50376
1987
109

année 1987

THESE

présentée

A L'UNIVERSITE DES SCIENCES ET TECHNIQUES FLANDRES-ARTOIS DE LILLE

pour obtenir le

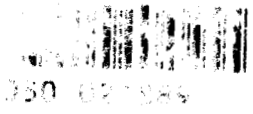
DIPLOME DE DOCTEUR EN AUTOMATIQUE

par

Eric BIENFAIT

PRATIC:

PROGRAMMATION DE ROBOT ASSISTEE PAR TRAITEMENT D'IMAGE ET CAMERA



SOMMAIRE

INTRODUCTION

CHAPITRE I : LES LANGAGES DE PROGRAMMATION ROBOTIQUE

I.1 : INTRODUCTION

I.2 : PROGRAMMATION PAR APPRENTISSAGE

I.3 : PROGRAMMATION PAR LANGAGE TEXTUEL

- I.3.1 : Classification
- I.3.2 : Le niveau actionneur
- I.3.3 : Le niveau effecteur
- I.3.4 : Le niveau objet
- I.3.5 : Le niveau objectif

I.4 : PROGRAMMATION GRAPHIQUE

- I.4.1 : Programmation géomètrique
- I.4.2 : Programmation graphique

I.5 : CONCLUSIONS

CHAPITRE II : LES OUTILS DE LA PROGRAMMATION GRAPHIQUE DES ROBOTS.

II.1 : PRESENTATION

II.2 : REPRESENTATION GRAPHIQUE

II.2.1 : Les éléments de base

II.2.2 : Structure du logiciel graphique

II.2.3 : La modélisation des objets

II.3 : LA VISION ARTIFICIELLE

II.3.1 : Extraction des contours

II.3.2 : Reconnaissance et positionnement

II.3.3 : Séparation des objets

II.4 : LA MORPHOLOGIE MATHEMATIQUE

II.4.1 : Dilatation et érosion

II.4.2 : Ouverture et fermeture

II.4.3 : Squelettisation

II.4.4 : Méthodologie employée

II.4.5 : Règle d'élaboration de l'algorithme

II.5 : CONCLUSIONS

CHAPITRE III : PRATIC: PROGRAMMATION DE ROBOT ASSISTEE PAR TRAITEMENT D'IMAGE ET CAMERA.

III.1 : INTRODUCTION

III.2 : ARCHITECTURE DU SYSTEME

III.2.1 : Architecture matérielle

III.2.2 : Architecture logicielle

III.3 : LES TROIS MODULES FONDAMENTAUX DE PRATIC

III.3.1 : Le module de définition

III.3.2 : Le module de programmation

III.3.3 : Le module d'exécution

III.4 : EXEMPLE D'APPLICATION

III.4.1 : Introduction

III.4.2 : Programmation

III.4.3 : Amélioration du dialogue Homme-Machine

III.4.4 : Exécution du programme

III.5 : CONCLUSIONS

CHAPITRE IV : METHODOLOGIE ET DEVELOPPEMENT

IV.1 : INTRODUCTION

IV.2 : LE GENERATEUR DE PLAN

IV.3 : EXECUTION TEMPS-REEL

IV.4 : CONCLUSIONS

IV.5 : DEVELOPPEMENT ENVISAGE

IV.5.1 : Modélisation

IV.5.2 : Programmation

IV.5.3 : Exécutif

CONCLUSIONS GENERALES

BIBLIOGRAPHIE

INTRODUCTION GENERALE

1. Généralités

Les dernières années ont montré que la vision artificielle rentre en force dans tous les éléments nécessaires à l'automatisation de l'usine intégrée soit pour des tâches d'inspection automatique soit pour la manipulation robotisée.

Une nouvelle génération de manipulateurs, les "robots intelligents" apparaît : ceux-ci viennent remplacer ou prolonger les manipulateurs actuels à cycle préréglé en se dotant d'organes sensoriels. Ces organes doivent permettre aux robots de modifier et d'adapter leur comportement en fonction de leur environnement.

Dans chaque cas, l'introduction d'un système de vision rend le poste plus performant et polyvalent : plus performant puisque des tâches plus complexes sont automatisées, polyvalent puisqu'une nouvelle tâche est définie par simple adaptation de logiciel, c'est-à-dire sans investissement de matériel nouveau. La plupart des tâches de manipulations automatiques requiert une reconnaissance a priori précise de l'identité et de la position des objets en présence. Or dans beaucoup de situations, le nom et la position des objets sont perdus au cours du processus de fabrication. La tâche première du système de vision est alors de réaliser une analyse de scène, c'est-à-dire identifier et localiser précisément les objets en présence.

Parallèlement au développement de la vision artificielle, la programmation des robots s'est enrichie d'une nouvelle classe de langages : les langages graphiques (cf figure 1).

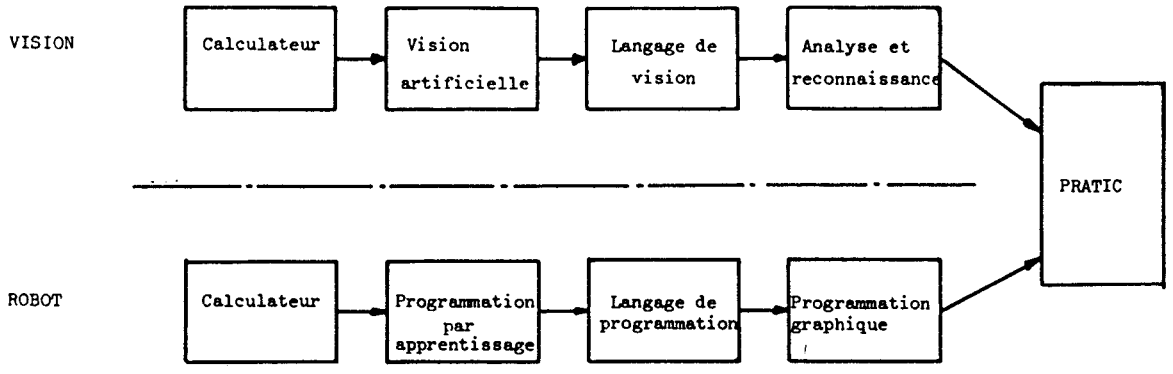


Figure 1 : Développement suivi en robotique et vision artificielle.

Ces langages ont été développés dans le but de simplifier la tâche du programmeur en lui offrant une représentation graphique de l'univers robotique. Ils sont d'excellents outils pour élaborer des programmes de commande en dehors du site d'opérations et permettent ainsi de diminuer les temps d'immobilisation. Ils se sont très vite développés pour donner naissance aux simulateurs graphiques. Ces simulateurs offrent une sécurité accrue puisque les programmes élaborés sont mis au point en n'utilisant pas le robot mais en ayant recours à la représentation de l'enchaînement des actions.

Pour atteindre ces objectifs, ces langages ont tous recours aux bases de données de la C.A.O pour disposer :

- des modélisations des objets
- d'un modèle structuré de l'univers
- des modèles des robots

ce qui augmente considérablement l'investissement nécessaire. D'autre part, ces simulateurs, pour s'approcher d'une représentation réelle permettant la détection des collisions sont obligés d'utiliser des représentations en trois dimensions. Ces représentations nécessitent encore, malgré la puissance de calcul des gros ordinateurs, plusieurs secondes pour engendrer une représentation de type fil de fer et un temps de calcul de l'ordre de la minute pour représenter un dessin avec élimination des lignes cachées. De plus les graphismes utilisés sont souvent éloignés de la réalité ce qui a pour conséquence de diminuer la créditibilité du contrôle visuel.

