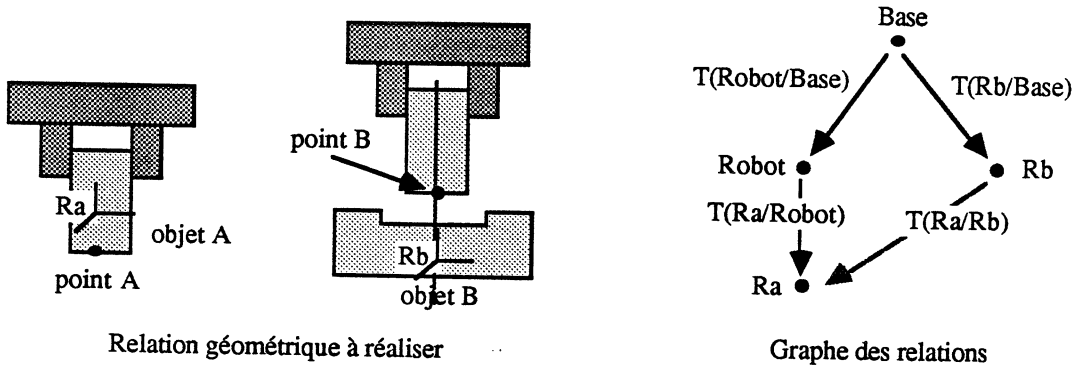


Figure 9.9: Modifications apportées au modèle par une action de déplacement du type: *REALISER A SUR B.*

9.6.3 Principe de vérification/correction:

Concept de plan contraint:

Conformément aux hypothèses énoncées précédemment, un plan de manipulation peut être vu comme une séquence d'actions A_1, $A_2 \cdots A_n$, conduisant à faire passer progressivement l'univers du robot d'un état initial E_0 à un état final E_n:

$$E_0 \xrightarrow{A_1} E_1 \xrightarrow{A_2} E_2 \cdots E_{n-1} \xrightarrow{A_n} E_n$$

Chaque état se différencie du précédent par les modifications apportées aux éléments du modèle, c'est-à-dire aux couples (position nominale, incertitude). Les actions du plan sont celles définies dans le paragraphe précédent (actions simples et macro-actions).

Une première nécessité induite par le problème de vérification, consiste à faire apparaître explicitement toutes les contraintes de la tâche au niveau de la spécification du plan. Ces contraintes représentent des conditions qui doivent impérativement être vérifiées par les variables de position (terme nominal et/ou terme d'incertitude), pour que les actions qui suivent soient exécutables, ou pour

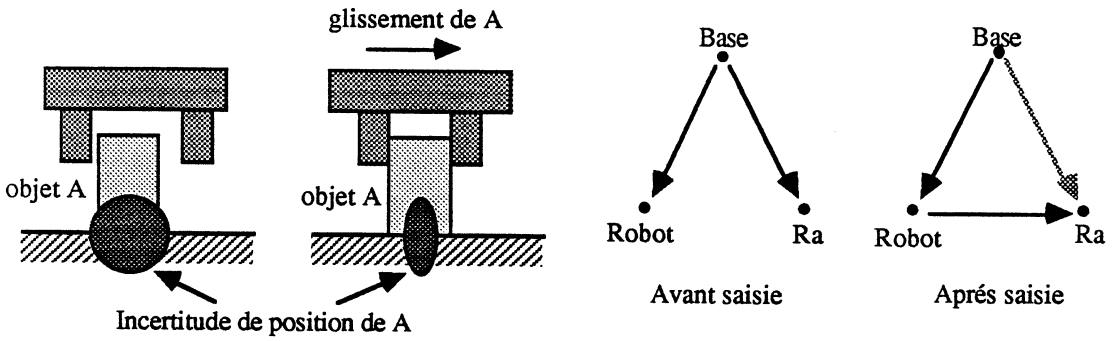


Figure 9.10: Modification des incertitudes de position à la suite d'une opération de saisie.

que certains objectifs de la tâche soient considérés comme atteints.

Chaque contrainte formulée s'exprime relativement à un état E_k particulier. Elle porte de ce fait sur la valeur dans E_k, des variables de position concernées. Si $I(R_a/R_b)$ est une incertitude définie par une expression conforme au formalisme développé, l'évaluation de cette expression dans l'état E_k donne un volume d'incertitude que nous noterons $I_k(R_a/R_b)$. La formulation d'une contrainte portant sur les positions relatives dans E_k des objets A et B, pourra alors être représentée par les expressions suivantes:

$$T(R_a/R_b) = t_0$$
$$I_k(R_a/R_b) < I_0 \qquad \text{avec:} \quad I_0 = Tr_0 * U_0 * A_0$$

où $T(R_a/R_b)$ et t_0 sont les transformations géométriques qui caractérisent les positions nominales de R_a par rapport à R_b, et la relation d'ordre utilisée pour comparer les incertitudes est définie comme suit:

$$I_1 > I_2 \iff (Tr_1 \subset Tr_2) \wedge (U_1 \subset U_2) \wedge (A_1 \subset A_2)$$

Définition 9.6.1 *Soit CO_k $(k = 0 \cdots n)$ les contraintes imposées dans l'état E_k. Nous appelons "plan contraint" toute séquence alternée d'actions et de contraintes portant sur les variables de position modifiées par ces actions. Nous représenterons*

un tel plan par l'expression suivante:

$$[CO_0]\, A_1\, [CO_1]\, A_2 \cdots [CO_{n-1}]\, A_n\, [CO_n]$$

Définition 9.6.2 *Un plan contraint est déclaré correct relativement à un état initial E_0, ssi toutes les contraintes de CO_k, pour k variant de 0 à n, sont satisfaites dans chaque état E_k résultant de l'exécution des actions $A_1 \ldots A_{k-1}$.*

Propagation descendante des contraintes et vérification du plan

On note C_i les conditions vérifiées dans l'état E_i, par les variables de position utilisées dans le plan. Ces conditions résultent de l'exécution des actions $A_1 \cdots A_{i-1}$; elles peuvent aussi dériver directement de conditions présentes dans l'état initial E_0. Par exemple, l'incertitude de position $I(R_a/Base)$ d'un objet A peut provenir des mécanismes d'alimentation employés pour placer A, d'une localisation par un capteur, de manipulations réalisées par des actions antérieures, ou encore d'un mélange des trois.

Afin de calculer les C_i, le système est donc amené à propager symboliquement les contraintes d'incertitudes de l'état E_0 vers l'état E_n. Ce mode de propagation est appelé *propagation descendante.* Il conduit à calculer à chaque étape les plus fortes conditions qui portent sur les variables de position. Ces conditions sont alors directement dérivées de la sémantique de l'action exécutée, et des conditions qui étaient antérieurement vérifiées. Nous parlerons de *plus fortes postconditions* $POST_i(C)$, associées aux conditions C et à l'action A_i. Par exemple, la condition "$I(R_a/Base) < i_0$" sera transformée en "$I(R_a/Base) < i_0 + I_{dep}$", par l'action *"DEPLACER A DE T"*. Dans le cas des actions de manipulation que nous avons définies, ce calcul est réalisé sur la base des expressions décrites précédemment.

