Titre: Title:	Conception d'un système robotisé intelligent pour la prise et le rangement de pièces dans une armoire
Auteur: Author:	Mihai Corneliu Murgulescu
Date:	1989
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Murgulescu, M. C. (1989). Conception d'un système robotisé intelligent pour la prise et le rangement de pièces dans une armoire [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/58263/

UNIVERSITE DE MONTREAL

CONCEPTION D'UN SYSTEME ROBOTISE INTELLIGENT POUR LA PRISE ET LE RANGEMENT DE PIECES DANS UNE ARMOIRE

par

Mihai Corneliu MURGULESCU

DEPARTEMENT DE GENIE ELECTRIQUE

ECOLE POLYTECHNIQUE

MEMOIRE PRESENTE EN VUE DE L'OBTENTION

DU GRADE DE MAITRE EN INGENIERIE (M.Ing.)

août 1989

ISBN 0-315-58120-4

UNIVERSITE DE MONTREAL

ECOLE POLYTECHNIQUE

Ce mémoire intitulé:

CONCEPTION D'UN SYSTEME ROBOTISE INTELLIGENT POUR LA PRISE ET LE RANGEMENT DE PIECES DANS UNE ARMOIRE.

présenté par: MURGULESCU Mihai

en vue de l'obtention du grade de: M.Ing.

a été dûment accepté par le jury d'examen constitué de:

M.	J. O'Shea, D.Ing.,	président-rapporteur
M.	R. De Santis, Ph.D.,	directeur de recherche
M.	A. B. Turgeon, D.SC.,	membre du jury

SOMMAIRE

Ce mémoire présente l'architecture conceptuelle, les composantes matérielles et logicielles et les principes d'opération d'un système robotisé intelligent de recherche et de rangement des appareils dans un laboratoire d'asservissements.

Les éléments principaux de ce système sont représentés par un robot mobile, un système d'identification d'objets basé sur une rétroaction visuelle, un système de navigation et un générateur de plan.

Une attention particulière a été consacrée aux problèmes de navigation et de guidage du robot ainsi qu'aux techniques d'intelligence artificielle utilisées pour la génération de plan.

ABSTRACT

This thesis presents the conceptual architecture, the hardware and software components, and the operating principles of an intelligent robotic system for the selection and the lay out of instruments used in a control laboratory.

The main elements of the system are represented by a mobile robot, a system for objects recognition based on visual feedback, a navigation system and a plan generator.

A special attention has been paid to the problems involved in robot navigation and guidance, as well as to the artificial intelligence techniques used in plan generation.

REMERCIEMENTS

Je désire exprimer ma reconnaissance à mon directeur de thèse, Dr.Romano M.DE SANTIS professeur au département de Génie Electrique, pour m'avoir dirigé. Tout au long de mes travaux, par ses précieux conseils, il m'a aidé à m'orienter vers une approche originale du sujet développé.

Un grand remerciement est également du à monsieur Daniel DELMAS professeur au département de Génie Mécanique - co-directeur de thèse,(actuellement chez COGNITECH - France) qui m'a beaucoup motivé par sa grande disponibilité.

Je tiens également à remercier à monsieur Jules O'SHEA, professeur au département de Génie Electrique pour ses précieux conseils.

TABLE DES MATIERES

CHAPITRE 6

CHAPITRE 7

CHAPITRE 8

LISTE DES FIGURES

LISTE DES TABLEAUX

CHAPITRE 1

INTRODUCTION

1.1 INTRODUCTION

Ce projet concerne la conception d'un système robotisé intelligent de recherche et d'arrangement des appareils dans le laboratoire d'asservissements de la section automatique. Le système robotisé sera constitué d'un bras manipulateur monté sur un chariot mobile (l'ensemble constitue un robot mobile), un sous-système de navigation pour permettre au robot la circulation dans le laboratoire (l'environnement de travail) et un sous-système de guidage pour commander le bras manipulateur pour assurer la recherche et la distribution des appareils sur les tables de travail.

L'étude de robots mobiles possédant en même temps des capacités décisionnelles leur conférant l'autonomie de fonctionnement pose le problème dans toute sa généralité car elle comporte l'intégration des divers aspects de perception, de décision et d'action.

L'importance de notre projet consiste dans son potentiel d'illustration, d'analyse et d'expérimentation, dans un cadre de laboratoire bien défini et contrôlé, d'une grande variété de problèmes qui se posent lors de la robotisation des opérations diverses en milieu industriel.

Nous mentionnons entre autres: la reconnaissance des objets à rechercher, la détermination de l'agencement des opérations nécessaires, le guidage du robot, l'architecture conceptuelle, matérielle et logicielle du système, la navigation du robot, la planification des tâches, etc...

Une solution pour ces problèmes requiert souvent l'emploi de plusieurs éléments technologiques avancés: robot mobile, vision par ordinateur, capteurs, liens par radio, etc... Elle requiert aussi l'utilisation d'un nombre de techniques d'une grande actualité et au niveau de l'état de l'art: description sémantique de la scène, contrôle sous rétroaction visuelle d'un manipulateur articulé, algorithmes de reconnaissance de forme, algorithmes de prise de décision, génération de plans, résolution des problèmes, modalités d'interfaçage, etc... Tous ces éléments peuvent être retrouvés d'une façon naturelle dans le développement d'un système robotisé intelligent de recherche et d'arrangement des appareils dans un laboratoire d'asservissements. D'où l'intérêt de notre projet.

Un projet d'une telle envergure peut difficilement être réalisé par une seule personne. Ainsi, dans le cadre spécifique qui nous concerne, le projet est à son début. La réalisation de notre système de robotisation va bénéficier des efforts de plusieurs étudiants dans le sens que certains résultats des projets antérieurs seront adaptés et utilisés; Il nous est agréable de mentionner entre autres: René Côté (Application

d'un robot à la distribution du courrier); P.Souphandavong [21] (Exécution robotisé du jeu de la Tour d'Hanoi par rétroaction visuelle). De la même manière, nous pouvons croire que la conception de notre système va générer des sujets pour d'autres projets.

Le travail présenté dans ce mémoire est le début de nombreux efforts qui seront nécessaires jusqu'à l'obtention d'un système global opérationnel. Notre système robotisé de recherche et d'arrangement des appareils est destiné à être un support expérimental pour la recherche en robotique, point de convergence de l'Automatique, de l'Instrumentation, de l'Informatique et de l'Intelligence Artificielle.

En conséquence, ce projet comprend l'étude de la perception multisenseurs pour concevoir le système de guidage du robot (sous rétroaction visuelle), de la navigation optique pour assurer la circulation du robot dans le milieu de travail, de générateur de plans permettant au robot de prendre des décisions dans l'univers réel dans lequel il évolue. Le but final est d'intégrer dans un tout cohérent et opérationnel des technologies utilisées en Robotique et des techniques utilisées en Intelligence Artificielle.

1.2 UN BREF APERCU DE L'ETAT DE L'ART

Pour situer notre approche par rapport aux réalisations déjà existantes, nous allons donner un aperçu de l'état de l'art dans le domaine de la robotique. Nous allons ensuite dégager les tendances actuelles dans ce domaine, ce qui va nous permettre en même temps d'avoir une idée claire concernant les réalisations et les perspectives.

Pour commencer, aux deux dernières expositions importantes dans le domaine de la robotique: l'exposition ROBOTS 11 à Chicago et la Foire de Hanovre(1987), la conclusion qui se dégage est que la robotique va continuer à bénéficier des progrés d'autres technologies (notamment l'I.A.) et que les premiers systèmes experts apparaissent déjà pour la maintenance des robots. A l'exposition ROBOTS 11 sont apparus pour la première fois les robots de service qui sont des robots multiusages ou spécialisés. Voilà trois exemples:

- ALVIN (Autonome Lab Vehicle with Intelligent Navigation)[16]
- ROBIN (Robotic Insect)[16] est un engin marcheur à 6 pattes.
- Un autre robot (fabriqué par Unimation) [16] a servi à une démonstration spectaculaire à Chicago. Le robot, couplé à un scanographe (pour repérer une lésion et assurer l'orientation parfaite dans l'espace du guide-aiguille avant la trépanation) a permis l'exécution d'une opération (tumeur au cerveau) assistée par robot.

On peut citer également quelques exemples pour illustrer les tendances actuelles dans le domaine de la robotique [11]:

- utilisation de la rétroaction visuelle pour localiser un robot autonome dans un espace tridimensionnel [12];
- utilisation d'une technique permettant la coordination entre le contrôle et une base de connaissance (au niveau d'un système multi-senseurs) pour un robot mobile autonome [13];
- le contrôle intelligent pour un robot utilisant un système de senseurs en boucle fermée avec modèle de référence [14];
- système expert de guidage utilisé dans un véhicule autonome sous-marin pour le contrôle en temps réel [15].

Concernant spécifiquement les robots mobiles, l'état de l'art est exposé en détail dans le chap.2 pour les réalisations les plus significatives. Nous pouvons remarquer les tendances suivantes:

- Utilisation de la rétroaction visuelle pour le guidage des robots;
- Utilisation des systèmes multisenseurs pour la navigation des robots (surtout ultrasons et infrarouges);
- Intégration des systèmes à base de connaissances dans le contrôle intelligent des robots.

Pour ce qui concerne notre projet, nous nous plaçons dans le cadre de systèmes pour robots mobiles spécialisés pour une tâche et un environnemet bien déterminés ce qui fait que par exemple le sous-système de navigation que nous avons adopté est

un système de navigation optique. Par rapport au tendances vues notre système est prévu avec un sous-système de navigation optique, un sous-système de guidage par rétroaction visuelle et un système multisenseurs (ultrasons et infrarouges) pour assurer la fonction de sécurité. Au niveau décisionnel il est prévu avec un système à base de connaissances.

Sur la base de la documentation que nous avons consulté, à date nous n'avons pas trouvé un système semblable. Il y a quand même certains fabriquants de robots qui ont lancé des robots d'assemblage ou d'usinage équipés avec un magasin des pièces ou d'outils, capables de les utiliser d'une façon intelligente. Ce qui pourrait être une éventuelle application future pour notre système suite aux adaptations qui s'imposeront.

1.3 LE SYSTEME A ROBOTISER

La recherche et la distribution des appareils pour les travaux pratiques d'asservissements afférents au cours sous-gradué 3.520 - Asservissements - sont faits de la manière suivante.

Un technicien va chercher les appareils nécessaires pour la manipulation (TP) à exécuter dans deux armoires où les appareils sont déposés. Ensuite, il les dépose sur un chariot pour les transporter, déplace le chariot dans le laboratoire et distribue sur chaque table de travail les appareils prévus par le cahier de charge de la manipulation à exécuter. L'opération est répétée autant de fois que nécessaire, pour s'assurer que chaque table de travail soit équipée avec tous les appareils nécessaires. Le critère pour choisir les appareils de l'armoire (où ils sont déposés aléatoirement) est l'appartenance à la manipulation en cours d'exécution. Cette procédure est résumée par le schéma de la fig.1.1

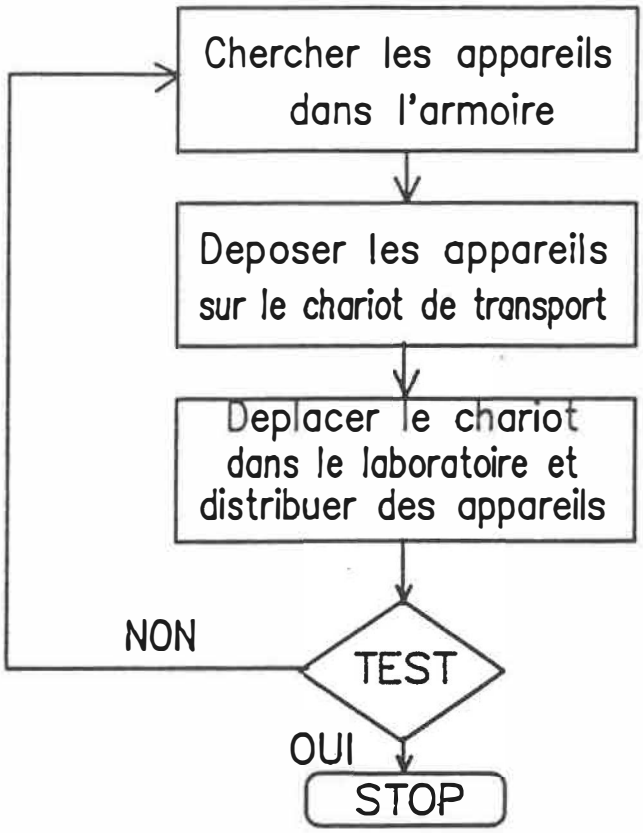


Fig.1.1 Les opérations exécutées par l'humain (solution actuelle)

Pour la résolution du problème de robotisation il y a quelques facteurs importants à considérer:

- La géométrie de l'environnement (du laboratoire) voir fig.1.2
- Le nombre de tables de travail (dix)
- Le nombre de manipulations différentes à exécuter (quatre)

Manipulation No.	Nombre d'appareils
1	6
2	11
3	4
4	3

Tableau 1.1: Le nombre d'appareils différents par manipulation.

Pour une meilleure compréhension du problème voici une courte description de l'environnement du travail. Les dimensions des différents éléments se trouvent dans le tableau 1.2. (exprimées en métres). Les tables de travail sont arrangées symétriquement par rapport à l'axe principale du laboratoire dirigée Nord - Sud; les distances entre les tables de travail sont égales. Au milieu du laboratoire il y a un couloir de circulation pour le robot. A l'extremité Sud du laboratoire se trouvent les deux armoires, et entre les deux armoires une table auxiliaire.

	Longueur	Largeur
Laboratoire	12	5.14
Couloir du milieu	12	1.50
Armoire	1.82	1.25
Table de travail	1.82	1.25
Table auxiliaire	1.50	1.25
Couloir entre tables	1.82	1.30

Tableau 1.2: Les dimensions des différents éléments de l'environnement de travail.

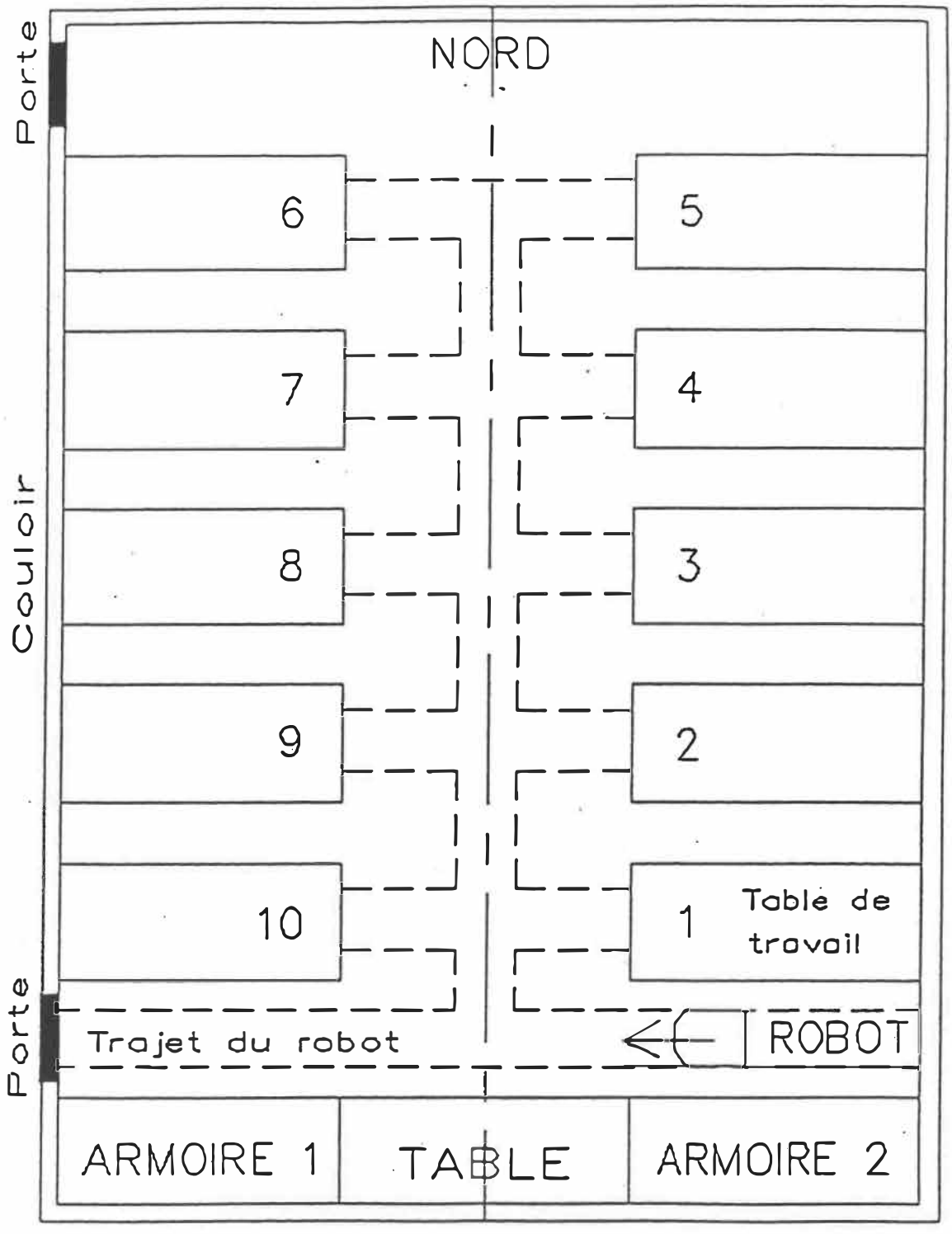


Fig.1.2 La géométrie de l'environnement de travail du robot (laboratoire)

1.4 ROBOTISATION ENVISAGEE

Nous allons automatiser la séquence des opérations exécutées actuellement par l'humain grâce au système robotisé à concevoir

Pour notre système robotisé, la solution envisagée est la suivante. Les éléments physiques du système seront: un robot mobile composé d'un bras manipulateur monté sur un chariot mobile, un sous-système de navigation optique, un sous-système de guidage composé d'un système de perception multisenseurs comprenant un système de vision, un système d'émetteurs-récepteurs optiques (infrarouge) et un micro-ordinateur IBM-PC embarqué pour assurer les ressources de calcul nécessaires pour la résolution des différents algorithmes. Au cours de l'exécution des tâches par le système robotisé, on peut dégager les modalités d'opération suivantes. Le robot mobile va remplacer l'humain pour exécuter d'une façon automatique la tâche de recherche/distribution des appareils. Le système de navigation assure le déplacement du robot dans l'espace de travail et le sous-système de guidage va assurer l'accomplissement de la tâche, c'est-à-dire la recherche des appareils dans l'armoire et ensuite leur distribution sur les tables de travail.

Concernant la structure conceptuelle du système le schéma bloc du système robotisé est représenté dans la fig.1.3. Elle représente un système asservi en boucle fermée à deux niveaux hiérarchiques. Au premier niveau se trouvent les boucles d'asservissement (en position et vitesse) du moteur qui vont

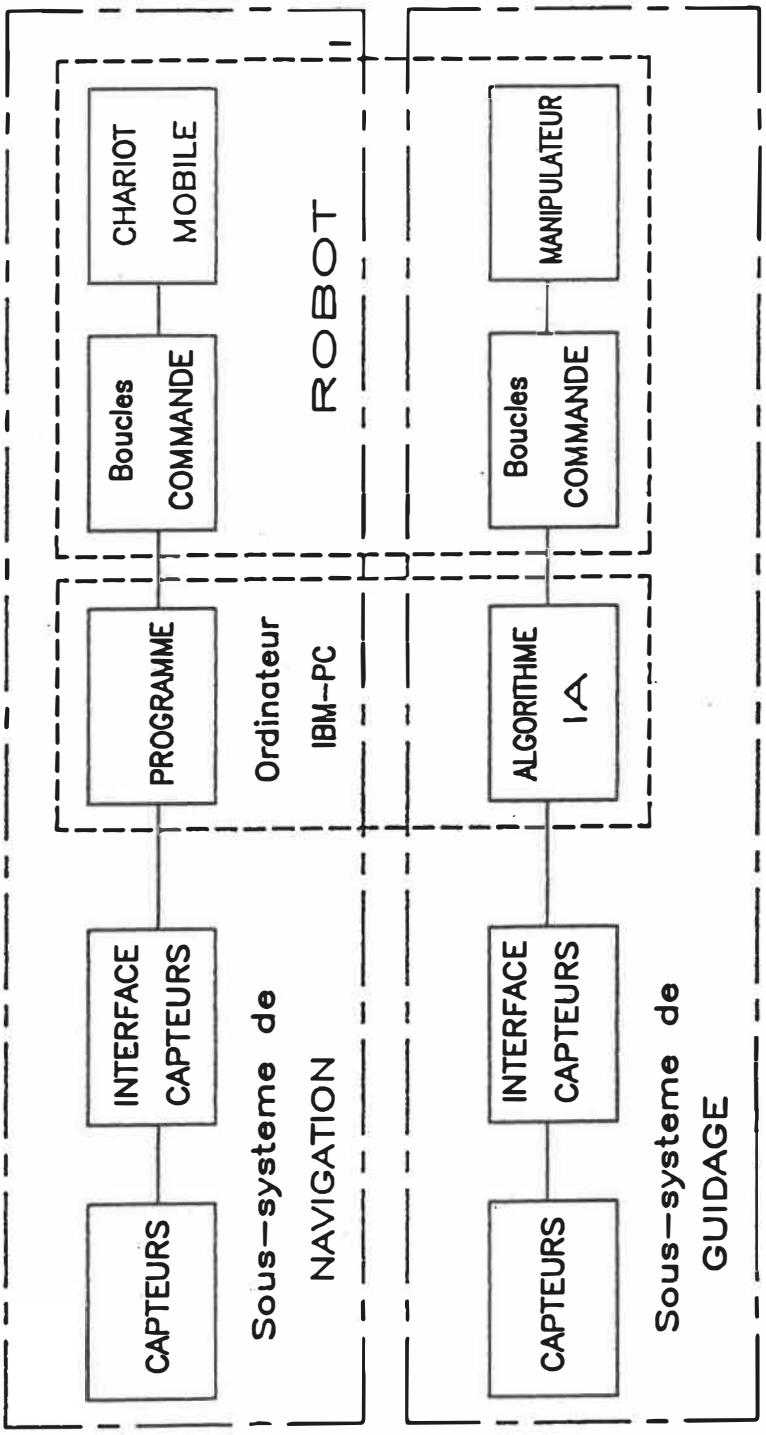


Fig.1.3 Le schéma bloc du système robotisé

équiper le système de locomotion du chariot mobile. Au niveau supérieur, se trouvent les boucles de commande du manipulateur, interfacées avec les capteurs prévus pour le guidage. L'interface doit supporter les logiciels d'intelligence artificielle qui commandent le bras du robot.

Cette solution présente plusieurs avantages:

- sur le plan économique, la solution robotisée va permettre de décharger les techniciens d'une tâche répétitive, donc ils pourront remplir d'autres travaux qui demandent plus de qualification comme par exemple l'exécution des vérifications qui conduiront à un meilleur déroulement des travaux pratiques des étudiants; ou simplement, ils pourront se dédier à d'autres travaux plus importants.
- sur le plan temporel, la solution robotisée va apporter plus de flexibilité dans l'horaire habituel de travail dans le sens que l'opération de recherche et de la distribution des appareils pourra s'exécuter en dehors des heures habituelles de travail (même pendant la nuit) ce qui aura comme conséquence la possibilité d'utiliser le laboratoire encore plus longtemps pour l'exécution des travaux des étudiants.
- sur le plan technique, la solution robotisée va apporter la nouveauté technologique, ce qui aura aussi un impact psychologique positif sur les utilisateurs du laboratoire.
- sur le plan éducatif, (étant donné que l'application sera faite dans le cadre d'un laboratoire universitaire) la solution robotisée aura un impact positif évident, étant un exemple concret et vivant d'application de la robotique

dans la vie quotidienne; on peut même envisager de l'insérer comme manipulation dans le cadre d'un laboratoire afférent à un cours de robotique, ce qui fera un double usage (donc encore plus rentable).

1.5 OBJECTIFS

L'objectif de notre travail est d'illustrer les modalités d'integration des techniques de l'I.A. avec les techniques de l'automatique pour la résolution du problème de robotisation. Dans cette idée le but de ce document est de présenter les éléments nécessaires (matériels et logiciels) pour concevoir un système robotisé intelligent pour la prise et le rangement de pièces dans un armoire. Plus en détail ce projet vise plusieurs objectifs:

- Identification des problèmes soulevés par la robotisation;
- La conception du système - identification des différents modules qui seront nécessaires pour le rendre fonctionnel;
- Développement de certains modules pour résoudre des problèmes liés à la robotisation envisagée.

La réalisation du projet prévoit le développement de plusieurs étapes:

- La conception du système robotisé dans son ensemble
- Le choix d'un bras manipulateur et du chariot mobile
- La conception des sous-systèmes de navigation et de guidage; le développement des algorithmes afférents.

Nous allons donner des solutions au niveau de la conception

pour: le système dans son ensemble, le robot mobile et le sous-système de guidage; nous allons développer plus en détail le sous-système de navigation et l'algorithme de génération de plan pour l'accomplissement de la tâche; on touche à la fois le domaine de la robotique ainsi que le domaine de l'intelligence artificielle.

1.6 ORGANISATION DU MEMOIRE

Le mémoire est divisé en 8 chapitres. Après la présentation générale de la problématique concernant le système en étude, l'état de l'art et les objectifs exposés dans le chapitre 1, on introduit dans le chapitre 2 les notions de "robot intelligent" "contrôle intelligent" et "robot complet" (exemples les plus significatifs) dans le but de faire la conjonction entre la robotique et l'intelligence artificielle. Le chapitre 3 est consacré à la présentation des composantes matérielles du système en étude. Dans le chapitre 4 on peut voir exposé le sous-système de navigation et le chapitre 5 présente le sous-système de guidage. Le chapitre 6 est consacré au générateur de plan qui représente la composante logicielle la plus importante du système en étude. Le chapitre 7 propose une autre solution possible pour le générateur de plan, la solution système expert Le développement du système expert pour la génération de plan a été fait avec un outil de développement, le VP-Expert. Finalement, les conclusions de ce travail sont exposées dans le chapitre 8 où nous suggérons également des extensions possibles

CHAPITRE 2

LES ROBOTS INTELLIGENTS

2.1 DEFINITION DU "ROBOT INTELLIGENT"

Dans la nouvelle phase de développement de la robotique, l'objectif majeur qui se dégage est la volonté de rendre le robot "intelligent". L'adjectif qu'on vient d'utiliser jusque là, réservé à l'homme est d'une ambiguité remarquable parce que, même pour l'homme, on n'a pas trouvé une définition universellement acceptée, ou une mesure de l'intelligence [17]. Alors, il faut remarquer que les spécialistes ne sont pas tous d'accord, ce qui n'empêche pas l'apparition d'une nouvelle génération des robots appelés "intelligents" par rapport aux robots actuels qualifiés en "automate programmable".

Dans le secteur de la recherche intitulé "Intelligence Artificielle", reporté aux thèmes retenus par l'Association for Computing Machinery (ACM) on considère partie integrante du domaine de l'Intelligence Artificielle (l'IA):

- la résolution des problèmes
- la reconnaissance des formes
- la théorie de la décision
- la théorie des jeux
- la preuve des théorèmes
- la composition musicale par ordinateur

Nous allons essayer de dégager les points essentiels sur cette voie d'évolution des robots modernes qui est l'utilisation exhaustive des techniques de l'I.A.

2.2 LES ROBOTS UTILISANTS LES TECHNIQUES DE l'IA - Exemples

Nous allons faire le point sur ce qui existe au niveau du laboratoire comme robots utilisants les techniques de l'IA pour avoir un aperçu plus clair du problème. Il y a deux grandes catégories [3]:

1. Les robots "logiciels" - des robots sans existence physique - des représentations sur un ordinateur de l'organisation et du fonctionnement des "cerveaux" de robots évolués.
2. Les robots "complets" - ceux qui mettent leurs décisions en pratique parce qu'ils possèdent en plus un "corps" mobilisable et des sens artificiels.

Fait très significatif: si les robots "logiciels" semblent nombreux, plusieures dizaines, les robots "complets" sont très rares: environ trois aux Etats-Unis et un en France. Même si ce dénombrement n'est pas exhaustif, il donne un bon ordre de grandeur.

On peut dégager deux conclusions:

1. La rélation décision - action est très difficile à maîtriser
2. Elle est très interactive.

Voila les quatre robots "complets":

SHANKEY (1968-1973) construit à Standford Research Institute

Robot mobile doté d'une caméra TV et de proximètres. Les ordres étaient donnés à partir d'un ordinateur, par radio. Son univers perceptible se réduisait à des chambres avec des portes par lesquelles il pouvait passer, à des blocs qu'il pouvait déplacer en les poussant [19].

JASON (1970-75) construit à Berkeley. Robot mobile doté d'un bras manipulateur et d'un synthétiseur de parole. Il recevait les ordres envoyés par radio depuis un ordinateur. Il ne comprenait que les langages FORTRAN et LISP. Ses capteurs comprenaient une caméra TV, des capteurs de contact, des proximètres infra-rouges et ultra-sonores. Il a permis de tester deux générateurs de plan fonctionnant en univers certain et incertain [19].

Le robot d'exploration de la planète MARS - [19] étudié par le Jet Propulsion Laboratory a été abandonné en 1980 pour des motifs financiers. La plate-forme mobile supportait un bras manipulateur, un télémètre laser et une caméra stéréoscopique. Il aurait fallu faire:

- de l'analyse de scène générale, puis locale;
- de la navigation;
- de la locomotion et de la manipulation.

Le robot possède des générateurs de plan pour agir et se déplacer suivant les instructions envoyées de la Terre. Il comprenait les langages FORTRAN, SAIL et CSA.

HILAIRE est un robot mobile français qui semble inspiré par Jason et qui est étudié depuis 1977 à Toulouse au LAAS (Laboratoire d'Automatique et d'Analyse des Systèmes). Son premier objectif concerne la navigation. Pour cela, il possède une caméra TV associée à un télémètre laser pour "voir" en relief, des proximètres infra-rouges et à ultra-sons.

Il devrait posséder trois niveaux de décision:

- prétraitement des informations captées et contrôle des déplacements par microprocesseurs.
- organisation, suivi des tâches de navigation et supervision des communications par mini-ordinateur embarqué.
- extension des ressources de calcul ou de réflection par liaison radio avec un gros ordinateur.

Remarque importante: l'intéractivité très présente dans le cas des robots "complets" n'apparait pas dans le robot "logiciel" où l'on se contente de résoudre des problèmes très déconnectés de la réalité matérielle et ajoutons qu'un robot "complet" coûte excessivement cher, alors qu'un robot "logiciel" n'exige que du temps de calcul sur un ordinateur (souvent gros)

2.3 PARTICULARISATION DANS LE CAS DE NOTRE PROJET

Après avoir dégagé les points essentiels sur la nouvelle voie d'évolution des robots (utilisation des techniques de l'I.A.) nous allons voir les implications pour notre projet. Pour concevoir un système robotisé intelligent capable de remplir la tâche ennoncée, notre système doit avoir deux propriétés:

1. Une certaine autonomie (dans le sens décisionnel)
2. La mobilité

Pour réaliser ces deux propriétés nous devons tenir compte de trois aspects différents:

- La perception de l'environnement par notre système et ensuite sa modélisation interne;
- L'utilisation du modèle pour la prise des décisions en fonction d'un but à atteindre;
- L'exécution de cette décision par les moyens de mouvement du robot.

Il est évident que ces trois aspects sont dans un rapport d'interdépendance. Ainsi, la prise des décisions dépend de la façon dont l'environnement a été perçu et également du modèle interne que le robot possède de lui-même.

Il est évident aussi, que ces aspects sont liés à des questions importantes en Robotique et Intelligence Artificielle

Il s'agit notamment d'abord de la représentation de la connaissance et ensuite de son traitement d'une manière efficace. En fait, si nous pensons notre système muni de ces deux propriétés: autonomie + mobilité, ça nous oblige à penser que notre système robotisé doit avoir un contrôle d'un type spécial qui sera le résultat de l'integration des techniques de l'Intelligence Articielle et de la théorie du contrôle.

La théorie du contrôle offre des outils très puissants pour la résolution des problèmes complexes liés au contrôle des processus. Pour appliquer ces outils, le problème doit être généralement:

- formulé analytiquement;
- suffisamment connu à priori;
- muni des restrictions (linéarité par exemple) qui permettent de prouver la convergence, l'optimalité, etc.

Alors, le contrôle est efficace, le comportement du système est interprétable et il peut être utilisé dans des applications en temps réel.

L'intelligence artificielle offre des outils puissants pour la prise des décisions d'une façon intelligente. Le problème attaqué est généralement:

- pas formulé analytiquement;
- méme pas complétement spécifié.