Ces deux défauts dégradent les performances du système et n'améliorent pas l'interactivité requise lors des manipulations.

2. Démarche

La démarche qui a conduit à l'élaboration du langage PRATIC (Programmation de Robot Assistée par Traitement d'Images et Caméra) tente de réduire au maximum les coûts d'investissement en C.A.O, calculateur et station graphique. La diminution de l'investissement graphique a été rendue possible en restreignant le champ possible de la programmation aux manipulateurs 2 D 1/2, ce qui engendre une représentation en deux dimensions et nécessite des performances de calcul moindres. Ceci n'est pas à contre courant de l'état actuel du marché. En effet, l'étude réalisée par Axe Robotique (N 16 février 1986) montre que l'implantation des robots SCARA, qui réalisent des tâches planes à grande vitesse et grande précision sera plus importante que celle des 6 axes et que les domaines d'application tels que manutention, conditionnement, emballage et parfois assemblage sont dédiés de plus en plus à des robots de type SCARA.

Dans cet objectif le système de C.A.O est remplacé par l'utilisation d'un système de vision artificielle capable d'acquérir et de traiter les informations nécessaires à la modélisation des objets. Cette modélisation sera ensuite utilisée pour décrire graphiquement l'objectif final souhaité. Là encore, une étude de l'ADI-BIPE (Industries et Techniques) montre que moins de 10 % des bureaux d'études sont équipés de système de C.A.O.

L'accent, lors du développement, a été mis sur l'élaboration d'un système modulaire permettant par la suite de séparer le logiciel de programmation et le logiciel d'exécution, ceci devant conduire d'une part à une réduction de l'investissement lors de l'acquisition de plusieurs robots et d'autre part à une diminution du coût et de la structure du système capable d'assurer l'exécutif. De plus cette modularité permet une adaptation aisée lors des changements de ressources (console de visualisation, robot). Enfin, cette philosophie permet également d'envisager une structure d'atelier robotique simplifiée par l'utilisation des réseaux (cf figure 2).

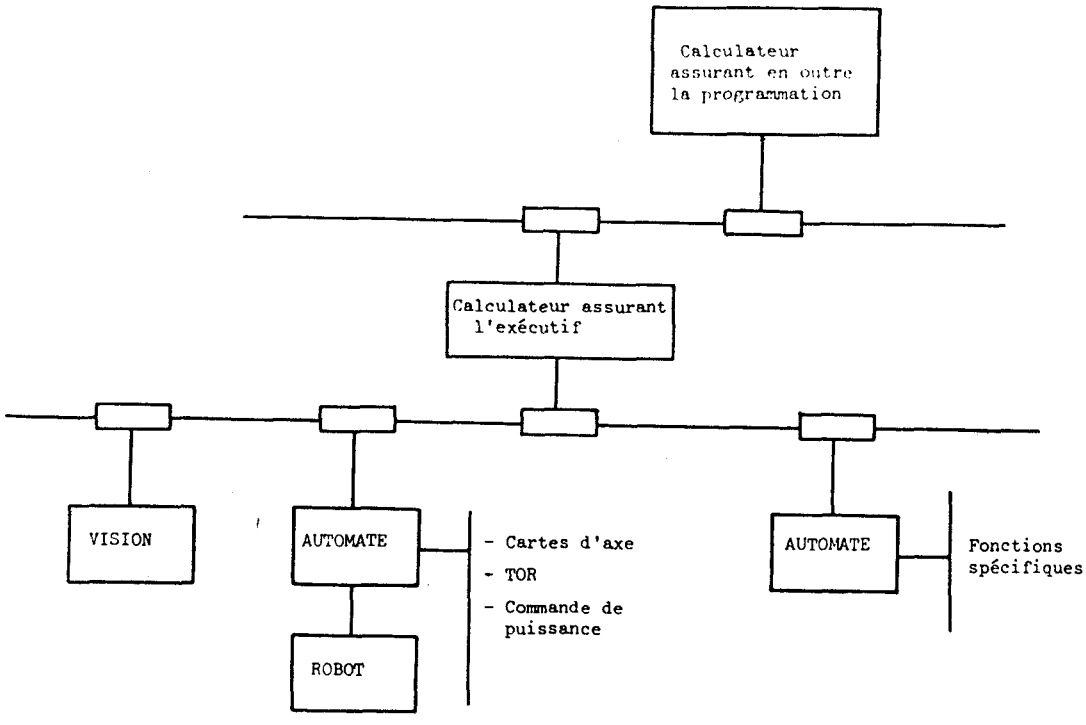


Figure I.2 · Exemple de répartition des différents moyens de calcul et de commande.

3. Plan de l'étude

Notre étude, après un bref rappel des différents langages de programmation, présenté au chapitre I, définit les bases nécessaires pour un langage de programmation graphique (chapitre II). La deuxième partie de ce chapitre expose la méthodologie employée pour utiliser en premier lieu les informations de contour des objets, puis dans une seconde phase l'accent est mis sur la résolution du vrac planaire. Le chapitre III expose le langage PRATIC et montre en détail un exemple d'application. Le chapitre IV est consacré au générateur de plans d'actions inclus dans PRATIC et au développement qu'il a déjà subi pour permettre l'exécution multi-robots-multi-préhenseurs. La deuxième partie de ce chapitre montre les développements qui peuvent encore être envisagés. Enfin et pour clore cette étude, la conclusion met en évidence les avantages et les inconvénients de PRATIC.

CHAPITRE I

LES LANGAGES DE PROGRAMMATION ROBOTIQUE

I.1 Présentation

Ce chapitre présente les différents concepts de programmation des robots [LOZ 83] et fournit une analyse critique de ceux-ci, à partir de laquelle le langage PRATIC est introduit. Comme toute classification, celle-ci peut être qualifiée d'arbitraire. Néanmoins elle fait apparaître clairement l'évolution de la commande. En effet, celle-ci est intimement liée au degré d'évolution des capteurs. Cette remarque conduit tout naturellement à présenter l'intégration d'un capteur particulièrement performant : la vision artificielle.

La classification des modes de programmation [BON 82] (cf figure I.1) repose en premier lieu sur la manière d'effectuer la programmation :

- par apprentissage

- par langage de haut niveau (programmation textuel)

- à l'aide de représentations visuelles (programmation graphique).