Le procédé de propagation permettant de calculer les C_i est alors le suivant:

C_0 = Conditions initiales de E_0
Pour $i = 1 \cdots n$: $\quad C_i = POST_i(C_{i-1})$

Le plan contraint est déclaré correct, ssi toutes les conditions de C_i impliquent celles de CO_i, pour i variant de 0 à n.

Le procédé de calcul sur lequel repose le processus de vérification de plans, permet de garantir qu' un plan déclaré correct, l'est réellement (le calcul des plus fortes postconditions étant toujours basé sur les hypothèses les plus défavorables en ce qui concerne les erreurs). Il peut, par contre conduire à déclarer incorrect un plan qui ne l'est pas, à cause des défauts de modélisation et des phénomènes de compensation d'erreurs. Il est à noter que la propriété de "correction" ne signifie pas que le plan est réellement exécutable (cf. paragraphe 9.1).

Propagation ascendante des contraintes et amendement du plan

Le principe de la propagation ascendante des contraintes consiste à "régresser" dans le plan, afin d'établir les contraintes qui devraient être vérifiées dans les états antérieurs pour que l'échec rencontré soit levé. Ce mode de propagation repose sur le calcul des *plus faibles préconditions* $PRE_i(C)$" associées aux conditions C et aux actions A_i. En d'autres termes, il s'agit de déterminer à chaque étape les plus faibles conditions $PRE_i(C)$ qui permettent de garantir que C sera toujours vérifié après exécution de A_i. Par exemple, la condition "$I(R_a/Base) < i_0$" sera toujours vérifiée après exécution de l'action *"DEPLACER A DE T"*, si $I(R_a/Base)$ vérifie la condition "$I(R_a/Base)+I_{dep} < i_0$" dans l'état qui précède l'exécution de l'action. Comme précédemment, le mode de calcul des plus faibles préconditions représente une partie de la sémantique des actions de manipulation.

On note C_j^i, $j < i$, les contraintes de E_j dérivées de CO_i par le processus de propagation ascendant des contraintes. Ces contraintes garantissent si elles sont vérifiées, que la séquence $A_{j-1} \cdots A_i$ est "correcte" (et en particulier que les contraintes de CO_i sont satisfaites). Elles sont calculées en appliquant l'algorithme suivant:

$C_i^i = CO_i$
Pour $j = i-1 \cdots 0$: $\quad C_j^i = PRE_{j-1}(C_{j-1}^i) \wedge CO_j$

Le procédé décrit ci-dessus permet de calculer les contraintes qu'il faudrait satisfaire dans les différents états antérieurs à celui où l'echec a été détecté. Il ne permet cependant pas de déterminer *où* corriger ni *comment* corriger.

Il est donc nécessaire de choisir dans un premier temps, l'état E_k qui paraît être le plus propice à la correction. Une possibilité consiste à rechercher pour chaque contrainte à satisfaire, l'endroit où la différence entre l'état souhaité et l'état obtenu est la plus facile à maîtriser. Il n'existe cependant (dans l'état actuel de nos connaissances) aucune règle précise qui permette de réaliser ce choix.

Le deuxième problème consiste à rechercher parmi les corrections possibles, celle qui semble être la mieux adaptée à la situation. Chaque correction apportée possède alors un caractère local. Elle se traduit, soit par une modification de l'état initial de certaines variables, soit par l'insertion d'une procédure de correction appelée "rustine". Le premier type de correction est appliqué en priorité lorsqu'il est possible; il conduit à contraindre un peu plus l'environnement initial de la tâche. L'autre type de modification se traduit par l'insertion d'une action ou d'une macro-action permettant de réduire certaines incertitudes; le choix réalisé est alors conditionné par plusieurs facteurs: conditions d'applicabilité de la rustine, nature de l'incertitude à réduire et contraintes à satisfaire. Cet aspect est encore mal maîtrisé.

Conclusion

Dans ce mémoire, nous avons montré comment les modèles et les raisonnements géométriques autorisés par les données CAO, pouvaient transformer le concept de programmation des robots.

Dans la première partie, nous avons abordé le problème sous l'angle des systèmes informatiques traditionnels, c'est à dire des systèmes ne possédant aucune capacité décisionnelle. Nous avons décrit pour cela deux formes de connexion CAO-robot, permettant respectivement de mettre en œuvre des robots sur des tâches de type "Prendre-Poser" (c'est à dire des tâches insensibles aux effets des erreurs de commande et de perception), et de programmer des opérations d'assemblage nécessitant d'exécuter des mouvements fins de montage. Les deux types de connexion que nous avons développés, possèdent les principales caractéristiques suivantes:

- Le premier type de connexion CAO-robot est orienté vers *la programmation graphique.* Il repose sur un simulateur graphique "complet", permettant d'interfacer l'opérateur et un langage de programmation de robot (le langage LM). Quatre fonctions ont été développées dans ce but: une fonction de simulation associée à la cinématique du robot, une fonction de visualisation graphique permettant de produire des images de synthèse, une fonction de simulation de l'univers physique, et une fonction permettant de décrire graphiquement les mouvements à faire exécuter par le robot. Ce simulateur a été testé sur différents exemples. Un film de synthèse a ainsi été réalisé.

- Le deuxième type de connexion CAO-robot est orienté vers *la programmation géométrique.* Il repose sur un langage géométrique et sur un mécanisme

"d'apprentissage interactif". Le langage a été conçu de manière à décrire les opérations du robot, en termes de relations simples portant sur les éléments du modèle géométrique. Une particularité importante de ce langage, est de pouvoir spécifier des mouvements qui mettent en jeu des contacts (et donc des forces de réaction). Le mécanisme d'apprentissage a été conçu comme une extension du langage. Il permet de programmer une tâche de montage, sans qu'il soit nécessaire d'évaluer exactement toutes les contraintes d'incertitudes liées à cette tâche. Il procède pour cela à une phase d'expérimentation en-ligne, suivie d'une phase d'induction permettant de faire la synthèse des informations recueillies au cours des différents essais. Certains aspects liés à la gestion des forces n'ont pas encore été implantés dans l'interpréteur du langage, ce qui limite la portée des expérimentations que nous avons faites.

Les principales contributions que nous avons apportées dans le contexte de la connexion CAO-robot sont les suivantes:

1. Réalisation d'une véritable connexion entre un langage de programmation de robot et des bases de données CAO.

2. Intégration d'un mécanisme de simulation des capteurs et des événements de nature physique.

3. Développement d'un langage géométrique permettant de décrire les actions du robot en termes de relations géométriques et de contacts.

4. Développement d'un mécanisme d'expérimentation en-ligne, facilitant la détection et la correction des échecs dus aux incertitudes.

Dans la deuxième partie du mémoire, nous avons abordé le problème de la programmation des robots sous l'angle de l'Intelligence Artificielle. Il s'agit alors d'automatiser le processus de programmation, en dotant le robot des capacités d'analyse et de planification qui lui font défaut. Après une étude détaillée des problèmes posés, nous avons décrit les modèles et les types de raisonnement sur lesquels reposent ces nouvelles fonctions:

- *Le modèle de l'univers du robot* que nous avons développé, intègre des données de différentes natures. Il combine des représentations qui portent sur la géométrie des objets, sur leurs relations physiques et spatiales, et sur les incertitudes de position qui leur sont associées. Il comporte également une modélisation des structures cinématiques, et des données mécaniques qui interviennent dans l'exécution des mouvements du robot (forces, frottements et type de commande). Ces données ont été utilisées pour implanter les

algorithmes de calculs géométriques, sur lesquels reposent les fonctions de planification.