Alors, les systèmes résultants sont typiquement larges, les logiciels complexes, défiant le traitement analytique,

impliquant un considérable effort de compréhension et demandant des ressources informatiques importantes.

Donc, il y a deux observations à faire:

1. Les deux disciplines travaillent avec des classes des problèmes différents:
   - une première classe des problèmes avec des contraintes et traitable analytiquement;
   - une deuxième classe des problèmes sans contraintes et difficiles à formaliser.
2. Les systèmes de contrôle intelligents mettent le concepteur en présence de ces deux types de problèmes simultanément. Alors nous nous proposons de tenir compte de toutes ces considérations dans la conception de notre système robotisé.

CHAPITRE 3

COMPOSANTES MATERIELLES DU SYSTEME EN ETUDE

Le système robotisé étudié est constitué par cinq modules principaux: manipulation, locomotion, perception, support énergétique et support informatique (fig.3.1). Chacun des modules sera décrit du point de vue matériel et fonctionnel. Ce chapitre est consacré surtout à la description des modules du point de vue matériel et trés brièvement à leurs fonctions, leurs spécifications et le choix fait pour les satisfaire. Les autres modules seront développés dans les chapitres suivants.

3.1 LE BRAS MANIPULATEUR

Le module de manipulation a la fonction de chercher les appareils choisis dans les armoires, les déposer sur le plateau du robot les arranger sur les tables de travail. Pour réaliser cette fonction, il doit répondre aux exigences suivantes:

- être capable de soulever une charge utile de 10 Kg (le poids maximum d'un appareil se trouvant dans l'armoire);
- être le plus léger possible (capable de satisfaire la première condition);
- avoir un minimum de quatre degrés de liberté;
- facilités d'interfaçage avec l'ordinateur et les senseurs;
- disponibilité éventuelle d'un langage de programmation.

Etant donné les spécifications nécessaires vues précédemment, le bras manipulateur a été choisi parmi la douzaine de

robots manipulateurs etudies pour etre utilises dans ce type d'application (voir fig.3.2).

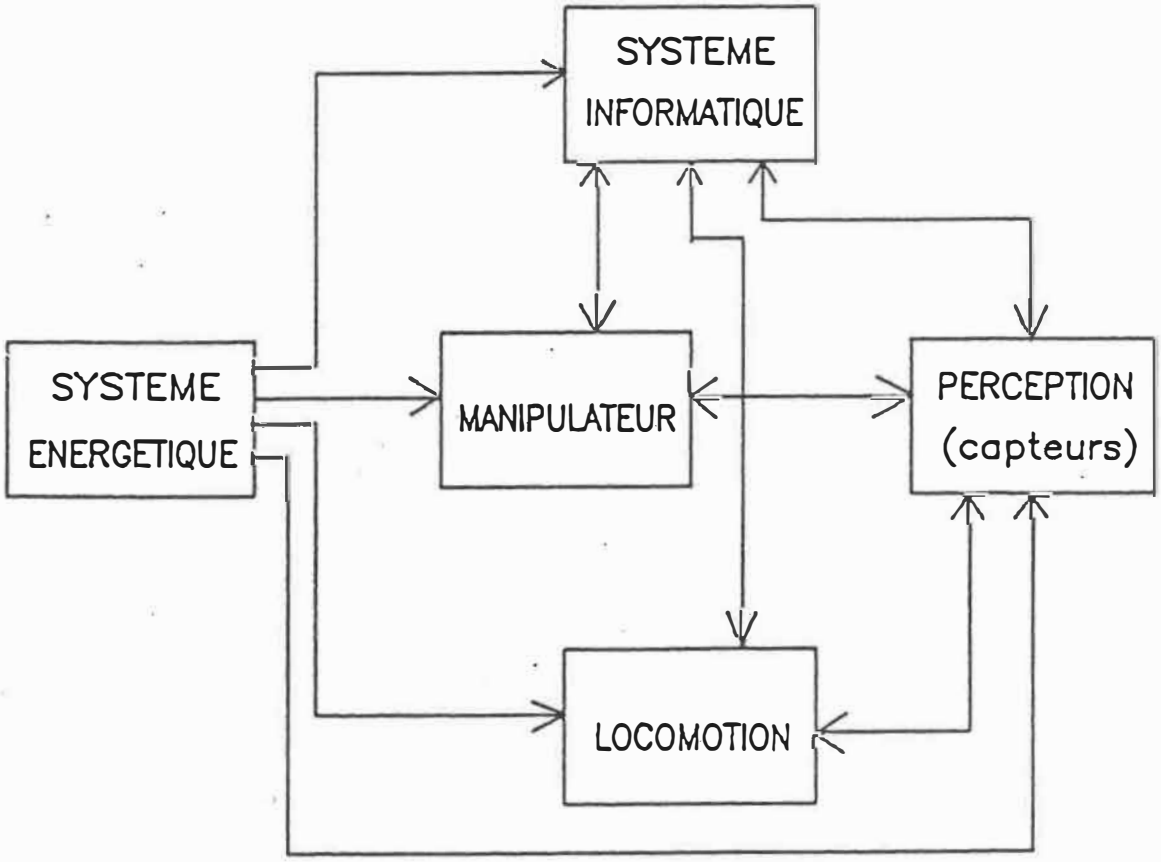


Fig.3.1 Structure du système robotise (modules).

No.	Type de ROBOT	Poids du robot (Kg)	Charge utile (Kg)	Degrés liberté	Langage utilisé	Gabarit robot (mm)
1	TOSHIBA SR-653/654H	80	5 10/20	3/4 axes	SCOL	1000/ 800
2	TOSHIBA SR-606V	80	3	6 axes	SCOL	1230/ 650
3	TOSHIBA SR-1053H/ 1054H	140	10 20 40	3/4 axes	SCOL	1480/ 1000
4	AKROBOTICS R77P7S		30	5 axes	Langage off-line	2000/ 2300
5	SYKE robot 600-5		2	5 axes		800/ 630
6	IRI M50E AC SERVO ROBOT	150	20	3 axes	RCL	1200/ 700
7	SCORBOT Educ. robot	12	1	5 axes	ELITE	400/ 550
8	CYBOTECH	320	10	6 axes		800/ 1390
9	ASEA IRB L6/2	145	6	6 axes		1500/ 1600
10	CYRO - 1000 Robot Manip	280	10	5 axes		1037/ 1194
11	L-10 Robert Roberts Corp	208	10	5/6 axes		1500/ 1750
12	PRAB G-06 Robots Inc.	380	15	5/6/7 axes	Robomac off-line	1220/ 1525

Tableau 3.1: Tableau comparatif des bras manipulateurs.

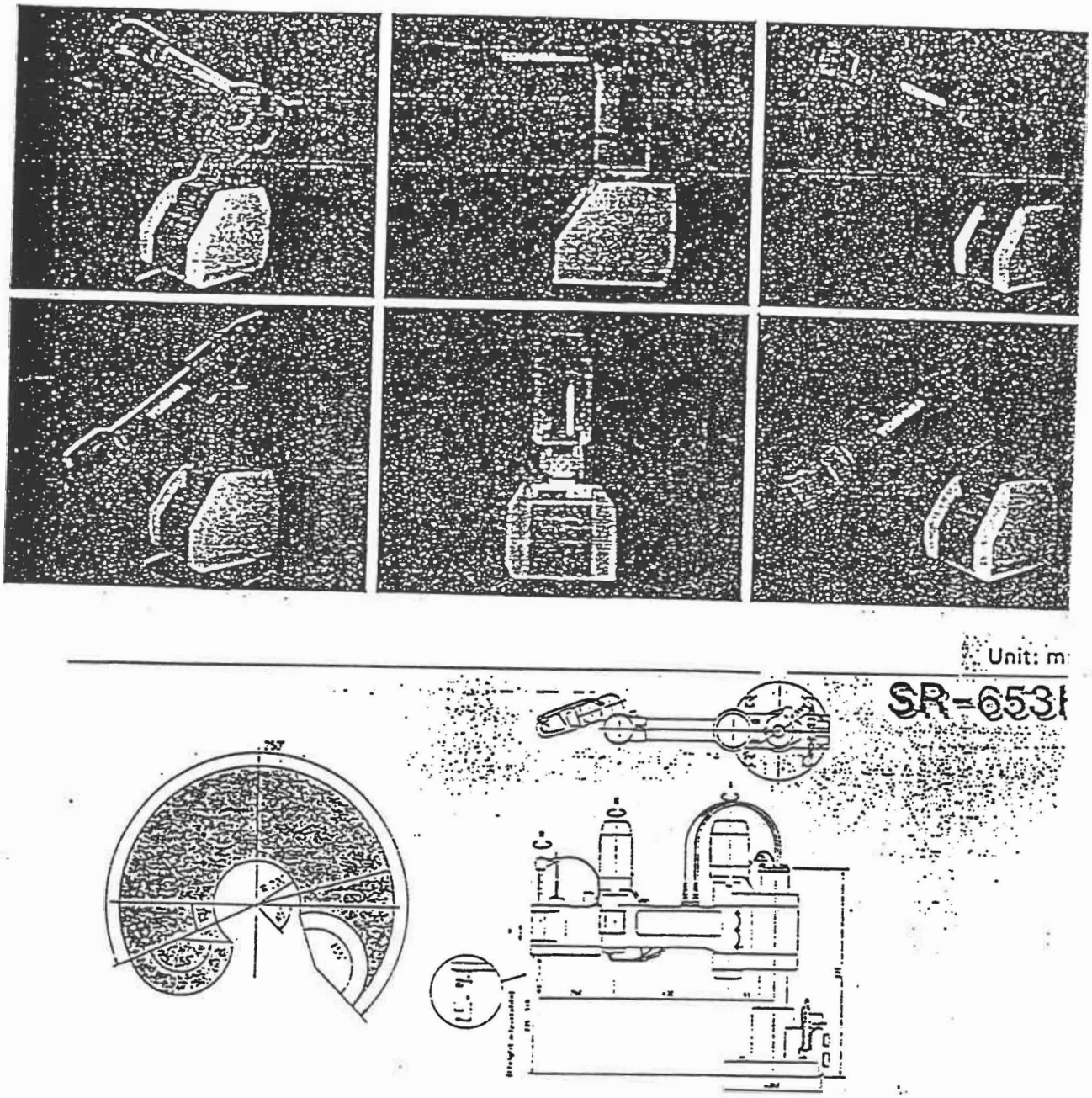


Fig.3.2 Le bras manipulateur TOSHIBA SR-654H

En étudiant le tableau comparatif des bras manipulateurs extraits des différents catalogues (tableau 3.1),le choix a été fixé sur le robot TOSHIBA SR-653/654H. Le robot choisi, TOSHIBA SR-654H, est illustré à la fig.3.2 et ses caractéristiques se trouvent en annexe. Le bras manipulateur sera monté sur le module de locomotion qui est un chariot mobile (une adaptation d'un fauteuil roulant motorisé de la compagnie FORTRESS SCIENTIFIC). Il servira pour remplir la tâche de manipulation des appareils.

3.2 LE MODULE DE LOCOMOTION

Le module de locomotion a la fonction d'assurer le déplacement du robot dans l'environnement de travail pour transporter les appareils de mesure de l'armoire vers les tables de travail et inversement. Il assure la mobilité du robot. Pour réaliser cette fonction, il doit répondre aux exigences suivantes:

- pouvoir transporter un poids de minimum 150 Kg (bras manipulateur, batteries, micro-ordinateur, senseurs);
- sa géométrie doit permettre son passage entre les tables de travail dans le laboratoire (couloir de 1,50 m de largeur);
- sa maniabilité doit lui permettre d'exécuter des virages à 90° pour pouvoir circuler au long des armoires et entre les tables de travail;
- sa vitesse doit être variable pour permettre des manoeuvres de rapprochement en douceur et une circulation plus rapide

quand cela est nécessaire (retour au point de départ);

- être capable d'assurer une précision d'arrêt de l'ordre de quelques centimètres près.

Pour satisfaire aux spécifications énumérées, nous adopterons une solution qui integrera aux éléments developpés dans les travaux de Mongy Rabemanantsoa [chariot mobile] [22] et René Coté [suivi de trajectoires] un certain nombre d'éléments nouveaux tels que:

- deux roues motrices indépendantes a l'arrière, chacune actionnée par un moteur DC commandé en vitesse par le module informatique;
- une roue folle en avant;
- un système de freinage avec freins électro-magnétiques mis en marche par des ressorts et relâchés électriquement.

Les roues arrière du chariot sont entraînées directement par deux moteurs autonomes à courant continu, à aimant permanent, alimentés par un système 24 volts-courant continu. Ils sont compacts et conçus pour économiser l'énergie; ils comportent leur propre boîtier de transmission à rapport de 16:1 ou 20:1. Les moteurs sont équipés de freins à disques électromagnétiques intégrés avec un excellent couple de démarrage et de blocage. Le contrôleur de vitesse se trouve dans le module informatique.

Le déplacement du module de locomotion est assuré par les deux roues motrices et la roue folle. La trace des roues forme au sol un triangle équilatéral de 65 cm de coté. Le choix d'une base triangulaire permet d'éliminer les problèmes de suspension

et de stabilité. Cette structure permet au robot d'accomplir des trajectoires en forme d'arcs de cercle centrés en un point quelconque de l'axe des roues motrices, la ligne droite étant un cas particulier où le centre est rejeté à l'infini (supérieur à une valeur limite). Le système permet aussi, par le contrôle indépendant des deux moteurs, d'exécuter des virages à 90° (pivot de 90°) par blocage d'une roue et commande de l'autre. Les deux roues motrices commandées par des moteurs DC, peuvent être contrôlées en vitesse et en direction (sens de rotation) ce, qui permet les manoeuvres suivantes:

- avancer
- réculer (changement de sens de rotation des moteurs)
- s'arréter (freinage)
- pivoter (dans les deux sens, commande d'avance pour un moteur et de freinage pour l'autre).

Chaque roue peut avoir six vitesses différentes avec une vitesse maximum de 8,8 Km/h dans chaque sens. Il est possible de faire avancer les roues très lentement (pour la phase d'approche d'une table de travail) ou bien plus vite (pour la phase de déplacement dans le laboratoire). Dans tous les cas, c'est le programme de navigation (implanté dans le module informatique) via le contróleur de vitesse qui s'occupe du contróle des moteurs, chaque fois qu'il reçoit l'ordre parvenu de l'IBM-PC qui contróle le robot dans son ensemble.

Le module de locomotion est doté de deux modes de freinage: dynamique et mécanique. Le freinage dynamique intervient

lorsqu'on envoie la commande d'arrêt, quand le chariot est en mouvement. Les freins à disque électromécaniques sont appliqués automatiquement lorsque le chariot s'arrête, quelle que soit la direction de sa course, ce sont des freins de sureté. Ils peuvent être rélâchés si l'on veut rouler en roue libre. Le chariot mobile est prévu avec un boîtier de commande à manette. Le boîtier sera utilisé pour la commande manuelle de notre robot (mesure préventive de sécurité) et il va doubler le programme implanté sur micro-ordinateur dans le module informatique.

Le chariot mobile a les dimensions de gabarit suivantes: 86 cm de longueur et 66 cm de largeur. La distance entre les axes des roues est de 45,5 cm et l'axe des roues motrices se trouve à 27 cm de la barre de protection arrière du chariot. Les roues ont un diamètre de 25 cm et le chariot seul a une hauteur de 44 cm. Sur le chariot, on va monter une structure en bois qui va supporter l'ordinateur (IBM-PC) et les éléments afférents au système de guidage sur la première plate-forme. Sur une deuxième plate-forme sera monté le bras manipulateur. L'ensemble (module de locomotion (chariot mobile), bras manipulateur, module de perception, module informatique et module de support énergétique) monté sur la structure spécialement construite, constitue un robot mobile.

Le robot mobile ainsi construit devra circuler dans le laboratoire en suivant le ruban guide. Il va circuler d'abord en direction est-ouest le long des armoires pour ramasser les appareils nécessaires et ensuite au long de l'axe nord-sud pour

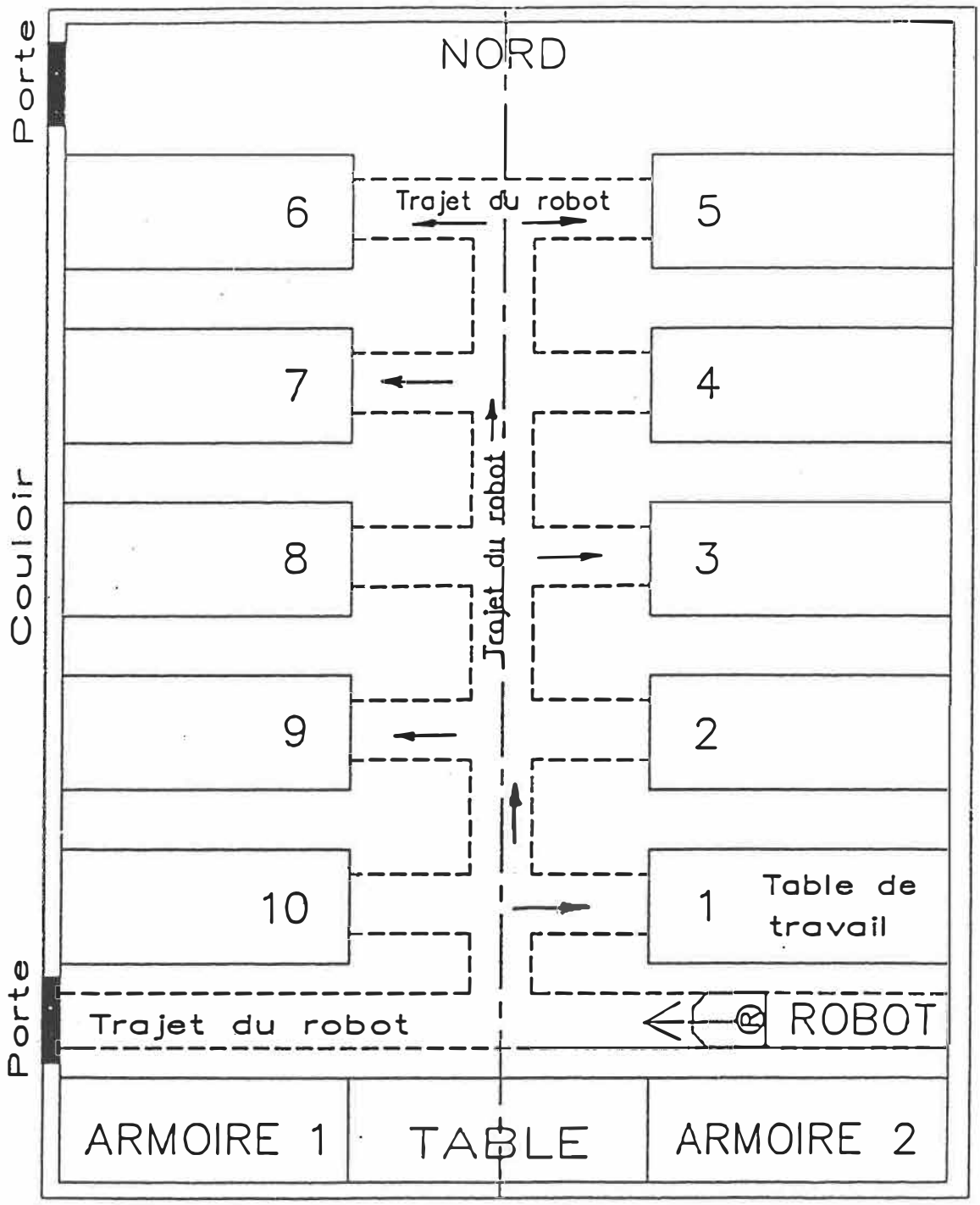


Fig.3.3 La géométrie de l'environnement de travail

(le trajet de circulation du robot)

se rendre à une destination, tourner (pivoter) de 90 ° dans un sens ou dans l'autre pour s'arrêter le plus près possible de la table de travail qui constitue sa destination. Etant donné que la largeur du robot est de 0,65m et la largeur du couloir entre les tables de 1,50m, il y a amplement d'espace de manoeuvre. Le robot va s'arrêter au bout de chaque table de travail et ensuite son bras manipulateur va déposer et ramasser les appareils sur les tables de travail. Le trajet de circulation est représenté sur la fig.3.3.

3.3 LE MODULE DE PERCEPTION

Le module de perception fournit toutes les informations perçues par ses capteurs, concernant l'environnement de travail, dans le but de les utiliser pour la commande du robot. C'est un système multisenseurs constitué par:

- un dispositif de navigation optique;
- un système de vision;
- un système d'émetteurs-récepteurs ultrasons;
- un système d'émetteurs-récepteurs optiques (infra-rouge).

Le dispositif de navigation optique permet le déplacement du robot dans l'environnement de travail. Il doit pouvoir suivre un trajet déterminé (voir fig.3.4) de manière à assurer les spécifications prévues pour le module de locomotion concernant notamment la maniabilité et la précision. Du point de vue matériel, il est constitué de deux lampes éclairantes et de quatre photo-transistors montés au niveau du sol pour détecter

différents niveaux de réflection lumineuse du sol. On utilise 4 photo-transistors montés 2 par 2 comme on voit sur la fig.3.4

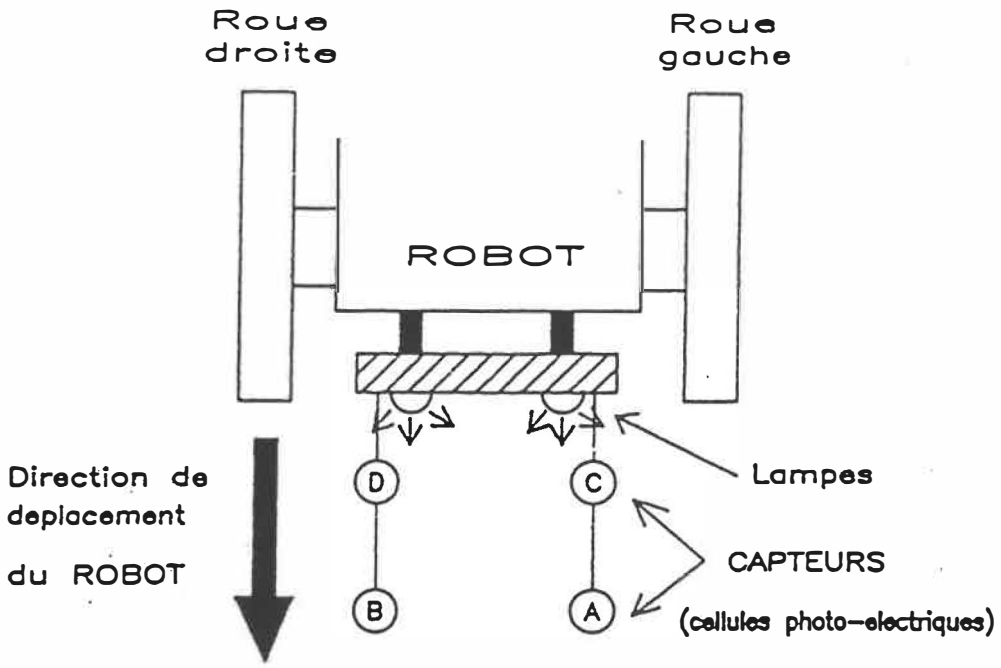


Fig.3.4 Dispositif de navigation avec capteurs photo-électriques

L'utilisation de quatre photo-transistors permet un meilleur contrôle de la direction suivie par le robot. Si l'axe du chariot fait un angle avec l'axe du ruban guide, on arrive à une situation où il y a trois cellules allumées et une éteinte, ce qui permet plus de précision de contrôle. La largeur du ruban guide (L) dépend de la distance (d) qui sépare les paires de photo-transistors. Evidemment, il faut que L>d pour pouvoir couvrir les deux en même temps. Les détails concernant l'influence de la largeur du ruban guide se trouvent dans le

chapitre 4, paragraphe 2. On a adopté une largeur de deux fois la distance qui sépare les photo-transistors, L = 2d. Les oscillations par rapport à la trajectoire voulue (ligne droite par exemple) dépendent de d. Les photo-transistors constituent les capteurs qui commandent le système de contrôle du chariot mobile. Chaque paire de photo-transistors commande un moteur via une unité de traitement de signal qui interprète les signaux envoyés par les quatre capteurs.

Le système de vision doit fournir les informations concernant l'environnement de travail pour permettre au robot de remplir sa tâche, la manipulation d'objets, c'est-à-dire la recherche et la distribution des appareils. Sans pouvoir donner des spécifications précises pour le système de vision, nous avons fait le choix du système tenant compte des certains critères géneraux: reconnaissance des objets dans un univers 3D (les appareils dans l'armoire), possibilités de traitement d'image sans faire appel aux ressources centrales de calcul (système informatique distribué), traitement en temps réel.

Après l'étude de plusieurs systèmes de vision, notre choix s'est arrêté sur le système de vision IRI P256 d'International Robomation/Intelligence qui nous semble plus approprié à nos besoins.

Voilà les principales caractéristiques du système IRI P256:

ORDINATEUR	
Microprocesseur	MOTOROLA MC68000, 12 MHz
Mémoire	64 Kbyte SRAM, 64 Kbyte EPROM
Portes E/S	Syn/Asyn RS232/RS422 Prt1; RS232 Prt2
Système d'opération	RT/M real-time monitor/debugger
Timers	2 timers independants
Interruptions	8 single line vectored interrupts
Special	Self-test + diagnostic
DISCRETISEUR d'IMAGE	
Entree	4 cameras vidéo RS170
Sortie	Moniteur RS330
Résolution	256*256 pixels, 8 bits de profondeur
Frame Buffer Memory	256 Kbyte - peut accommoder 4 images
Sampling Rate	6 millions pixels par seconde
ENVIRONNEMENT DE PROGRAMMATION	
VCL = Vision Command Language	
Editeur on-line	
FORTH - programming environment	
Exécution des programmes de vision via RT/M	
Real-time moniteur à partir des modules compilés off-line	
C based (VCL) library	

Tableau 3.2: Les caractéristiques du système IRI P256

Le systeme d'émetteurs-recepteurs à ultrasons doit assurer une fonction de sécurité en détectant tout objet se trouvant en-deça d'une distance donnee (la distance de sécurité) pour éviter les collisions et déclencher l'alarme. Le système choisi est "Ultrasonic Ranging System" de la compagnie POLAROID. Ce système contient le capteur jouant le rôle d'émetteur

récepteur d'ultrasons et toute la partie électronique afférente pour pouvoir contrôler le fonctionnement du système dans son ensemble.

Le système d'émetteurs - récepteurs optiques (infra-rouges) doit assurer une deuxième fonction de sécurité dans une situation d'alerte en permettant au robot de se retrouver dans son environnement de travail quand il est "perdu" accidentellement (système de navigation défectueux par exemple) Le système choisi est un proximètre opto-électronique (infra-rouge) de la compagnie BALLUFF model BLS et BLE M18. Le système doit permettre au robot de connaître sa position et son orientation dans l'environnement de travail qui sera balisé. Le sous-système de guidage avec ses trois parties composantes sera détaillé dans le chapitre 5.

3.4 LE SUPPORT ENERGETIQUE

Le module de support énergétique a pour rôle d'assurer l'autonomie énergétique du robot pendant au moins 5 heures, soit le temps nécessaire pour la préparation et le déroulement du laboratoire. Il est constitué de:

- un groupe de batteries 24 VDC;
- un chargeur de batteries;
- un inverseur 24 VDC-115 VAC 50 c/s.

Les batteries assurent l'alimentation énergétique du système via l'inverseur 24 VDC-115 VAC. Pour assurer l'indépendance énergétique du robot, le module est prévu aussi avec un

chargeur de batteries capable de contrôler et d'indiquer l'état de charge des batteries et avertir lorsqu'il faut les récharger. Le système peut être alimenté également par le réseau électrique, ce qui pourra être utile quand il s'agit de l'utilisation du module informatique indépendamment pour des besoins de calcul.

3.5 LE SUPPORT INFORMATIQUE

Le module informatique constitue le cerveau de tout le système puisqu'au niveau décisionnel, tout se produit dans ce module. Le module est conçu de façon distribuée. L'une des raisons pour le choix d'une structure distribuée et non centralisée dans un seul gros processeur a été la suivante: par sa conception, le système étant un système physiquement distribué avec des processeurs multiples, une structure similaire au niveau du logiciel contribue à une efficacité et une robustesse accrues. Dans un système centralisé, il aurait été nécessaire d'avoir une vitesse d'entrée-sortie importante pour assurer un temps de réponse faible, ce qui est essentiel en robotique, alors qu'un système distribué peut avoir cette capacité par le traitement en temps différé. Donc notre structure est une structure hiérarchisée modulaire avec un coordinateur de haut niveau qui est notre module informatique.

La fonction essentielle de ce module est la coordination des opérations: navigation, contrôle des moteurs, guidage du bras manipulateur par rétroaction visuelle, reconnaissance des

appareils, planification des tâches. Du point de vue materiel, le module informatique est constitue par les elements suivants:

Micro-ordinateur	IBM-XT compatible
Compilateurs	Prolog, Pascal, C, Basic, Forth
Convertisseurs	A/N et N/A
Microprocesseur	MOTOROLA MC68000, 12 MHz et 8086
Mémoires	64K RAM, 64K EPROM
Discrétiseur d'image	4 cameras video RS170
Interface	RS232C
Logiciels	Navigation et guidage du robot
Logiciel	Planification des tâches

Tableau 3.3: Les composantes du module informatique

La structure informatique (fig.3.5) est celle qui contribue le plus à donner au robot sa capacité d'action dans le monde physique où il évolue. La nécessité d'action (et de réaction) en temps réel et avec un temps de réponse faible nous a amené à rejeter une structure constituée d'un seul microprocesseur et à concevoir plutôt une architecture multiprocesseurs.

Ainsi, un microprocesseur (8086) est utilisé pour la commande des moteurs des roues (navigation), un autre (MC68000) pour le traitement de l'image vidéo et la gestion du système de vision, et le IBM-XT sert comme ressource de calcul pour l'élaboration et la génération de plans et aussi ce dernier permet la communication avec l'utilisateur.

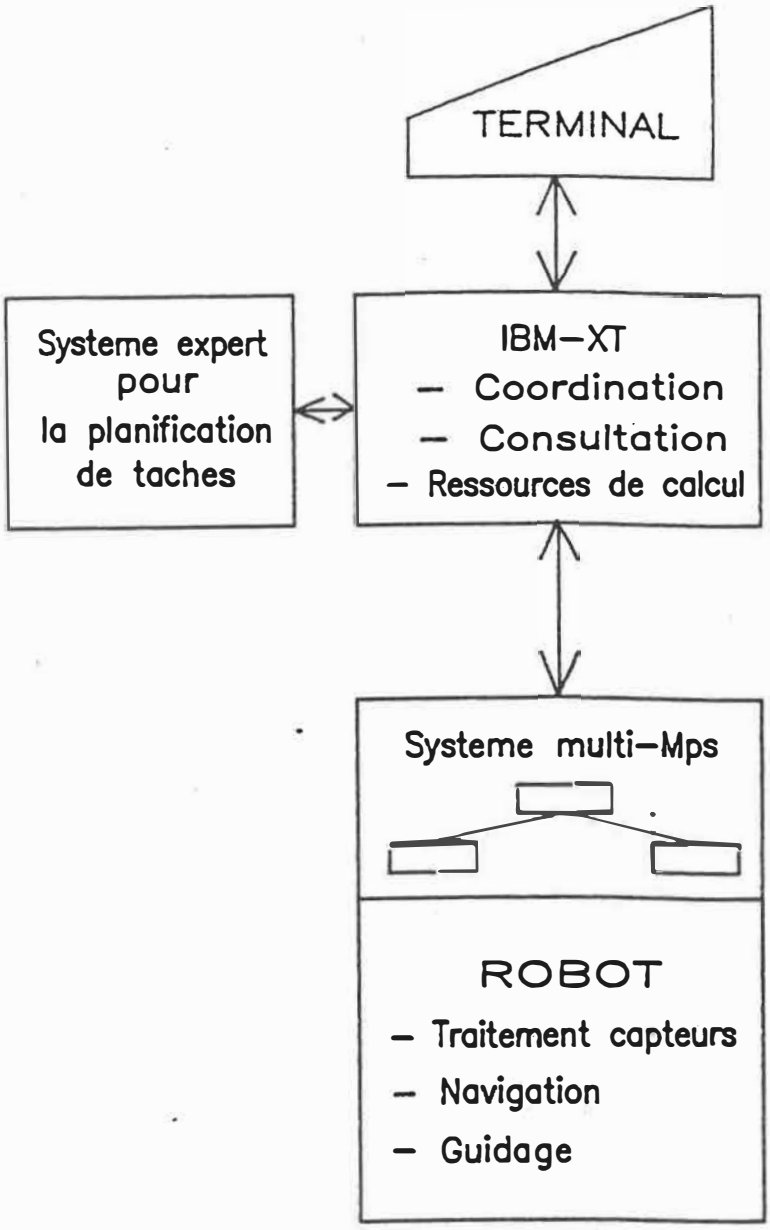


Fig.3.5 Structure informatique du système robotisé.