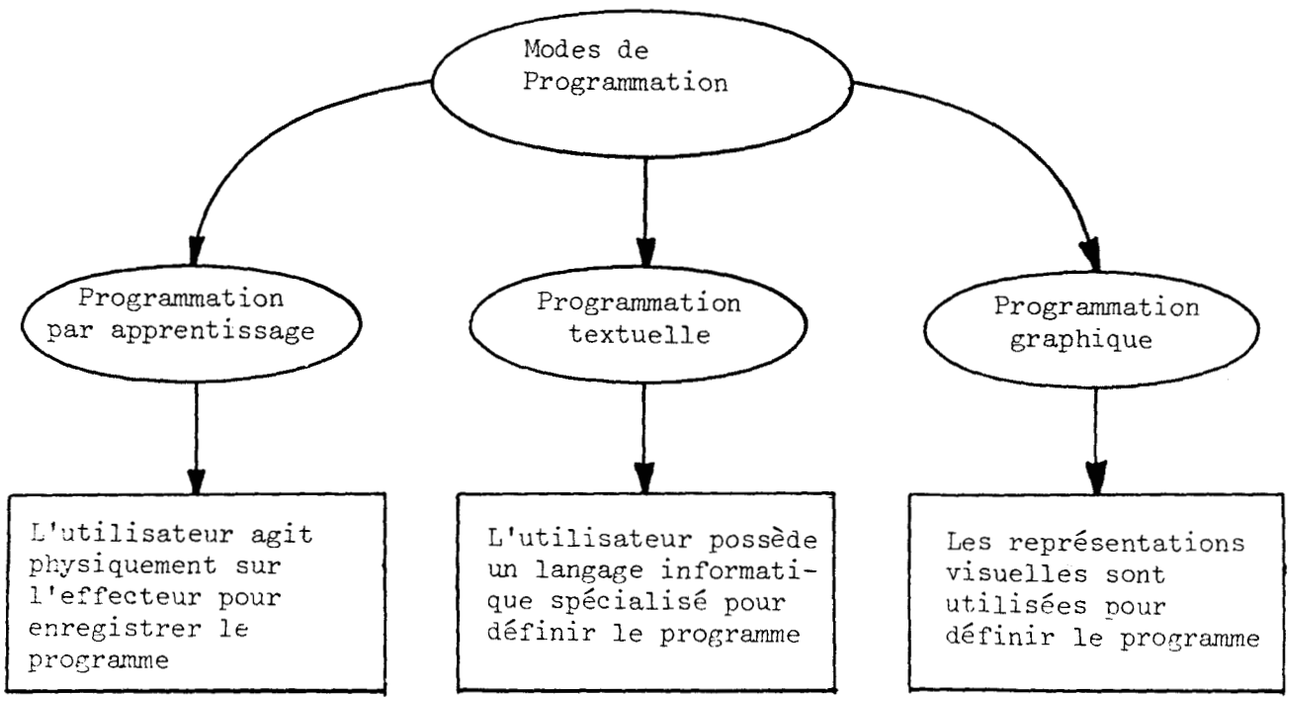


figure I.1 : Modes de programmation

La programmation par apprentissage regroupe tous les systèmes utilisant une action physique (directe ou indirecte) de l'opérateur sur l'outil terminal du robot [ASE 79], [HOL 77]. L'enregistrement des programmes correspond à la succession des déplacements validés.

Dans la classe des langages textuels, l'opérateur, c'est-à-dire le programmeur, doit concevoir un programme informatique incluant des instructions spécifiques permettant les actions et les déplacements de l'effecteur.

La dernière classe utilise les représentations visuelles, soit en deux ou trois dimensions suivant leur degré d'évolution, pour modéliser les déplacements et les actions de l'outil terminal.

Plusieurs sous-classes apparaissent suivant le mode opératoire pour les systèmes par apprentissage et suivant le niveau de développement pour les langages textuels et graphiques.

I.2 La programmation par apprentissage

Dans ce cas, la trajectoire de l'organe terminal est enregistrée par échantillonage [DAV 79]. Chaque relevé, représentant la position d'un point de la trajectoire, est inscrit en mémoire. Deux modes d'enregistrement sont possibles :

- enregistrement point-à-point

L'opérateur déplace l'outil terminal du robot par action directe sur les actionneurs. Lorsque la position désirée est atteinte le point est validé. La manutention, le soudage sont des applications privilégiées de la programmmation point-à-point. La commande s'assimile dans ce cas à un asservissement de position.

- enregistrement continu

Les instants d'enregistrement sont fixés par horloge interne. Ce mode de programmation s'effectue par l'utilisation d'un esclave représentant le robot simplifié (pantin) que l'opérateur déplace suivant les mouvements à effectuer. A l'exécution, le robot recopie l'enregistrement effectué. Cette programmmation trouve son utilisation la plus fréquente dans les opérations de mise en peinture robotisée.

La programmation par apprentissage est encore largement utilisée aujourd'hui dans l'industrie en raison de sa simplicité, de sa rapidité de mise en oeuvre et parce qu'elle ne

nécessite pas l'intervention de spécialistes. Néanmoins, elle reste cantonée à des domaines d'application bien définis tel que le soudage, la peinture ...,pour en tirer le maximun de profit car ces opérations peuvent être considérées comme simples dans la mesure où elles requièrent peu de contraintes de précision (projection de peinture, soudure par point).

Les inconvénients majeurs en sont :

- le manque d'instructions conditionnelles
- l'impossibilité d'itération
- le manque de synchronisation entre les robots
- la difficulté d'éditer un programme
- puis surtout l'immobilisation du matériel.

Ces derniéres années ont vu apparaître plusieurs langages pouvant être classés d'une certaine façon dans la programmation par apprentissage et innovant dans plusieurs directions :

- programmation graphique avec représentations visuelles du robot sur un écran permettant une programmation hors-ligne.

- utilisation du robot pour définir la position alors que la séquence d'exécution est spécifiée par programme.

I.3.1.Classification des langages textuels

La différenciation entre ces différents langages est effectuée suivant les différents niveaux de description d'une tâche. J.C. LATOMBE [LAT 79],[LAT 82] propose quatre niveaux (cf figure I.2) :

- le niveau actionneur

- le niveau effecteur

- le niveau objet

- le niveau objectif.