- *Le raisonnement spatial* a pour objet de planifier les mouvements du robot. Il a été développé sur la base de fonctions permettant de calculer des débattements valides, d'exécuter des calculs de grossissements d'obstacles, de construire une représentation approchée de l'espace libre, et de rechercher des solutions dans cet espace.

- *Le raisonnement morphologique* a pour objet de planifier les opérations de robot qui mettent en jeu des contacts (saisie et mouvements fins). Il a été développé comme une "extension" du raisonnement spatial, afin de réduire la complexité algorithmique inhérente à ce type de raisonnement. Il opère comme un premier filtre, permettant d'exhiber des solutions potentielles. Ces solutions sont alors établies sur la base d'une analyse des propriétés locales des contacts. Les méthodes que nous avons développées ont donc été orientées d'une part vers l'analyse des propriétés topologiques des contacts, et d'autre part vers la détermination des contraintes de mouvements imposées par ces contacts.

Toutes les méthodes décrites ont été implantées, avec les quelques limitations énoncées dans le texte.

Le dernier chapitre de la thèse montre comment toutes les techniques présentées peuvent être combinées, en vue de constituer un système de programmation automatique de robots appelé SHARP. Une architecture basée sur un mécanisme de propagation de contraintes a été proposée. Elle repose sur des développements partiels, réalisés dans le contexte de la vérification/correction de plans. La description des modules de planification qui a été donnée, s'appuie par contre sur des réalisations concrètes. Quelques résultats d'expérimentations ont ainsi été décrits.

Les principales contributions que nous avons apportées dans le contexte de la programmation automatique, sont les suivantes:

1. Unification des approches liées à la planification des mouvements d'un robot. Nous avons formalisé pour cela les concepts de "raisonnement spatial" et de "raisonnement morphologique", ainsi que le rôle joué par chacun dans le processus de programmation.

2. Développement des algorithmes de base nécessaires à la conduite des raisonnements précédents. Nous avons décrit le principe de ces algorithmes, ainsi que les termes de complexité qui leur sont associés.

3. Développement d'une forme de raisonnement basée sur une représentation explicite des mouvements potentiels associés aux contacts. Cette technique a été utilisée pour planifier des séquences de mouvements fins de montage.

4. Proposition d'une architecture intégrée pour un système de programmation automatique de robots. Cette architecture repose sur un processus de filtrage réparti dans les modules de planification, un mécanisme élémentaire de backtrack, un procédé de vérification/correction de plan, un langage de description incluant des constructions pour spécifier les contraintes d'incertitudes, et un langage géométrique interprétable par le robot.

Cette thèse montre que la réalisation d'un système de programmation automatique de robots n'est pas une utopie, bien que l'état actuel de nos connaissances conduit à apporter de sévères restrictions sur la classe des tâches que l'on peut envisager de traiter. Il est donc plus raisonnable dans un proche avenir, de considérer le problème sous l'angle d'un système semi-automatique, c'est à dire d'un système capable de proposer interactivement des solutions aux trois sous-problèmes de base (saisie, transfert, montage). Le rôle de l'opérateur serait alors de valider les choix proposés, et de guider le système en cas d'échecs.

Cette remarque "pratique", ne remet évidemment pas en cause les objectifs de recherche que nous nous sommes fixés. Ces objectifs conduisent à approfondir et à améliorer les méthodes que nous avons développées, et à aborder des problèmes peu ou mal traités tels que l'intégration des rotations dans le processus de planification (orientations du poignet et mouvements fins basés sur des rotations correctives en particulier), l'amendement automatique de plans de manipulation, ou l'apprentissage en robotique.

Un autre aspect que nous avons commencé à aborder, consiste à supprimer l'hypothèse de "monde clos" (c'est à dire d'un univers entièrement modélisé, dans lequel le seul acteur est le robot). Bien que cette hypothèse s'applique à la plupart des applications manufacturières, elle s'avère totalement inacceptable dans les autres domaines d'application (espace, sous-marin, nucléaire, agriculture, service ...). Il devient alors essentiel de baser le processus de planification sur les mécanismes de perception. Ceci nécessite de remettre en cause certaines techniques que nous avons développées dans le cadre de la programmation automatique. En particulier, les méthodes "globales" de planification de trajectoires sont très mal adaptées à ce nouveau contexte. Par contre, le procédé de saisie automatique que nous avons développé se prête bien au couplage avec des données sensorielles [Troccaz 87]. Ces travaux viennent en complément des recherches mentionnées précédemment.

Bibliographie

[André, Boulic 85] G.André, R.Boulic: *"CAO de systèmes multicapteurs et simulation de commandes basées sur la perception de l'environnement en robotique"*, Colloque AFRI-MICADO sur la CAO-Robotique, La Grande - Motte, Janvier 1985.

[Ahuja et al. 80] N.Ahuja, R.T.Chien, N.Bridwell: *"Interference detection and collision avoidance among three dimensional objects"*, 1st American Association for Artificial Intelligence conference, Stanford, Août 198O.

[Alami, Chochon 85] R.Alami, H.Chochon: *"Programming of Flexible Assembly Cells: task modeling and system integration"*, IEEE International Conference on Robotics and Automation, Saint-Louis, Avril 1985.

[Ambler, Popplestone 75] A.P.Ambler, R.J.Popplestone: *"Inferring the positions of bodies from specified spatial relationships"*, Artificial Intelligence 6(2), 1975.

[Arai 83] T.Arai: *"A robot language system with colour graphic simulator"*, International Meeting on Advanced Software in Robotics, Liège, Mai 1983.

[Bansard 83] J.P.Bansard: *"Le système LM-EX"*, IMAG, Grenoble, Juin 1983.

[Baumeister 78] T.Baumeister: *"Marks Standard Handbook of Mechanical Engineers"*, Mc Graw-Hill, 1978.

[Binford 79] T.O.Binford: *"Computer integrated assembly systems"*, Artificial Intelligence Laboratory, Stanford, 1979.

[Binford 82] T.O.Binford: *"Survey of model-based image analysis systems"*, International Journal of Robotics Research, 1(1), 1982.

[Birk et al. 81] J.R.Birk et al.: *"An orienting robot for feeding workpieces stored in bins"*, IEEE Transactions on System, Man and Cybernetics, vol. SMC-11, Février 1981.

[Boissonnat 82] J.D.Boissonnat: *"Stable matching between a hand structure and an object silhouette"*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-4, no 6, Novembre 1982.

[Bolle, Cooper 86] R.M.Bolle, D.B. Cooper: *"On optimally combining pieces of information, with application to estimating 3D complex object position from range data"*, IEEE Trans. Pattern Anal. Machine Intell., PAMI-8, pp. 619-638, Sept. 1986.

[Bond, Mott 81] A.H.Bond, D.H.Mott: *"Learning of sensory-motor schemas in a mobile robot"*, IJCAI-7, Vancouver, 1981.