CHAPITRE 4

LE SOUS-SYSTEME DE NAVIGATION

4.1 INTRODUCTION

Le sous-système de navigation a la fonction d'assurer la circulation du robot pour:

- le déplacement dans l'espace de travail (laboratoire);
- chercher les appareils dans l'armoire;
- distribuer ensuite les appareils sur les tables de travail;
- eviter les obstacles.

Pour résoudre le problème de la navigation du robot, il y a plusieurs solutions possibles:

- la navigation inertielle qui utilise des gyromoniteurs pour contrôler les changements de direction, des capteurs sonars pour la détection des obstacles et la prévention des collisions; tout peut être contrôlé par microprocesseur pour guider le robot au long d'un trajet programmé d'avance. C'est une solution coûteuse.
- la navigation en infra-rouge qui utilise des capteurs à rayons infra-rouges pour mesurer des distances, pour la détection des obstacles et des réflecteurs pour trouver la position exacte du robot par triangulation. De la même manière, on peut utiliser des capteurs à rayons laser (le robot HILAIRE de LAAS par exemple) ou des capteurs à ultra-sons permettant des mesures de distance et même des mesures de vitesse (par exemple l'utilisation de l'effet Doppler).

- la navigation en champ magnétique qui utilise un "trajet guide" fabriqué avec un fil collé sur le plancher par exemple, celui-ci étant le siège d'un champ magnétique, et une antenne montée sur le robot, couplée avec un contrôleur à microprocesseur pour guider le robot au long du trajet. C'est une solution compliquée et coûteuse.
- la navigation optique qui utilise des capteurs actifs émettant un rayonnement lumineux. Leur portée, de quelques centimètres à quelques dizaines de centimètres, se trouve dans la zone intéressante pour la proximétrie du robot (fig.4.1).

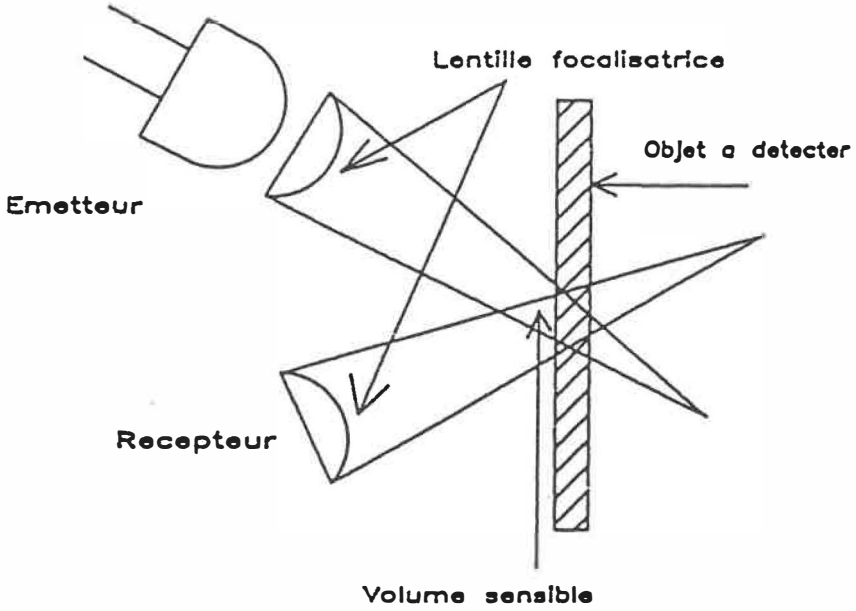


Fig.4.1 Principe d'un capteur proximètrique photoélectrique

Ils permettent la détection des obstacles et des mesures de distance. C'est la solution que nous avons choisie pour raison de simplicité, efficacité, et aussi parce que la méthode a été déja expérimentée à l'école avec des bons résultats sur un robot Scorpio.

La méthode qui a été retenue pour la navigation est une adaptation de la navigation optique à notre application. On utilise un ruban lumineux fixé au sol (ruban adhésif blanc) et on fait suivre cette ligne blanche par le robot. Alors il suivra la trajectoire voulue, quel que soit le nombre de fois que l'itinéraire est répété. Le robot est alors confiné à un itinéraire déterminé d'avance qui est tracé sur le plancher.

4.2 LE PRINCIPE DE NAVIGATION

En dessous du chariot mobile, qui supporte le bras manipulateur, entre les roues d'en avant est monté un dispositif de navigation au sol (voir fig.4.2 et 3.4) Il est constitué de deux lampes éclairantes et de quatre capteurs photo-électriques disposés d'une part et d'autre du dispositif. Le bon fonctionnement est assuré par un contraste suffisant. Alors on utilise un ruban adhésif reflétant et un autre non reflétant. Etant donné que le robot doit suivre le tracé blanc (ruban reflétant) l'algorithme de commande prend des mesures correctives lorsque le robot quitte sa trajectoire. Il y a neuf états possibles que le dispositif de navigation peut transmettre et pour lesquels il faudra agir. Ces cas sont illustrés dans le tableau 4.1. La flèche indique le sens du déplacement du robot, les vues sont en plan horizontal. Les cercles avec les lettres A, B, C et D représentent les capteurs (gauche et droite). L'algorithme de commande doit avoir des renseignements concernant l'état des capteurs. Alors, suite à

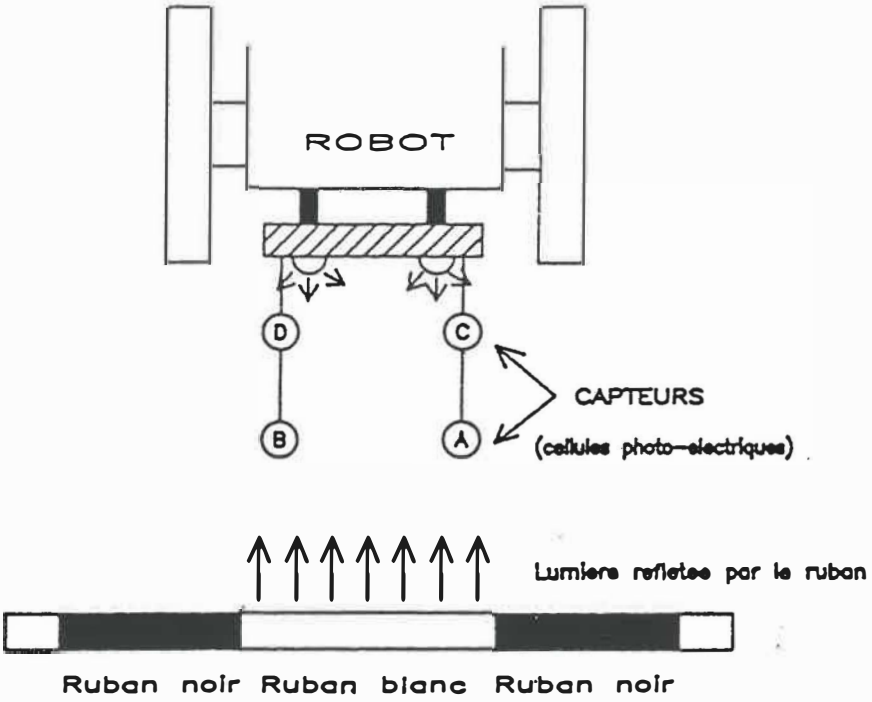


Fig.4.2 Illustration du principe de navigation.

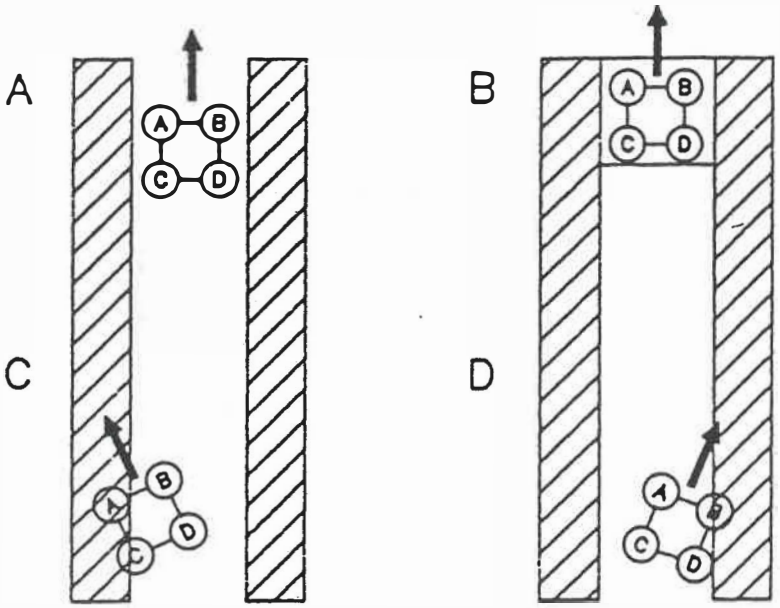


Fig.4.3 Les quatre situations possibles sur la voie de navigation

une commande d'interrogation des capteurs, on aura les situations suivantes:

Capteur	1	2	3	4	5	6	7	8	9
A	B	N	N	B	B	N	N	N	N
B	B	B	B	N	N	N	N	N	N
C	B	B	N	B	B	B	N	N	B
D	B	B	B	B	N	B	N	B	N
Correction	OK	-->	-->	<--	<--	STOP	STOP	-->	<--

Tableau 4.1 Les différents états des capteurs

D'abord les actions possibles du robot:

- avancer
- reculer
- déplacement vers la droite
- déplacement vers la gauche
- tourner vers la droite
- tourner vers la gauche

Signification des abréviations utilisées dans la fig.4.5:

B - le robot se trouve sur le ruban blanc (reflétant);

N - le robot se trouve sur le ruban noir (non reflétant);

OK - la correction n'est pas nécessaire, on est en plein centre

du ruban blanc;

--> - correction nécessaire vers la droite, on est trop à gauche du ruban blanc (débordement par la gauche);

<-- - correction nécessaire vers la gauche, on est trop à droite du ruban blanc (débordement par la droite);

STOP - on est au bout du ruban blanc ou sur une discontinuité du ruban guide, il faut arrêter le robot.

Les cas 6 et 7 sont très importants parce qu'il y a deux possibilités:

1. Le robot est arrivé au bout du ruban réflectif;
2. Le robot a atteint une discontinuité dans le ruban.

Alors le robot doit être arrêté. Il est impossible d'en arriver au cas 6 ou 7 en s'écartant du ruban vers la droite ou gauche, parce que dans une telle situation, on atteindrait automatiquement un des cas intermédiaires. Ces deux discontinuités sont utilisées pour indiquer au robot:

- les points de halte aux différentes destinations;
- les points où il faut faire un tournant.

4.3 LES MESURES CORRECTIVES

Voilà maintenant le problème des mesures correctives pour les différentes situations dans lesquelles le robot peut se trouver. Il y a quatre situations différentes pour le robot par rapport au ruban (voir fig.4.3). Dans chacune des situations, il va falloir agir en conséquence:

a) Les quatre cellules photo-électriques sont allumées, car

elles sont encore au-dessus du ruban reflétant (blanc); on est sur la bonne trajectoire. Il n'y a pas de manoeuvre requise, on continue à avancer.

b) On est sur une discontinuité du ruban. Alors, il y a plusieurs actions possibles:
   - Faire une halte temporaire;
   - Vérifier si on est à la destination - c'est-à-dire si on est à une table de travail où il faut s'arrêter pour décharger ou charger des appareils;
   - Sinon, trouver la direction de la prochaine station;
   - Si le robot n'est pas dans la bonne direction, effectuer un pivot;
   - Se remettre en marche.

c) On vient de déborder du ruban par la gauche:
   - Il faut s'arrêter;
   - Reculer d'environ 3 cm;
   - Faire un pivot de 10° ou 20° vers la droite (10° dans le cas 3, 20° dans le cas 2);
   - Se remettre en marche.

d) On vient de déborder du ruban par la droite, alors la procédure est la même, sauf qu'on fait le pivot vers la gauche avant de se remettre en marche.

Sur la fig.4.4, on illustre la méthode de recouvrement du cas c). Il y a deux facteurs déterminants dans la performance des mesures correctives:

a) La largeur du ruban de guidage;

b) L'angle de pivot à l'étape c).

Pour une étude approfondie des problèmes soulevés par la navigation du robot, concernant les mesures correctives nécessaires pour assurer l'orientation sur le trajet à suivre, la seule solution possible dans notre cas est la simulation sur l'ordinateur.

Un programme complet de simulation doit permettre la variation des paramètres suivants: la largeur du ruban, la vitesse de déplacement du robot, le delai d'arrêt, la géométrie des capteurs et la vitesse de correction. Une interface graphique permettant l'animation du robot sera très souhaitable sinon indispensable pour voir les mouvements de déplacement, les déviations de la trajectoire et les mouvements de correction nécessaires pour ramener le robot sur la bonne trajectoire. Un tel programme dépasse le cadre de notre projet et pourra faire l'objet d'un développement ultérieur.

Néanmoins, nous avons développé un programme simplifié de simulation, permettant l'apréciation qualitative des différentes solutions possibles concernant la navigation du robot. Le programme de simulation nous a permis de tirer certaines conclusions utiles que nous allons exposer par la suite. La marche à suivre pour le développement du programme de simulation est montré sur la fig.4.5 et le programme se trouve dans l'annexe.

a) Influence de la largeur du ruban

Pour déterminer l'influence de la largeur du ruban-guide, nous avons procédé par simulation sur l'ordinateur. Le programme de simulation permet la variation des paramètres

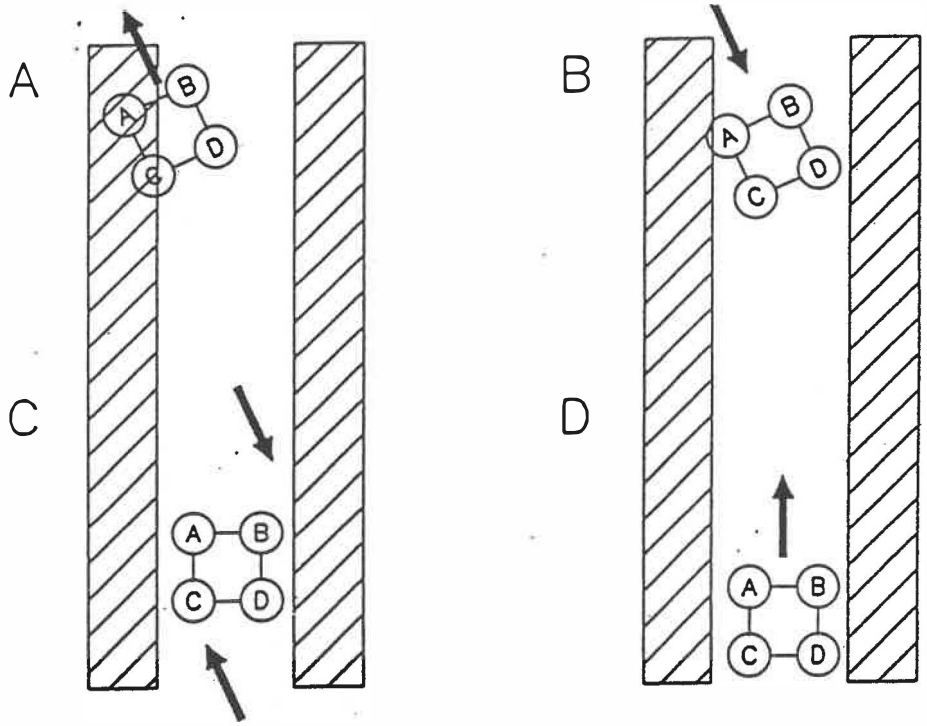


Fig.4.4 Procédure de recouvrement

a) S'arrêter

b) Reculer de 3 cm

c) Pivoter

d) Se remettre en marche

suivants: la largeur du ruban, la vitesse de déplacement du robot, la géométrie des capteurs et la vitesse de correction.

Après la simulation pour déterminer la géométrie des capteurs, nous avons adopté la solution suivante: les quatre

capteurs sont placés dans les sommets d un rectangle de 4cm de longueur et de 2cm de largeur pour avoir une bonne sensibilite et un temps de reponse court du système. Une autre configuration géométrique (carré ou rectangle longitudinal par rapport au ruban) implique un recul plus important avant d'appliquer le pivot de correction.

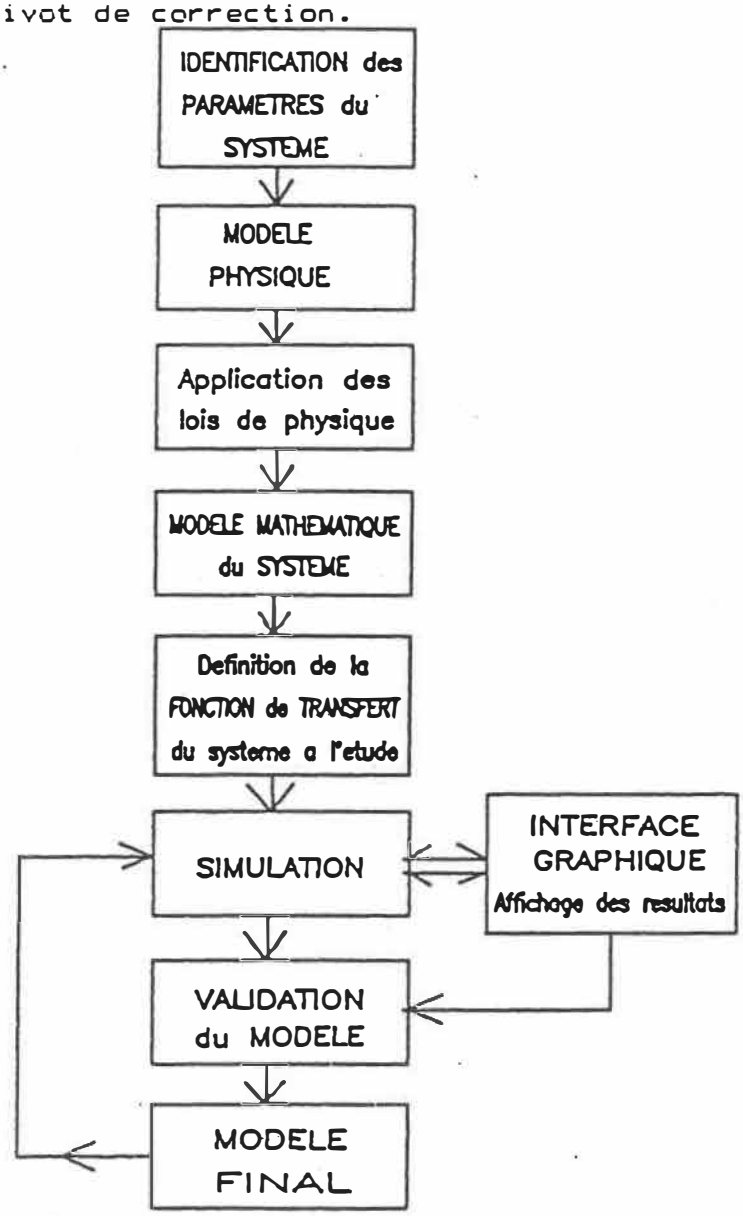


Fig.4.5 Marche a suivre pour developper le programme de simulation

Concernant la largeur du ruban-guide (ruban blanc), il était évident, après la simulation, qu'elle avait un effet assez important sur la performance du système. Lorsque le robot quitte sa trajectoire, la largeur du ruban influencera la fréquence des manoeuvres de recouvrement et la capacité du système à réagir convenablement pour assurer la correction de la trajectoire. Voilà les conclusions après avoir fait la simulation pour déterminer la largeur du ruban: si le ruban est très serré par rapport aux capteurs, il y a risque d'avoir deux capteurs désactivés quand le robot se met à travers le ruban, et alors le système ne pourra prendre une décision. Si le ruban est large et qu'après une correction le robot se trouve plus proche d'un ruban noire (droite ou gauche) et qu'il avance vers le ruban oposé, l'erreur d'orientation s'accumule ce qui va imposer une correction plus forte (de l'angle de déviation), donc plus de temps de recouvrement. Si le débordement se fait par l'avant (cellule A ou B), la correction se fait par recul, suivie de pivotement; si le débordement se produit par l'arrière (ce qui se produit quand le robot avance presque tangent au ruban noir), la correction se fait par avance suivie de pivotement. Un ruban large permet plus de déviation par rapport à la bonne trajectoire, surtout quand le robot avance plus vite. Si le ruban-guide dépasse seulement avec 0.5 cm d'un coté et de l'autre, ce qui donne une largeur du ruban de 5 cm pour une distance de 4 cm entre les capteurs, on obtient la meilleure situation du point de vue correction de la

meilleure situation du point de vue correction de la trajectoire, en ce sens que le robot n'aura pas le temps de se mettre à travers et les manoeuvres de recouvrement seront plus rapides (moins de recul et angle de pivotement plus petit). Alors le recouvrement se fera par recul ou avance de 3cm, et pivotement de 10° ou 20°. La fig.4.6 montre l'influence de largeur du ruban-guide.

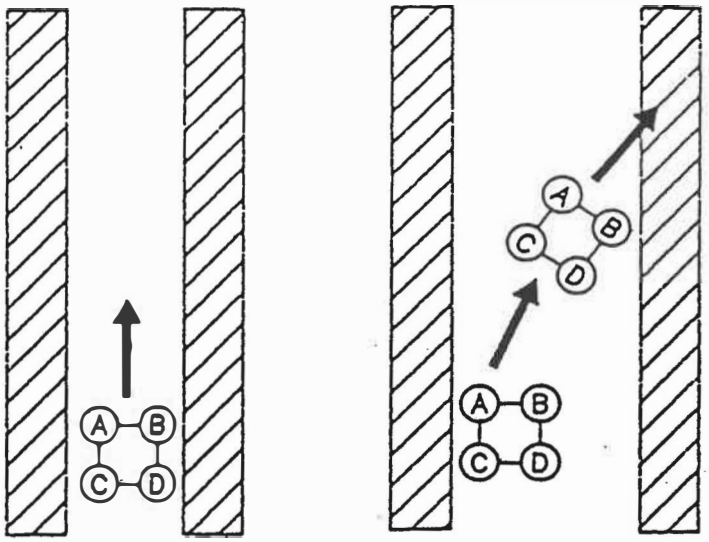
b) Influence de l'angle de pivotement

L'angle de la manoeuvre de pivotement influence également la performance du système. Il y a deux possibilités (voir fig.4.7 et fig.4.8):

1) Un angle trop grand va augmenter le nombre des manoeuvres étant donné qu'on peut dépasser la position correcte de l'autre côté et alors il faut revenir avec une autre correction en sens inverse, ce qui complique la manoeuvre de recouvrement.

2) Un angle trop faible va imposer de refaire la manoeuvre plusieurs fois avant d'être de nouveau sur la trajectoire, mais par contre la précision sera meilleure. Dans le cas du débordement du ruban-guide (par la gauche ou par la droite), l'angle de pivotement est soit de 20° quand il y a un seul capteur dans la zone noire, soit de 10° quand il y a deux capteurs dans la zone noire parce qu'alors l'angle de déviation et plus petit (déviation par glissement). Quand le robot arrive sur une discontinuité du ruban, il doit faire un STOP et alors soit qu'il fasse un pivot de 90° (dans un sens ou dans l'autre)

simplement s'il s'agit d'une intersection.



Ruban mince Ruban trop large.

Fig.4.6 Influence de la largeur du ruban guide.

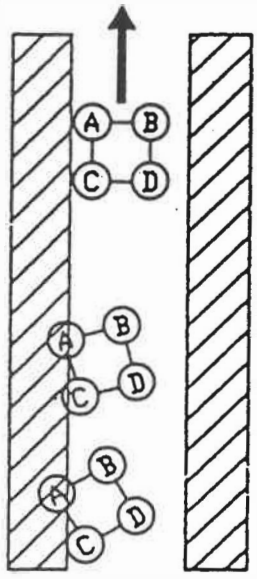
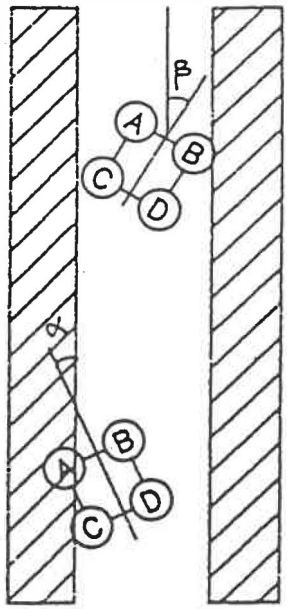


Fig.4.7 Inconvénients d'un angle trop faible.



L'angle de correction = $\alpha + \beta$

Fig.4.8 Inconvénients d'un angle trop grand.

Jusqu'ici, nous avons vu:

- Le principe de navigation - qui nous montre comment le robot peut circuler d'un point à l'autre en suivant le ruban;
- Les mesures correctives qu'on doit appliquer;
- L'influence de deux paramètres:

1) la largeur du ruban guide

2) l'angle de pivotement

4.4 LA CIRCULATION DU ROBOT DANS LE LABORATOIRE

Le robot doit être capable de "prendre des décisions" qui vont lui permettre de circuler dans le laboratoire pour s'acquitter de sa tâche. Pour cela, il faut analyser ce que le robot doit faire au niveau de la circulation et essayer de

décomposer cette tâche en tâches élémentaires (de deplacement). Nous avons vu le principe de navigation du robot pour permettre son déplacement. On va étudier et solutionner les problemes liés à la circulation du robot dans l'environnement de travail (laboratoire). Si nous regardons la fig.7 qui représente le trajet de circulation du robot (géometrie de l'environnement de travail), nous remarquons les choses suivantes:

- Le trajet principal de circulation est dirigé nord-sud sur l'axe géométrique du laboratoire.
- Les trajets secondaires de circulation sont dirigés perpendiculairement sur le trajet principal en direction est-ouest. Il y a 6 trajets secondaires, c'est-à-dire 5 pour la circulation entre les tables de travail, et 1 pour la circulation au long des deux armoires pour permettre au robot de faire la recherche des appareils et ensuite la distribution sur les tables de travail.
- L'origine du trajet de circulation se trouve à l'intersection du trajet principal avec le premier trajet secondaire (parallèle avec les armoires).
- Les 6 intersections dont une est particulière - l'origine.

Elles portent des numéros d'identification qui servent de code pour identifier chaque endroit où le robot peut se trouver.

Le code sera lu par une caméra dans chaque intersection et transmis au programme de navigation. La caméra sera une des quatre caméras du système de vision affecté spécialement pour le sous-système de navigation.Elle sert uniquement à identifier

l intersection, est fixe, montée sur le côte droit de la deuxième plate-forme du robot de maniere a pouvoir lire les codes inscrits sur les tables de travail.

L'origine est identifiée par 00, et les autres par deux chiffres representant les numéros des tables adjacentes au trajet: 01 - pour la premiere (tables 1 et 10), 29 pour la deuxième (tables 2 et 9), 38 pour la troisième (tables 3 et 8), 47 pour la quatrième (tables 4 et 7), et 56 pour la cinquième (tables 5 et 6). Nous allons appeler les intersections - stations. Il y a 6 stations: l'origine (#00), 4 intersections (#01, #29, #38, #47) et le terminus (#56).

Du point de vue de la circulation du robot, les stations sont différentes:
Dans la station-origine, il faudra effectuer une manoeuvre: un pivotement de 90° dans le sens horaire pour changer de direction.

Dans les stations-intersections, il faudra effectuer deux manoeuvres: un pivotement de 90° pour les tables du coté droit du laboratoire (1,2,3,4,5) ou de -90° pour celles du coté gauche du laboratoire (6,7,8,9,10) pour décider dans quelle direction se diriger, et l'inverse pour le retour.

Dans la station terminus (#56), il faudra effectuer une manoeuvre de plus, donc à part les pivotements nécessaires pour se rendre aux tables de travail, il faut, à la fin, faire un pivotement de 90° ou -90° dépendamment de la table #5 ou #6 d'où il vient pour changer la direction pour revenir au point

de départ. Toujours, du point de vue circulation, le robot doit pouvoir effectuer les manoeuvres suivantes:

- Avancer
- Reculer
- s'Arrêter
- Pivoter de 90° (dans le sens horaire) ou de -90°.

L'algorithme du programme principal est presente sur la fig.4.9

Du point de vue decisionnel, il faut faire la distinction entre l'intersection et la destination. L'intersection doit exister forcément pour pouvoir permettre au robot le changement de direction pour se rendre à la destination - une table de travail ou l'armoire. Avant d'arriver à la destination, il faut passer par une intersection appelée "intersection-destination" pour faire la différence par rapport à une "intersection-ordinaire" qu'on doit simplement traverser. Alors, on se rend compte que la décision de qualifier une intersection comme "ordinaire" (à traverser) ou comme "destination" est prise d avance au niveau de la tâche (à un niveau décisionnel plus haut). Pour savoir toujours dans quelle direction se déplace le robot, il faut définir une variable gardée tout au long du programme, qui contient la direction à laquelle le robot fait face et qu'on doit changer à chaque manoeuvre de pivot. Pour permettre au robot de reconnaître une intersection (en géneral) une discontinuité du ruban va indiquer une station. La discontinuité est créée avec un ruban noir qui marque chaque intersection.

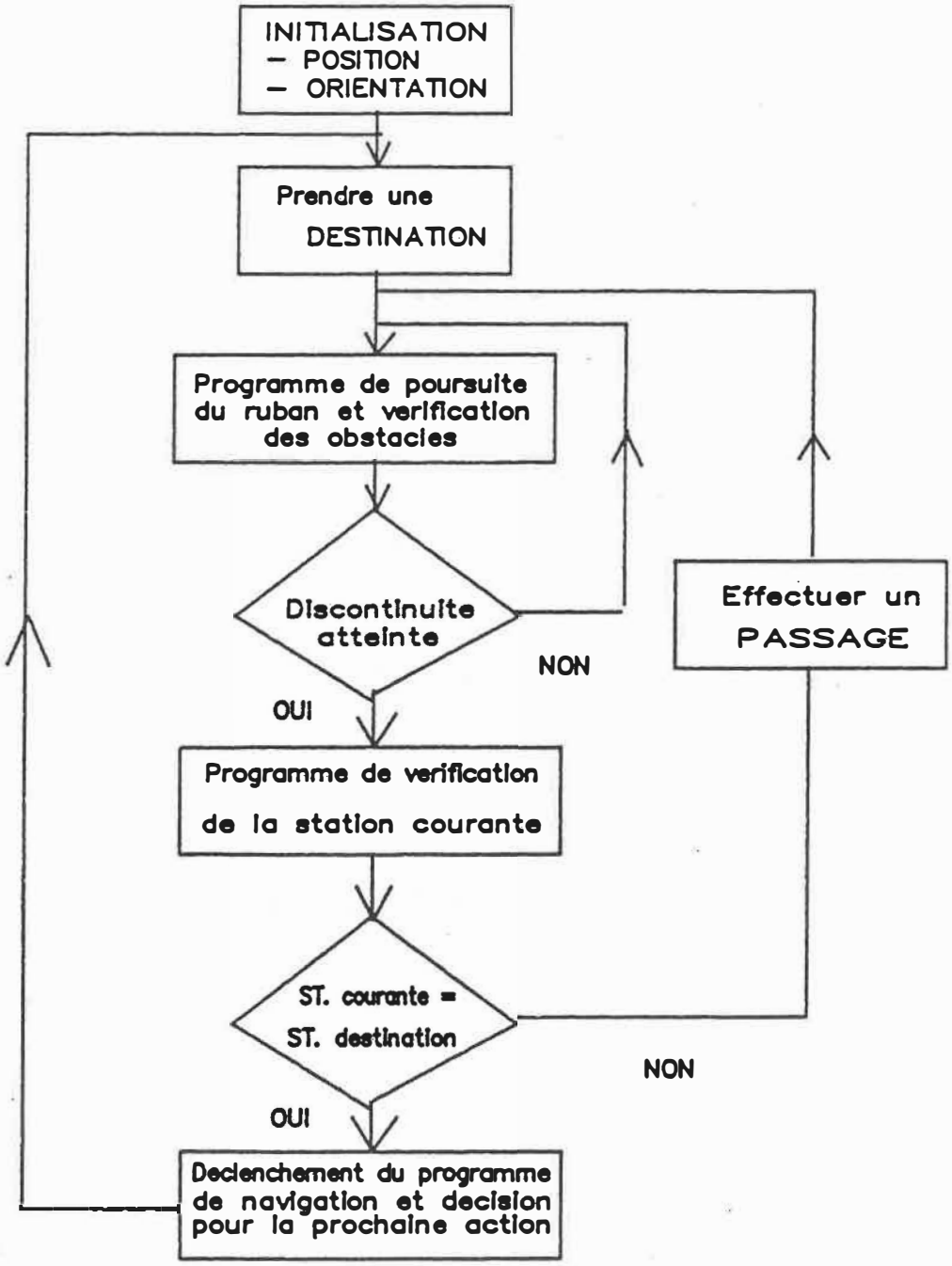


Fig.4.9 Algorithme du programme principal

Dans une intersection ordinaire, lorsqu'on arrive à une discontinuité du ruban indiquant une station, mais qui n'est pas une destination, et qui doit être simplement traversée, on suit la procédure suivante:

- En prenant le numéro de la station courante (là où on est arrêté - arrivé), le programme se branche à une sous-routine réservée à cette station.
- A l'aide du numéro de la destination qu'on doit atteindre, la sous-routine trouve la direction dans laquelle il faut se diriger à partir de la station courante. C'est-à-dire, soit faire un pivot de +/-90° (dépendamment de la table de travail où le robot doit se rendre) si la station courante est la destination, soit continuer le trajet dans le cas contraire.
- Les deux directions: la direction actuelle et la direction à prendre (trouvées par la sous-routine) sont envoyées à une sous-routine de pivot.