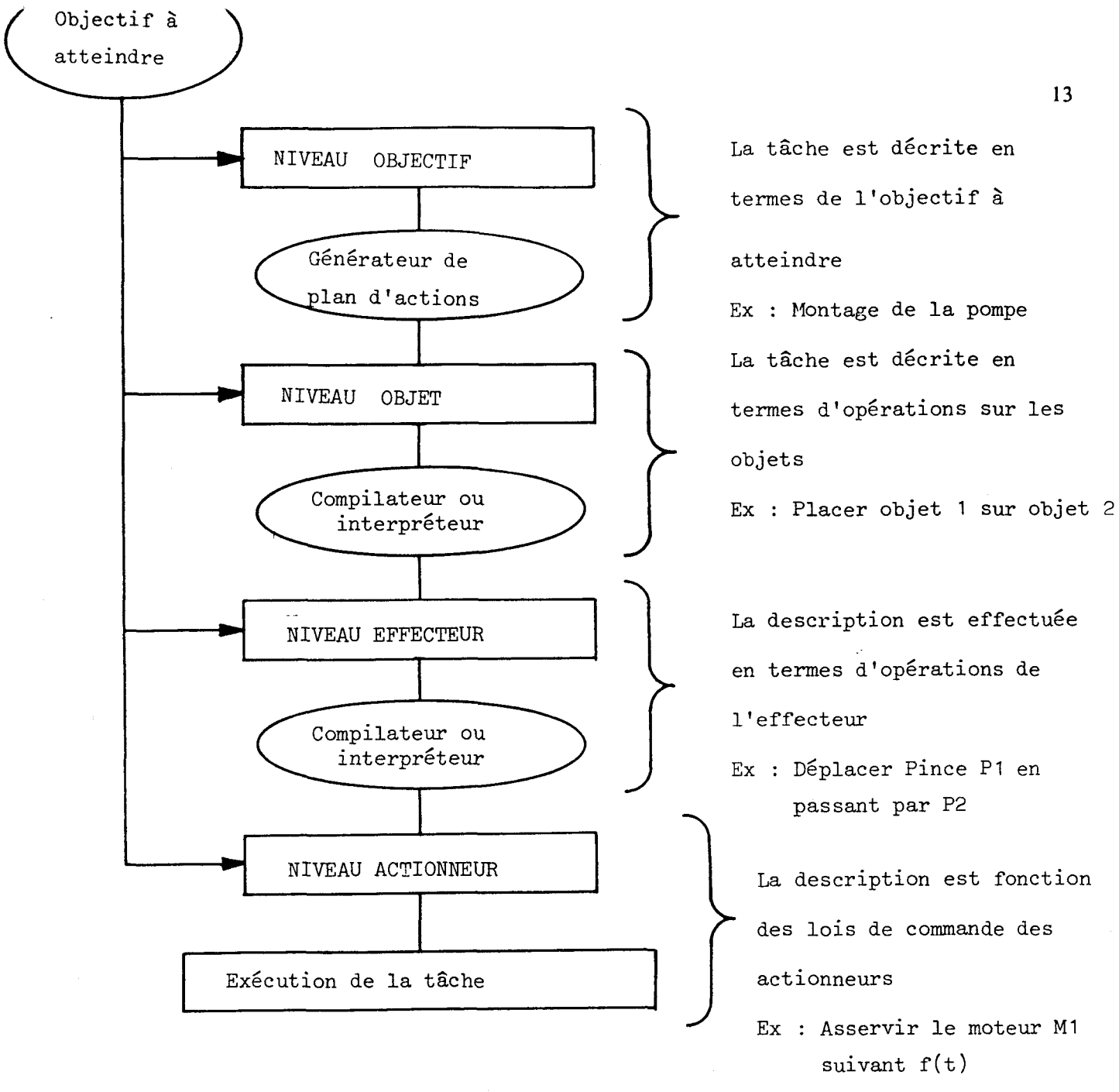


figure I.2: Classification des différents niveaux de description d'une tâche.

I.3.2. Le niveau actionneur

La programmation d'une tâche au niveau actionneur ne présente pas grand intérêt car la description de la tâche devient rapidement fastidieuse. En effet, cette programmation s'effectue en termes de loi de commande des actionneurs. Ces langages sont comparables aux langages de programmation des machines-outils à commande numérique (langage APT, PROMO), ou s'apparentent à la programmation hexadécimale sur les calculateurs.

I.3.3. Le niveau effecteur

La description d'une tâche au niveau effecteur consiste à définir la séquence des déplacements et des opérations de l'organe terminal du robot. Ces langages reposent sur une représentation de l'univers perçu à partir de repères cartésiens [MIR 82].

Le principe consiste à modéliser l'outil terminal du robot et les objets par un ensemble de repères judicieusement choisis (cf figure I.3). Le programmeur raisonne alors en termes de positions et d'orientations de ces repères. Programmer un robot au niveau effecteur c'est d'abord définir l'ensemble des repères représentatifs de la tâche à effectuer. Chaque objet est ainsi modélisé par la localisation de son repère défini dans un système de référence fixe commun au programmeur et au langage. Les déplacements de l'outil terminal du robot sont également définis par rapport à ce repère fixe.

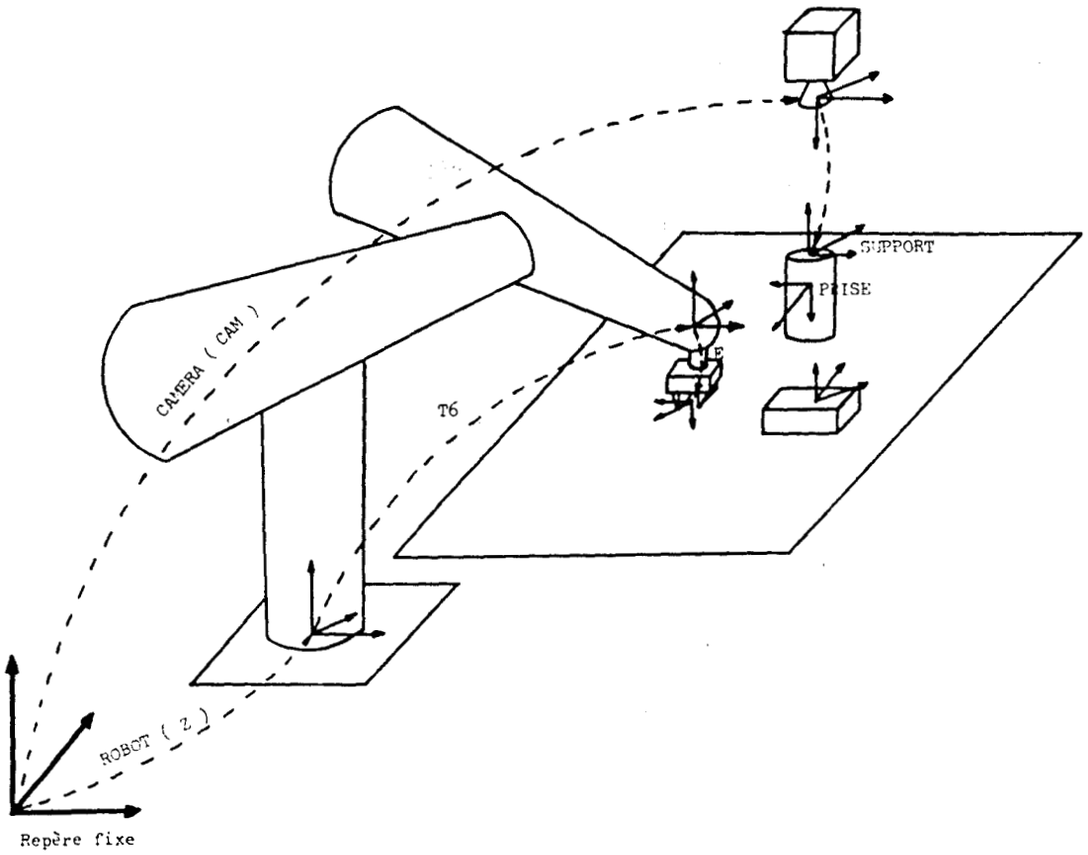


figure I.3: représentation de l'univers perçu à partir de repères cartésiens.

Le premier de ces langages fut le langage MHI (1960) [ERN 61] dont, par exemple, une primitive de déplacement était MOVE suivi de la direction et de la vitesse plus quelques instructions de test de capteurs. Mais le premier langage vraiment conçu pour des applications générales en robotique fut le langage WAVE (1970) [PAU 77]. La syntaxe en a été

inspirée du langage assembleur du calculateur PDP. Ce langage innova en permettant la description des positions par les coordonnées de l'effecteur (x, y, z et les trois angles d'EULER) et la coordination des mouvements pour réaliser des accélérations continues.

Parallèlement au développement de WAVE est apparu MINI (1972) [SIL 73] qui était une extension du langage LISP. Les fonctions de base permettaient de spécifier la position et la force de chacun des degrés de liberté. Inversement, d'autres instructions autorisaient la lecture de la position et de la force ainsi que le test de conditions pour continuer le déroulement du programme.