[Borrel et al. 83] P.Borrel et al.: *"The robotics facilities in the CAD/CAM CATIA system"*, édité par Brooks, IFS Publications, 1983.

[Boyse 79] J.W.Boyse: *"Interference detection among solids and surfaces"*, CACM, vol.22, nb.1, Janvier 1979.

[Bo Zhang et al. 84] Bo Zhang et al.: *"Planning collision-free paths for robotic arm among obstacles"*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol PAMI-6, no 1, Janvier 1984

[Brady et al. 82] J.M.Brady et al.: *"Robot Motion: Planning and Control"*, MIT Press, Cambridge, MA, 1982.

[Brooks 82a] R.A.Brooks: *"Solving the find-path problem by good representation of free space"*, 2nd AAAI conference, Carnegie Mellon University, Août 1982.

[Brooks 82b] R.A.Brooks: *"Symbolic error analysis and robot planning"*, International Journal of Robotics Research, vol 1, no 4, Decembre 1982.

[Brooks 83] R.A.Brooks: *"Planning collision free motions for pick and place operations"*, International Journal of Robotics Research, vol 2, no 4, Hiver 1983/ 1st International Symposium of Robotics Research, Bretton Woods, Août 1983.

[Brooks, Lozano-Perez 83] R.A.Brooks, T.Lozano-Perez: *"A subdivision algorithm in Configuration Space for findpath with rotation"*, International Joint Conference on Artificial Intelligence, Karlsruhe, Août 1983.

[Brooks 84] R.A.Brooks: *"Aspects of mobile robot visual map making"*, 2nd International Symposium of Robotics Research, Kyoto, Août 1984.

[Brost 85] R.C.Brost: *"Planning robot grasping motions in the presence of uncertainty"*, Technical Report CMU-RI-TR-85-12, Carneggie Mellon University, The Robotics Institute, Pittsburgh, Juillet 1985.

[Buckley 87] S.J.Buckley: *"Planning and teaching compliant motion strategies"*, Ph.D Thesis, Artificial Intelligence Laboratory, MIT, Janvier 1987.

[Chatila 81] R.Chatila: *"Système de navigation pour un robot mobile autonome: modélisation et processus décisionnels"*, Thèse de Docteur Ingénieur, Toulouse, Juillet 1981.

[Chatila, Laumond 85] R.Chatila, J.P.Laumond: *"Position referencing and consistent world modeling formobile robots"*, Proc. IEEE Int. Conf. Robotics and Automation, St.Louis, 1985.

[Chochon 86] M.Chochon: *"Programmation et conduite de tâches d'assemblage robotisées: modélisation et processus décisionnels"*, Thèse de l'Université Paul Sabatier, Toulouse, Novembre 1986.

[Cohen, Feigenbaum] P.R.Cohen, E.A.Feigenbaum: *"The handbook of Artificial Intelligence"*, Tome 3, pp 513/563, William Kaufmann Inc.

[Crowley 85] J.L.Crowley: *"Navigation for an Intelligent Mobile Robot"*, IEEE Journal of Robotics and Automation, 1(1), Mars 1985.

[Demazeau 85] Y.Demazeau: *"La programmation des jeux: Programmation classique et Intelligence Artificielle"*, Rapport de Recherche LIFIA/IMAG no. 502, Avril 1985.

[Descotte, Latombe 84] Y.Descotte, J.C.Latombe: *"GARI: an expert system for process planning"*, in "Solid modeling by computers: from theory to applications", edited by J.W.Boyse and M.S.Pickett, Plenum Press, New-York, 1984.

[Dillmann, Huck 85] R.Dillmann, M.Huck: *"Intelligent simulation of robot application"*, Symposium on Robot Control 85, Barcelona, Novembre 1985.

[Donald 84] B.R.Donald: *"Motion Planning with Six Degrees of Freedom"*, AI-TR-791. Cambridge, Mass.: Massachusetts Institute of Technology Artificial Intelligence Laboratory, 1984.

[Donald 86] B.Donald: *"Robot motion planning with uncertainty in the geometric models of the robot and environnement: a formal framework for error detection and recovery"*, PROC. IEEE Int. Conf. on Robotics and Automation, San Francisco, Avril 1986.

[Dooner et al. 82] M.Dooner, N.K.Taylor, M.C.Bonney: *"Planning robot installations by CAD"*, Computer-Aided Design Conference, Brighton, Mars 1982.

[Doran 70] J.Doran: *"Planning and robots"*, Machine Intelligence, no. 5, 1970.

[Drysdale 79] R.L.Drysdale: *"Generalized voronoi digrams and geometric searching"*, Department of Computer Science, Stanford University, 1979.

[Dufay 83] B.Dufay: *"Apprentissage par induction enRobotique: Application à la synthèse de programmes de montage"*,Thèse de 3ème cycle, INPG, Grenoble, Juin 1983.

[Dufay, Latombe 84] B.Dufay, J.C.Latombe: *"An approach to automatic robot programming based on inductive learning"*, Rapport de recherche IMAG no 433, Février 1984/ Publié aussi dans:1st International Symposium on Robotics Research, Bretton Woods, Août 1983.

[Erdmann 84] M.Erdmann: *"On motion planning withuncertainty"*, AI-TR-810, Artificial Intelligence Laboratory, MIT,1984.

[Fahlmann 74] S.E.Fahlmann: *"A planning system for robot construction tasks"*, Artificial Intelligence Journal, vol. 5, 1974.

[Faugeras, Hebert 86] O.D.Faugeras,M.Hebert: *"The representation, Recognition, and locating of 3D Objects"*, Int. Journal of Robotics Research, vol.5, nb.3, 1986.

[Farreny 80] H.Farreny: *"Un système pour l'expression et la résolution de problèmes orientés vres le contrôle de robots"*, Thèse d'état, Université Paul Sabatier, Toulouse, Septembre 1980.

[Faverjon 84] B.Faverjon: *"Obstacle avoidance using anoctree in the configuration space of a manipulator"*, IEEE International Conference on Robotics and Automation, Atlanta, Mars 1984.

[Faverjon 86] B.Faverjon: *"Object level programming of industrial robots"*, IEEE International Conference on Robotics and Automation, San Francisco, Avril 1986.

[Faverjon, Tournassoud 87] B.Faverjon, P.Tournassoud: *"A local based method for path planning of manipulators with a high number of degrees of freedom"*, IEEE International Conference on Robotics and Automation, Raleigh, Avril 1987.

[Fearing 84] R.S.Fearing: *"Simplified grasping and manipulation with dextrous robot hands"*, AI-Memo-809, Artificial Intelligence Lab., M.I.T., Cambridge, Novembre 1984.

[Feldman et al. 71] J.Feldman et al.: *"The stanford Hand-Eye project"*, Proceedings IJCAI 71, London, England, Septembre 1971.

[Fikes, Nilsson 71] R.E.Fikes, N.J.Nilsson: *"STRIPS: A new approach to the application of theorem proving to problem solving"*, Artificial Intelligence Journal, no 2 (3/4), 1971.

[Fikes et al. 72] R.E.Fikes, P.E.Hart, N.J.Nilsson: *"Learning and executing generalized robot plans"*, Artificial Intelligence Journal, no 3, 1972.