La sous-routine de pivot doit déterminer la manoeuvre à exécuter:

- pivot de +/-90° dans le cas de la station d'origine (#00) venant d'une armoire;
- pivot de 0° (dans le cas d'une station "ordinaire" à traverser) c'est-à-dire ignorer la discontinuité et traverser l'intersection pour se rendre à la destination;
- pivot de +/-90° en venant d'une table vers le point de départ pour déplacement vers les armoires et traversée des stations "ordinaires".

La procedure est recursive et on recommence a chaque station la décision est prise à chaque station. Pour s'assurer que les manoeuvres placent le robot correctement pour pouvoir suivre son chemin, il faut aussi une procédure de sûrete:

1. détection de la discontinuite;
2. délai de temps avant l'arrêt pour que le robot arrive au milieu de la station;
3. arrêt suivi de la procédure intersection (pivotement si nécessaire);
4. démarrage et, après un certain délai (nécessaire pour dépasser la discontinuité), interrogation des photo-transistors pour s'assurer que le robot est de nouveau sur le ruban blanc.

Dans une intersection-destination, on applique la procédure destination:

- Si le numéro de la station courante est égal au numéro de la station destination (test), alors le programme se branchera à une sous-routine "guidage" qui est à un niveau décisionnel plus haut (niveau de la tâche) et qui doit répondre à une question du type:
  - vers quelle table faut-il se déplacer? (gauche ou droite)
  - pivot de +90° si la réponse est "table-est" (à droite).
  - pivot de -90° si la réponse est "table-ouest" (à gauche).

Remarques:

a) La nécessité de la procédure-destination démontre clairement l'affirmation faite au chapitre 2.2 que la rélation décision - action est très difficile à maîtriser et qu'elle est très intéractive.

b) A l'application de la procédure-destination, nous faisons la connexion entre les deux niveaux décisionnels:
   - au niveau de la circulation du robot (niveau bas);
   - au niveau du guidage (niveau haut - niveau de la tâche) où il y a l'intervention des "sens" du robot, et c'est là l'intervention de l'aspect "intelligent".

CHAPITRE 5

LE SOUS-SYSTEME DE GUIDAGE

La fonction du sous-système est d'assurer le guidage du bras manipulateur par rétroaction visuelle pour permettre l'exécution de la tâche et d'assurer la sécurité de déplacement du robot dans l'environnement de travail. Le guidage du robot est assuré par le biais de trois systèmes:

- un système de vision
- un système d'émetteurs-récepteurs à ultrasons (sonar)
- un système d'émetteurs-récepteurs optiques (infra-rouge)

5.1 LE SYSTEME DE VISION

Le système de vision permettra l'analyse de la scène. Il y a deux scènes qui devront être analysées: une première scène qui est constituée de deux armoires et une deuxième scène constituée d'une table de travail.

La première sera analysée pour identifier les objets (appareils) dans l'espace de recherche (armoires). La deuxième sera analysée pour établir si la table est complète. La table est considérée complète si tous les appareils nécessaires à la manipulation en cours se trouvent sur la table.

Les résultats obtenus seront utilisés pour commander le bras manipulateur, soit pour prendre des objets, soit pour déposer des objets. Pour remplir cette tâche, le système de vision doit

être muni d'une interface avec le bras manipulateur assuré par le programme de guidage. Ce programme aura comme entrées les résultats des analyses de la scène, et les sorties seront des commandes envoyées au bras manipulateur. Pour l'analyse de la scène, le système de vision fait d'abord l'apprentissage et ensuite la reconnaissance.

Pour simplifier le problème d'identification des objets de la scène, les appareils seront marqués par un code. Le code doit permettre l'identification du type d'appareil et l'appartenance à une manipulation. Il sera formé par deux marques: la première pour indiquer la manipulation, et la deuxième pour indiquer le type de l'appareil. Les marques seront en matériel reflétant blanc pour assurer un bon contraste et elles seront collées sur chaque face de l'appareil au centre géométrique de la face. La marque pour identifier la manipulation sera: un carré (1cm de coté) pour la manipulation No 1, un triangle isocèle (1cm de coté) pour la manipulation No 2, un cercle (avec un rayon de 0,7cm) pour la manipulation No 3, et un demi-cercle (de même rayon) pour la manipulation No 4. Pour identifier le type de l'appareil, la deuxième marque placée à droite de la première sera un chiffre romain. De cette façon, le système sera capable de faire l'identification des appareils pour répondre à une requête du système.

Le système de vision IRI P256 est un système d'analyse des images destinées aux applications industrielles. Le système

offre un rapport prix/performance très intéressant. Les hautes performances du système sont obtenues grâce à une architecture-multiprocesseurs dans laquelle on retrouve deux microprocesseurs dédiés:

- Un microprocesseur MOTOROLA MC68000 rapide (12 MHz) qui sert à supporter les fonctions du système d'opération et à l'exécution des programmes d'application; Il est capable d'exécuter 1 million d'instructions par seconde (MIPS).
- Un préprocesseur qui permet différentes fonctions de prétraitement de l'image comme par exemple la transformation des points, la segmentation de l'image et des opérations d'extraction. Il est capable d'exécuter 40 millions d'opérations par seconde (MOPS) en opèrant en mode SIMD (Single Instruction - Multiple Data) et de transformer une image de 256*256 pixels dans 17ms.

5.2 LES SYSTEMES DE SECURITE (SONAR et INFRA-ROUGE)

Le système de guidage est muni de deux systèmes de sécurité:
- le premier permet la détection des obstacles et donne l'alarme;
- le deuxième permet au robot de se retrouver dans son environnement de travail quand il est "perdu" pour une raison quelquonque.

Le premier système de sécurité qui permet d'éviter les collisions est un système d'émetteurs-récepteurs à ultrasons. Ce système permet la détection des obstacles se trouvant sur le chemin du robot (une chaise, un étudiant, etc.). Après la détection de l'obstacle, le robot envoie un signal sonore d'avertissement et freine son mouvement. Si l'obstacle reste toujours sur son chemin, il s'arrête pendant un délai de temps (60 secondes par exemple), il envoie le signal sonore d'alerte (une autre fréquence audio que celle d'avertissement). Si la situation persiste après ce délai, l'alimentation est coupée. Ce système sonar est conçu pour des situations imprévues dans l'environnement de travail pour assurer la sécurité du robot.

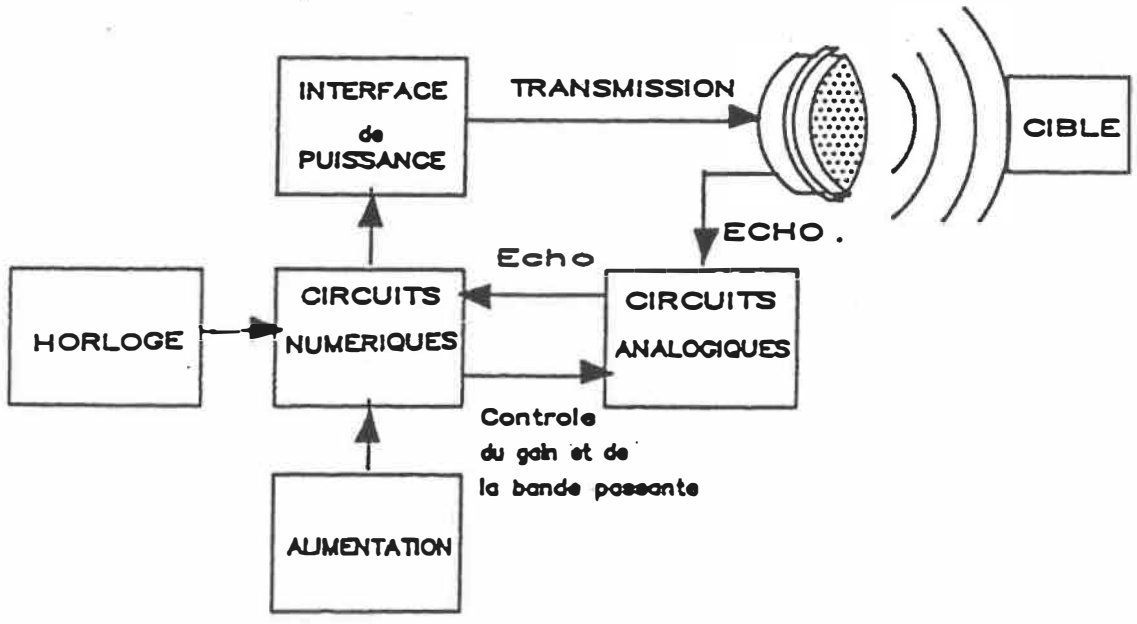


Fig.5.1 Schema bloc de l'emmeteur-recepteur a ultrasons.

L'autre système de sécurité permet au robot de se retrouver dans son environnement; il est un complément du système de navigation pour des situations d'alerte. Ce système utilise des émetteurs-récepteurs optiques fonctionnant en infra-rouge. Trois émetteurs optiques sont disposés de manière à réaliser deux axes hortogonaux dans l'environnement de travail: deux émetteurs seront fixés dans les coins du laboratoire, et le troisième dans le coin nord pour réaliser l'axe vertical au long du mur parallèle avec l'axe nord-sud du laboratoire. En faisant tourner le système optique à partir de sa position de repos (qui est celle de l'axe de symétrie du robot) et en mesurant les angles entre cette position et celle où les émetteurs sont détectés, on peut déterminer les coordonnées et l'orientation du robot dans le référentiel des émetteurs pour pouvoir prendre la décision appropriée.

Le système sera capable de donner l'alerte si l'angle limite est dépassé, ce qui correspond à une sortie du robot en dehors du ruban guidé (voir fig.5.2). Si l'alerte est déclenchée alors la procédure d'alerte (I) doit s'exécuter. Cette procédure doit stopper d'abord le robot et ensuite si le débordement a été produit au bout du ruban en direction nord, de faire un pivotement de 180° pour changer la direction.Alors le contrôle sera fait par les deux autres émetteurs placés sur l'axe horizontal. Dans ce cas, il y a deux paramètres à surveiller:
- la variation de l'angle avec lequel le système optique tourne autour de son propre axe de symétrie sera:

$$\alpha_{min} \leq \alpha \leq 180^{o}$$

où α_{min} correspond à la station 56 - la table la plus éloignée- et où 180^{o} correspond au point de départ quand le robot sera sur l'axe horizontal; cela veut dire que les deux émetteurs seront à 90^{o} par rapport à l'axe du système optique d'un côté et de l'autre (voir fig.5.2).

- l'égalité des angles α et β correspondant aux deux émetteurs placés sur l'axe horizontal pour le cas de déplacement correct du robot sur le ruban guide. Si par exemple le robot, pour une raison quelquonque, ne peut plus être ramené par son propre système de navigation sur le ruban, alors un des deux angles sera sensiblement plus grand que l'autre (on accepte une erreur de 5%) et le système devra donner l'alerte. Cette fois-ci, il y aura une autre procédure d'alerte (II) qui sera exécutée parce que le débordement du ruban a été produit sur le ruban (d'un coté ou de l'autre) pendant le déplacement sur l'axe nord-sud en direction du point de départ. Alors, il faudra simplement faire la correction nécessaire pour ramener le robot sur le ruban-guide.

Ce système optique travaillant en infra-rouge complète d'une façon éficace le système optique de navigation décrit au chapitre 4 pour des situations de détresse en assurant le déplacement du robot d'une façon sécuritaire.

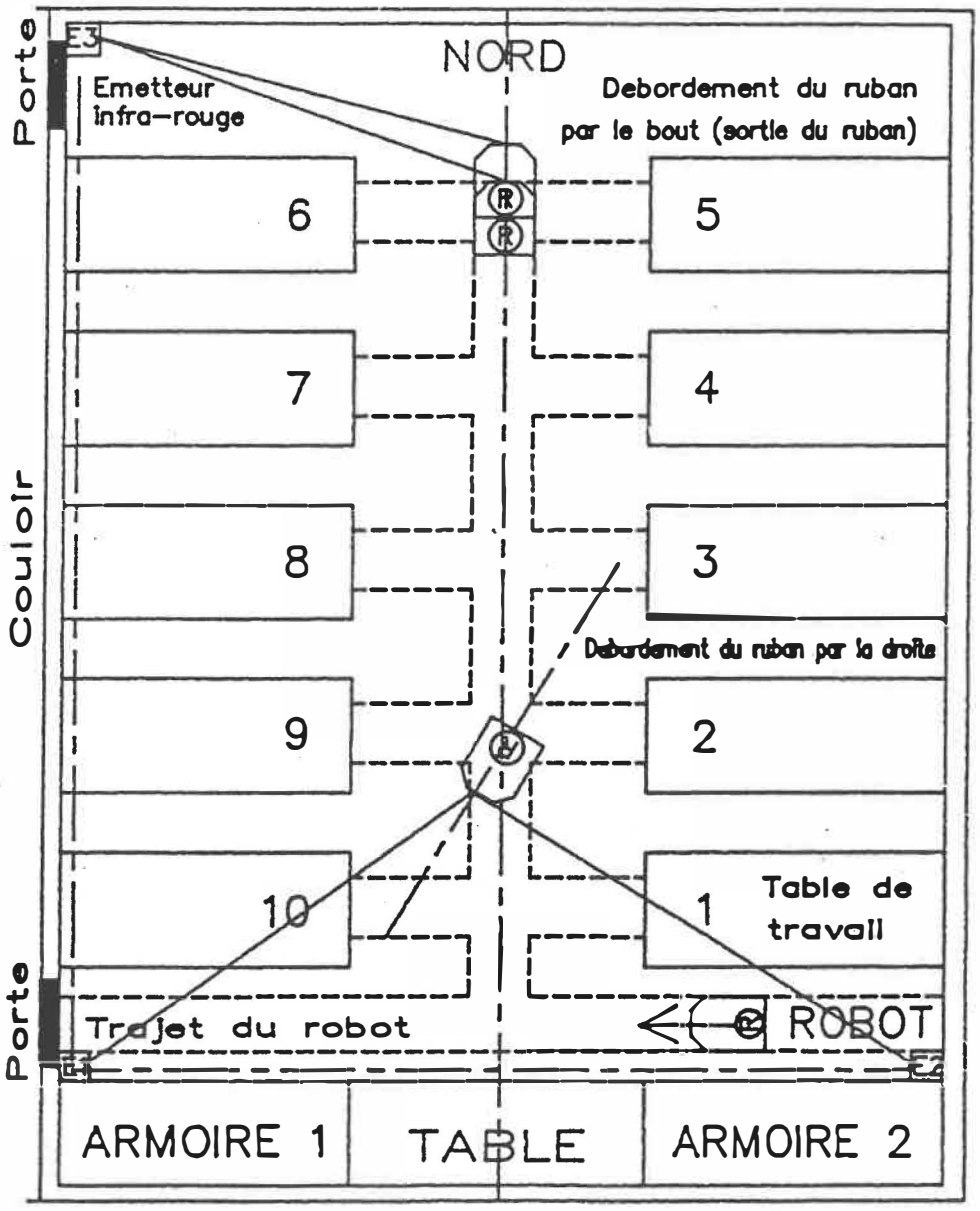


Fig.5.2 Système a emmeteurs-recepteurs infra-rouge

CHAPITRE 6

COMPOSANTES LOGICIELLES DU SYSTEME EN ETUDE

GENERATEUR DE PLANS DIRIGE PAR LE BUT

6.1 INTRODUCTION

Avant d'entrer dans les détails de ce chapitre, nous tenons a apporter quelques précisions. Dans le chapitre 6, ainsi que dans le chapitre 7 nous allons aborder essentiellement le problème de génération de plan par deux angles de vue différents. Toutefois, nous avons utilisé dans la mesure du possible le même cheminement dans les deux cas.

Le plan sera dans les deux cas une séquence d'opérateurs (d'actions) permettant au système de passer de l'état initial à un état final où le but proposé est satisfait.

Nous avons procédé dans les deux cas de façon structurée en respectant le cheminement suivant:

- Enoncé du problème (voir section 6.4 et 7.2)
- Solution du problème (voir section 6.5 et 7.3)
- Formalisation mathématique du problème (voir section 6.2, 6.5 et 7.4)
- Implantation du programme à l'aide de Turbo-Prolog dans le premier cas et avec VP-Expert dans le cas du système expert.

Dans ce chapitre nous allons traiter la planification de tâches et la génération de plan en faisant l'analogie avec le

problème de blocs. Dans la section 6.2 nous avons introduit la notion de "planificateur de tâches" qui peut être interpréter comme un régulateur d'état avec un formalisme mathématique particulier. Le générateur de plan universel Warplan (section 6.6) a été étudié à titre d'exemple avec sa propre formalisation pour saisir différents aspects du problème. Evidement l'implantation d'un générateur de plan peut être faite avec d'autres langages, l'utilisation du Prolog étant un choix de l'auteur tout simplement.

Dans le chapitre suivant, il s'agit d'un système à base de connaissance traité de la même façon du point de vue démarche mais avec d'autres moyens au niveau de la formalisation, l'approche étant différente dans le sens qu'on utilise un "scénario" et des règles de production qui supposent une expertise préalable qui sera enregistrée dans la base de connaissance du système.

Nous allons aborder dans ce chapitre le problème des capacités de déduction d'un système robotisé intelligent comme celui qui fait l'objet de notre projet. Nous allons traiter plusieurs points: la planification de tâches (dans un monde de blocs) faite par un générateur de plans, sa nécessité et son importance, le problème de blocs et sa résolution vue comme approche pour illustrer la faisabilité d'un générateur de plans Warplan (le générateur de plans universel), l'implantation de notre générateur de plans (une version Turbo-Prolog de Warplan)

Pour parler de la planification de tâches et de la génération de plans, nous allons aborder un exemple concret issu de notre projet. Une tâche importante de notre système robotisé est la recherche des appareils dans l'armoire pour satisfaire une requête du système qui est du type: "trouver les appareils appartenant à la manipulation no 1 dans l'armoire, charger les appareils sur le plateau du robot, déplacer le robot à une table de travail, déposer les appareils sur une table de travail et revenir à l'armoire avec le robot".

Si nous analysons cette phrase qui représente une commande donnée à notre système, nous remarquons que les actions: trouver, charger, déplacer, déposer et revenir représentent autant de tâches que le système robotisé doit exécuter pour remplir en fait sa fonction principale de recherche et de distribution des appareils.

La première tâche: "chercher les appareils appartenant à la manipulation no...." implique presque toujours un réarrangement des appareils dans l'armoire pour faciliter la deuxième tâche: "charger les appareils (trouvés) sur le plateau du robot". Les appareils sont distribués aléatoirement dans l'armoire, donc la probabilité de trouver du prèmier coup les appareils qu'on cherche, reste faible. Cette action préalable de réarrangement nous amène à la résolution d'un problème semblable au problème de blocs. Dans ce type de problème, on empile les blocs un par dessus l'autre; dans notre cas, il s'agit de placer les appareils un en face de l'autre étant donné que les appareils ne sont jamais arrangés un par-dessus l'autre.

Alors notre problème est analogue au problème de blocs. La résolution du problème de blocs signifie la conversion d un état initial donné en un état cible désiré. Elle sera vue dans le paragraphe 6.4.

6.2 LA GENERATION DE PLANS

Le problème de génération de plans peut être envisagé aussi d'un autre point de vue, celui de l'automaticien. C'est-à-dire que, en utilisant le langage de l'automaticien, un générateur de plans (planificateur de tâche) peut être interprété comme un régulateur d'états du point de vue conceptuel.

Etant donné un système robotique complexe, avec capacités décisionnelles, il y a toujours un générateur de plans qui utilise des informations concernant l'environnement de travail en provenance des capteurs et qui décide des actions a déclancher pour atteindre des buts désirés. Le générateur de plans qu'on peut également appeler Planificateur s'occupe de la génération des commandes (actions) sur le système robotique quand il s'agit d'un système intelligent et il ferme la boucle.

Donc, par-dessus le système, il y a trois espaces: l'espace des états du système, l'espace des sorties et l'espace des actions (voir fig.6.1). Si U est l'ensemble des actions possibles, S est l'ensemble d'états. Alors u(i) $\in$ U est une action possible et s(i) $\in$ S est un état quelconque. Si nous avons G:U*S -> S, donc une relation définie sur le produit cartésien U*S avec des valeurs en S, nous pouvons construire

H:S*S -> U une relation définie sur le produit cartésien S*S avec des valeurs en U (l'ensemble des actions possibles).

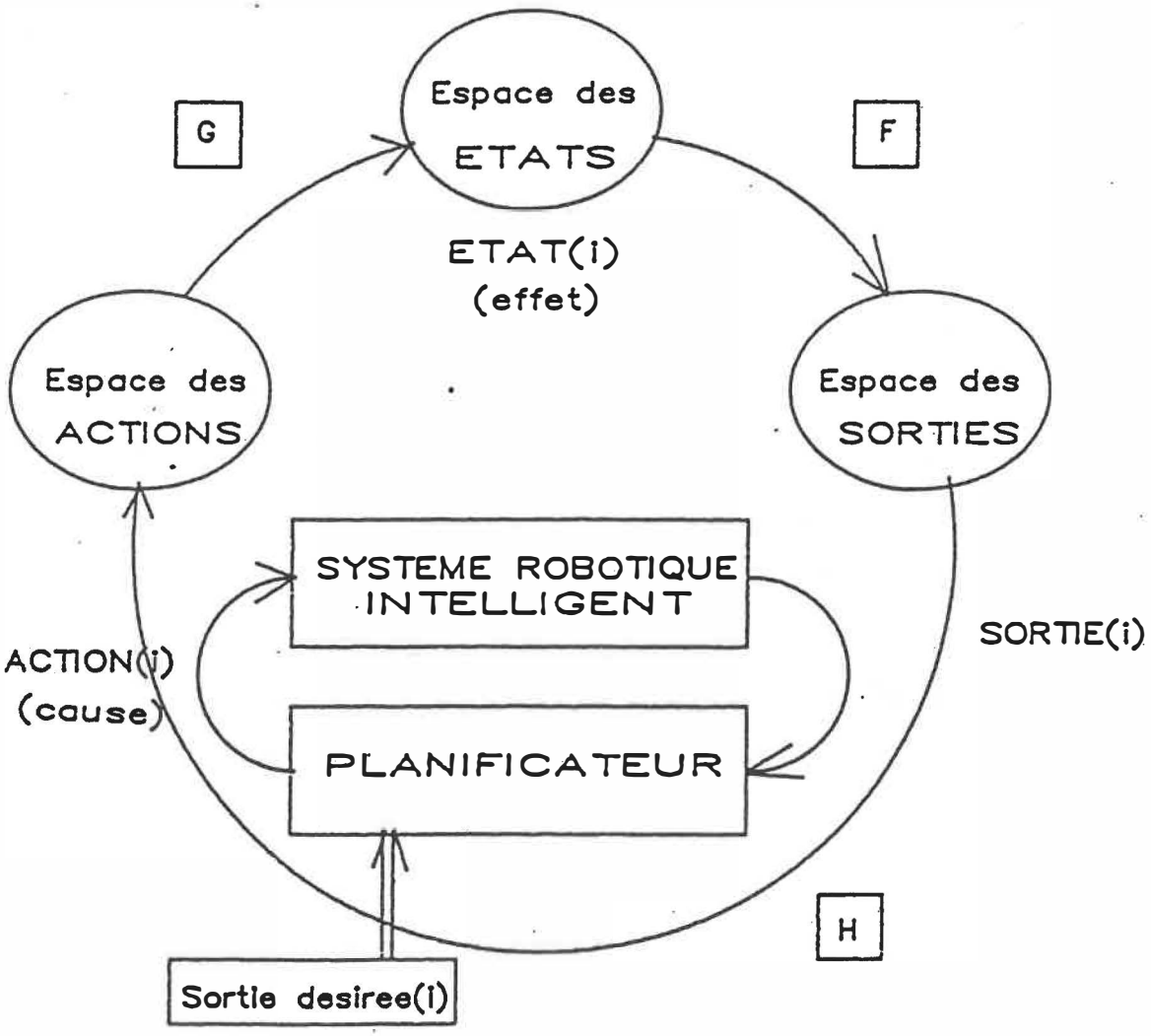


Fig.6.1 Schéma du système robotique intelligent.

Notre système avec planificateur peut être modélisé de la façon suivante:

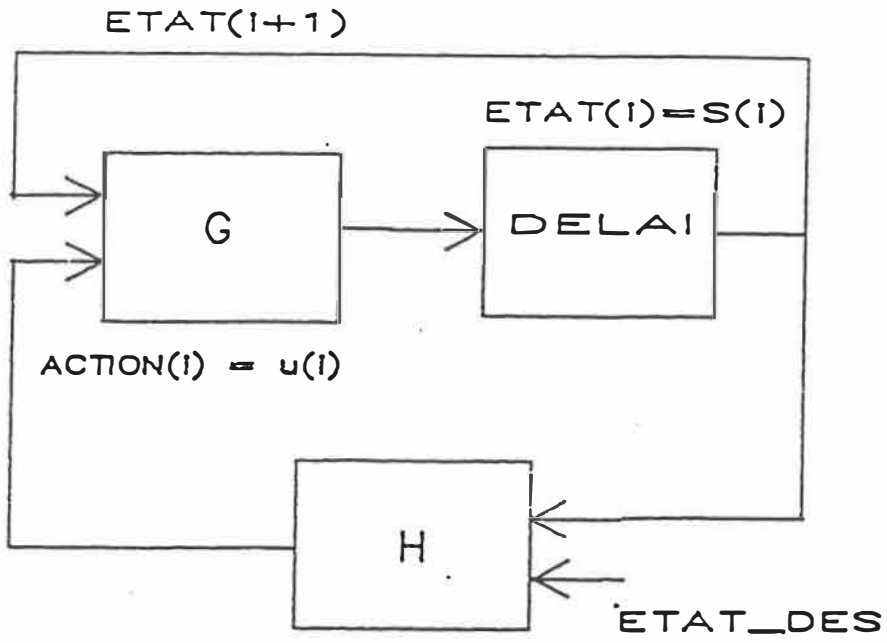


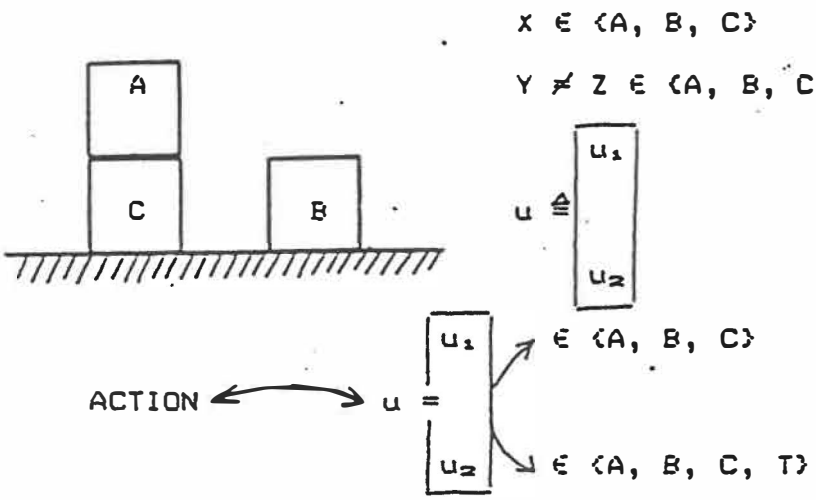
Fig.6.2 Modèle du système robotisé avec planificateur

Le modèle d'état sera: s(i+1) = G [s(i), u(i)]
Le modèle du régulateur sera: u(i) = H [s(i), ETAT_DES].
Alors en donnant à U, S, G et H la structure appropriée, nous retrouvons la représentation d'un système de commande. Dans ces conditions, nous pouvons formuler le problème typique de planification de tâches de la manière suivante: étant donné G, U, S, ETAT_IN, ETAT_FIN, C [H, ETAT_IN, ETAT_FIN] déterminer H telle que:

$$s(1) = ETAT\_IN$$

$$s(N) = ETAT\_FIN$$

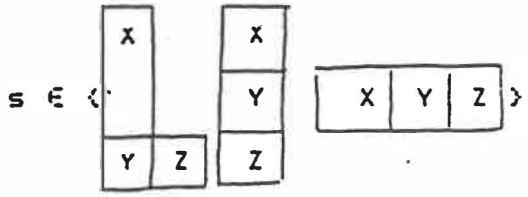
Concernant la représentation des actions possibles, elle sera faite par un vecteur avec deux entrees: la premiere dénote l'élément qui sera soumis a l'action, et la deuxieme denote l'effet de l'action. Si nous prenons comme exemple le problème de blocs, alors nous allons avoir: U = l'ensemble des actions possibles, u ∈ U -> u ∈ ⟨MOVE(X, Y, Z)⟩ signifie que le bloc X est déplacé de Y sur Z.



correspondance

biunivoque

Pour ce qui concerne la representation des etats possibles nous avons: S = l'ensemble des états possibles, s ∈ S -> s = A B C



X ≠ Y ≠ Z ∈ (A, B, C)

L'état s est un vecteur avec quatre entrees: les trois premières entrées donnent l'ordre dans lequel les blocs sont arrangés, et la quatrième donne la structure de l'arrangement.

$$S \triangleq \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \quad x_1 \neq x_2 \neq x_3 \in (A, B, C)$$

$x_4 \in (1, 2, 3)$ - signifie $x_4 = 1$ - un bloc sur la table, $x_4 = 2$ - deux blocs sur la table et $x_3 = 3$ - trois blocs sur la table. La representation de S, U et G peut être faite avec la logique des predicats ou par un graphe. Ce qui est important est que, avec une formalisation comme celle qu'on vient d'exposer, on peut tenter de rapprocher les deux points de vue: automatique classique et intelligence artificielle, pour trouver un langage commun qui sera valable pour l'analyse des sytemes complexes qui englobent des techniques de l'Intelligence Artificielle.

Nous allons illustrer le problème de generation de plans avec un exemple issu de notre projet. La fig.6.3 représente l'état initial (vu par le systeme de vision) de l'armoire avec un arrangement particulier des appareils et l'état cible qu'on cherche à obtenir. Les deux etats peuvent être formalisés de la façon suivante:

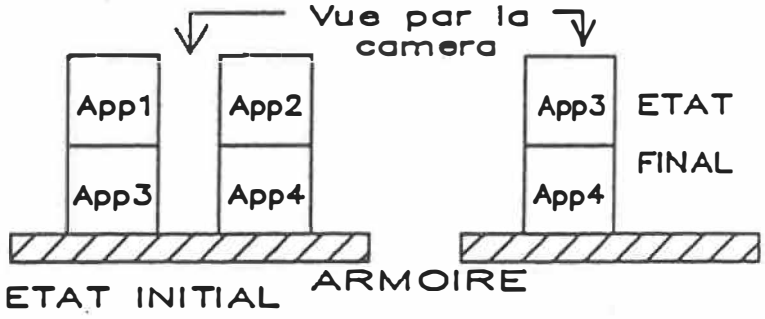


Fig.6.3 Etat initial et état final (vus par la camera)

Etat initial	Etat cible
DEV (App_3, arm)	DEV (App_3, App_4)
DEV (App_4, arm)	VUE (App_3)
DEV (App_1, App_3)	Pour App_1 et App_2
DEV (App_2, App_4)	rien n'est spécifié,
VUE (App_1)	alors ils peuvent se
VUE (App_2)	trouver aussi dans une
LIBRE (pince) le robot ne	autre position.
saisit rien avec l'effecteur.	(arm signifie armoire)

Avec des opérateurs définis convenablement et en appliquant un algorithme de résolution, notre problème sera résolu par la technique de générateur de plans. Générer un plan signifie alors trouver une séquence d'opérateurs (d'actions) qui permet au système de passer de l'état initial à un état final où le but est satisfait.