Les chercheurs de Stanford ont développé sur les bases de WAVE, le langage AL (1974) [FIN 74]. AL s'inspire pour sa structure de Algol et de Pascal. Il a été conçu pour recevoir des spécifications au niveau de la tâche. AL, comme WAVE et MINI, tourne sur deux machines. La première effectue la compilation de AL vers un langage de bas niveau qui est ensuite interprété par la deuxième machine en temps réel. Les possibilités d'AL sont la spécification cartésienne du mouvement, de la trajectoire et de la compliance. Les structures de contrôle et de données sont celles d'Algol (vecteur, rotation, ...). De plus, il offre la possibilité d'une modélisation de l'environnement et d'une exécution en temps réel.

Dans cette chronologie est ensuite apparu le langage VAL (1975) [UNI 79]. Ce langage est interprété. Les possibilités de bases en sont :

- pour les déplacements, la spécification des trajectoires en point-à-point, interpolées et cartésiennes

- pour les données, la manipulation des systèmes de coordonnées cartésiennes,des variables entières et arithmétiques.

La gestion des capteurs se ramène simplement à la désignation de lignes binaires et de leur test, et à la capacité de les surveiller et d'exécuter une procédure lorsqu'un évènement est détecté.

Le langage AML (1977) [TAY 82] est ensuite apparu, mais le premier langage à permettre l'exécution de tâches multiples fut le langage TEACH (1975) [RUO 79]. Il innova aussi en permettant une définition du programme indépendamment du robot. Les programmes étaient décomposés en séquences élémentaires d'états pouvant être chainées ou exécutées en parallèle.

Un autre langage mérite d'être cité : PAL (1978) [TAK 79] car sa conception est totalement différente. En effet, le programme contient une séquence d'équations résolvant la localisation des objets et de l'effecteur. Si Z est la base du robot (cf figure I.3), T6 l'extrémité du dernier bras du robot relativement à Z et E la position de l'effecteur relative à T6, les mouvements sont accomplis en spécifiant la valeur de Z+T6+E (+ indiquant la composition de transformation). L'équation suivante Z+T6+E=CAM+SUPP+PRISE spécifie que l'effecteur doit être placé en position de prise sur le support dont la position est connue relativement à la caméra.

Plusieurs autres langages sont ensuite apparus mais tous s'inspirent des principales caractéristiques qui viennent d'être citées. Ce fut le cas de :

- MCL [DON 79] extension du langage APT

- ML [WIL 75] innova en permettant la commande de deux robots en parallèle et la synchronisation des tâches

- MAPPLE [DAR 75] AL interprété

- SIGLA [SAL 78] comparable à ML

- MAL [GIN 79] exécution de tâches multiple et synchronisation par l'utilisation de sémaphores

- LAMA [FAL 80] exécution pseudo parallèle des tâches

- LM [LAT 81] comparable à AL implémenté sur mini-calculateur

- RAIL [FRA 82] s'inspire de la structure de PASCAL

I.3.4 Le niveau objet

La description de la tâche au niveau objet se traduit par l'énoncé d'un ensemble d'opérations sur les objets (poser un objet sur support en précisant l'orientation de l'un par rapport à l'autre) [PAR 72]. Le nombre d'opérations sur les objets étant réduit, le travail de préparation est alors simplifié. L'interpréteur se réfère aux informations concernant les objets et les actioneurs pour transformer le programme du niveau objet au niveau effecteur.

Dans ce type de langage on peut classer :

- HAND EYE [FEL 71] qui était capable de choisir les positions stables de préhension sur un polyédre et de planifier les mouvements pour approcher et déplacer les objets.

- LAMA [LOZ 76] : ce langage a été conçu comme un langage de niveau objet mais n'a été que partiellement implanté. LAMA inclut les relations de spécifications de la tâche, des obstacles, la préhension, une stratégie de base et de détection des erreurs.

- AUTOPASS [LIB 77] : ce langage possède la déclaration PLACER. L'effort du développement a également porté sur la réalisation d'un planificateur de chemin sans collision pour un robot cartésien.

I.3.5 Le niveau objectif

Le niveau objectif vise à substituer à la phase de description de la tâche une phase de description de l'objectif final souhaité. Cette description est suivie d'un générateur de plan d'actions capable, à partir de la description de l'objectif de générer un ensemble de tâches exécutables au niveau effecteur. Pour effectuer cette génération, le planificateur doit posséder une description des objets à manipuler de l'environnement et de l'état final désiré.

La planification du générateur peut être divisée en trois phases :

a) modélisation
b) spécification de la tâche
c) programme de synthèse

a) modélisation

La modélisation doit fournir les informations suivantes :

* description géométrique de tous les objets et du robot [BAE 79]

* description physique de tous les objets [BRA 78]

* description cinématique de toutes les liaisons [MAS 81]

* description des caractéristiques du robot [WHI 82]

* description de l'environnement de périrobotique

b) spécification de la tâche

La réalisation de l'objectif est obtenue en effectuant un séquencement d'objectifs intermédiaires. A chaque étape de ce séquencement est associé un modèle de l'état de l'environnement [TAY 76]. La spécification correspondante, qui relève d'une démarche récursive, est donnée au vu du résultat de l'analyse de l'objectif final souhaité. Cette analyse de l'objectif est réalisée soit par l'opérateur soit par le planificateur lui-même. Dans ce dernier cas le planificateur n'a besoin que d'une description de l'objectif final souhaité et de l'état initial. Les systèmes associés produisent alors une stratégie dont les primitives de commande sont du type prendre, déplacer, déposer.

c) programme de synthèse

C'est la phase cruciale car chaque action définie par le planificateur doit satisfaire différents critères avant de pouvoir être exécutée :

- critère de faisabilité (limitations dues au robot, obstacle de trajectoire) [BRA 83],[LOZ 79]
- critère de progression (la tâche exécutée va-t-elle vers le but final?)
- critère d'optimalisation (par exemple, ce chemin est-il le plus court?).

Un des intérêts majeurs de la programmation des tâches au niveau objectif, outre l'extrême simplicité de la programmation, est la possibilité de traiter les incidents intervenant durant la phase d'exécution et de planifier de façon quasi optimale le déroulement de la tâche. Toutefois l'implémentation de générateurs de plans d'action en robotique industrielle n'est actuellement pas ressentie comme une nécessité pour les tâches qui sont robotisées.

L'introduction des différents concepts exposés ci-dessus a permis d'accroitre le domaine d'application des robots. En particulier ces langages ont permis :

- de construire des programmes de manipulations non linéaires (présence d'aiguillage et d'interactions)

- d'utiliser des données en provenance de capteurs afin d'adapter le comportement du robot à son environnement

- de faciliter l'édition des programmes (modification, mise au point)

- d'autoriser la coordination de plusieurs robots.

Néanmoins programmer un robot à l'aide d'un langage tel que ceux décrits précédemment nécessite une bonne maîtrise de l'univers tridimensionnel. La mise au point du programme de commande reste alors longue et difficile. Elle nécessite en outre l'immobilisation du matériel et une attention soutenue de la part du programmeur pour appréhender le problème des collisions et de surcroît veiller à ce que celles-ci ne se produisent pas lors de la première exécution. C'est pour pallier ces inconvénients que sont apparus des langages à base de représentation graphique du robot. Ces langages permettent lors de la programmation de simplifier le raisonnement tridimensionnel en fournissant des entités géométriques concrètes qui ne nécessitent pas un esprit d'abstraction poussé.