[Finkel et al. 74] R.A.Finkel et al.: *"AL, a programming system for automation"*, Memo AIM 243, Artificial Intelligence Laboratory, Stanford, Novembre 1974.

[Finkel 76] R.A.Finkel: *"Constructing and debugging manipulator programs"*, AIM 284, Artificial Intelligence Laboratory, Stanford University, Août 1976.

[Franklin, Vanderbrug 82] J.W.Franklin, G.J.Vanderbrug: *"Programming vision and robotics systems with RAIL"*, SME Robots VI, Mars 1982.

[Gandon 86] C.Gandon: *"Introduction de la compliance dans la programmation des robots"*, Thèse de 3ème Cycle, INPG, Grenoble, Octobre 1986.

[Germain 84] F.Germain: *"Planification de trajectoires sans collision"*, Rapport de DEA, INPG, Juin 1984.

[Germain 85] F.Germain: *"Génération statique de trajectoires: un algorithme paramètrable par le contexte"*, 5ème Congrès AFCET Reconnaissance des Formes et Intelligence Artificielle, Grenoble, Novembre 1985.

[Gini 78] G.Gini, M.Gini: *"Object representation with a manipulator"*, The Industrial Robot, Mars 1978.

[Giralt 83] G.Giralt: *"Mobile robots"*, NATO Advanced Study Institute on Robotics and Artificial Intelligence, Castelvecchio Pascoli, Barga, Juin 1983.

[Giraud 83] M.Giraud: *"Generalized active compliance for part-mating with assembly robots"*, 1st International Symposium of Robotics Research, Bretton Woods, 1983.

[Gouzenes 84] L.Gouzenes: *"Strategies for solving collision-free trajectories problems for mobile and manipulator robots"*, International Journal of Robotics Reserch, vol 3, no 4, Hiver1984.

[Grossman 77] D.D.Grossman: *"Programming a computer controlled manipulator by guiding through the motions"*, IBM T.J.Watson Research Center, Research Report RC6393, 1977.

[Guinot 85] J.C.Guinot: *"Modelisation and simulation of force-position control for a manipulator-gripper"*, 3rd International Symposium of Robotics Research, Gouvieux, Septembre 1985.

[Hanafusa, Asada 77a] H.Hanafusa, H.Asada: *"Stable prehension by a robot hand with elastic fingers"*, 7th International Symposium on Industrial Robots Tokyo, Octobre 1977/ publié dans "Robot Motion: Planning and Control", édité par M.Brady et al., M.I.T. Press, 1982.

[Hanafusa, Asada 77b] H.Hanafusa, H.Asada: *"A robot hand with elastic fingers and its application to assembly process"*, IFAC Symposium on Information and Control Problems in Manufacturing Technology, Tokyo, 1977/ publié dans "Robot Motion: Planning and Control", édité par M.Brady et al., M.I.T. Press, 1982.

[Hayward, Paul 84] V.Hayward, R.P.Paul: *"Introduction to RCCL: a robot control C library"*, Proc. IEEE Internationnal Conference on Robotics, Atlanta, Mars 1984.

[Heginbotham et al. 79] W.B.Heginbotham, M.Dooner, K.Case: *"Rapid assessment of robot performance by interactive computer graphics"*, 9th International Symposium on Industrial Robots, Washington D.C., 1979.

[Hopcroft et al. 82] J.E.Hopcroft, D.A.Joseph, S.H.Whitesides: *"On the movements of robot arms in 2-dimensional regions"*, Department of Computer Science, Cornell University, TR 82-486, Mars 1982.

[Hopcroft et al. 84] J.E.Hopcroft, J.T.Schwartz, M.Sharir: *"On the complexity of motion planning for multiple independant objects; PSPACE-Hardness of the Warehouseman's problem"*, International Journal of Robotics Research, 3(4), 1984.

[Ijel 86] A.Ijel: *"Modélisation des aspects physiques de l'univers d'un robot"*, Rapport de DEA, INPG, Septembre 1986.

[Ikeuchi et al. 84] K.Ikeuchi et al.: *"Determining grasp points using photometric stereo and the PRISM binocular stereo system"*, AI-Memo-772, Artificial Intelligence Lab., M.I.T., Cambridge, Août 1984.

[Inoue 74] H.Inoue: *"Force feedback in precise assembly tasks"*, Artificial Intelligence Laboratory, MIT, AIM-308, Août 1974.

[Joyal, Provost 66] M.Joyal, P.Provost: *"Statique"*, Masson & Cie, 1966.

[Khatib 80] O.Khatib: *"Commande dynamique dans l'espace opérationnel des robots manipulateurs en présence d'obstacles"*,Thèse de Docteur-Ingénieur, Ecole Nationale Supérieure de l'Aéronautique et de l'Espace, Toulouse, Décembre 1980.

[Krogh 84] B.H.Krogh: *"A generalized potential field approach to obstacle avoidance control"*, Robotics Research: The Next Five Years and Beyond, SME, One SME Drive, Dearborn Mich, 1984.

[Lagoude, Tsang 85] Y.Lagoude, J.P.Tsang: *"A plan representation structure for expert planning systems"*, Symposium on Computer Aided Process Planning, Miami Beach, Florida, Novembre 1985.

[Latombe, Mazer 81] J.C.Latombe, E.Mazer: *"LM 1981: a high-level programming language for controlling assembly robots"*, 11th International Symposium on Industrial Robots, Tokyo, Octobre 1981.

[Latombe 82] J.C.Latombe: *"Survey of advanced general-purpose software for robot manipulators"*, RR n0. 330, IMAG, Grenoble, Novembre 1982.

[Latombe 83] J.C.Latombe: *"Automatic synthesis of robot programs from CAD specifications"*, NATO Advanced Study Institute on Robotics and Artificial Intelligence, EL Coccio, Italy, Juin 1983.

[Latombe et al. 84] J.C.Latombe, C.Laugier, J.M.Lefebvre, E.Mazer, J.F.Miribel: *"The LM robot programming system"*, 2nd International Symposium on Robotics Research, Kyoto, Août 1984.

[Laugier 79] C.Laugier: *"Visualisation d'objets tridimensionnels dans un contexte interactif"*, Rapport sur contrat DGRST n0 787-039, Juillet 1979.

[Laugier 81] C.Laugier: *"A program for automatic grasping of objects with a robot arm"*, 11th International Symposium on Industrial Robots, Tokyo, Octobre 1981.

[Laugier, Dufay 82] C.Laugier, B.Dufay: *"Geometrical reasoning in automatic grasping and contact analysis"*, PROLAMAT 82, Leningrad, Mai 1982.

[Laugier 82] C.Laugier: *"LISP-3D: logiciel graphique pour la manipulation et la visualisation de scènes tridimensionnelles"*, Rapport de Recherche IMAG, no 328, Septembre 1982.

[Laugier, Pertin 83] C.Laugier, J.Pertin: *"Automatic grasping: a case study in accessibility analysis"*, International Conference on Advanced Software in Robotics, Liège, Mai 1983/publié dans "Advanced Software in Robotics", édité par A.Danthine et M.Géradin, North Holland, 1984.

[Laugier 83] C.Laugier: *"Influence du raisonnement géométrique dans le choix d'une prise d'objet"*, Rapport de Recherche IMAG no. 414, Décembre 1983.