Le plan qui a été généré est représenté par une séquence d'opérateurs: DEPLACE_APP (App_2, App_4, arm) -->
DEPLACE_APP (App_1, App_3, armoire) -->
--> DEPLACE_APP (App_3, arm, App_4). Le plan est alors représenté par une succession des opérateurs qui représente physiquement des actions exécutées par le robot pour réaliser le but proposé: DEV (App_3, App_4) c'est-à-dire, poser l'appareil no 3 devant l'appareil no 4.

Cet exemple concret issu de notre projet nous permet de saisir la nécessité et l'importance de la génération de plans pour le système. Notre système doit agir sur l'environnement

pour construire un nouvel environnement qui sera conforme à l'ordre émis par le système décisionnel, ce qui signifie en fait exécuter sa tâche principale, recherche et distribution des appareils. Avant d'agir, le système doit raisonner pour générer un plan d'action qui signifie un enchaînement cohérent de transformations dont la réalisation produira l'effet attendu

La génération de plans se rapporte en robotique à la commande des robots. L'objet des plans est de donner aux robots les moyens de décider eux-mêmes des actions physiques à exécuter pour atteindre des objectifs formulés par un opérateur humain. Ces moyens consistent essentiellement en un programme de résolution de problèmes, appelé générateur de plans d'actions, parfois complété par un autre programme pour contrôler l'exécution des plans engendrés. Un tel programme doit avoir une certaine connaissance qui se réfère à deux aspects:

- l'environnement physique dans lequel le robot agit: topologie propriété des objets, lois de l'univers, etc.
- les actions qui peuvent être exécutées par le robot.

En ce qui concerne l'acquisition de cette connaissance, elle peut être réalisée par deux voies:

- interprétation des données transmises par des capteurs
- par un processus d'apprentissage.

Le générateur de plans utilise cette connaissance comme modèle de simulation pour déterminer l'enchaînement des actions (plan d'actions) qui permettront au robot d'atteindre les

objectifs spécifiés.

Après l'analyse que nous avons fait, nous pouvons fixer les étapes de développement d'un générateur de plans:

- formalisation mathématique des états (initial et cible);
- caractérisation mathématique des opérateurs permettant la transition d'un état à un autre;
- formulation du problème par le choix d'un but à atteindre;
- choix de la méthode de résolution;
- choix de l'outil de développement du générateur;
- implantation, tests, validation du générateur.

Dans le cas concret de notre projet, supposons que, à un instant donné, nous prenions une image du monde dans lequel évolue notre robot; une telle vue instantanée, nous l'appelons un état (du monde). Evidemment, un état réel du monde dans notre cas particulier va représenter un arrangement particulier des appareils dans l'armoire. Cet état réel du monde peut être représenté de différentes façons. Alors, l'état sur lequel on raisonne est un modèle idéalisé et simplifié de l'état réel et correspond à un niveau de description abstrait. Ainsi, nous allons décrire une situation de départ par un état initial qui représente une situation donnée d'arrangement des appareils dans l'armoire, et un état final qui représente un but à atteindre. Pour atteindre le but, plusieurs changements d'états successifs doivent être provoqués par les actions du robot; ces actions sont modélisées par des opérateurs. Un opérateur de notre générateur indique d'abord quelles propriétés doit avoir

un état pour que l'opérateur lui soit applicable (par exemple, pour pouvoir saisir un appareil, le robot doit être à portée de celui-ci et sa pince doit être vide), donc les conditions de déclenchement de l'action et ensuite les données qui doivent être effacées et ajoutées à cet état pour décrire le nouvel état. Générer un plan, signifie alors trouver une séquence d'opérateurs (d'actions) qui permet au système de passer de l'état initial à un état final où le but est satisfait.

6.3.UN BREF APERCU DES DIFFERENTES APPROCHES CONCERNANT LA GENERATION DES PLANS

Le problème de génération de plans a donné naissance à plusieurs études qui ont tenté de concevoir des générateurs universels, c'est-à-dire des générateurs qui soient applicables à de nombreux modèles de monde. Voilà quelques exemples des systèmes: STRIPS [Fikes] - réalisé au début des années soixante-dix au Standford Research Institute pour construire les plans d'action du robot Shakey, conçu pour évoluer dans des univers physiques typiquement constitués de salles, portes et boites, ABSTRIPS [Sacerdoti] - l'amélioration du STRIPS, NOAH [Sacerdoti] - utilisé pour construire des plans d'action permettant le montage et le démontage d'appareils électromécaniques, en particulier des compresseurs d'air, WARPLAN [Warren] ecrit en PROLOG - utilisé pour la génération des plans dans le monde des blocs, et BUILD qui engendre des plans d'action pour construire des structures complexes

d'objets parallélépipédiques et prismatiques en tenant compte des forces de contact entre objets pour tester la stabilité de sous-structures intermédiaires.

D'abord, nous allons faire une précision importante. La génération de plan, telle qu'elle est appliquée en robotique, utilise de plus en plus un type de programmation particulière spécifique à l'intelligence artificielle et appelée "déclarative", car le mode de programmation "impératif" traditionnel est difficilement applicable étant donné que le programmeur doit prévoir explicitement d'avance tous les cas possibles, ce qui n'est pas toujours facile. La programmation déclarative est basée sur un ensemble de règles du type: "SI telle(s) condition(s) vraie(s), ALORS exécuter telle(s) action(s)". Les règles sont données en vrac, sans préciser l'ordre dans lequel elles seront utilisées. Les avantages de la programmation déclarative sont la puissance expressive et les possibilités aisées de modifications, mais ses inconvénients restent la lenteur et le fait qu'on ne garantie pas l'obtention du résultat.

Pour avoir un aperçu de différentes approches concernant la génération des plans, nous allons exposer brièvement quelques méthodes parmi les plus couramment utilisées.

L'une des méthodes les plus classiques de génération de plan est celle du système GPS(General Problem Solver) qui est basée sur un plan de réduction des différences les plus importantes entre un état et un autre état. Ces différences sont une sorte de "mesure" des écarts entre deux états; elles ne correspondent

à aucune définition mathématique précise, ce qui fait qu'on parle alors d'"heuristiques", et elles sont particulières pour chaque type de problème. GPS génère des plans par choix des opérateurs sur la base des réductions successives de différences qu'ils rendent possibles. Alors il définit des sous-buts qui permettent le rapprochement du but final.

Une autre méthode de génération de plan est celle faite par sélection d'un opérateur et recherche en profondeur. Les opérateurs définissent des actions par lesquelles on transforme des états par l'application des préconditions et de l'opérateur Les préconditions représentent les conditions nécessaires pour pouvoir appliquer l'opérateur. Pour une définition complète, il faut décrire les modifications apportées par l'application de l'opérateur. Ces modifications sont décrites par deux listes: une liste de suppressions qui contient ce qui doit être effacé de l'état initial, et une liste des adjonctions qui contient toutes les modifications apportées par l'application de l'opérateur et qui caractérise l'état final. Donc, l'état final est obtenu de l'état initial par l'application de l'opérateur en respectant les préconditions et les deux listes. Le principe de cette méthode est le suivant: à partir du but, on sélectionne un opérateur qui contient le but dans sa liste des adjonctions. Alors, les préconditions du même opérateur constituent des sous-buts. On sélectionne ensuite un autre opérateur d'après le même principe, cette fois, pour réaliser les sous-buts; le processus continue jusqu'au moment où l'état obtenu est satisfait par l'état initial, suite à une

substitution convenable. Le processus de recherche est fait d'abord en profondeur.

Si, lors du processus de résolution, il y a une interférence de plusieurs buts, il faut appliquer une méthode par laquelle plusieurs buts (clauses) peuvent être résolus indépendamment. Quand nous parlons d'interférence de plusieurs buts, nous parlons d'un processus de résolution avec plusieurs sous-buts à atteindre en même temps. Par exemple nous devons atteindre deux buts en même temps: DEV(x,y) ^ DEV(y,z). Alors, le problème est décomposé en plusieurs clauses, qui sont ensuite résolues indépendamment et représentées par un arbre ET/OU. Pour tout opérateur O_i devant être appliqué, il y a deux possibilités: soit que les opérateurs le précédant dans la séquence ne doivent pas être inclus ni dans la liste des adjonctions ni dans la liste des suppressions de O_i, soit que le dernier opérateur de la séquence doit figurer dans la liste des adjonctions.

Une autre méthode de génération de plan est celle du générateur dirigé par le but. Ce type de générateur procède par génération descendante. C'est-à-dire que pour réaliser un but, on cherche parmi toutes les actions possibles une action qui peut avoir pour conséquence le but qu'on s'est proposé de réaliser. Alors les conditions de déclenchement de l'action choisie deviendront les nouveaux sous-buts à réaliser. Ce générateur applique une stratégie de réduction de l'espace de recherche.

6.4. ENONCE DU PROBLEME DE BLOCS

Le problème de blocs traite la manipulation des objets qui appartiennent au "monde des blocs". Pour la première fois, la notion a été utilisée par T. Winograd dans ses travaux sur la compréhension du langage naturel et par la suite, un tel monde est devenu le champ d'expérimentation pour différents types de systèmes intelligents dont des systèmes robotisés intelligents, des systèmes de vision, etc. Le monde des blocs est un monde idéalisé, irréel et simplifié qui contient seulement certains types d'objets, et qui est gouverné par certaines relations. Evidemment, il est apparu par la nécessité de limiter la complexité du contenu sémantique et syntaxique des représentations visuelles ou linguistiques, et pour avoir une bonne approximation de tout monde "simple". Tout système qui opère adéquatement dans un monde de blocs pourra être étendu à un environnement plus complexe, mais toujours limité. Le milieu d'opération d'un robot et les types d'objets qu'il doit manipuler en sont deux exemples.

C'est d'ailleurs l'analogie entre le problème de blocs et notre problème de manipulation des appareils que nous allons utiliser au cours de notre projet. Le monde de blocs défini pour les besoins de ce projet est ainsi constitué:

- une surface de référence, la table de travail ou l'armoire parfaitement horizontale;
- un type d'objet: le bloc, dans notre cas, un appareil. Tous

les objets sont physiquement identiques, sauf ceux qui ont des caractéristiques permettant leur idéntification par un module de perception. La caractéristique d'idéntification est le gabarit; le module de perception est le système de vision.

Une série de contraintes spécifie généralement les relations permises entre les objets:

- tous les objets peuvent reposer sur la table ou dans l'armoire;
- un bloc peut supporter tous les autres;
- aucune limite préétablie de hauteur d'empilement des objets.

La configuration de tous les blocs existants et des relations entre eux à un moment donné, définit ce qu'on appèle l'état du monde. Il y a deux états importants: l'état initial et l'état cible (final). L'état initial constitue l'état de départ; l'état cible appartient à un ensemble d'états possibles et représente un but à atteindre. Ces états ont les propriétés suivantes:

- les états sont stables, seule l'intervention du robot peut modifier l'état des objets.
- chaque objet est décrit par deux éléments: son support (l'objet qui est sous lui ou en face de lui) et son emplacement.

Le monde de blocs est une représentation simplifiée de ce qui pourrait être l'environnement réel de travail d'un robot; il peut cependant servir pour illustrer la faisabilité d'un système de planification de tâches, tout en réduisant la complexité de l'étude, sans affecter pour autant la nature du

problème. Dans un problème de blocs, on empile les blocs un par dessus l'autre; dans notre cas, il s'agit de placer les appareils un en face de l'autre, étant donné que les appareils ne sont jamais arrangés un par-dessus l'autre. Alors, notre problème est analogue au problème de blocs. Une restriction que nous allons imposer à notre problème est qu'on ne peut pas empiler plus de trois appareils, donc, un appareil qui se trouve au fond de l'armoire peut avoir devant lui, au maximum, deux autres appareils.

6.5. SOLUTION DU PROBLEME DE BLOCS

Le problème de blocs peut être résolu par différentes méthodes. Pour illustrer la procédure de résolution, nous allons exposer la méthode de sélection de l'opérateur et de recherche en profondeur. Les étapes à franchir pour résoudre le problème ont été exposé à la page 78.

Formalisation mathématique des états:

Pour une plus grande clarté, nous allons formaliser le problème de blocs de manière à le rendre compatible avec notre problème particulier et avec notre solution. Ainsi, pour décrire les états nous allons utiliser les notations suivantes:

DEV(X, Y) - pour exprimer que l'objet X se trouve en face de l'objet Y;

DEV(X,arm) - pour exprimer que l'objet X se trouve au fond de l'armoire.

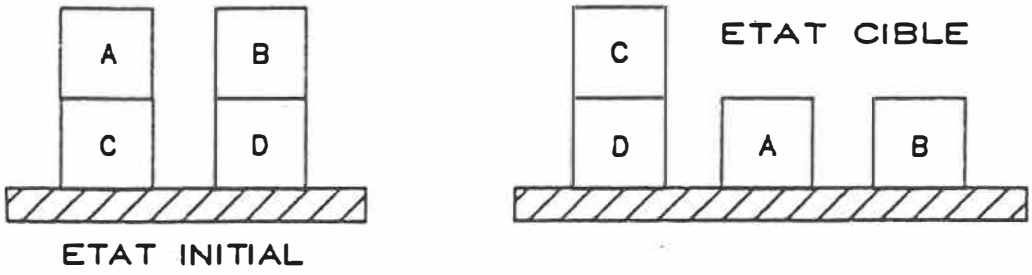


Fig.6.4 Etat initial et état cible (formalisation)

DEV(A, C)	DEV(C, D)
DEV(B, D)	DEV(D, arm)
DEV(C, arm)	DEV(A, arm)
DEV(D, arm)	DEV(B, arm)
VUE(A)	VUE(C)
VUE(B)	VUE(A), VUE(B)
LIBRE(pince)	LIBRE(pince)

L'état cible représente le but à atteindre.

Formalisation mathématique des opérateurs:

Pour passer d'un état à un autre, nous allons définir des opérateurs qui représentent les actions qu'on doit exécuter. Il y a deux situations possibles:

a) L'objet A se trouve en face de l'objet B qui est au fond de l'armoire. Le but est de rendre visible l'objet caché (B). L'opérateur sera: DEPLACE_APP(A, B, arm) ce qui signifie prendre l'objet A qui se trouve en face de l'objet B et le deposer dans l'armoire.

b) L'objet A se trouve en face de l'objet B qui n'est pas au fond de l'armoire. Le but est de déplacer l'objet A en face d'un autre objet C qui n'est pas au fond de l'armoire. L'opérateur sera: DEPLACE_APP(A, B, C), ce qui signifie prendre l'objet A qui se trouve en face de l'objet B et le placer en face de l'objet C.

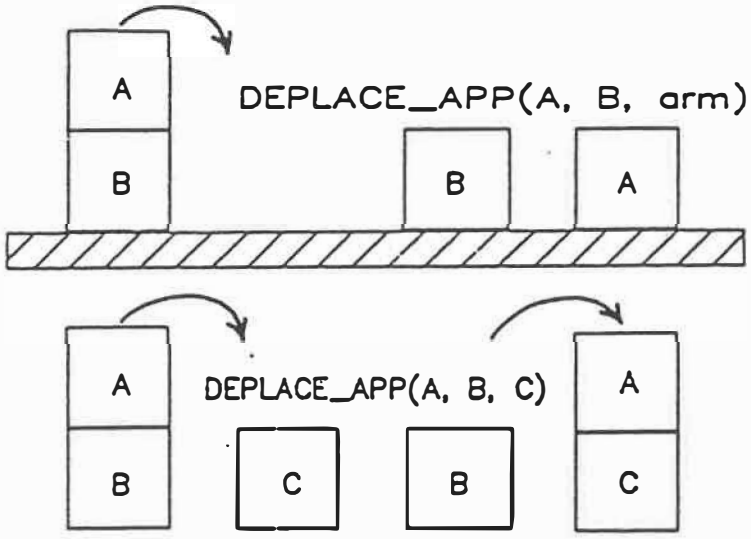


Fig.6.5 Définition des opérateurs

DEPLACE_APP(x, y, arm) = saisir l'objet x qui se trouve en face de l'objet y et le déposer dans l'armoire.

préconditions: DEV(x, y), VUE(x), dif(y, arm)

liste des suppressions: DEV(x, y)

liste des adjonctions: VUE(y)

DEPLACE_APP(x, y ,z) = déplacer l'objet x qui se trouve en face de l'objet y, en face de z.

préconditions: VUE(z), DEV(x, y), VUE(x), dif(x, z)

liste des suppressions: VUE(z)

liste des adjonctions: DEV(x, z)

Résolution du problème par sélection de l'opérateur:

Dans notre exemple, le but est: DEV(C, D). Le but n'est pas satisfait évidemment par l'état initial; alors, il faut sélectionner un opérateur qui contient le but dans sa liste d'adjonctions et le prendre comme sous-but. L'opérateur qui satisfait cette condition est DEPLACE_APP(x, y, z). Cette procédure utilise le principe de résolution de Robinson: Si la clause vide peut être dérivée de la négation ~DEV(C, D) du but et de la liste des adjonctions de l'opérateur DEV(x, y), alors l'opérateur peut être sélectionné.

Dans cet exemple, la clause vide est obtenue par la substitution: (x <- C), (y <- D). Le sous-but est alors constitué des préconditions de l'opérateur choisi DEPLACE_APP(C, D): VUE(C), DEV(A, B), VUE(A). Partons maintenant de ces clauses et par application du même principe sélectionnons un opérateur. Mais, pour sélectionner correctement l'opérateur, il faut tenir compte du principe suivant: si le sous-but comporte plus d'une clause (comme dans notre cas), un opérateur qui satisfait une de ces clauses ne doit pas être choisi si l'autre clause est incompatible avec la liste des suppressions et des adjonctions de l'opérateur. Après un certain nombre d'applications de la procédure, on arrive à un état qui est satisfait par l'état initial. Alors la solution du problème sera obtenue en remontant l'arbre:

DEPLACE_APP(B, D, arm) -> DEPLACE_APP(A, C, arm) ->

-> DEPLACE_APP(C, arm, D)

6.6 WARPLAN - LE GENERATEUR DE PLAN UNIVERSEL

WARPLAN: A System for Generating Plans, est un générateur universel de plans, ce qui signifie applicable à de nombreux modèles de monde; le programme du générateur a été écrit par D.H.Warren. Beaucoup de problèmes peuvent être formalisés sous forme d'un monde avec différents états et d'un ensemble d'actions qui transforme ce monde d'un état à un autre. Alors, pour spécifier un problème particulier, il faut faire la description d'un état initial et d'un état but désiré; pour résoudre le problème de transformation d'un état dans un autre, il faut générer un plan. Le plan est représenté par une séquence d'actions qui transforment le monde de l'état initial à l'état but désiré. Warplan est un générateur de plans indépendant du problème, capable de générer un plan d'actions pour n'importe quelle base de données, qui représente la description du monde [20].

Le programme contient plusieurs prédicats prédéfinis:

1. Le prédicat central **plan(C, P, T, T_1)** a quatre arguments:
   - C représente la conjonction de buts à résoudre;
   - P représente la conjonction de buts résolus par T et qui doivent être protégés;
   - T est un plan partiel;
   - T_1 est un nouveau plan qui contient T comme sous-plan; il protège les buts P et résout les nouveaux buts C.

   Ce prédicat produit un plan par résolution des buts dans un certain ordre. Les variables d'entrées sont: C, P et T; la

variable de sortie est T_1.

2. La résolution des buts est faite par le prédicat solve(X, P, T, P_1, T_1). Les arguments de ce prédicat sont:
   - X est un but atomique;
   - T est un plan partiel;
   - P représente la conjonction des buts achevés par T;
   - P_1 représente la conjonction entre P et X, ne doit pas se répéter
   - T_1 est un plan ayant T comme sous-plan et qui résout P_1.
3. Pour réaliser une action, il y a le prédicat **achieve**. Le prédicat achieve(X, U, P, T, T_1) permet de réaliser une action par extension ou par insertion. La méthode par extension vérifie si l'action U "préserve" les faits protegés P, et si la liste C est conforme à la liste P. La méthode par insertion vérifie si la dernière action V dans le plan courant n'a pas effacé le but courant X. Alors on peut insérer l'action U avant V, pourvu qu'on retire la liste de faits protegés qui correspond au point d'insertion.
4. Le prédicat **holds**(X, Y) vérifie si un fait se trouve dans un certain état.
5. Le prédicat **retrace**(X, V, P) retire de la liste de faits preservés tous les faits établis par V, mais qui diffèrent des préconditions du V; c'est-à-dire qu'il retire les faits ajoutés par V et les faits qui constituent les préconditions du V sont réinserés par **append**.

Les actions sont définies par trois procédures:

- can (Action, Précondition)

  cette procédure sert à cataloguer une clause par action; Précondition représente une conjonction de faits nécessaires pour que Action soit applicable.

- add (Fact, Action)
- del (Fact, Action)

  ces deux procédures donnent les listes de faits ajoutés et effacés par une action.

  Les combinaisons impossibles de faits sont listées par:

- imposs (conjonction)

  Les faits qui restent dans l'état initial et qui ne sont pas affectés par aucune action sont listés par:

- always (Fact)

  D'autres faits qui restent dans l'état initial sont identifiés par:

- given (Initial State Name, Fact)

  L'état initial est noté par son nom, par exemple start. Un état dérivé de l'état initial par l'application des actions $A_1,...,A_n$ est noté par: Initial State Name: A_1:...:A_n

  Start: move (c, a, floor): (a, floor, b): move (c, floor, a).

Fonctionnement du WARPLAN.

Le programme travaille indépendamment de la description spécifique du monde, à condition de disposer d'une base de données appropriées pour laquelle la cohérence doit être assurée par l'usager. Le programme commence avec une conjonction de faits, par exemple la description de l'état final désiré et un plan vide. A chaque pas, la conjonction rétrécit et le plan grossit de façon telle que des états intermédiaires successifs aboutissent à l'état final.

WARPLAN est muni d'un moteur d'inférence qui travaille en chaînage arrière: pour réaliser un but, il cherche une action pouvant avoir pour conséquence le but, et les préconditions de l'action choisie deviendront les nouveaux sous-buts à réaliser. A moins qu'un fait se trouve dans un état intermédiaire, le programme choisit une action qui ajoute ce fait, insère l'action dans un sous-plan, extrait le fait de la conjonction de buts et l'ajoute aux préconditions d'une autre action.

Le prédicat principal "plan" est appelé seulement si la déscription de l'état final est consistante, c'est-à-dire si elle n'implique pas une des combinaisons impossibles de faits. Il y a trois paramètres d'entrée: les faits qu'on doit exécuter, les faits déjà exécutés (initialement vrais) et le plan courant, et un paramètre de sortie: le plan final. Pour une meilleure comprehénsion de la façon de travailler du Warplan, voici un exemple.

Exemple d application du programme Warplan.

La fig.6.6 represente un probleme de blocs avec l etat initial et l etat cible (but) desire.

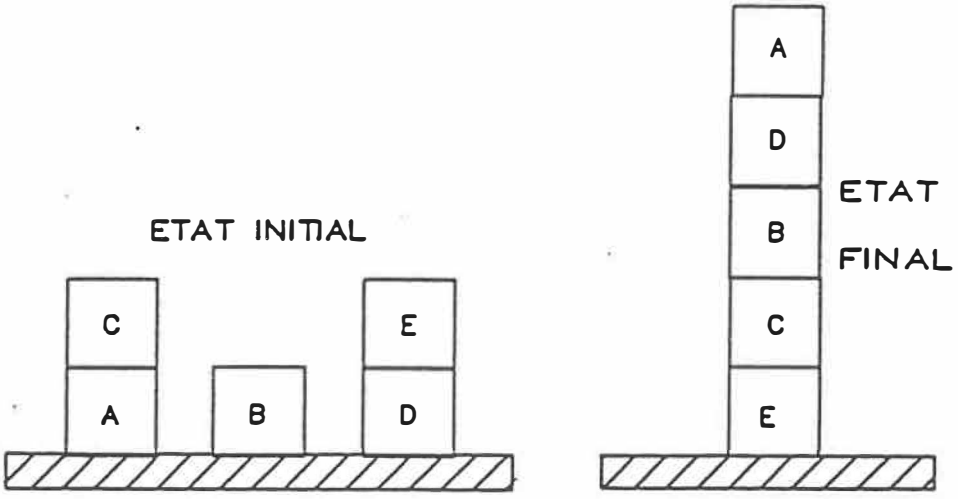


Fig.6.6 Exemple de l'état initial et de l'etat final.

D'abord, nous allons préciser la signification de chaque prédicat utilise pour la modélisation mathématique des états et des actions.

given(T, X) - signifie que la clause X est vraie dans l'état initial T;

impss(X) - la clause X est impossible dans tous les états;

always(X) - la clause X est vraie dans tous les états;

can(U, C) - la conjonction de faits C est la précondition de l'action U; l'action U est possible dans tous les états dans lesquels C est vraie;

del(X, U) - la clause X est effacee par l action U;

add(X, U) - la clause X est ajoutee par l'action U;

La modélisation mathématique des états et des actions donne la formalisation suivante:

L'état initial:

```
given(start, on(a, floor)).
given(start, on(b, floor)).
given(start, on(d, floor)).
given(start, on(c, a)).
given(start, on(e, d)).
given(start, clear(b)).
given(start, clear(c)).
given(start, clear(e)).
imposs(on(X, Y) & clear(Y)).
imposs(on(X, Y)&on(X, Z) & not_equal(Y, Z)).
imposs(on(X, X)).
```

Les actions:

```
can(move(U,V,floor),on(U,V) & not_equal(V,floor) & clear(U)).
can(move(U,V,W),clear(W) & on(U,V) & not_equal(U,W) & clear(U))
add(on(U, W), move(U, V, W)).
add(clear(V), move(U, V, W)).
add(on(U, floor), move(U, V, floor)).
add(clear(V), move(U, V, floor)).
del(on(U, V), move(U, V, W)).
del(clear(W), move(U, V, W)).
del(on(U, V), move(U, V, floor)).
```

L execution du programme est declenchee par la commande:

:- plan(on(c, e), on(b, c), on(d, b), on(a, d), start).

Apres l execution, la solution obtenue est:

start;

move(e, d, floor);

move(c, a, floor);

move(c, floor, e);

move(b, floor, c);

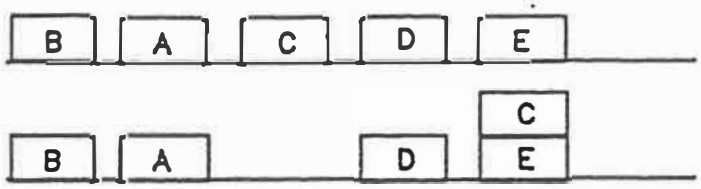
move(d, floor, b);

move(a, floor, d);

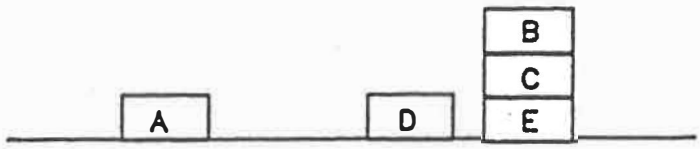
Fig.6.7 Exemple d'execution du programme Warplan.

1. move(e, d, floor)

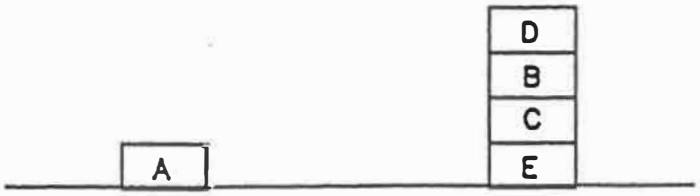
2. move(c, a, floor)



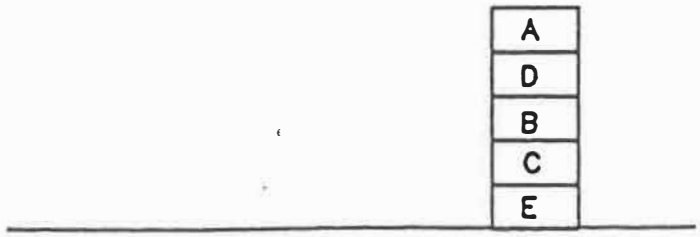
3. move(c, floor, e)



4. move(b, floor, c)



5. move(d, floor, b)



6. move(a, floor, d)

6.7. GENERATEUR DE PLAN DIRIGE PAR LE BUT (implantation PROLOG)

Nous allons écrire un générateur de plans en Prolog.

6.7.1 Représentation des actions

La représentation d'une action implique deux spécifications:

- la spécification des conditions de déclenchement
- la spécification des conséquences faite par:
  - l'énumération des effets positifs - les faits ajoutés par l'action
  - l'enumération des effets négatifs - les faits supprimés par l'action

Ca veut dire qu'une action quelconque sera représentée par une assertion de la forme suivante:

ACTION(act,cond,supprime,ajoute)

act - est libellé de l'action

cond - est la liste des conditions de déclenchement

supprime - est la liste des faits supprimés par l'action

ajoute - est la liste des faits ajoutés par l'action

La représentation des conséquences d'une action reste un problème délicat en génération de plans dans le sens qu'il faut décrire:

- tout ce qui devient vrai
- tout ce qui n'est plus vrai
- tout ce qui reste vrai après l'action

Ce problème peut nécessiter la quantification des symboles de prédicat, donc l'utilisation d'éléments de logique du deuxième ordre. Alors, pour éviter cela, nous allons faire la convention suivante: tout ce qui n'est pas déclaré effacé d'une façon explicite (dans le champ "supprime") reste vrai. La description des conséquences des actions néccessite beaucoup d'attention.

6.7.2 Représentation d'un état du monde

L'état du monde est représenté explicitement par une liste des faits décrivant le monde, toujours rélativement à la situation initiale nommée "init". Tout état est représenté par une liste d'actions issue de la situation initiale. Par exemple:

$$[ACTION_3\ ACTION_2\ ACTION_1\ init]$$

représente la situation qu'on obtient à partir de la situation initiale après l'exécution de l'$ACTION_1$ suivie de l'$ACTION_2$ et puis de l'$ACTION_3$. Alors, il nous reste à décrire explicitement l'état initial, chose réalisée par une liste

d'assertions qui correspondent au prédicat "donnée".

6.7.3 Le générateur de plans

Il y a trois aspects à envisager:

1. Le but - qui sera caractérisé par une liste, la liste de faits qu'on souhaite réalisés.
2. Le plan - qui est représenté explicitement par la description de la situation finale, dans le sens d'enchaînement d'actions appliquées à l'état initial.
3. Le générateur - qui sera bâti autour des prédicats de base
   - planif (B, S_i, S_f)
   - réalise (B_o, S_i, S_f)
   - exact (F, S)

Les prédicats de base:

- planif (B, S_i, S_f) - exprime que le but B est réalisé quand on évolue de la situation initiale S_i à la situation finale S_f.
- realise (B_o, S_i, S_f) - exprime que le but isolé B_o fait unique, est réalisé quand on passe de la situation S_i à la situation S_f.
- exact (F, S) - exprime que le fait F correspond exactement à la situation S (qui peut être une action).

Prédicats auxiliaires:

- donnée - pour représenter les faits spécifiques à l'état initial.
- dif (X,Y) - pour éviter la génération d'actions non significatives. Il vérifie si l'argument X et différent de l'argument Y.

Les deux prédicats de base: planif et realise fonctionnent par "génération descendante en profondeur", c'est-à-dire: pour réaliser un but, on cherche une action qui peut avoir pour conséquence le but, et alors les conditions de l'action deviendront les nouveaux sous-buts à réaliser.

Le générateur sera:

PLAN (G,S) if
planif (G,S,S_1) \ écrire (S_1)

il va générer le PLAN pour réaliser le but G à partir de la situation S.

Les prédicats de base:

Le prédicat planif: planif ([B|R], S, S_2) if
realise (B, S, S_1) and
planif (R, S_1, S_2).
planif ([ ], S, S).

Le prédicat realise: realise (B, S, S) if

```
                    exact (B, S).
          realise (B, S, [Act: S1]) if
                    ajoute (B, Act) and
                    action (Act, Cond) and
                    planif (Cond, S, S1).
Le prédicat exact: exact (F, [Act:_]) if
                    ajoute (F, Act).
                    exact (F, [Act: S]) if
                          exact (F, S) and
                          not (supprime (F, Act)).
          exact(F,[depart (init)]) if donnée (F).
```

Les prédicats auxiliaires:

```
Le prédicat dif:    dif (X, X) if
                          !, fail.
                    dif (_, _).
Le prédicat donnée: donnée (dev(a, arm)).
                    donnée (dev(b, a)).
                    donnée (vue(b)).
```

Pour déclencher notre générateur de plans il faut définir les actions que nous souhaitons réaliser. Nous avons déjà vu quand nous avons parlé de la représentation des actions, qu'une clause action a quatre paramètres:

1. act - qui signifie le nom de l'action, par exemple avance(A,B)

2. cond - qui représente la liste des conditions de déclenchement
3. supprime - qui représente la liste des faits supprimés par l'action et qui a deux paramètres: supprime ([F|_], Act): [F|_] - la liste des faits à supprimer et l'action Act qui les supprime.
4. ajoute - qui représente la liste des faits ajoutés par l'action et qui a aussi deux paramétres: ajoute ([F|_], Act) [F|_] - la liste des faits à ajouter et l'action Act qui les ajoute.