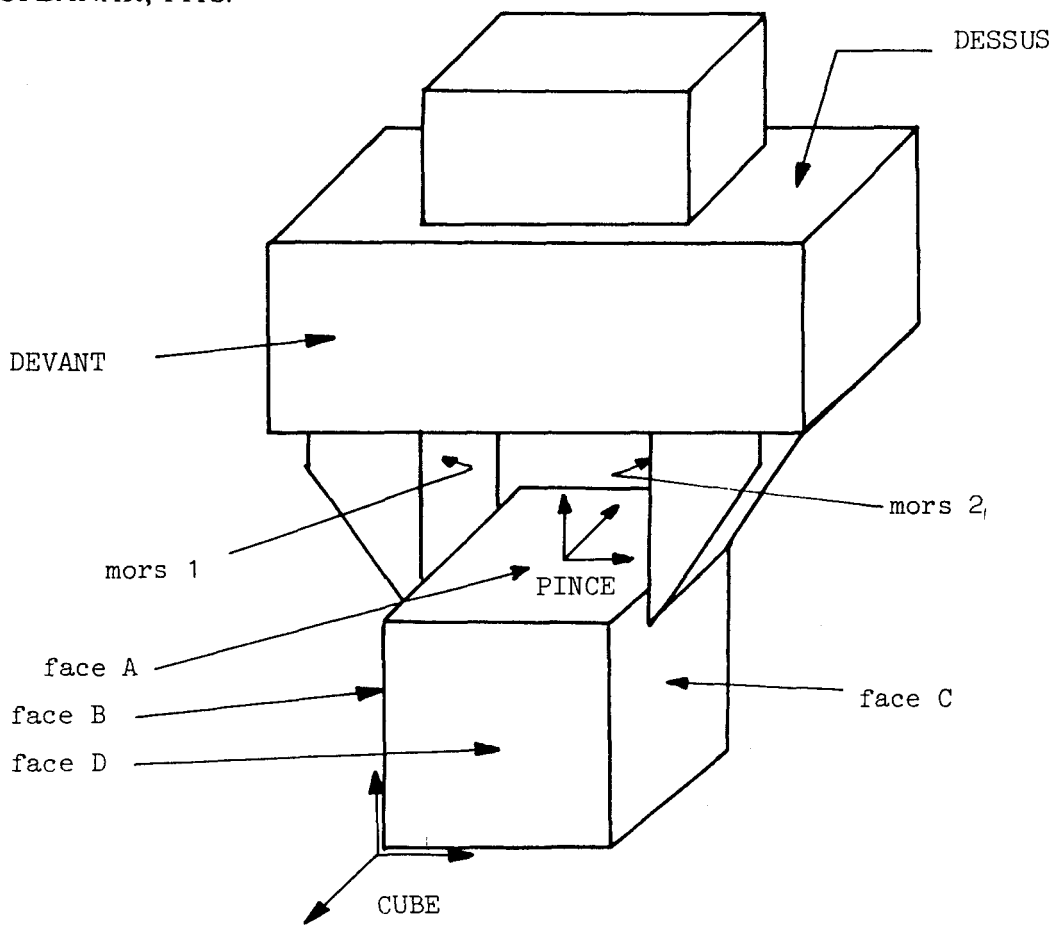
I.4. Progammation graphique

L'intérêt de tels systèmes de programmation [LAU 83], [LAU 82] réside principalement dans le fait qu'il facilite le raisonnement dans l'espace tridimensionnel en permettant au programmeur d'exprimer les positions prises par le robot au cours de l'exécution d'une tâche en termes de relations géométriques élémentaires. Le mode d'expression utilisé par ce type de système peut être textuel : on parle alors de programmation géométrique, ou graphique : on parle dans ce cas de programmation graphique.

I.4.1. La programmation géométrique

En programmation géométrique, le programmeur décrit les mouvements et les actions du robot à l'aide d'un ensemble de positions clés appelées "situations géométriques". Ces situations sont décrites symboliquement par des relations géométriques du type : "la face a de l'objet A est contre la face de la face b de l'objet B" (cf figure I.4). C'est le cas du langage LM.GEO [MAZ 82] qui utilise quatre relations élémentaires : CONTRE, COPLANAIRE, ALIGNE, SUR alors que le langage RAPT [POP 78] n'en considère que trois AGAINST,

COPLANAR, FITS.



programmation effective:

DESSUS DE PINCE coplanaire FACE A DE CUBE
MORS1 DE PINCE contre FACE B DE CUBE
MORS2 DE PINCE contre FACE C DE CUBE
DEVANT DE PINCE coplanaire FACE D DE CUBE

figure I.4: Description d'une situation géometrique

I.4.2. la programmation graphique

L' utilisation d'un poste travail graphique permet de simplifier la description des positions par lesquelles doit passer le robot pour exécuter une tâche. L'opérateur spécifie alors graphiquement sa destination pour chaque mouvement. Cette programmation consiste donc à amener en coincidence une entité géométrique représentive de l'outil terminal du robot et un élément de l'environnement (cf figure I.5). Le calcul du mouvement nécessaire à la satisfaction de la relation est effectué en appliquant la règle du déplacement minimum.