[Laugier et al. 84] C.Laugier, C.Evieux, J.Pertin-Troccaz: *"Un système de simulation graphique de robots incluant une gestion élémentaire des incidents et des capteurs"*, RR n0.421, IMAG, Grenoble, Mars 1984.

[Laugier, Pertin 84] C.Laugier, J.Pertin-Troccaz: *"Graphic simulation as a tool for debugging robot control programs"*, 1st International Symposium on Design and Synthesis, Tokyo, Juillet 1984.

[Laugier 84] C.Laugier: *"Robot programming using a high-level language and CAD facilities"*, Robotics Europe Conference, Brussels, Juin1984.

[Laugier, Germain 85] C.Laugier, F.Germain: *"An adaptative collision-free trajectory planner"*, '85 International Conference on Advanced Robotics, Tokyo, Septembre 1985.

[Laugier, Troccaz 85] C.Laugier, J.Troccaz: *"S.H.A.R.P.: A system for automatic programming of manipulation robots"*, 3rd International Symposium of Robotics Research, Gouvieux, Octobre 1985.

[Laugier 85] C.Laugier: *"Les Apports respectifs des langages symboliques et de la CAO en programmation des robots"*, Colloque "Robotique et CFAO", La Grande Motte, Fev. 1985.

[Laugier, Theveneau 86] C.Laugier, P.Theveneau: *"Planning sensor-based motions for part-mating using geometric reasonning"*, ECAI'86, Brighton, Juillet 1986.

[Laugier 87] C.Laugier: *"Traitement des incertitudes en programmation automatique des robots"*, Journées Robotique, INRIA, Sophia-Antipolis, Juin 1987.

[Lespinard 85] P.Lespinard: *"Problèmes liés à la programmation automatique des robots"*, Rapport de DEA, LIFIA/IMAG Laboratory, Septembre1985.

[Lewis 74] R.A.Lewis: *"Autonomous manipulation on a robot: summary of manipulator software functions"*, Jet Propulsion Laboratory, California Institute of Technology, TM 33-679, Mars 1974.

[Liebermann, Wesley 77] L.I.Liebermann, M.A.Wesley: *"AUTOPASS: an automatic programming system for computer controlled mechanical assembly"*, IBM Journal of Research and Development, Juillet 1977.

[Liégeois et al. 82] A.Liégeois, E.Dombre, P.Borrel: *"Développement d'un système de CAO et de simulation de robots manipulateurs"*, Premières Journées ARA, Poitiers, Septembre 1982.

[Liégeois et al. 84] A.Liégeois, E.Dombre, P.Borrel: *"Programming, simulating and evaluating robot actions"*, 2nd International Symposium of Robotics Research, Tokyo, Août 1984.

[Liégeois et al. 86] A.Liégeois, E.Dombre, P.Borrel: *"Développement d'un système de CAO et de simulation de robots manipulateurs"*, Journées bilan du projet ARA, Paris, Mai 1986.

[Lozano-Perez 76] T.Lozano-Perez: *"The design of a mechanical assembly system"*, AI-TR-397, M.I.T. Artificial Intelligence Laboratory, Cambridge, Decembre 1976.

[Lozano-Perez, Wesley 79] T.Lozano-Perez, M.A.Wesley: *"An algorithm for planning collision- free paths among polyhedral obstacles"*, CACM, Vol.22, nb.1, Octobre 1979.

[Lozano-Perez 81] T.Lozano-Perez: *"Automatic planning of manipulator transfer movements"*, IEEE Transactions on System, Man and Cybernetics, SMC-11, no. 10, Octobre 1981/ publié aussi dans "Robot Motion: Planning and Control", édité par M.Brady et al., MIT Press, 1982.

[Lozano-Perez 82] T.Lozano-Perez: *"Robot Programming"*, MIT AI memo 698, Décembre 1982.

[Lozano-Perez et al. 83] T.Lozano-Perez, M.T.Mason, R.H.Taylor: *"Automatic synthesis of fine-motions strategies for robots"*, 1st International Symposium of Robotics Research, Bretton Woods, Août 1983.

[Lozano-Perez 83] T.Lozano-Perez: *"Spatial planning: a configuration space approach"*, IEEE Transactions on Computers, vol. C-32, no. 2, Février 1983 / publié aussi dans MIT AI memo 605, Décembre 1980.

[Lozano-Perez, Brooks 85] T.Lozano-Perez, R.A.Brooks: *"An approach to automatic robot programming"*, AI Memo 842, Artificial Intelligence Laboratory, M.I.T., Avril 1985.

[Lozano-Perez 85] T.Lozano-Perez: *"Motion planning for simple manipulator robots"*, 3rd International Symposium of Robotics Research Gouvieux, Octobre 1985.

[Lozano-Perez et al. 87] T.Lozano-Perez et al.: *"Handey: a robot system that recognizes, plans and manipulates"*, IEEE International Conference on Robotics and Automation, Raleigh, Avril 1987.

[Lyons 85] D.M.Lyons: *"A simple set of grasps for a dextrous hand"*, IEEE International Conference on Robotics and Automation, St Louis, Mars 1985.

[Martinez, Ferreira 81] F.Martinez, F.Ferreira: *"HELIOS: terminal interactif pour la synthèse d"images réalistes"*, congrés AFCET-TTI, Gyf sur Yvette, 1981.

[Maruyama 72] K.Maruyama: *"A procedure to determine intersections between polyedral objects"*, Int. Journal of Computer and Information Sciences", vol.1, nb.3, 1972.

[Mason 81] M.T.Mason: *"Compliance and force control for computer controlled manipulatòrs"*, IEEE International Conference on Robotics and Automation, Juin 1981.

[Mason 82] M.T.Mason: *"Manipulator grasping and pushing operations"*, AI-TR-690, Artificial Intelligence Lab., M.I.T., Cambridge, Juin1982.

[Mason 84] M.T.Mason: *"Mechanics of pushing"*, 2n International Symposium of Robotics Research, Kyoto, Août 1984.

[Matsushima et al. 82] K.Matsushima, N.Okada, T.Sata: *"The integration of CAD and CAM by application of Artificial Intelligence techniques"*, Annals of the CIRP, Vol. 31, n0. 1, 1982.

[Mazer 81] E.Mazer: *"Réalisation d'un support expérimental de recherche pour le projet robotique PANDORE. Définition et implantation du langage LM"*, Thèse de 3ème cycle, INPG, Grenoble, Janvier 1981.

[Mazer 82] E.Mazer: *"An algorithm for computing relative positions between two objects from symbolical specifications"*, IMAG, Rapport de Recherche no 297, Mars 1982.

[Mazer 83] E.Mazer: *"Geometric programming of assembly robots (LM-GEO)"*, International Symposium on Advanced software in Robotics, Liège, Mai 1983.

[Mazer 87] E.Mazer: *"HANDEY: Un modèle de planificateur pour la programmation automatique des robots"*, Thèse d'Etat, INPG, Grenoble, Décembre 1987.

[Merlet 86] J.P.Merlet: *"Contribution à la commande par retour d'effort en Robotique. Application à la commande de robots parallèles"*, Thèse de Doctorat, Paris 6, Juin 1986.

[Meyer 81] J.Meyer: *"An emulation system for programmable sensory robots"*, IBM Journal of Research and Development, Vol. 25, n0. 6, Novembre 1981.