Après avoir défini les actions, les ajouter et les supprimer il faut définir aussi les faits initiaux à l'aide du prédicat auxiliaire "donnee".

A la fin du chapitre il y a quelques exemples de déroulement du programme, ainsi que le listing du programme de génération de plan.

A remarquer une chose importante - l'ordre des conditions de déclenchement au moment de la description d'une action. Par exemple, si on permute les conditions d'une action on pourra tomber dans une boucle infinie par le raisonnement du moteur d'inférence. C'est une difficulté majeure de générateurs de plan. Pour éviter l'apparition de boucles infinies, il faut avoir d'outils de méta-contrôle de la progression dans l'espace de recherche. La solution sera de définir des heuristiques de contrôle à l'aide de métaclauses pour pallier ce genre de difficultés.

CHAPITRE 7

GENERATION DE PLAN EN UTILISANT UN SYSTEME EXPERT

7.1 INTRODUCTION

Dans ce chapitre, nous allons aborder le problème de la génération de plans d'un autre point de vue. Nous avons traité le générateur de plans comme un système expert capable de planifier les tâches de notre système robotisé. Plusieurs points sont abordés: l'énoncé du problème, la solution du problème, la formalisation mathématique du problème, le scénario de recherche et distribution des appareils, la représentation de la connaissance sous forme de règles de production, le mécanisme d'inférence et le contrôle du processus d'inférence (toujours en liaison directe avec notre projet). Ensuite, nous allons faire reférence à l'outil de développement et on terminera avec l'explication de notre implantation et son fonctionnement.

Nous avons réalisé un système expert pour la planification des tâches, que nous avons appelé de façon abrégée SEPT, Système Expert pour la Planification des Tâches. Ce système expert doit servir de logiciel de décision pour notre Système Robotisé pour Recherche et Distribution des appareils dans un laboratoire d'asservissements (abrégé par SRRD).

Cette approche est essentiellement différente de celle qui avait à la base l'analogie de notre système avec le monde de blocs. La différence est qu'ici nous voyons le générateur de plans comme un système expert capable de planifier les tâches de notre système robotisé. C'est-à-dire que le système contient l'expertise nécessaire pour réaliser la planification de tâches sur la base des informations envoyées par le système de senseurs. A l'heure actuelle, le système travaille sur la base d'un dialogue avec l'usager. Dans une étape ultérieure d'implantation opérationnelle, le dialogue s'établira entre le système expert et le module de perception. Alors, le système expert interrogera le module de perception et en fonction des informations reçues et de ses connaissances a priori, il sera capable de planifier les actions du SRRD.

Les entrées de notre SEPT seront des données concernant l'environnement de travail. Exemple: le numéro de la manipulation à exécuter, les états vus par le système de vision, le code d'identification des appareils vu par la caméra

Les sorties du SEPT sont constituées par des listes qui donnent les séquences d'actions du SRRD et qui aboutiront au but proposé. Dans la version opérationnelle, les séquences d'actions envoyées par SEPT représentent le plan d'actions qui seront executées physiquement par SRRD pour accomplir la tâche. Alors la tâche sera réalisée par accomplissement des sous--tâches qui seront comprises dans le plan généré par le système expert, et exécutées par SRRD.

7.2 ENONCE DU PROBLEME

Le système robotisé de recherche et distribution des appareils dans un laboratoire d'asservissements, qu'on va désormais désigner par l'abréviation SRRD, a la tâche de chercher d'abord les appareils appartenant à la manipulation qu'on doit exécuter et ensuite de les distribuer sur les tables de travail. Pour réaliser cette tâche, le SRRD doit répondre aux exigences du problème de recherche et distribution des objets. Voilà les données du problème. Il y a 18 appareils différents en plusieurs exemplaires (10 pour chacun), donc un total de 180 appareils. Ils sont disposés sur 12 tablettes dans deux armoires (6 tablettes dans chaque armoire). Sur chaque tablette sont placés 16 appareils sur 4 rangées et 4 colonnes.

Le système de vision est capable de "voir" une tablette et quatre appareils à la fois. Donc il peut identifier les quatre appareils situés sur la première rangée de chaque tablette, les autres quatre rangées étant cachées par la première. Entre les deux armoires, il y a une table auxiliaire sur laquelle peuvent être déposés en attente des appareils (jusqu'à 32 appareils sur quatre rangées et huit colonnes). Les appareils doivent être distribués sur les 10 tables de travail dans le laboratoire. Il y a quatre manipulations qu'on doit exécuter une à la fois. Pour chaque manipulation, le nombre des appareils est différent Ainsi, pour la première manipulation, sont nécessaires 6 appareils, pour la deuxième, 10 appareils, pour la troisième, 9 appareils et pour la quatrième, 8 appareils. Pour chaque

manipulation, il y a 5 appareils qui sont toujours les mémes (l'amplificateur KEPCO, le moteur DC, le voltmètre analogique, le voltmètre numérique et l'ampèremètre numérique). A ces cinq appareils de base, s'ajoutent pour chaque manipulation d'autres appareils, changés en fonction de la manipulation à exécuter (un pour la première, cinq pour la deuxième, quatre pour la troisième et trois pour la quatrième). Pour solutionner notre problème, nous allons d'abord décomposer le problème en deux phases: la première phase concerne la recherche des appareils (appartenant à une certaine manipulation) dans les deux armoires; la deuxième phase concerne la distribution des appareils sur les tables de travail. Chaque des phases passe nécessairement par une opération d'identification, opération qui doit répondre à deux questions:

1. L'appareil (vu par la caméra) appartient à la manipulation no...? (en phase de recherche)
2. L'appareil appartient à la même catégorie d'appareils? (en phase de distribution)

7.3 SOLUTION DU PROBLEME

Pour solutionner le problème que nous venons d'énoncer, nous allons décrire en détail la procédure d'identification et ensuite les phases nécessaires pour accomplir la tâche du SRRD. Les appareils sont identifiés par deux codes, un pour indiquer l'appartenance à une manipulation, et l'autre pour indiquer l'appartenance à une catégorie d'appareils (moteurs, voltmètres

amplificateurs, etc.) Les marques indiquant les codes seront en matériel reflétant blanc pour assurer un bon contraste, et elles seront collées sur chaque face de l'appareil pour être facilement "vues" par le système de vision. La marque pour identifier la manipulation sera: un carré (2 cm de côté) pour la manipulation No 1, un triangle équilatéral (2 cm de côté) pour la manipulation No 2, un cercle (2 cm diamètre) pour la manipulation No 3 et un demi-cercle (de même diamètre) pour la manipulation No 4. L'appareil de la manipulation No 1 qui ne fait pas partie des cinq appareils de base (retrouvés dans toutes les manipulations) sera identifié par un code spécial- une croix de 2cm pour éviter toute éventuelle confusion. Pour identifier la catégorie à laquelle appartient l'appareil, la deuxième marque placée à côté de la première sera un chiffre (de 1 à 18). A part les codes, nous avons associé à chaque catégorie d'appareil, un attribut qui permettra au système de vision de les identifier. Il s'agit de leur gabarit. Alors, il y a trois types d'appareils: grand, moyen et petit. Voilà la nécessité de ces moyens d'identification.

- Le premier code (géométrique) permet de trouver l'appartenance d'un appareil à une certaine manipulation pour satisfaire une requête du système. Il sera nécessaire dans la phase de recherche des appareils;
- Le deuxième code (numérique) permet de trouver l'appartenance d'un appareil à une catégorie. Il sera nécessaire dans la phase de distribution des appareils (sur les tables de travail) pour vérifier si tous les appareils ont été

distribués et pour éviter d'avoir deux appareils de la même catégorie sur la même table;

- Le troisième moyen d'identification (attribut) permet l'application de règles de production. Il sera nécessaire dans la phase de recherche des appareils pour pouvoir charger le plateau du robot.

La tâche du SRRD peut être décomposée en deux phases:

- La phase de recherche des appareils qui aura pour but de chercher dans les deux armoires tous les appareils appartenant à une certaine manipulation. Cette phase peut être décomposée aussi en deux étapes: une première étape d'identification des appareils appartenant à la manipulation qu'on veut exécuter, et une deuxième étape de chargement du plateau du robot en vue de transporter et de distribuer les appareils.
- La phase de distribution des appareils qui aura pour but de distribuer sur chaque table de travail tous les appareils nécessaires à l'exécution de la manipulation. Cette phase implique deux étapes d'identification: la première identification qui appartient à la phase de recherche, et une deuxième pour répondre à une question du type: cette table est-elle complète?

Les phases et les étapes nécessaires à l'accomplissement de la tâche de notre système robotisé sont représentées sur la fig.7.1.

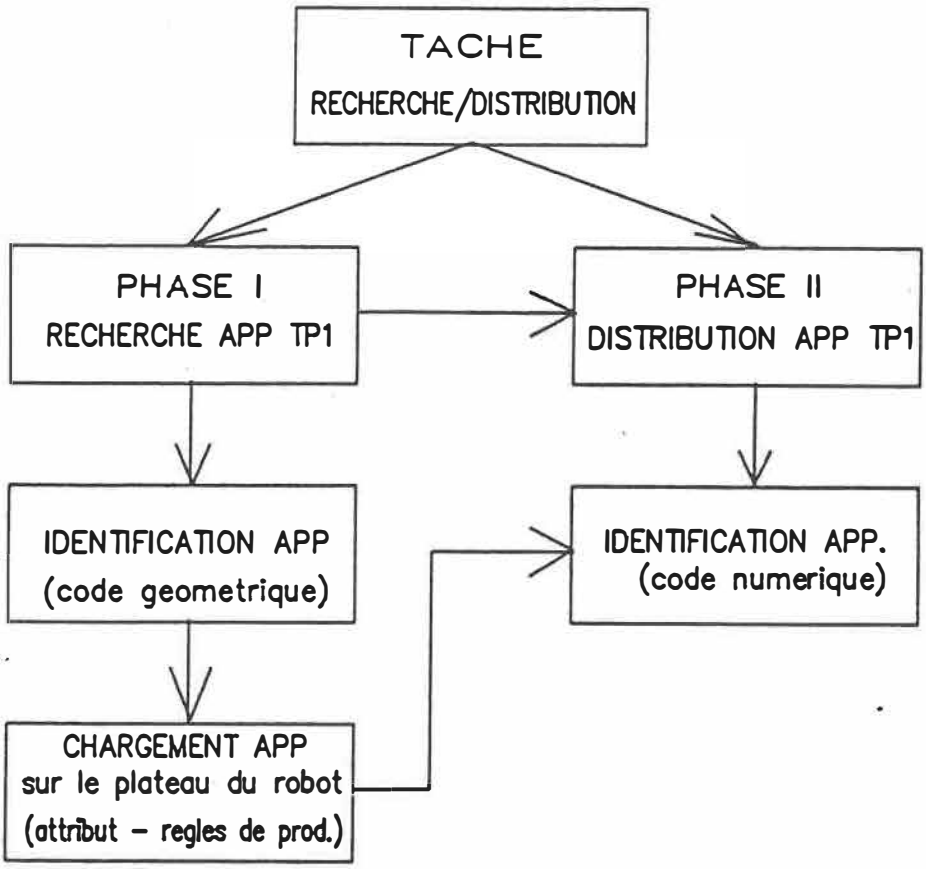


Fig.7.1 Les phases et les étapes nécessaires pour accomplir la tâche du SRRD.

7.4 FORMALISATION MATHEMATIQUE DU PROBLEME

Nous allons voir d'abord les organigrammes des programmes pour réaliser les deux phases nécessaires à l'accomplissement de la tâche, et ensuite la formalisation mathématique du problème.

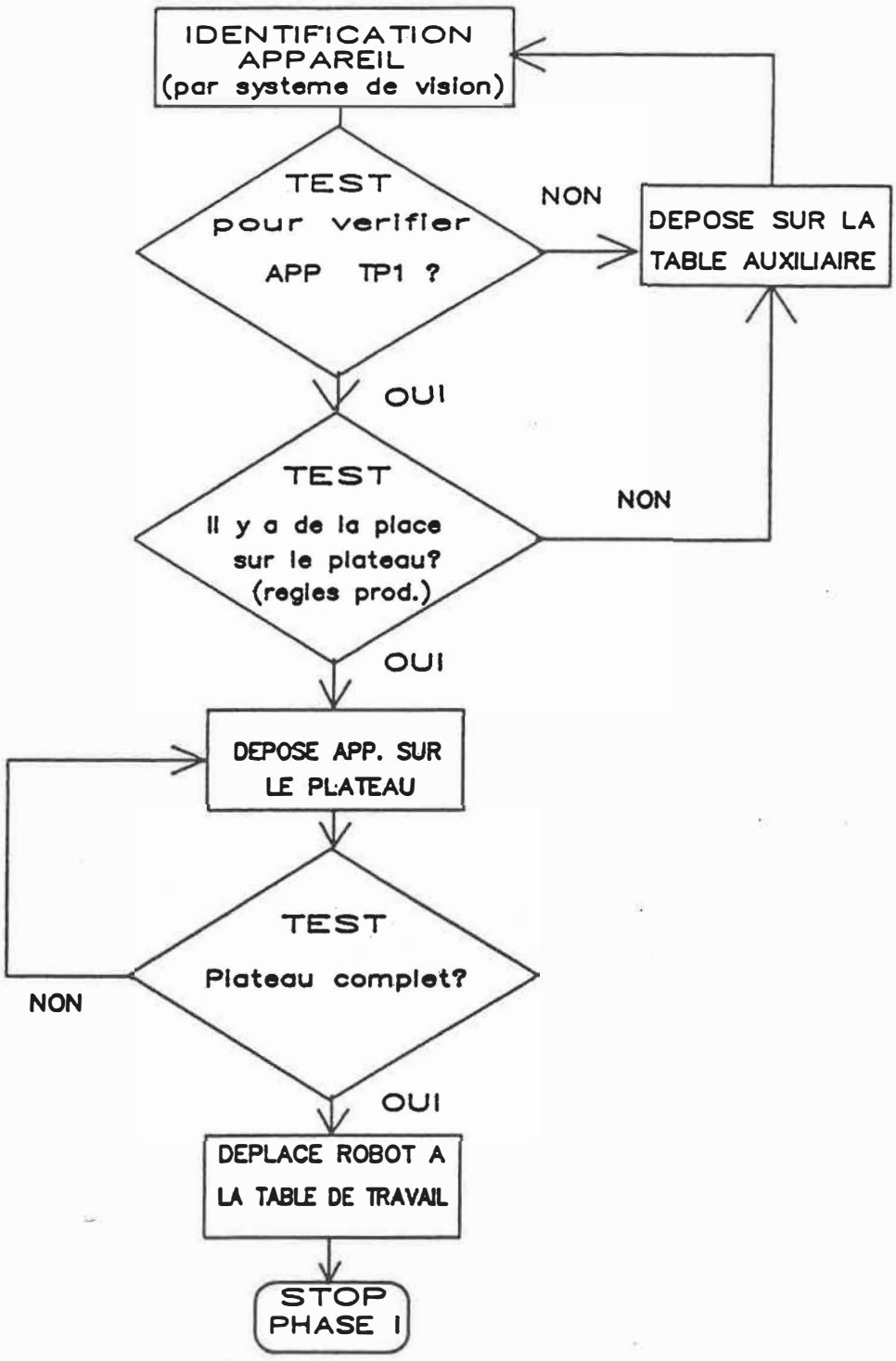


Fig.7.2 PHASE I - RECHERCHE DES APPAREILS APP TP_1

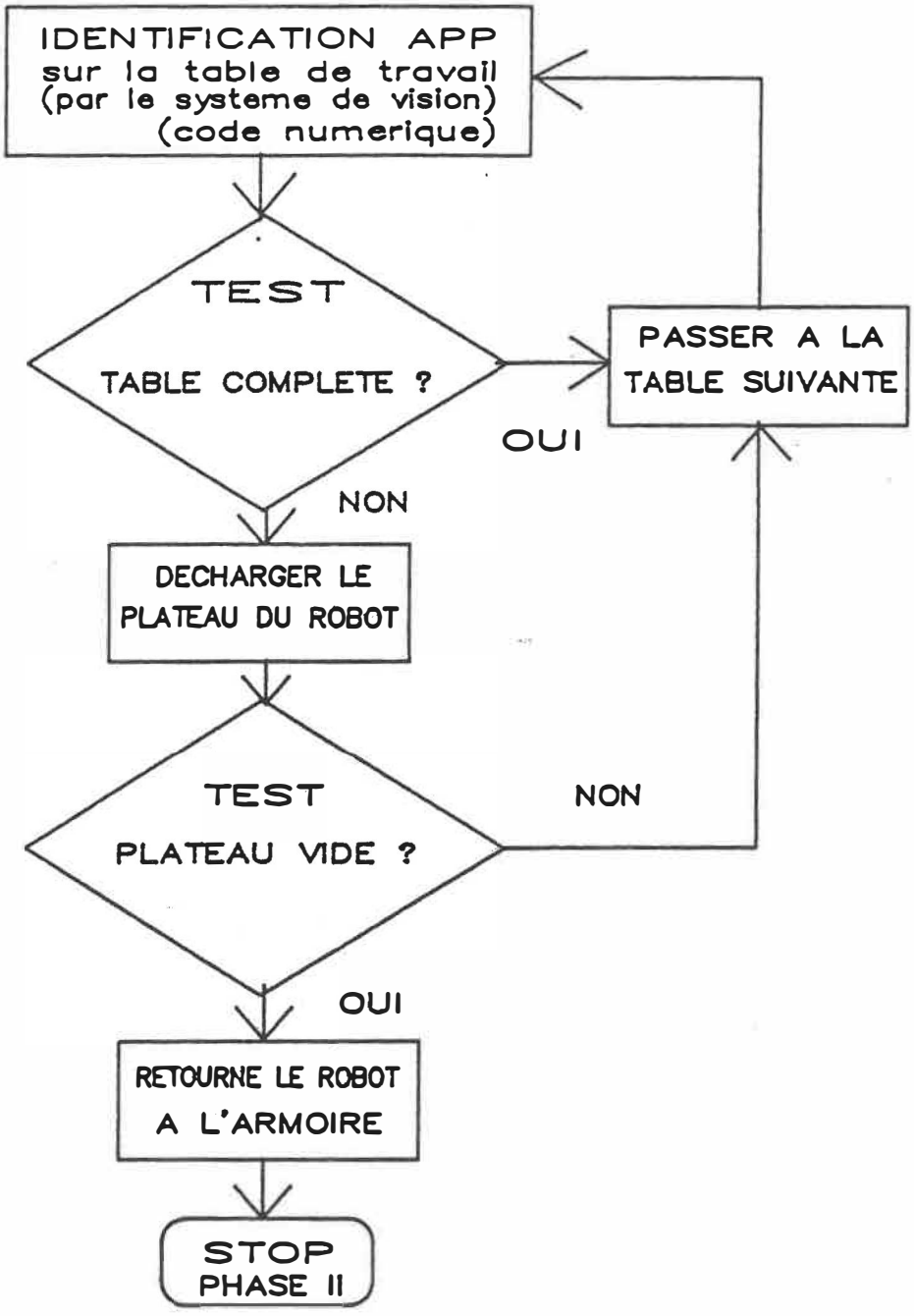


Fig.7.3 PHASE II - DISTRIBUTION DES APPAREILS APP TP.

Pour une résolution correcte de notre problème, nous allons la formaliser mathématiquement, ce qui va nous permettre une vision globale, structurée et claire du problème. Notre système opère sur l'espace des états constitué par les états initiaux pour arriver à des états buts (finals). Pour réaliser cette opération, il utilise une série d'actions.

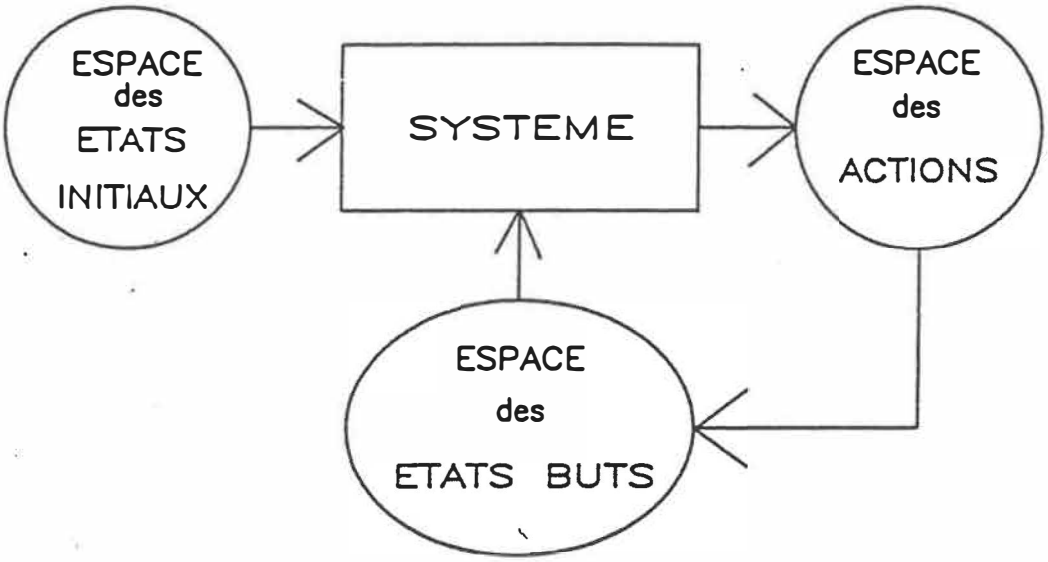
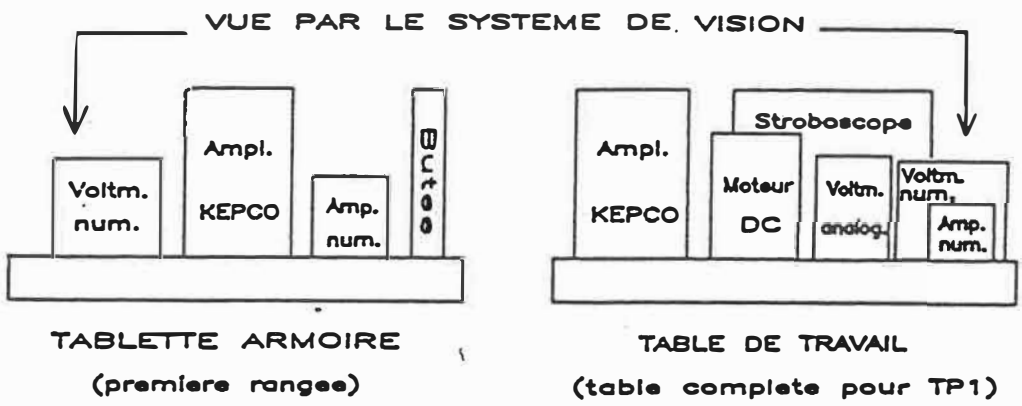


Fig.7.4 Les espaces de travail du système

Les entrées de notre système seront les états initiaux et les états buts désirés, les sorties seront des enchaînements (listes) des actions capables de faire passer le système d'un état à un autre. L'espace des états initiaux est constitué par l'ensemble des états des appareils sur les tablettes des armoires, et aussi par l'ensemble des états intermédiaires des appareils sur les tables de travail. L'espace des états buts

est constitué par l'ensemble des états désirés concernant l'arrangement des appareils sur les tables de travail.

Finalement, l'espace des actions est constitué par l'ensemble des actions nécessaires pour faire passer le système d'un état à un autre. Voilà un exemple d'état initial et final:



ETAT INITIAL	ETAT FINAL
VUE(Voltm_Num)	VUE(Ampl_KEPCO)
VUE(Ampl_KEPCO)	VUE(Moteur_DC)
VUE(Amp_Num)	VUE(Voltm_Num)
VUE(Butée)	VUE(Amp_Num)
LIBRE(pince)	DEV(Moteur_DC, Strob.)
	LIBRE(pince)

Fig.7.5 Exemple des états initial et final du système

Voilà les actions exécutées par le système, et qui sont nécessaires pour passer d'un état à un autre:

IDENTIFICATION_APP	action exécutée par le système de vision du SRRD
PREND_APP	
DEPOSE_APP_PLATEAU	actions exécutées par le
DEPOSE_APP_TAB_AUX	bras manipulateur du SRRD
DEPOSE_APP_TABLE	
DEPLACE_ROBOT_TABLE	actions exécutées par le système de
RETOURNE_ROBOT_ARM	navigation du SRRD

Du point de vue mathématique, une action est une fonction qui permet de passer d'un état à un autre; un état est obtenu par l'application d'une fonction (action) à un état précédent.

$$action(i) = f_1[e(i), e(i+1)]$$

$$e(i+1) = f_2[e(i), action(i)]$$

Par exemple, l'action "DEPOSE_APP_PLATEAU" est appliquée à l'état "APP_pince" pour obtenir l'état "APP_PLATEAU". Aussi, l'état "APP_TABLE" est obtenu de l'état "APP_PLATEAU" par l'application de l'action "PREND_APP" suivie de l'action "DEPOSE_APP_TABLE". Il faut remarquer que l'action "PREND_APP" est toujours suivie d'une action "DEPOSE_APP_..." étant donné que le bras manipulateur doit être libéré pour une autre action (LIBRE_pince). La formalisation mathématique que nous avons exposée, permet l'encadrement du problème avec un certain degré de généralité de façon à pouvoir l'utiliser aussi à d'autres problèmes semblables.

7.5 SCENARIO

Nous allons décrire un "scénario" de recherche et distribution des appareils exécuté par SRRD. Ensuite, notre système expert (SEPT) doit réproduire le méme scénario. D'abord, nous allons formuler quelques hypothèses:

- Le système est muni d'un système de vision, donc il est capable d'identifier les appareils par vision.
- La caméra peut identifier seulement les appareils qui se trouvent à "vue", c'est-à-dire, sur la première rangée de l'armoire. Il sont au nombre de quatre sur une tablette de l'armoire. Pour identifier les appareils qui sont "cachés" sur une rangée en arrière, le robot doit enlever l'appareil qui cache l'autre.
- Pour la distribution, le système de vision doit identifier les appareils existant sur la table de travail, mémoriser et comparer la scène avec une "scène standard" (table complète). Alors, le systène sera capable d'ajouter seulement les appareils qui manquent, pour que la table soit complète (tous les appareils nécessaires pour la manipulation sont sur la table). Les appareils sont distribués en "vrac" sur les tables de travail (sans arrangement particulier précis).

Le SRRD reçoit une commande: "CHERCHER ET DISTRIBUER LES APPAREILS POUR LA MANIPULATION NO...!" Suite à cette commande, le système va enchaîner une série d'actions.

Action I - Le SRRD "regarde" l'armoire avec son système de vision jusqu'au moment où il identifie un appareil

appartenant à la liste des appareils pour la manipulation no...

Action II - Si, parmi les quatre appareils de la première rangée, il n'y en a aucun qui soit dans la liste, il "prend" un appareil, "dépose" l'appareil sur la table auxiliaire pour pouvoir "regarder" la deuxième rangée.

Action III - Une fois identifié l'appareil appartenant à la liste, le robot "prend" l'appareil et le "dépose" sur son plateau.

Action IV - Le système exécute le cycle: regarde-prend-dépose jusqu'au moment où le plateau du robot sera plein.

Action V - Le SRRD se "déplace" jusqu'à une table de travail.

Action VI - Le SRRD "regarde" la table de travail pour établir s'il y a des appareils sur la table. S'il n'y en a pas, il passe à l'action suivante. S'il y en a, alors, soit qu'il passe à la table suivante si la table est complète, soit qu'il "dépose" sur la table les appareils qui manquent.

Action VII - Le SRRD "dépose" les appareils sur la table de travail.

Action VIII - Le SRRD "retourne" à l'armoire avec le plateau vide et recommence le cycle.

Action IX - Le SRRD "répète" le cycle d'actions, de l'action I à l'action VIII, autant de fois que nécessaire pour assurer la fourniture des appareils sur les 10 tables de travail

Action X - Le SRRD "arrête" jusqu'à une nouvelle commande.

7.6 LA REPRESENTATION DE LA CONNAISSANCE

Dans un système expert, il y a trois éléments principaux:

- La base de connaissance constituée par la base de faits et la base de règles. Elle contient les jugements, l'intuition et l'expérience de celui qui l'a construit;
- Le mécanisme d'inférence pour réaliser l'interprétation de la connaissance, les inférences logiques et les modifications dans la base de connaissance;
- Le mécanisme de contrôle constitué par les stratégies de contrôle du processus d'inférence.

Etant donné que nous avons utilisé une "coquille" (VP-EXPERT) pour développer notre système expert, le mécanisme d'inférence et le mécanisme de contrôle viennent avec l'outil de développement. Ainsi, nous avons travaillé sur la base de connaissance. Il y a trois formalismes utilisés plus souvent pour la représentation de la connaissance: les règles de production, les objets structurés et la logique des prédicats. Ces formalismes sont souvent désignés par le terme "Pattern Directed Inference Systems" (Watterman and Hayes-Roth, 1978) soit, systèmes d'inférence dirigés par les filtres.

Nous avons adopté comme formalisme de représentation de la connaissance les règles de production parce que c'est le formalisme le plus utilisé (Wendy B., Rauch-Hindin, 1986) et le plus adapté à notre problème particulier. Ce formalisme a été

utilisé dans la théorie des automates, les grammaires formelles les langages de programmation avant d'être utilisés dans les modèles psychologiques (Newell and Simon, 1972) et les systèmes experts (Buchanan, Feigenbaum, 1978). Notre système utilise la représentation déclarative de la connaissance sous forme de règles de production. Nous avons choisi ce mode de représentation parce qu'il permet de créer des fragments de connaissances indépendants les uns des autres, donc facilement modifiables. Les avantages seront: la lisibilité, l'économie, la souplesse et la facilité de modification de règles.

"Le principe de base de la programmation en règles de production est que chaque règle est un morceau indépendant de connaissance (granularité de la connaissance), c'est-à-dire qu'elle contient toutes les conditions de son application" (Bonnet A, 1984). Alors une règle peut traduire une relation, une action conditionnelle ou une information sémantique.

Une **règle de production** est en général de la forme:

Si prémisse (condition)

Alors conséquence (action)

La partie gauche exprime les conditions d'applicabilité de la règle. Elle peut contenir une conjonction de propositions logiques, de prédicats ou de relations. La partie droite représente la conclusion de la règle qui peut être une action à effectuer ou une assertion à ajouter dans la base de faits. La comparaison entre les conditions d'applicabilité (de déclenchement) d'une règle et les faits existants permet de "filtrer" les règles pour ne retenir que celles qui sont

applicables. La partie prémisse est également appelée filtre de la règle.

Dans un programme informatique classique,"les granules" sont en fait des procédures qui ont des noms leur permettant de s'appeler mutuellement. Le problème réside dans le fait qu'en cas d'ajout ou de suppression de procédures, il faut modifier tous les appels à cette procédure.

Dans un programme déclaratif avec des règles de production, chaque règle consiste en principe à rédiger en ignorant l'existence des autres. Donc les règles ne sont pas déclenchées par leurs noms, mais par leurs conditions d'applicabilité (filtrage). Alors, on peut ajouter ou supprimer des règles sans s'occuper des conséquences de ces modifications.

Voilà ce qui explique pourquoi nous avons adopté la représentation déclarative de la connaissance. D'ailleurs, ce n'est pas par hasard que cette méthode de représentation est utilisée dans la majorité des SE.

Les avantages dus à cette méthode:

- Facilité de modification à cause de la modularité des connaissances.
- Très grande lisibilité des règles et donc facilité d'écriture
- Plus le système possède des règles, plus il est puissant.
- Possibilité d'introduire de coefficients de vraisemblance permettant de pondérer les connaissances.

7.7 LE MECANISME D'INFERENCE ET SON CONTROLE

La base de connaissance, peu importe le formalisme utilisé pour la bâtir, constitue la perception que le système possède concernant son environnement de travail. Pour faire fonctionner le système expert, il faut ajouter à la base de connaissance un mécanisme d'inférence ("moteur d'inférence"). Il est alimenté par la base de connaissance, il construit dynamiquement le raisonnement et décide quelles sont les règles à déclencher et dans quel ordre; donc il utilise la connaissance pour résoudre les problèmes.