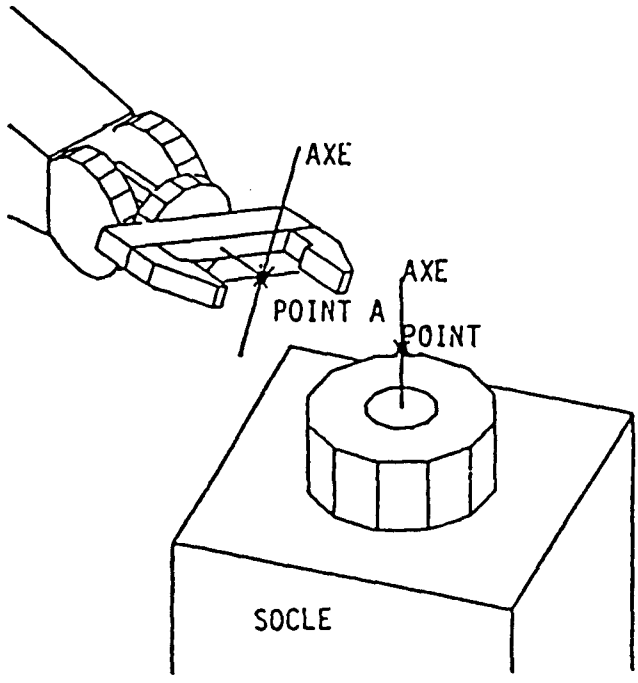
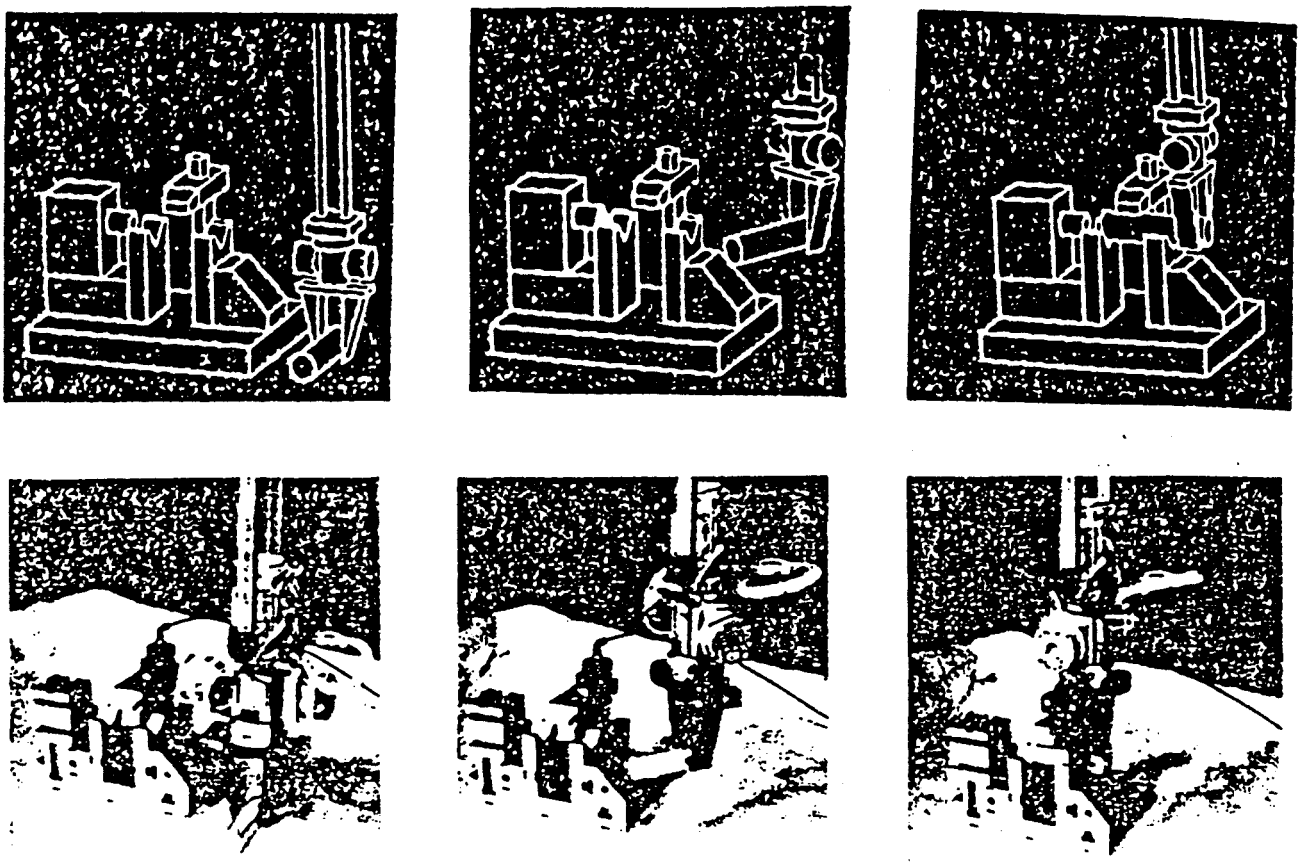


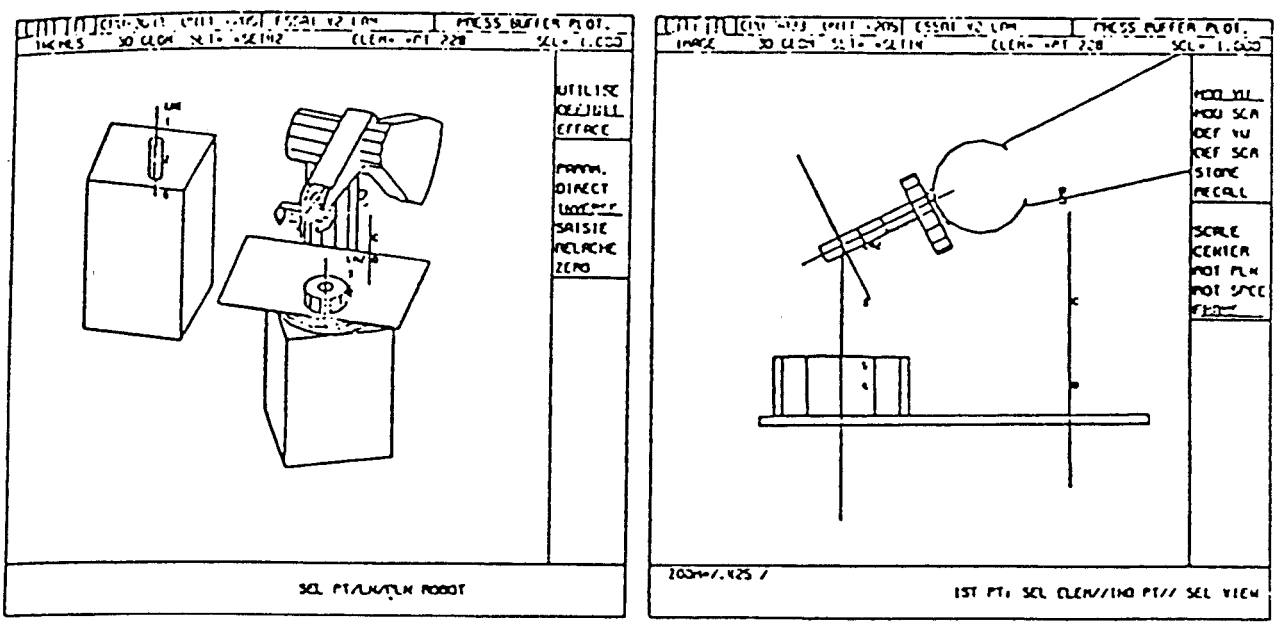
figure I.5:Programmation graphique: mise en coïncidence des représentations graphiques

La programmation graphique des robots nécessite l'utilisation de modèles géométriques qui comportent une composante graphique et une composante fonctionnelle. La composante graphique permet de visualiser le robot et son environnement. La composante fonctionnelle vient en complément pour la description de la tâche. Elle est constituée d'un ensemble d'entités géométriques élémentaires qui représentent les parties fonctionnelles des objets à manipuler (partie à saisir, partie à assembler). Les langages tels que CATIA, MODBULL fournissent cette faculté (cf figure I.6).

L'expression graphique est un excellent mode de communication homme-machine. Elle ne peut cependant se substituer totalement à l'expression textuelle qui est sémantiquement plus riche. En particulier, le graphique ne permet pas de manipuler aisément les données des capteurs. Il est de plus mal adapté à la description des aiguillages, des itérations et du paralléllisme. Ces critiques sont valables si la programmation graphique se situe au niveau effecteur. Par contre si la programmation graphique se situe au niveau de la tâche, c'est le générateur de plans d'actions qui doit prendre en charge l'élaboration de la structure informatique, seuls certains recours à la gestion des capteurs ou actionneurs spécifiques restant à envisager.