[Meyer, Jayaraman 83] J.Meyer, R.Jayaraman: *"Simulating robotic applications on a personal computer"*, Computer in Mechanical Engineering, Juillet 1983.

[Miribel, Mazer 83] J.F.Miribel, E.Mazer: *"The LM Reference Manual Vl.7"*, ITMI, Meylan, Octobre 1983.

[Miribel 84] J.F.Miribel: *"Conception et implantation d'un système de programmation de robots"*, Thèse de 3ème cycle, INPG, Grenoble, Octobre 1984.

[Muntean et al. 85] T.Muntean et al.: *"CONKER, Noyau réparti pour OCCAM"*, Rapport de Recherche IMAG no. 529, 1985.

[Nilsson 69] N.J.Nilsson: *"A mobile automaton: An application of artificial intelligence techniques"*, Proceedings IJCAI 69, Mai 1969.

[Nilsson 80] N.J.Nilsson: *"Principles of Artificial Intelligence"*, Tioga, Palo Alto, 1980.

[O'Dunlaing, Yap 82] C.O'Dunlaing, C.Yap: *"The voronoi diagram method of motion planning: I. The Case of a Disc"*, Courant Institute of Mathematical Sciences, 1982 / Technical Report 53, Mars 1983.

[Paul 77] R.P.Paul: *"WAVE: a model-based language for manipulator control"*, The Industrial Robot, Mars 1977.

[Paul et al. 80] R.P.Paul et al.: *"Advanced industrial robot control system"*, School of Electrical Engineering, Purdue University, TR-EE 80-29, Juillet 1980.

[Paul 81] R.P.Paul: *"Robot manipulators: mathematics, programming and control"*, The MIT Press, 1981.

[Pertin-Troccaz 84] J.Pertin-Troccaz: *"S.M.G.R.: un système de modélisation géométrique et relationnelle pour la Robotique"*, Rapport de recherche IMAG no 422, Juin 1984.

[Popplestone et al.78] R.J.Popplestone, A.P.Ambler, I.Bellos: *"RAPT: a language for describing assemblies"*, The Industrial Robot, Septembre 1978.

[Powell 82] E.G.Powell: *"An efficient collision warning algorithm for robot arms"*, 2nd American Association for Artificial Intelligence Conference, Carnegie-Mellon, Août 1982.

[Puget 85] P.Puget: *"Problèmes de prise en compte d'incertitudes en Robotique d'assemblage"*, Rapport de DEA, LIFIA/IMAG Laboratory, Juin 1985.

[Puget, Troccaz 86] P.Puget, J.Troccaz: *"Contrôle dans le système de programmation automatique SHARP: gestion des interdépendances liées aux contraintes d'accessibilité et d'incertitude"*, Rapport de Recherche IMAG/LIFIA, no. 615, Juin 1986.

[Puget 88] P.Puget: *"Utilisation de techniques de preuve de programme pour la programmation automatique des robots"*, Thèse de l'Institut National Polytechnique de Grenoble, 1988 (à paraître).

[Queromes 82] J.G.Queromes: *"Computer-Aided Design and Robotics: a full promise of cooperation"*, 12th International Symposium on Industrial Robots, Paris, Juin 1982.

[Raibert, Graig 81] M.Raibert, J.Graig: *"Hybrid position/force control of manipulators"*, Journal of Dynamic Systems, Measurement and control, no. 102, Juin 1981.

[Reboulet, Robert 85] C.Reboulet, A.Robert: *"Hybrid control of a manipulator with an active compliant wrist"*, 3rd ISRR, Gouvieux, pp76-80, od 1985.

[Reif 79] J.H.Reif: *"The complexity of the mover's problem and generalizations"*, Proceedings, 20th Symposium on the Foundations of Computer Science, 1979.

[Reif, Sharir 85] J.H.Reif, M.Sharir: *"Motion planning in the presence of moving obstacles"*, Proceedings, 26th IEEE Symposium on the Foundations of Computer Science, 1985.

[Requicha 80] A.A.G.Requicha: *"Representations for rigids solids: theorys, methods and systems"*, ACM Computing Surveys, vol. 12, no 4, Décembre 1980.

[Richard 68] *"Intelligence"*, Encyclopaedia Universalis, vol. 8, 1968.

[Romiti et al. 81] A.Romiti et al.: *"Picking from a bin through tactile sensing"*, 11th Int. Symposium on Industrial Robots, Tokyo, Octobre 1981.

[Sacerdoti 75] E.D.Sacerdoti: *"The non-linear structure of plans"*, IJCAI-4, 1975.

[Sata et al. 81] T.Sata, F.Kimura, A.Amano: *"Robot simulation system as a task programming tool"*, 11th International Symposium on Industrial Robots, Tokyo, Octobre 1981.

[Schwartz, Sharir 82] J.T.Schwartz, M.Sharir: *"On the piano movers' problem II: General properties for computing topological properties of real algebraic manifolds"*, Dep. of Computer Science, Courant Institute of Mathematical Sciences, NYU, Report 41, Février 1982.

[Schwartz, Sharir 83] J.T.Schwartz, M.Sharir: *"On the piano movers' problem III: Coordinating the motion of several independant bodies. The special case of circular bodies moving amidst polygonal barriers"*, International Journal of Robotics Research 2(3), 1983.

[Shimano et al. 84] B.E.Shimano, C.C.Geschke, C.H.Splalding: *"VAL-II: a new robot control system for automatic manufacturing"*, IEEE International Conference on Robotics, Atlanta, Mars 1984.

[Simunovic 75] S.N.Simunovic: *"Force information in assembly processes"*, Proc. 5th International Symposium on Industrial Robots, Chicago, Sept. 1975.

[Smith et al. 86] R.Smith, M.Self, P.Cheeseman: *"A stochastic map for uncertain spatial relationships"*, IEEE Conf. on Robotics and Automation, Raleigh, Mars 1987.

[Soroka 80] B.J.Soroka: *"Debugging robot programs with a simulator"*, CAD-CAM 8 Conference, Anaheim, California, Novembre 1980.

[Sussman 75] G.J.Sussman: *"A computer model of skill acquisition"*, North-Holland, 1975.

[Taylor 76] R.H.Taylor: *"Synthesis of manipulator control programs from task-level specifications"*, AIM 228, Stanford Artificial Intelligence Laboratory, Juillet 1976.

[Taylor et al. 82] R.H.Taylor, P.D.Summers, J.M.Meyer: *"AML: a manufacturing language"*, The International Journal of Robotics Research, Vol1, No3, 1982.

[Taylor, Grossman 83] R.H.Taylor, D.D.Grossman: *"The architecture of an integrated robot system"*, International Conference on Advanced Robotics, Tokyo, Septembre 1983.

[Tengvald 84] E.Tengvald: *"The design of expert planning systems: an experimental operations planning system for turning"*, Thesis Dissertation, University of Linkoping, Sweden, Mai 1984.

[Theveneau 85] P.Theveneau: *"Une méthode de génération de mouvements fins basée sur une analyse des contacts"*, Rapport de DEA, LIFIA/IMAG Laboratory, Juin 1985.