Les mécanismes de raisonnement utilisant les règles de production, sont ceux de la logique formelle: le "modus ponens" et son homologue, le "modus tollens" (J.N.Chatain,A.Dussauchoy, 1987). Le "modus ponens" permet à partir de la proposition "si A alors B" et de l'assertion A, d'en déduire B. Il est utilisé en chaînage avant. Le "modus tollens" déduit de la même règle, et de non B, non A. Il est utilisé en chaînage arrière. Le moteur d'inférence de VP-EXPERT utilise les deux mécanismes de raisonnement étant donné qu'il travaille en chaînage arrière (backward chaining) ou en chaînage avant (forward chaining).

Quel que soit le mode de raisonnement utilisé, le moteur d'inférence possède un cycle de base qu'on peut définir comme des cycles de travail enchaînés pour aboutir au résultat; le cycle de base comprend quatre phases (Farreny H, 1985):

- Phase de sélection réalisée sur un sous-ensemble de la base de faits et de la base de règles qui mérite plus d'attention

que le reste de la base. Ce choix tient à une stratégie du moteur qui décide qu'un tel groupe de règles est privilégié par rapport à un autre.

- Phase de filtrage c'est l'étape clé du cycle de base d'un moteur d'inférence qui consiste à comparer la partie prémisse (en chaînage avant) ou la partie conclusion (en chaînage arrière) des règles, avec les faits de la base de faits (en chaînage avant) ou avec l'ensemble des sous-buts (en chaînage arrière) pour déterminer l'ensemble des règles applicables. Le moteur de VP-EXPERT applique le filtrage des règles.

- Phase de résolution de conflits suite à cette phase, on choisit la règle qui va être appliquée effectivement. Pour cette phase, VP-EXPERT utilise le chaînage arrière. Le moteur d'inférence part du but et essaie de remonter aux faits pour le démontrer. Les règles sélectionnées sont celles dont la partie droite correspond au but recherché. Alors les conditions inconnues (partie gauche) de ces règles deviennent des sous-buts à démontrer.

- Phase d'exécution cette phase consiste en l'application de la règle choisie précédemment. L'application de la règle peut avoir plusieurs effets: modification de la base de faits, appel des procédures externes, etc. Il peut également, dans cette phase, questionner l'usager du système. Dans la phase d'exécution, VP-EXPERT utilise le backtracking, le filtrage et la génération de faits suivie de tests (AI-Expert, sept 1987). L'arrêt du cycle de base du moteur d'inférence dépend du mode de raisonnement utilisé. Le moteur du VP-EXPERT travaille en

chaînage avant, dirigé par les données, ou en chaînage arrière, dirigé par le but.

7.8 L'OUTIL DE DEVELOPPEMENT - VP-EXPERT

Concernant le choix d'un outil d'aide au développement du SE, il y a deux solutions:

- employer un outil de développement déjà existant;
- écrire son propre moteur d'inférence avec tout ce qui va autour.

Chacune des solutions possède ses avantages et ses désavantages

Evidemment, écrire son propre moteur d'inférence avec en plus, tout ce qui va autour (éditeur, traceur, etc.) est sûrement la solution la plus longue, la plus coûteuse et la plus difficile, mais pas nécéssairement la meilleure. Par contre, cela permet la construction d'un moteur d'inférence adapté au projet, ce qui pourrait s'avérer nécessaire dans certains cas spéciaux. Si l'ingénieur de la connaissance envisage l'utilisation d'un outil de développement de SE (la première solution), alors il disposera d'un système doté de tous ses utilitaires, prêt à recevoir des connaissances afin de devenir un système expert. Mais, il y a dans ce cas un autre problème à résoudre qui n'est pas facile - le choix d'un outil. Alors, la question qui nous vient à l'esprit immédiatement est: sur quels critères faut-il choisir un outil de développement?

Voilà quelques principes essentiels concernant le choix d'un outil de développement de SE, tirés du livre "Building Expert

Systems" de F.Hayes-Roth chez Addison Wesley, 1983:

- L'outil doit posséder seulement le degré de généralité nécessaire pour résoudre le problème donné, car si l'outil est de type plus général, l'accroissement de ses capacités se fait au détriment de ses performances.
- Tester le logiciel dès le départ en construisant un très petit prototype avant de se lancer dans la réalisation proprement dite.
- L'outil choisi devra posséder les caractéristiques suivantes:
    - Le langage de représentation des connaissances doit être aussi simple et universel que possible;
    - Un moyen d'accéder au mécanisme de contrôle si la généralité est plus importante que l'efficacité;
    - Un système de contrôle très contraint si l'apprentissage, l'automodification ou des explications élaborées sont recherchés;
    - Des capacités de dialogue élaborées (langage quasi-naturel, dictionnaire ...) si le temps de développement est un facteur critique.
- Il faut utiliser (dans la mesure du possible!) un outil qui a déjà servi pour une application comparable.

Une fois qu'on a décidé de la solution à adopter concernant la nature du moteur d'inférence, le principal reste à faire; il faut développer la base de connaissances, ce qui signifie effectuer le transfert d'expertise entre l'homme et la machine. C'est l'étape la plus importante et la plus délicate dans la construction d'un système expert. Le transfert d'expertise

s'effectue de l'expert vers le SE à l'aide du cogniticien. L'enchaînement des phases du processus de transfert de l'expertise est représenté sur la fig.7.6.

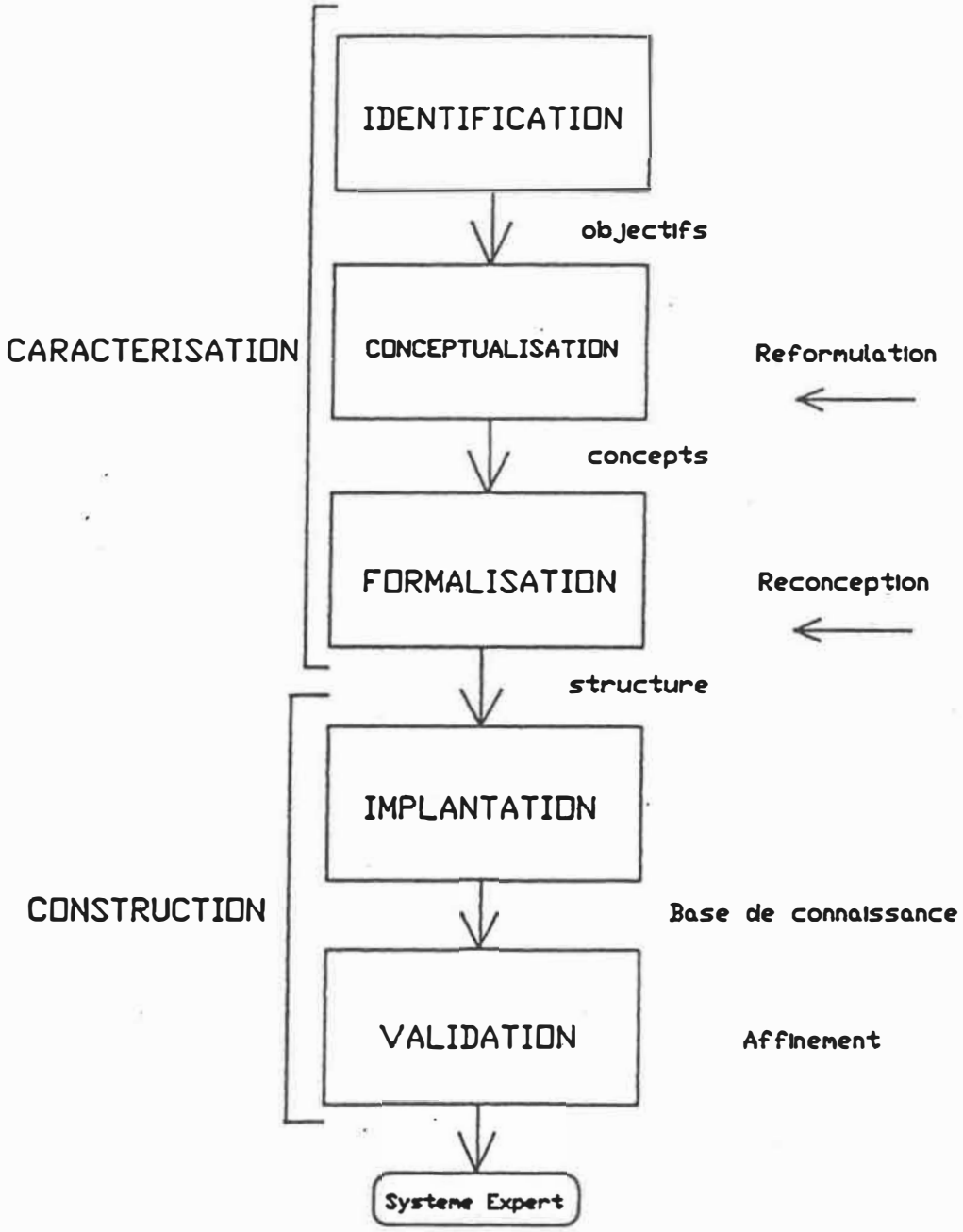


Fig.7.6 Les phases du transfert d'expertise dans la conception d'un système expert.

Ce schéma représente une caractérisation approximative d'une activité complexe et mal structurée qui est de loin la plus importante et la plus délicate dans le processus d'élaboration d'un système expert - la création de la base de connaissance. Après avoir dépassé la phase de caractérisation, la phase de formalisation nous a obligé de choisir un outil d'aide à la construction de SE.

En ce qui concerne l'outil de développement, nous avons testé plusieurs outils: Personal Consultant de Texas Instruments, M1 de Teknowledge Inc. et VP-Expert de Paperback Software International. Les principaux critères qui nous ont permis de déterminer le choix du VP-EXPERT comme outil de développement: l'interface usager dont il dispose est très bien pensé, ce qui permet une utilisation agréable et efficace; le fait que le logiciel est écrit en langage C le rend intéressant à plusieurs points de vue; ses possibilités d'interfaçage avec d'autres programmes écrits en Pascal, Prolog, C, Fortran, Basic dBase, etc.; et en fin de compte, son prix plus qu'accessible.

7.8.1 Représentation des connaissances avec VP-EXPERT

- Les faits

Dans les règles, les faits utilisés sont de la forme:

<OBJET> = <VALEUR>

```
Exemple: IF nombre = nbr_grand AND tp = 1
         THEN nbr_grand = 2
```

Avec les clauses ASK et CHOICES, on peut passer par un menu

pour changer les faits dans la base de connaissances. Par exemple, la suite d'instructions:

ASK tp: "Quelle manipulation (tp) voulez-vous exécuter?";

CHOICES tp: 1, 2, 3, 4;

nous permet de choisir le numéro de la manipulation qu'on veut exécuter. Le choix fait, le programme se charge de modifier les faits dans la base de connaissance.

- Les règles

Les règles sont formalisées de la façon suivante:

REGLE (Nom de la règle)
SI (PREMISSE) ALORS (CONCLUSION) (avec coefficient de vraisemblance)
(optionnel) SI NON (CONCLUSION) (avec coefficient de vraisemblance)
(optionnel) PARCE QUE (TEXT).

Chaque prémisse est de la forme: <VARIABLE> <OPERATEUR LOGIQUE> <VALEUR>. Exemple: IF nbr_grand = 0 AND nbr_moyenne<4

THEN act = PREND_APPAREIL;

La prémisse peut être une conjonction ou une disjonction de clauses. La conclusion de la règle peut être une liste d'une ou plusieurs expressions de la forme: <VARIABLE> = <VALEUR> avec coefficient de vraisemblance et avec une éventuelle conclusion alternative. Il y a aussi la possibilité d'ajouter un texte explicatif avec l'option PARCE QUE (BECAUSE) qu'on peut utiliser à la fin de la règle.

7.8.2 La structure d'un programme écrit a l'aide du VP-EXPERT

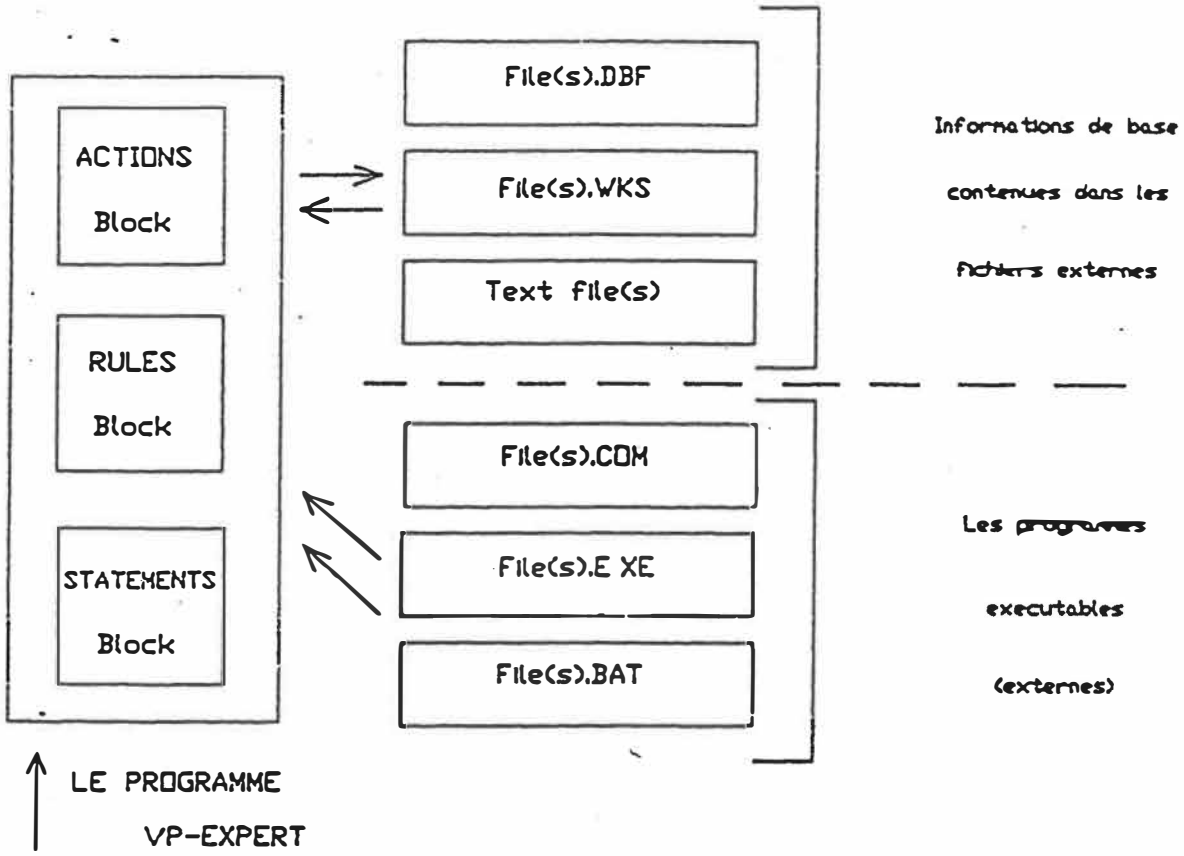


Fig.7.7 La structure d'un programme écrit a l'aide du VP-EXPERT

Un programme doit avoir une structure précise: un bloc ACTIONS suivi d'un bloc RULES et, a la fin du programme, les instructions (STATEMENTS).

Le bloc ACTIONS joue le rôle du programme principal. Il contient le keyword "ACTIONS" suivi de plusieurs clauses. C'est le mécanisme qui dirige les actions à exécuter pendant la consultation. Les clauses sont exécutées dans l'ordre. Voici quelques exemples de clauses qui font partie du bloc "ACTIONS":

La clause DISPLAY "<text>" permet l'affichage des messages sur l'écran de l'ordinateur. D'habitude, c'est la première clause du bloc ACTIONS étant donné que le programme est intéractif.

La clause FIND "<variable>" identifie par son argument un but que le moteur cherche à évaluer. Par exemple, quand la clause FIND action est exécutée, le moteur d'inférence survole la base de connaissance pour trouver la valeur de la variable "action". Une fois qu'il a trouvé une valeur de la variable, le but est atteint et la consultation, terminée. Pour que le moteur soit capable de trouver toutes les valeurs de la variable "action", il faut la déclarer dans le bloc STATEMENTS, qui est une variable PLURAL. La clause FIND peut être utilisée aussi dans une règle (dans la partie conclusion). Le but est d'instruire le moteur pour trouver une variable particulière quand la règle est passée.

```
Exemple: RULE 15
         IF act = VUE_APPAREIL
         THEN action = pince
         FIND pince;

         ASK pince: "La pince du robot est libre?";
         CHOICES pince: Oui, Non;
```

La clause MENU génère un menu avec les options qui accompagnent une question affichée par une instruction ASK. Exemple:

```
ACTIONS
MENU tp_numéro, ALL, tp, numéro
FIND tp_numéro;
```

ASK tp_numéro: "Quel numéro de tp voulez-vous exécuter?";

CHOICES tp_numéro: 1, 2, 3, 4;

Il y a d'autres clauses qui peuvent être placées dans le bloc ACTIONS (voir VP-EXPERT Rule-Based Expert System Development Tool, 1987).

Le bloc RULES qui suit le bloc ACTIONS contient la base de règles du système expert. Il représente la partie du système contenant toute "l'expertise". La syntaxe des règles, nous l'avons expliquée dans le paragraphe concernant la représentation des connaissances. Voici des détails concernant l'utilisation des règles pendant la consultation. Exemple:

IF nbr_grand=0 AND nbr_moyenne<4

THEN act=PREND_APP;

Si, pendant la consultation, la valeur de la variable nbr_grand est trouvée égale à zéro, et la valeur de la variable nbr_moyenne est trouvée plus petite que 4, alors la variable act sera assignée avec la valeur PREND_APP. Si, pendant la consultation, la prémisse d'une règle est trouvée vraie, on assigne à la variable qui se trouve dans la conclusion la valeur correspondante (celle donnée par la règle) et on dit que la règle a "passé". Cela signifie que la règle en question a été testée par le mécanisme d'inférence.

Le bloc STATEMENTS contient les instructions: ASK, CHOICES et PLURAL. L'instruction ASK crée un message qui demande à l'usager l'introduction d'une valeur pour une variable donnée. Si une variable est utilisée comme argument pour l'instruction ASK et associée à un message, alors pendant la consultation, la

première fois que la variable est rencontrée, le message associé sera envoyé sur l'écran. L'instruction CHOICES permet la création d'un menu avec plusieurs options. Elle accompagne toujours l'instruction ASK pour permettre à l'usager le choix d'une valeur pour la variable. Une fois que l'usager a fait le choix, la consultation se poursuit. L'instruction PLURAL identifie les variables pour lesquelles on assigne plusieurs valeurs durant la consultation. Elle permet de déclarer une variable d'une manière telle que le moteur d'inférence puisse la chercher plusieurs fois; sinon, la consultation sera arrêtée dès que la variable aura été trouvée pour la première fois. Quand la phase de tests et de corrections du système est terminée, on ajoute deux instructions juste avant le bloc ACTIONS: RUNTIME et EXECUTE. La première instruction: RUNTIME est ajoutée à la base de connaissance quand le développement est terminé, et la base de connaissance est prête pour la consultation par l'usager. Alors VP-EXPERT n'affiche plus sur l'écran les actions du système pendant la consultation.

L'instruction EXECUTE permet le déclenchement automatique de la consultation juste après l'exécution de la commande Consult du Main Menu. Si elle n'existe pas après RUNTIME et juste avant ACTIONS, l'usager doit utiliser la commande "GO" pour lancer la consultation.

7.9 IMPLANTATION ET FONCTONNEMENT DU PROGRAMME

L'utilisation même du VP-EXPERT comme outil de développement fait que le programme soit intéractif, ce qui facilite beaucoup la tâche de l'utilisateur. Quand on lance le programme, il y a un message sur l'écran avec une brève description d'utilisation Voici les principales directives. Pour charger le VP-EXPERT, il faut taper "VPX". Dès que le logiciel est chargé, il y a un menu de commandes qui apparaît en bas de l'écran. A l'aide des flèches on choisit le FileName pour donner le nom du programme: ROBOT.KBS qu'on désire charger. Suite au choix d'une option, on fait "ENTER" pour envoyer la commande choisie. Les noms des programmes (.KBS) qui se trouvent dans la directory VP-EXPERT seront affichés dès qu'on aura choisi FileName. Les flèches permettent le choix du nom du programme et avec "ENTER", on charge le programme. Après que le chargement a été accompli, un autre menu de commandes sera affiché. Alors on choisit "Consult" et le programme commence à se dérouler. Pendant le déroulement du programme (option "GO" après "Consult"), le menu de commande (en bas de l'écran) ne s'affiche pas. Si le programme s'arrête (et le menu de commandes n'est toujours pas affiché), il faut taper "ENTER" pour continuer. Si par contre, le programme est arrêté et le menu de commandes est affiché, alors la consultation est terminée. Pour continuer une autre consultation, il faut choisir l'option "GO" suivie de "ENTER". Pour quitter le VP-EXPERT, il faut choisir l'option "QUIT".

En ce qui concerne l'implantation de la base de connaissance nous avons utilisé les règles de production de la façon que nous l'avons expliqué plus haut. L'hypothèse fondamentale qui se trouve à la base du programme est la suivante: les informations qui doivent être fournies par le module de perceptions (capteurs) sont remplacées par les réponses de l'usager.

A la fin du chapitre nous avons annexé le listing d'une partie du programme écrit à l'aide du VP-EXPERT. Le programme est intéractif et contient les messages nécessaires pour guider l'utilisateur pendant la consultation du système expert.

CHAPITRE 8

CONCLUSION

Ce projet a permis d'étudier les possibilités de robotiser des processus de recherche et distribution des objets en utilisant des techniques classiques et des techniques d'Intelligence Artificielle sur un exemple pratique offert par un laboratoire universitaire. En particulier, il met en évidence la possibilité d'utiliser un système expert pour résoudre le problème de génération de plan pour un robot.

8.1 RESULTATS OBTENUS

Les principales conclusions qui se dégagent de ce projet sont les suivantes:

a) La conception du système dans son ensemble nous a amené à une solution à plusieurs niveaux:

- au niveau inférieur se trouve le **système de locomotion** constitué par le chariot mobile muni d'un **sous-système de navigation.** Etant donné les particularités de notre problème au niveau de la navigation (trajectoires précises topologie de l'environnement connue et non variable), nous avons choisi comme solution la **navigation optique** avec ruban guide pour des raisons entre autres de simplicité, d'efficacité, et parce que la méthode a été déja expérimentée à l'école avec des bons résultats dans un autre projet.

- au niveau supérieur, se trouve le système de perception multisenseurs qui constitue le sous-système de guidage qui permettra au robot (bras manipulateur) de remplir sa tâche concernant la manipulation d'objets. Nous avons prévu dans le cadre du système multisenseurs:
  - un système de vision qui permet le guidage du bras manipulateur pour remplir sa tâche;
  - un système d'émetteurs-récepteurs à ultrasons qui assure une fonction de sécurité en permettant d'éviter les collisions;
  - un système d'émetteurs-récepteurs optiques (infra-rouges) qui assure une deuxième fonction de sécurité permettant au robot de connaître sa position et son orientation dans l'environnement de travail (balisé) quand il est "perdu" suite à une défectuosité du système de navigation.

b) Le robot mobile nécessite une adaptation, étant donné qu'il est constitué d'un bras manipulateur monté sur un chariot mobile.

c) Concernant la génération de plan pour l'accomplissement de la tâche, nous avons donné deux solutions:

- un générateur de plan dirigé par le but - une version de base écrite en PROLOG (Turbo-Prolog);

- un système expert pour la planification des tâches du SRRD (système robotisé pour recherche\distribution) développé avec VP-Expert, outil de développement assez puissant avec des possibilités très intéressantes au niveau de l'interfaçage avec d'autres langages ou logiciels. En outre, nous avons fait l'interface avec une base de données écrite en Turbo-Prolog et une autre en dBase III-Plus. Le développement du système expert en utilisant la représentation de la connaissance sous forme de règles de production s'est avéré bien adapté à notre problème particulier et nous a permis d'explorer une voie originale pour la génération de plan.

En conclusion générale, ce projet est une des représentations des robots de la troisième génération dotés de l'intelligence artificielle qui est un domaine de plus en plus adopté présentement par des industries ouvrant dans le domaine de la robotique.

8.2 EXTENSIONS POSSIBLES

Pour une première approche globale du système robotisé pour recherche/distribution, la présente étude a atteint les objectifs fixés. Il serait néanmoins intéressant d'investiguer les approches suivantes:

- faire une implantation du générateur de plan (écrit en LISP) sur une machine LISP pour investiguer une autre approche et exploiter d'avantage les possibilités offertes par une machine spécialisée;

- étudier d'avantage le système de guidage au niveau du système de vision pour résoudre le problème de reconnaissance des objets dans un contexte plus général;

- étudier la possibilité de la génération des règles à partir directement de l'information fournie par les capteurs, donc faire une étude de l'interface entre le système de capteurs et le générateur de plan.

REFERENCES

[1] E.RICH, "ARTIFICIAL INTELLIGENCE", McGraw - Hill Book Company (1985).

[2] M. CONDILLAC, "Prolog - fondements et applications", AFCET Informatique - DUNOD (1985).

[3] P.COIFFET, "Les Robots - interaction avec l'environnement", HERMES (1981).

[4] Y.SHIRAI,J.TSUJII, "ARTIFICIAL INTELLIGENCE CONCEPTS, TECHNIQUES AND APPLICATIONS", John Wiley & Sons (1985)

[5] P.H.WINSTON, "ARTIFICIAL INTELLIGENCE", Addison Wesley (1984)

[6] N.NILSSON, "PRINCIPLES OF ARTIFICIAL INTELLIGENCE", Tioga Press (1985).

[7] P.CHARNIACK, Mc.DERMOTT, "INTRODUCTION TO ARTIFICIAL INTELLIGENCE", Addison Wesley (1985).

[8] R.M.DeSANTIS, "Commande des systèmes robotiques", Ecole Polytechnique de Montréal (1986).

[9] J.ALLEN, "PLANING PROBLEMES", Journal of Artificial Intelligence - january 1984.

[10] H.SCHILDT, "ADVANCED TURBO PROLOG", Osborne, McGraw - Hill (1986).

[11] PROCEEDINGS TRENDS & APPLICATIONS 1983 AUTOMATING INTELLIGENT BEHAVIOR APPLICATIONS AND FRONTIERS May 25-26 1983.

[12] J.W.COURTNEY, J.K.AGGARWAL, "ROBOT GUIDANCE USING COMPUTER VISION", The University of Texas at Austin.

[13] S.Y.HARMON, "COORDINATION BETWEEN CONTROL AND KNOWLEDGE BASED SYSTEMS FOR AUTONOMOUS VEHICLE GUIDANCE", Naval Ocean Systems Center San Diego,CA.

[14] N.S.RAJARAM, "DESIGN OF INTELLIGENT SYSTEMS WITH COOPERATING KNOWLEDGE BASED COMPONENTS", University of Houston (1985).

[15] D.R.BLIDBERG, A.S.WESTNEAT,R.W.CORELL, "EXPERT SYSTEMS A TOOL FOR AUTONOMOUS UNDERWATER VEHICLES", Marine Systems Engineering Laboratory University of New Hampshire (1987).

[16] M.SOREL,"La robotique americaine a l'heure des bilans" Sciences & Techniques No.39 Juillet-Août 1987

[17] LAROUSSE de la langue française - LEXIS 1986

[18] ROBERT METHODIQUE 1986

[19] V.D.HUNT, "SMART ROBOTS A HANDBOOK OF INTELLIGENT ROBOTIC SYSTEMS", Chapman and Hall, New York, London.

[20] H.COELHO, J.C.COTTA, L.M.PEREIRA, "HOW TO SOLVE IT WITH PROLOG", Laboratorio Nacional de Engenharia Civil Lisboa, Julho de 1985.

[21] RMDS.6 SOUPHANDAVONG, P., DeSANTIS, R.M, "Exécution robotisé du jeu de la Tour d'Hanoi, EMP/RT -87/15, Mai 1987.

[22] RABEMANANTSOA, M., "Analyse par voie de simulation du système d'autopilotage d'un vehicule mobile." Mémoire M.Sc.A. Eté 1989.

LES ANNEXES

ANNEXE A

LE PROGRAMME DE SIMULATION "SIMULROB.PAS"

```
program MovePolygon;

{$I typedef.sys}                    {these files must be}
{$I graphix.sys}                    {included in this order}
{$I kernel.sys}
{$I windows.sys}
{$I polygon.hgh}
{$I modpoly.hgh}

var ArrowAngle,i,Alpha,b,U: integer;
    Ch: char;
    Arrow: PlotArray;
    CurrX,CurrY,IncrX,IncrY,Size,L,V,Dim,Speed: real;
    ArrowIncr: array[0..7,1..2] of real;
 procedure MakeArrow;
begin
  Arrow[1,1]:=0;                    {PlotArray init for the arrowhead}
  Arrow[1,2]:=-800;
  Arrow[2,1]:=0;
  Arrow[2,2]:=Dim-800;
  Arrow[3,1]:=Size;
  Arrow[3,2]:=Dim-800;
  Arrow[4,1]:=Size;
  Arrow[4,2]:=-800;
  Arrow[5,1]:=0;
  Arrow[5,2]:=-800;
end;

procedure MakeMovelable;
begin
  ArrowIncr[0,1]:=0;                {component velocities for radial moves}
  ArrowIncr[0,2]:=1;
  ArrowIncr[1,1]:=-1;
  ArrowIncr[1,2]:=1;
  ArrowIncr[2,1]:=-1;
  ArrowIncr[2,2]:=0;
  ArrowIncr[3,1]:=-1;
  ArrowIncr[3,2]:=-1;
  ArrowIncr[4,1]:=0;
  ArrowIncr[4,2]:=-1;
  ArrowIncr[5,1]:=1;
  ArrowIncr[5,2]:=-1;
  ArrowIncr[6,1]:=1;
  ArrowIncr[6,2]:=0;
  ArrowIncr[7,1]:=1;
  ArrowIncr[7,2]:=1;
end;

procedure MoveForward;              {routine to move polygon forward}
begin
  SetColorBlack;                    {draw over old polygon to erase it}
  DrawPolygon(Arrow,1,5,0,0,0);
  CurrX:=CurrX+IncrX;               {move to new position}
  CurrY:=CurrY+IncrY;
  TranslatePolygon(Arrow,5,IncrX,IncrY);
  SetColorWhite;                    {draw polygon in new position}
  DrawPolygon(Arrow,1,5,0,0,0);
end;
```

```
procedure MoveBack;                    {routine to move polygon back}
begin
  SetColorBlack;                       {same as above}
  DrawPolygon(Arrow,1,5,0,0,0);
  CurrX:=CurrX-IncrX;
  CurrY:=CurrY-IncrY;
  TranslatePolygon(Arrow,5,-IncrX,-IncrY);
  SetColorWhite;
  DrawPolygon(Arrow,1,5,0,0,0);
end;

procedure TurnLeft;                    {rotate polygon counter-clockwise}
begin
  SetColorBlack;                       {undraw old polygon}
  DrawPolygon(Arrow,1,5,0,0,0);
  RotatePolygon(Arrow,5,Alpha);           {rotate it 45 degrees}
  ArrowAngle:=ArrowAngle+1;
  if ArrowAngle>7 then ArrowAngle:=0;
  IncrX:=Speed * ArrowIncr[ArrowAngle,1];  {get new velocity}
  IncrY:=Speed * ArrowIncr[ArrowAngle,2];
  SetColorWhite;                       {draw rotated polygon}
  DrawPolygon(Arrow,1,5,0,0,0);
end;

procedure TurnRight;                   {rotate polygon clockwise}
begin
  SetColorBlack;                       {same as above}
  DrawPolygon(Arrow,1,5,0,0,0);
  RotatePolygon(Arrow,5,-Alpha);
  ArrowAngle:=ArrowAngle-1;
  if ArrowAngle<0 then ArrowAngle:=7;
  IncrX:=Speed * ArrowIncr[ArrowAngle,1];
  IncrY:=Speed * ArrowIncr[ArrowAngle,2];
  SetColorWhite;
  DrawPolygon(Arrow,1,5,0,0,0);
end;

procedure MoveG;
var L:integer;
begin
     L:=G;
     repeat
          Speed:=V;
          MoveForward;
          TurnLeft;
          L:=L-1;
     until L=0;
 end;

procedure MoveD;
var L:integer;
begin
     L:=D;
     repeat
          Speed:=V;
          MoveForward;
          TurnRight;
```

```
          L:=L-1;
      until L=0;
end;

begin

InitGraphic;                          {initialize the graphics system}

DefineWindow(1,0,0,XMaxGlb,YMaxGlb);
DefineWorld(1,-1000,1000,1000,-1000);  {give it a world coordinate system}

SelectWorld(1);                       {select it's world}
SelectWindow(1);                      {select window}
SetBackground(0);                     {give it a black background}
Writeln('SIMULATION du DEPLACEMENT du ROBOT - NAVIGATION OPTIQUE');
Writeln('Les parametres de la simulation:');
Writeln('Largeur du ruban = L(350...500/100cm)');
Writeln('Vitesse du robot = V(0....100/40m/sec)');
Writeln('Perturbation = A(1,3,5,10,15 degres');
Writeln('Vitesse angulaire(gauche) = G(1,2,3*Alpha*Pi/k rad/sec)');
Writeln('Vitesse angulaire(droite) = D(1,2,3*Alpha*Pi/k rad/sec)');
Writeln('Largeur du ruban guide L=');
Readln(L);
Writeln('Vitesse du robot V=');
Readln(V);
Writeln('Perturbation Alpha=');
Readln(Alpha);
Writeln('Vitesse angulaire(gauche) G=');
Readln(G);
Writeln('Vitesse angulaire(droite) D=');
Readln(D);

DrawLine(-50,-800,-50,1000);
DrawLine(L,-800,L,1000);

Size:=350;
Dim:=400;
Speed:=V;
Curr1:=0;
CurrY:=0;
ArrowAngle:=0;
Incr1:=0;
IncrY:=Speed;

MakeArrow;                            {make the arrowhead}
MakeMovelable;                        {make the move table}
DrawPolygon(Arrow,1,5,0,0,0);         {draw it pointing up}
MoveG;
Speed:=V/2;
MoveBack;
MoveBack;
MoveBack;
MoveD;


repeat
  read(Kbd,Ch);                       {read the keystroke}
  case ord(Ch) of
```

```
      72 : MoveForward;              {up arrow?}
      75 : TurnLeft;                 {left arrow?}
      77 : TurnRight;                {right arrow?}
      80 : MoveBack;                 {down arrow?}
  end;
until Ch= ' ';                       {'space' char exits program}

LeaveGraphic;                        {leave the graphics system}

end.
```