[Theveneau 88] P.Theveneau: *Planification de mouvements fins de montage dans un système de programmation automatique de robots"*, Thèse de l'Institut National Polytechnique de Grenoble, 1988 (à paraître).

[Thorpe 84] C.E.Thorpe: *"Path Planning for a Mobile Robot"*, AAAI-84, Kaufman, Août 1984.

[Todd 81] D.J.Todd: *"A method for grasping randomly oriented objects using touch sensing"*, Artificial Intelligence Lab., Queen Mary College, Univ. of London, Juin 1981.

[Tournassoud 86] P.Tournassoud: *"A Strategy for Obstacle Avoidance and its Application to Multi-Robot Systems"*, Proceedings of the 1986 IEEE International Conference on Robotics and Automation, San Francisco, Avril 1986.

[Troccaz 85] J.Troccaz: *"S.M.G.R.: a geometric and relational modeller for robotics"*, International Conference on Advanced Robotics, Tokyo, Septembre 1985.

[Troccaz 86] J.Troccaz: *"Modélisation du raisonnement géométrique pour la programmation des robots"*, Thèse de l'Institut National Polytechnique de Grenoble, Avril 1986.

[Troccaz 87] J.Troccaz: *"On-line automatic robot programming: a case study in grasping"*, IEEE Conference on Robotics and Automation, Raleigh, Avril 1987.

[Troccaz, Puget 87] J.Troccaz, P.Puget: *"Dealing with uncertainties in robot planning using program proving techniques"*, 4 th International Symposium of Robotics Research, Santa Cruz, Août 1987.

[Tsujido et al. 83] Y.Tsujido, N.Kodaira, M.Oshina: *"Realtime motion simulator of robots"*, International Conference on Advanced Robotics, Tokyo, Septembre 1983.

[Udupa 77a] S.M.Udupa: *"Collision detection and avoidance in computer controlled manipulators"*, Proceedings IJCAI 77, Cambridge, Août 1977.

[Udupa 77b] S.M.Udupa: *"Collision detection and avoidance in computer controlled manipulators"*, Ph.D Thesis, Department of Electrical Engineering, California Institute of Technology, 1977.

[Valade 85] J.M.Valade: *"Raisonnement géométrique et synthèse de trajectoire d'assemblage"*, Thèse de Docteur-Ingénieur, Université Paul Sabatier, Laboratoire d'Automatique et d'Analyse des Systèmes, Toulouse, Janvier 1985.

[Weck, Zuhlke 81] M.Weck, D.Zuhlke: *"Fundamentals for the development of a high-level programming language for numerically controlled industrial robots"*, AUTOFACT West, Dearborn, Michigan, 1981.

[Weck et al. 81] M.Weck, W.Eversheim, D.Zuhlke: *"ROBEX: An off-line programming system for industrial robots"*, Proc. of the 11th ISIR, Tokyo, Octobre 1981.

[Whitney 77] D.E.Withney: *"Force feedback control of manipulator fine motions"*, Journal of Dynamic Systems Measurement and Control, Juin 1977.

[Whitney 82] D.E.Whitney: *"Quasi-Static Assembly of Compliantly Supported Rigid Parts"*, Journal of Dynamic Systems Measurement and Control, Mars 1982 / Publié aussi dans Robot Motion, MIT Press, 1982.

[Widdoes 74] C.Widdoes: *"A Heuristic Collision Avoider for the Stanford Robot Arm"*, Stanford Artificial Intelligence Laboratory, 1974.

[Will, Grossman 75] P.M.Will, D.D.Grossman: *"An experimental system for computer controlled mechanical assembly"*, IEEE Transactions on Computers, Vol. 24, n0. 9, Septembre 1975.

[Will 78] P.M.Will: *"Computer controlled mechanical assembly"*, The Industrial Robot, Mars 1978.

[Wingham 77] M.Wingham: *"Planning how to grasp objects in a cluttered environment"*, M. Phil. Thesis, University of Edinburgh, Scotland, 1977.

[Winston 77] P.Winston: *"Artificial Intelligence"*, Addison-Wesley Publishing Company, 1977.

[Wolter et al. 84] J.D.Wolter, R.A.Volz, A.C.Woo: *"Automatic generation of gripping positions"*, RSD-TR-2-84, Center for Robotics and Integrated Manufacturing, Robot Systems Division, Ann Arbor (Michigan), Février1984.

INPG département des études doctorales

AUTORISATION de SOUTENANCE

VU les dispositions de l'article 5 de l'arrêté du 16 avril 1974

VU les rapports de présentation de Messieurs

. G. GIRALT, Directeur de recherche
. J.C LATOMBE, Professeur
. A. LIEGEOIS, Professeur

Monsieur Christian LAUGIER

est autorisé à présenter une thèse en soutenance en vue de l'obtention du grade de DOCTEUR D'ETAT ES SCIENCES.

Fait à Grenoble, le 1er décembre 1987

Le Président de l'U.S.T.M.G

Le Président

A.J. FAUGIER

Le Président de l'I.N.P.-G

Georges LESPINARD
Président
de l'Institut National Polytechnique
de Grenoble

P.O. le Vice-Président,

RESUME

L'un des principaux obstacles au développement de la robotique, réside dans la difficulté qu'il y a à appréhender la complexité et la diversité du monde physique dans lequel le robot est appelé à évoluer. L'objet du présent travail est de montrer que la résolution de ce problème passe par le développement de deux catégories de techniques informatiques: des techniques traditionnelles orientées vers la "programmation des robots", et des méthodes d'Intelligence Artificielle destinées à doter la machine de "capacités décisionnelles". La première partie de la thèse présente les outils informatiques nécessaires au développement des fonctions de programmation. Elle montre de quelle manière nous avons fait évoluer ces fonctions, en les couplant avec des outils dérivés de la CAO: simulation graphique, programmation graphique, programmation géométrique et apprentissage interactif. La deuxième partie traite des mécanismes de la programmation automatique. Elle développe les méthodes de modélisation et de raisonnement, sur lesquelles reposent les fonctions de planification: modélisation géométrique du robot et de son environnement, raisonnement spatial et morphologique, raisonnement sur les incertitudes. Le dernier chapitre de la thèse montre comment toutes les techniques présentées peuvent être combinées, en vue de constituer un système de programmation automatique de robots appelé SHARP.

ABSTRACT

The major obstacle to the development of robotics is the fact that the real world is too complex to process using purely algorithmic tools. The purpose of this work is to show how to solve this problem, by combining two types of computing techniques: traditionnal techniques aimed at *programming assembly robots,* and Artificial Intelligence techniques allowing to equip the machine with *decision making capabilities*. The first part of the thesis presents the computer tools which are necessary for the effective programming of assembly robots. It shows the way we have developped these tools using CAD based facilities: graphic simulation, graphic programming, geometric programming and interactive learning. The second part of the thesis deals with the automatic robot programming problem. It describes modeling and reasoning techniques we have developped for implementing planning functions: geometric modeling, spatial and morphological reasoning, and reasoning about uncertainties. The last chapter shows how it is possible to combine the above techniques, in order to obtain an automatic robot programming system named SHARP.

MOTS CLES

Robotique, Intelligence Artificielle, Modélisation géométrique, Raisonnement spatial, Raisonnement morphologique, Planification de mouvements, Programmation automatique des robots.