ANNEXE B

LE PROGRAMME GENERATEUR DE PLAN - Version TURBO-PROLOG

(avec exemple de deroulement du programme)

```
/* PROGRAMME GENERATEUR DE PLAN */
/* Version TURBO-PROLOG */
/* Reference WARPLAN */

/*
Poser les questions suivantes :


plan([dev(e,arm),dev(f,arm),dev(g,arm),dev(b,e),dev(b,e),dev(d,b),vue
),vue(h),dev(c,g),dev(c,g),dev(a,c),vue(a)],[depart(init)])


Reponse :

      depart("init")
      deplacer("b","arm","c")
      deplacer("b","c","arm")
      deplacer("c","a","arm")
      deplacer("a","arm","b")
      True
*/


trail = 1000
domains
    a = deplace_app(p,p,p);depart(p)
    p = reference symbol
    l = dev(p,p);vue(p);dif(p,p)
    llist = l*
    alist = a*

predicates

    ecrire(alist)
    planif(llist,alist,alist)
    realise(l,alist,alist)
    exact(l,alist)
    ajoute(l,a)
    action(a,llist)
    supp(l,a)
    donnee(l)
    plan(llist,alist)
    dif(p,p)
```

```
clauses

    plan(G,S) :-
        planif(G,S,S1),!,
        ecrire(S1).

    ecrire([S1|S2])  :-
        ecrire(S2),
        write(S1), nl.
    ecrire([]).

    planif([B|R],S,S2) :-
        realise(B,S,S1),
        planif(R,S1,S2).
    planif([],S,S).

    realise(dif(B,C),S,S) :-
        dif(B,C).
    realise(B,S,S) :-
        exact(B,S).
    realise(B,S,[Act|S1])  :-
        ajoute(B,Act),
        action(Act,Cond),
        planif(Cond,S,S1).

    exact(F,[Act|_]) :-
        ajoute(F,Act).
    exact(F,[Act|S]) :-
        exact(F,S),
        not(supp(F,Act)).
    exact(F,[depart(init)]) :-
        donnee(F).
    dif(X,X) :-
        !,fail.
    dif(_,_).


/* DEFINITION DES ACTIONS */
/* La liste des actions et les listes des adjonctions et des suppressi


    action(deplace_app(A,B,arm),[dev(A,B), vue(A),dif(B,arm)]).
    action(deplace_app(A,B,C),[vue(C), dev(A,B), vue(A),dif(A,C)]).

    ajoute(dev(A,C),deplace_app(A,_,C)).
    ajoute(vue(B),deplace_app(_,B,_)).

    supp(dev(A,B),deplace_app(A,B,_)).
    supp(vue(C),deplace_app(_,_,C)).
```

```
/* ETAT INITIAL */
/* L'etat initial peut etre modifier en fonction du probleme */

    donnee(dev(a,arm)).
    donnee(dev(b,arm)).
    donnee(dev(d,arm)).
    donnee(dev(c,a)).
    donnee(dev(e,d)).
    donnee(vue(c)).
    donnee(vue(b)).
    donnee(vue(e)).
```

```
Goal: plan([dev(a,b),dev(c,a)],[depart(init)])
depart("init")
deplace_app("c","a","arm")
deplace_app("a","arm","b")
deplace_app("c","arm","a")
True
Goal: plan([dev(e,arm),dev(c,arm),dev(b,e),dev(b,e),dev(d,b),vue(d),dev
[depart(init)])
depart("init")
deplace_app("e","d","arm")
deplace_app("c","a","arm")
deplace_app("b","arm","e")
deplace_app("d","arm","b")
deplace_app("d","b","arm")
deplace_app("b","e","arm")
deplace_app("a","arm","e")
True
```

```
[depart(init)])
depart("init")
deplace_app("e","d","arm")
deplace_app("c","a","arm")
deplace_app("b","arm","e")
deplace_app("d","arm","b")
deplace_app("d","b","arm")
deplace_app("b","e","arm")
deplace_app("a","arm","e")
True
Goal: plan([dev(e,arm),dev(d,e),dev(c,d),dev(b,c),dev(a,b)],[depart(init
depart("init")
deplace_app("e","d","arm")
deplace_app("d","arm","e")
deplace_app("c","a","d")
deplace_app("b","arm","c")
deplace_app("a","arm","b")
True
```

ANNEXE C

LE LISTING DU PROGRAMME ROBOT21.KBS

(developé à l'aide du logiciel VP-EXPERT)

```
ACTIONS
DISPLAY "SYSTEME EXPERT pour la PLANIFICATION des TACHES du SYTEME ROBOTISE
pour RECHERCHE/DISTRIBUTION (SRRD) des appareils dans un
Laboratoire d'Asservissements.
Frapez une touche pour continue !"
DISPLAY
"Le SRRD est muni d'un systeme decisionnel qui represente son cerveau.
Le systeme decisionnel(logiciel) commande le bras manipulateur du robot
en fonction des informations concernant l'environnement de travail.
Les informations sont envoyer par le systeme des sensors.
En principal par le systeme de vision.
Frapez une cle pour continue !"
DISPLAY
"Le SYSTEME EXPERT permet la simulation du fonctionnement du
systeme decisionnel qui ete concu comme un 'systeme de production'.
L'operateur qui fait la consultation du systeme expert doit repondre
au questions pose par le systeme et aussi il doit tenir compte de
ses recommandations.Frapez une cle pour continue !"
DISPLAY
"CONSULTATION du SE: Voila la procedure de consultation du systeme.
La consultation commence par GO (menu principal)
Vous choissisez la reponse a la question pose par le systeme et
apres vous tapez 'Enter' suivie de 'End'. Alors le consultation est
lance, le moteur fait les inferences en survolant la base de connaissance
et renvoiesoit une autre question soit le resultat de la consultation.
Frapez une cle pour continue !"
DISPLAY
"Si la consultation est arrete mais le menu principal n'est pas sur l'ecran
vous devez tapez toujours sur 'Enter' pour continue.
Quand le menu principal est affiche la consultation est termine et vous
avez le choix entre GO pour recommancer ou QUIT pour quitter
ou une autre choix eventuellement. BONNE CHANCE !! Frapez une cle !".

FIND action;
RULE 1
IF tp=1
THEN action=etat
FIND act;
RULE 2
IF tp=2
THEN action=etat
FIND act;
RULE 3
IF tp=3
THEN action=etat
FIND act;
RULE 4
IF tp=4
THEN action=etat
FIND act;
RULE 5
IF etat=E1_G_G_G_M
THEN act=E1_G_G_G_M
DISPLAY "Etat initial=(etat)"
FIND act;
RULE 6
IF etat=E2_G_G_G_P
THEN act=E2_G_G_G_P
```

```
DISPLAY "Etat initial=(etat)"
FIND act;
RULE 7
IF etat=E3_G_G_M_P
THEN act=E3_G_G_M_P
DISPLAY "Etat initial=(etat)"
FIND act;
RULE 8
IF etat=E4_G_G_M_M
THEN act=E4_G_G_M_M
DISPLAY "Etat initial=(etat)"
FIND act;
RULE 9
IF etat=E5_G_G_P_P
THEN act=E5_G_G_P_P
DISPLAY "Etat initial=(etat)"
FIND act;
RULE 10
IF etat=E6_G_M_M_M
THEN act=E6_G_M_M_M
DISPLAY "Etat initial=(etat)"
FIND act;
RULE 11
IF etat=E7_G_M_M_P
THEN act=E7_G_M_M_P
DISPLAY "Etat initial=(etat)"
FIND act;
RULE 12
IF etat=E8_G_M_P_P
THEN act=E8_G_M_P_P
DISPLAY "Etat initial=(etat)"
FIND act;
RULE 13
IF etat=E9_M_M_M_M
THEN act=E9_M_M_M_M
DISPLAY "Etat initial=(etat)"
FIND act;
RULE 14
IF etat=E10_M_M_M_P
THEN act=E10_M_M_M_P
DISPLAY "Etat initial=(etat)"
FIND act;
RULE 15
IF etat=E11_M_M_P_P
THEN act=E11_M_M_P_P
DISPLAY "Etat initial=(etat)"
FIND act;
RULE 16
IF etat=E1_G_G_G_M
THEN act=CG
FIND premier;
RULE 17
IF etat=E1_G_G_G_M
THEN premier=G
DISPLAY "Premier appareil a la grandeur (premier)"
FIND deuxieme;
RULE 18
IF etat=E1_G_G_G_M
```

```
THEN deuxieme=6
DISPLAY "Deuxieme appareil a la grandeur (deuxieme)"
FIND troisieme;
RULE 19
IF etat=E1_6_6_6_M
THEN troisieme=6
DISPLAY "Troisieme appareil a la grandeur (troisieme)"
FIND quatrieme;
RULE 20
IF etat=E1_6_6_6_M
THEN quatrieme=M
DISPLAY "Quatrieme appareil a la grandeur (quatrieme)"
FIND act;
RULE 21
IF etat=E1_6_6_6_M
THEN act=grandeur
DISPLAY "Appareil app1=(premier),app2=(deuxieme),app3=(troisieme),app4=(quatrieme)"
FIND act;
RULE 22
IF act=grandeur
THEN act=appartient;
RULE 23
IF etat=E1_6_6_6_M AND CG=carre
THEN act=nombre
DISPLAY "App1 apartient au tp1,il a la grandeur (premier)";
RULE 24
IF etat=E1_6_6_6_M AND  CG=triangle
THEN act=nombre
DISPLAY "App1 appartient au tp2,il a la grandeur (premier)";
RULE 25
IF etat=E1_6_6_6_M AND CG=cercle
THEN act=nombre
DISPLAY "App1 appartient au tp3,il a la grandeur (premier)";
RULE 26
IF etat=E1_6_6_6_M AND CG=demicercle
THEN act=nombre
DISPLAY "App1 appartient au tp4,il a la grandeur (premier)"
FIND act;
RULE 27
IF etat=E1_6_6_6_M AND code=carre
THEN act=nombre
DISPLAY "App2 appartient au tp1,il a la grandeur (deuxieme)";
RULE 28
IF etat=E1_6_6_6_M AND code=triangle
THEN act=nombre
DISPLAY "App2 appartient au tp2,il a la grandeur (deuxieme)";
RULE 29
IF etat=E1_6_6_6_M AND code=cercle
THEN act=nombre
DISPLAY "App2 appartient au tp3,il a la grandeur (deuxieme)";
RULE 30
IF etat=E1_6_6_6_M AND code=demicercle
THEN act=nombre
DISPLAY "App2 appartient au tp4,il a la grandeur (deuxieme)";
RULE 31
IF etat=E1_6_6_6_M AND marque=carre
THEN act=nombre
DISPLAY "App3 appartient au tp1,il a la grandeur (troisieme)";
```

```
RULE 32
IF etat=E1_G_G_G_M AND marque=triangle
THEN act=nombre
DISPLAY "App3 appartient au tp2,il a la grandeur (troisieme)";
RULE 33
IF etat=E1_G_G_G_M AND marque=cercle
THEN act=nombre
DISPLAY "App3 appartient au tp3,il a la grandeur (troisieme)";
RULE 34
IF etat=E1_G_G_G_M AND marque=demicercle
THEN act=nombre
DISPLAY "App3 appartient au tp4,il a la grandeur (troisieme)";
RULE 35
IF etat=E1_G_G_G_M AND signe=carre OR signe=croix
THEN act=nombre
DISPLAY "App4 appartient au tp1,il a la grandeur (quatrieme)";
RULE 36
IF etat=E1_G_G_G_M AND signe=triangle
THEN act=nombre
DISPLAY "App4 appartient au tp2,il a la grandeur (quatrieme)";
RULE 37
IF etat=E1_G_G_G_M AND signe=cercle
THEN act=nombre
DISPLAY "App4 appartient au tp3,il a la grandeur (quatrieme)";
RULE 38
IF etat=E1_G_G_G_M AND signe=demicercle
THEN act=nombre
DISPLAY "App4 appartient au tp4,il a la grandeur (quatrieme)";
RULE 39
IF etat=E2_G_G_G_P
THEN act=CG
FIND premier;
RULE 40
IF etat=E2_G_G_G_P
THEN premier=G
DISPLAY "Premier appareil a la grandeur (premier)"
FIND deuxieme;
RULE 41
IF etat=E2_G_G_G_P
THEN deuxieme=G
DISPLAY "Deuxieme appareil a la grandeur (deuxieme)"
FIND troisieme;
RULE 42
IF etat=E2_G_G_G_P
THEN troisieme=G
DISPLAY "Troisieme appareil a la grandeur (troisieme)"
FIND quatrieme;
RULE 43
IF etat=E2_G_G_G_P
THEN quatrieme=P
DISPLAY "Quatrieme appareil a la grandeur (quatrieme)"
FIND act;
RULE 44
IF etat=E2_G_G_G_P
THEN act=grandeur
DISPLAY "Appareil app1=(premier),app2=(deuxieme),app3=(troisieme),app4=(quatrieme)"
FIND act;
RULE 45
```

```
IF act=grandeur
THEN act=appartient;
RULE 46
IF etat=E2_6_6_6_P AND CG=carre
THEN act=nombre
DISPLAY "App1 appartient au tp1,il a la grandeur (premier)";
RULE 47
IF etat=E2_6_6_6_P AND CG=triangle
THEN act=nombre
DISPLAY "App1 appartient au tp2,il a la grandeur (premier)";
RULE 48
IF etat=E2_6_6_6_P AND CG=cercle
THEN act=nombre
DISPLAY "App1 appartient au tp3,il a la grandeur (premier)";
RULE 49
IF etat=E2_6_6_6_P AND CG=demicercle
THEN act=nombre
DISPLAY "App1 appartient au tp4,il a la grandeur (premier)";
RULE 50
IF etat=E2_6_6_6_P AND code=carre
THEN act=nombre
DISPLAY "App2 appartient au tp1,il a la grandeur (deuxieme)";
RULE 51
IF etat=E2_6_6_6_P AND code=triangle
THEN act=nombre
DISPLAY "App2 appartient au tp2,il a la grandeur (deuxieme)";
RULE 52
IF etat=E2_6_6_6_P AND code=cercle
THEN act=nombre
DISPLAY "App2 appartient au tp3,il a la grandeur (deuxieme)";
RULE 53
IF etat=E2_6_6_6_P AND code=demicercle
THEN act=nombre
DISPLAY "App2 appartient au tp4,il a la grandeur (deuxieme)";
RULE 54
IF etat=E2_6_6_6_P AND marque=carre
THEN act=nombre
DISPLAY "App3 appartient au tp1,il a la grandeur (troisieme)";
RULE 55
IF etat=E2_6_6_6_P AND marque=triangle
THEN act=nombre
DISPLAY "App3 appartient au tp2,il a la grandeur (troisieme)";
RULE 56
IF etat=E2_6_6_6_P AND marque=cercle
THEN act=nombre
DISPLAY "App3 appartient au tp3,il a la grandeur (troisieme)";
RULE 57
IF etat=E2_6_6_6_P AND marque=demicercle
THEN act=nombre
DISPLAY "App3 appartient au tp4,il a la grandeur (troisieme)";
RULE 58
IF etat=E2_6_6_6_P AND signe=carre
THEN act=nombre
DISPLAY "App4 appartient au tp1,il a la grandeur (quatrieme)";
RULE 59
IF etat=E2_6_6_6_P AND signe=triangle
THEN act=nombre
DISPLAY "App4 appartient au tp2,il a la grandeur (quatrieme)";
```

```
RULE 60
IF etat=E2_6_6_6_P AND signe=cercle
THEN act=nombre
DISPLAY "App4 appartient au tp3,il a la grandeur (quatrieme)";
RULE 61
IF etat=E2_6_6_6_P AND signe=demicercle
THEN act=nombre
DISPLAY "App4 appartient au tp4,il a la grandeur (quatrieme)";
RULE 62
IF etat=E1_6_6_6_M AND CG=carre
THEN act=nbr nbr_grand=1
DISPLAY "tp=tp1,nbr_grand=(nbr_grand)";
RULE 63
IF etat=E1_6_6_6_M AND marque=carre
THEN act=nbr nbr_grand=1
DISPLAY "tp=tp1,nbr_grand=(nbr_grand)";
RULE 64
IF etat=E1_6_6_6_M AND CG=carre AND code=carre
THEN act=nbr nbr_grand=2
DISPLAY "tp=tp1,nbr_grand=(nbr_grand)";
RULE 65
IF etat=E1_6_6_6_M AND CG=carre AND marque=carre
THEN act=nbr nbr_grand=2
DISPLAY "tp=tp1,nbr_grand=(nbr_grand)";
RULE 66
IF etat=E1_6_6_6_M AND code=carre AND marque=carre
THEN act=nbr nbr_grand=2
DISPLAY "tp=tp1,nbr_grand=(nbr_grand)";
RULE 67
IF etat=E1_6_6_6_M AND CG=carre AND code=carre AND marque=carre
THEN act=nbr nbr_grand=3
DISPLAY "tp=tp1,nbr_grand=(nbr_grand)";
RULE 68
IF etat=E1_6_6_6_M AND signe=carre OR signe=croix
THEN act=nbr nbr_moyenne=1
DISPLAY "tp=tp1,nbr_moyenne=(nbr_moyenne)";
RULE 69
IF etat=E1_6_6_6_M AND CG=triangle
THEN act=nbr nbr_grand=1
DISPLAY "tp=tp2,nbr_grand=(nbr_grand)";
RULE 70
IF etat=E1_6_6_6_M AND marque=triangle
THEN act=nbr nbr_grand=1
DISPLAY "tp=tp2,nbr_grand=(nbr_grand)";
RULE 71
IF etat=E1_6_6_6_M AND CG=triangle AND code=triangle
THEN act=nbr nbr_grand=2
DISPLAY "tp=tp2,nbr_grand=(nbr_grand)";
RULE 72
IF etat=E1_6_6_6_M AND CG=triangle AND marque=triangle
THEN act=nbr nbr_grand=2
DISPLAY "tp=tp2,nbr_grand=(nbr_grand)";
RULE 73
IF etat=E1_6_6_6_M AND code=triangle AND marque=triangle
THEN act=nbr nbr_grand=2
DISPLAY "tp=tp2,nbr_grand=(nbr_grand)";
RULE 74
IF etat=E1_6_6_6_M AND CG=triangle AND code=triangle AND marque=triangle
```

```
THEN act=nbr nbr_grand=3
DISPLAY "tp=tp2,nbr_grand=(nbr_grand)";
RULE 75
IF etat=E1_6_6_6_M AND signe=triangle
THEN act=nbr nbr_moyenne=1
DISPLAY "tp=tp2,nbr_moyenne=(nbr_moyenne)";
RULE 76
IF etat=E1_6_6_6_M AND CG=cercle
THEN act=nbr nbr_grand=1
DISPLAY "tp=tp3,nbr_grand=(nbr_grand)";
RULE 77
IF etat=E1_6_6_6_M AND marque=cercle
THEN act=nbr nbr_grand=1
DISPLAY "tp=tp3,nbr_grand=(nbr_grand)";
RULE 78
IF etat=E1_6_6_6_M AND CG=cercle AND code=cercle
THEN act=nbr nbr_grand=2
DISPLAY "tp=tp3,nbr_grand=(nbr_grand)";
RULE 79
IF etat=E1_6_6_6_M AND CG=cercle AND marque=cercle
THEN act=nbr nbr_grand=2
DISPLAY "tp=tp3,nbr_grand=(nbr_grand)";
RULE 80
IF etat=E1_6_6_6_M AND code=cercle AND marque=cercle
THEN act=nbr nbr_grand=2
DISPLAY "tp=tp3,nbr_grand=(nbr_grand)";
RULE 81
IF etat=E1_6_6_6_M AND CG=cercle AND code=cercle AND marque=cercle
THEN act=nbr nbr_grand=3
DISPLAY "tp=tp3,nbr_grand=(nbr_grand)";
RULE 82
IF etat=E1_6_6_6_M AND signe=cercle
THEN act=nbr nbr_moyenne=1
DISPLAY "tp=tp3,nbr_moyenne=(nbr_moyenne)";
RULE 83
IF etat=E1_6_6_6_M AND CG=demicercle
THEN act=nbr nbr_grand=1
DISPLAY "tp=tp4,nbr_grand=(nbr_grand)";
RULE 84
IF etat=E1_6_6_6_M AND CG=demicercle AND code=demicercle
THEN act=nbr nbr_grand=2
DISPLAY "tp=tp4,nbr_grand=(nbr_grand)";
RULE 85
IF etat=E1_6_6_6_M AND CG=demicercle AND code=demicercle AND marque=demicercle
THEN act=nbr nbr_grand=3
DISPLAY "tp=tp4,nbr_grand=(nbr_grand)";
RULE 86
IF etat=E1_6_6_6_M AND signe=demicercle
THEN act=nbr nbr_moyenne=1
DISPLAY "tp=tp4,nbr_moyenne=(nbr_moyenne)";
RULE 87
IF etat=E2_6_6_6_P AND CG=carre
THEN act=nbr nbr_grand=1
DISPLAY "tp=(tp),nbr_grand=(nbr_grand)";
RULE 88
IF etat=E2_6_6_6_P AND CG=carre AND code=carre
THEN act=nbr nbr_grand=2
DISPLAY "tp=(tp),nbr_grand=(nbr_grand)";
```

```
RULE 89
IF etat=E2_G_G_G_P AND CG=carre AND code=carre AND marque=carre
THEN act=nbr_grand nbr_grand=3
DISPLAY "tp=(tp),nbr_grand=(nbr_grand)";
RULE 90
IF etat=E2_G_G_G_P AND signe=carre
THEN act=nbr nbr_petit=1
DISPLAY "tp=(tp),nbr_petit=(nbr_petit)";
RULE 91
IF etat=E3_G_G_M_P AND CG=carre
THEN act=nbr nbr_grand=1
DISPLAY "tp=(tp),nbr_grand=(nbr_grand)";
RULE 92
IF etat=E3_G_G_M_P AND CG=carre AND code=carre
THEN act=nbr nbr_grand=2
DISPLAY "tp=(tp),nbr_grand=(nbr_grand)";
RULE 93
IF etat=E3_G_G_M_P AND marque=carre
THEN act=nbr nbr_moyenne=1
DISPLAY "tp=(tp),nbr_moyenne=(nbr_moyenne)";
RULE 94
IF etat=E3_G_G_M_P AND signe=carre
THEN act=nbr nbr_petit=1
DISPLAY "tp=(tp),nbr_petit=(nbr_petit)";
RULE 95
IF etat=E4_G_G_M_M AND CG=carre
THEN act=nbr nbr_grand=1
DISPLAY "tp=(tp),nbr_grand=(nbr_grand)";
RULE 96
IF etat=E4_G_G_M_M AND CG=carre AND code=carre
THEN act=nbr nbr_grand=2
DISPLAY "tp=(tp),nbr_grand=(nbr_grand)";
RULE 97
IF etat=E4_G_G_M_M AND marque=carre
THEN act=nbr nbr_moyenne=1
DISPLAY "tp=(tp),nbr_moyenne=(nbr_moyenne)";
RULE 98
IF etat=E4_G_G_M_M AND marque=carre AND signe=carre
THEN act=nbr nbr_moyenne=2
DISPLAY "tp=(tp),nbr_moyenne=(nbr_moyenne)";
RULE 99
IF etat=E5_G_G_P_P AND CG=carre
THEN act=nbr nbr_grand=1
DISPLAY "tp=(tp),nbr_grand=(nbr_grand)";
RULE 100
IF etat=E5_G_G_P_P AND CG=carre AND code=carre
THEN act=nbr nbr_grand=2
DISPLAY "tp=(tp),nbr_grand=(nbr_grand)";
RULE 101
IF etat=E5_G_G_P_P AND marque=carre
THEN act=nbr nbr_petit=1
DISPLAY "tp=(tp),nbr_petit=(nbr_petit)";
RULE 102
IF etat=E5_G_G_P_P AND marque=carre AND signe=carre
THEN act=nbr nbr_petit=2
DISPLAY "tp=(tp),nbr_petit=(nbr_petit)";
RULE 103
IF etat=E6_G_M_M_M AND CG=carre
```

```
THEN act=nbr nbr_grand=1
DISPLAY "tp=(tp),nbr_grand=(nbr_grand)";
RULE 104
IF etat=E6_G_M_M_M AND code=carre
THEN act=nbr nbr_moyenne=1
DISPLAY "tp=(tp),nbr_moyenne=(nbr_moyenne)";
RULE 105
IF etat=E6_G_M_M_M AND code=carre AND marque=carre
THEN act=nbr nbr_moyenne=2
DISPLAY "tp=(tp),nbr_moyenne=(nbr_moyenne)";
RULE 106
IF etat=E6_G_M_M_M AND code=carre AND marque=carre AND signe=carre
THEN act=nbr nbr_moyenne=3
DISPLAY "tp=(tp),nbr_moyenne=(nbr_moyenne)";
RULE 107
IF etat=E7_G_M_M_P AND CG=carre
THEN act=nbr nbr_grand=1
DISPLAY "tp=(tp),nbr_grand=(nbr_grand)";
RULE 108
IF etat=E7_G_M_M_P AND code=carre
THEN act=nbr nbr_moyenne=1
DISPLAY "tp=(tp),nbr_moyenne=(nbr_moyenne)";
RULE 109
IF etat=E7_G_M_M_P AND code=carre AND marque=carre
THEN act=nbr nbr_moyenne=2
DISPLAY "tp=(tp),nbr_moyenne=(nbr_moyenne)";
RULE 110
IF etat=E7_G_M_M_P AND signe=carre
THEN act=nbr nbr_petit=1
DISPLAY "tp=(tp),nbr_petit=(nbr_petit)";
RULE 111
IF etat=E8_G_M_P_P AND CG=carre
THEN act=nbr nbr_grand=1
DISPLAY "tp=(tp),nbr_grand=(nbr_grand)";
RULE 112
IF etat=E8_G_M_P_P AND code=carre
THEN act=nbr nbr_moyenne=1
DISPLAY "tp=(tp),nbr_moyenne=(nbr_moyenne)";
RULE 113
IF etat=E8_G_M_P_P AND marque=carre
THEN act=nbr nbr_petit=1
DISPLAY "tp=(tp),nbr_petit=(nbr_petit)";
RULE 114
IF etat=E8_G_M_P_P AND marque=carre AND signe=carre
THEN act=nbr nbr_petit=2
DISPLAY "tp=(tp),nbr_petit=(nbr_petit)";
RULE 115
IF etat=E9_M_M_M_M AND CG=carre
THEN act=nbr nbr_moyenne=1
DISPLAY "tp=(tp),nbr_moyenne=(nbr_moyenne)";
RULE 116
IF nbr_grand=2 AND tp=1
THEN action=TEST TEST=plateau_complet
DISPLAY "TEST-Plateau complet(tp1)";
RULE 117
IF nbr_grand=2 AND tp=2
THEN action=TEST TEST=plateau_complet
DISPLAY "TEST-Plateau complet(tp2)";
```

```
RULE 118
IF nbr_grand=2 AND tp=3
THEN action=TEST TEST=plateau_complet
DISPLAY "TEST-Plateau complet(tp3)";
RULE 119
IF nbr_grand=2 AND tp=4
THEN action=TEST TEST=plateau_complet
DISPLAY "TEST-Plateau complet(tp4)";
RULE 120
IF nbr_grand=1 AND nbr_moyenne=2
THEN action=TEST TEST=plateau_complet
DISPLAY "TEST-Plateau complet";
RULE 121
IF nbr_moyenne=4
THEN action=TEST TEST=plateau_complet
DISPLAY "TEST-Plateau complet";
RULE 122
IF nbr_grand=1 AND nbr_moyenne=1 AND nbr_petit=2
THEN action=TEST TEST=plateau_complet
DISPLAY "TEST-Plateau complet";
RULE 123
IF nbr_moyenne=2 AND nbr_petit=2
THEN action=TEST TEST=plateau_complet
DISPLAY "TEST-Plateau complet";
RULE 124
IF action=TEST AND TEST=plateau_complet
THEN action=DEPLACE_ROBOT_TABLE
DISPLAY "Action executer par le robot (action)";
RULE 125
IF action=DEPLACE_ROBOT_TABLE
THEN action=DEPOSE_APP_TABLE
DISPLAY "Les actions executer par le robot:(action)";
RULE 126
IF action=DEPOSE_APP_TABLE
THEN action=RETOURNE_ROBOT_ARM
DISPLAY "Les actions executer par le robot:(action)";
RULE 127
IF action=RETOURNE_ROBOT_ARM
THEN action=FIN
DISPLAY "Recommancer le cycle par IDENTIFICATION DES APPAREILS (etat)";
RULE 128
IF etat=E1_6_6_6_M AND CG=carre AND signe=croix AND code=triangle OR
code=cercle OR code=demicercle AND marque=triangle OR marque=cercle OR marque=demicercle
THEN action=TEST_PLATEAU action=DEPOSE_APP_PLATEAU
DISPLAY "PLATEAU INCOMPLET action executer par le robot (action)
Continuer avec la range suivante (une autre etat) pour completer le plateau.
Recommancer par GO!";
RULE 129
IF nbr_grand>2 AND tp=1 OR tp=2 OR tp=4
THEN act=nombre
DISPLAY "IMPOSSIBLE !! ERREUR ! dans la premier la deuxieme ou la quatrieme
manipulation il y a seulement deux appareils grands.RECOMMANCER par GO!";
RULE 130
IF nbr_petit>0 AND tp=1 OR tp=2 OR tp=4
THEN act=nombre
DISPLAY "IMPOSSIBLE !! ERREUR ! dans la premier,la deuxieme ou la quatrieme
manipulation il n'y a pas des appareils petits.RECOMMANCER par GO!";
RULE 131
```

```
IF nbr_moyenne>1 AND tp=1 AND CG=croix OR code=croix OR marque=croix OR signe=croix
THEN act=nombre
DISPLAY "IMPOSSIBLE !! ERREUR ! dans la premier manipulation il y seulement
un appareil moyenne mise a part les appareils communs (trois) a toutes les
manipulations.RECOMMANCER par GO!";

ASK tp:"QUEL TP?";
ASK etat:"Quelle etat est vue par la camera?";
ASK CG:"Quelle code_geometrique est vue par la camera
pour le premier appareil?";
ASK code:"Quelle code_geometrique est vue par la camera
pour le deuxieme appareil?";
ASK marque:"Quelle code_geometrique est vue par la camera
pour le troisieme appareil?";
ASK signe:"Quelle code_geometrique est vue par la camera
pour le quatrieme appareil?";
CHOICES tp:1,2,3,4;
CHOICES coco:1,2,3,4;
CHOICES etat:E1_G_G_G_M,
             E2_G_G_G_P,
             E3_G_G_M_P,
             E4_G_G_M_M,
             E5_G_G_P_P,
             E6_G_M_M_M,
             E7_G_M_M_P,
             E8_G_M_P_P,
             E9_M_M_M_M,
             E10_M_M_M_P,
             E11_M_M_P_P;
CHOICES CG:carre,croix,triangle,cercle,demicercle;
CHOICES code:carre,croix,triangle,cercle,demicercle;
CHOICES marque:carre,croix,triangle,cercle,demicercle;
CHOICES signe:carre,croix,triangle,cercle,demicercle;
PLURAL:var,act,act1,act2,act3,act4,action,app,grandeur;
PLURAL:stop,actt,nbr;
```