- a :Programmation à l'aide du langage MODBULL



1. Etat initial

2. Le centre de la pince est amené sur l'axe du cylindre à saisir

- b : Programmation à l'aide du langage CATIA

figure I.6: Exemples de programmation graphique

I.5 Conclusions

Les langages graphiques nécessitent de gros investissements logiciels et matériels: calculateur, logiciels graphiques, station de développement. La puissance de calcul nécessaire est trés importante, car ces sytèmes doivent supporter les algorithmes des représentations en trois dimensions ainsi que ceux concernant l'amélioration des images : élimination des traits cachés en particulier [WAT 70]. La figure I.7 montre qu'une représentation de type " fil de fer " devient rapidement inexploitable pour représenter l'environnement total d'un robot [HEG 83]. La complexité de ces systèmes provient également du fait que le développement poursuivi s'est principalement intérressé à la représentation des robots 6 axes nécessitant l'utilisation des trois dimensions.

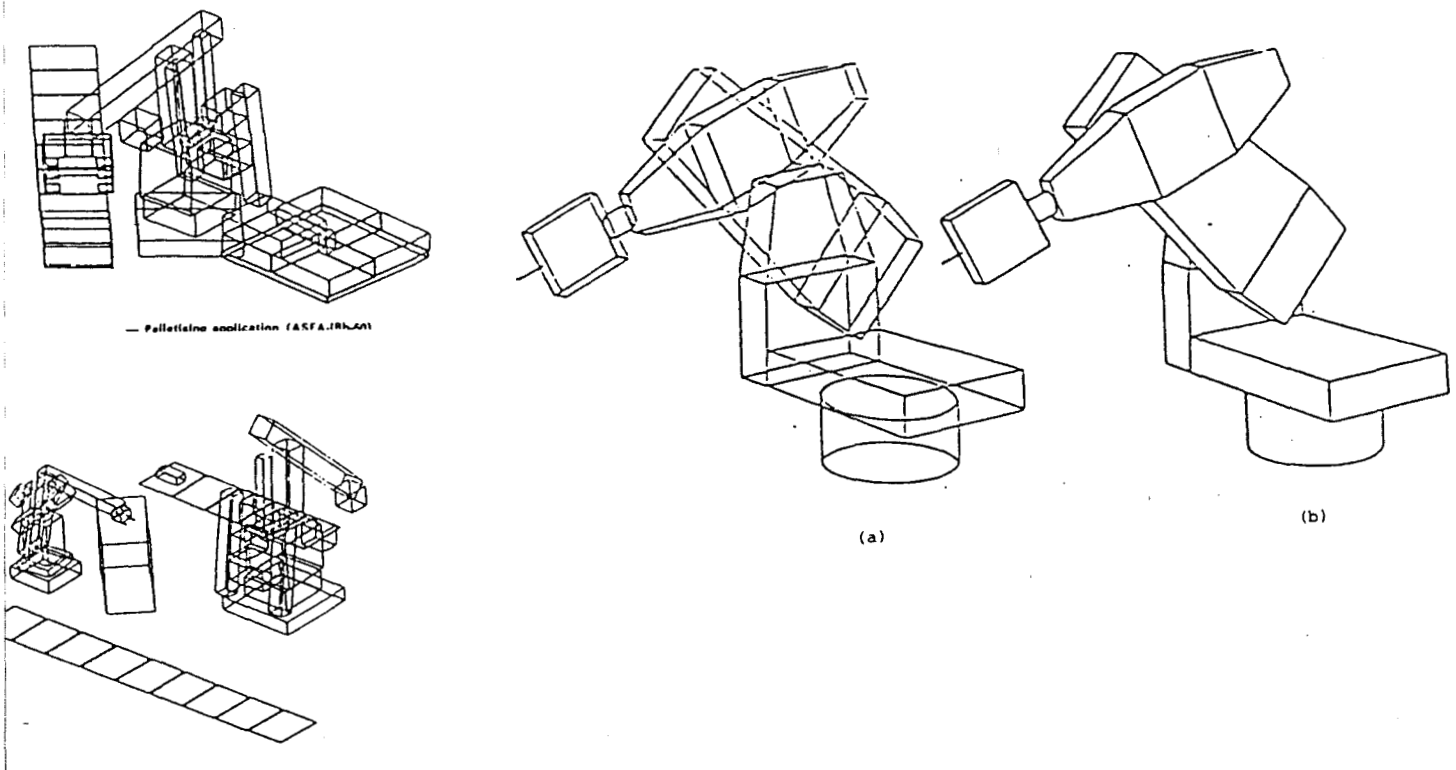


figure I.7: Représentations graphiques du type "fil de fer "

- a : sans élimination des traits cachés
- b : avec élimination

L'apparition des robots SCARA a changé le paysage du monde robotique. Plusieurs statistiques montrent leur implantation sans cesse croissante dans l'industrie pour effectuer des tâches planes.

Les langages de programmation actuels de ces robots ne tirent pas un grand profit de la simplification des tâches qui sont exécutées. C'est pour tenter de combler cette lacune, tout en profitant des avantages considérables apportés par la représentation visuelle lors de la phase de programmation, qu'a été développé le langage PRATIC présenté dans ce mémoire. Ce langage vise donc à programmer graphiquement les robots réalisant des tâches en 2D 1/2 (SCARA, CARTESIEN) sans pour cela limiter les applications aux seuls robots SCARA (cf chapître IV.5).

CHAPITRE II

LES OUTILS DE LA PROGRAMMATION GRAPHIQUE DES ROBOTS.

II.1.Présentation

Le chapitre précédent a permis de mettre en évidence les avantages et les inconvénients des différents modes de programmation actuels appliqués dans les domaines de la robotique. Cette analyse montre que l'apport des représentations visuelles est indéniable car celles-ci accélèrent considérablement le processus de perception de l'univers [BOU 82].

En permettant la saisie et l'intégration quasi immédiate d'informations globales, l'image apparaît comme un outil précieux pour l'analyse rapide d'un ensemble de données et pour la détermination des relations liant les différents composants représentés. En particulier l'image permet une description aisée des objets physiques sans faire appel à l'esprit d'abstraction de l'individu et facilite donc le raisonnement dans l'espace tridimensionnel.

Cependant, la création et l'animation d'images sont longues et difficiles. La puissance de calcul des ordinateurs limite rapidement la souplesse d'utilisation [BIE 84]. Il est donc nécessaire de s'appuyer au maximum sur les possibilités de communication interactive, le calculateur se chargeant d'afficher les images représentatives des données qu'il manipule, l'opérateur fournissant les ordres et les données à l'aide de commandes appropriées [MOR 76].

Toutefois la transformation des représentations réelles des objets en une modélisation est bien souvent une opération fastidieuse [VUL 83]. L'utilisation d'un système de vision artificielle va permettre de simplifier cette acquisition. Les principes de mise en oeuvre seront exposés dans la deuxième partie de ce chapitre, alors que la première partie s'intéresse au formalisme du logiciel utilisé pour représenter les objets traités par la vision artificielle et à leur modélisation.

II.2.Représentation graphique

2.1.LES ELEMENTS DE BASE

L'interaction souhaitée se caractérise par le fait que les images présentées sont des images "vivantes", c'est-à-dire susceptibles de modification. L'image présentée sur l'écran doit être le reflet de la situation du programme robotique à un instant donné, situation pouvant être modifiée au gré de l'opérateur. Si la situation

change, il est évident que l'image qui la compose doit également changer, de manière à ce que l'opérateur puisse suivre son évolution. Ceci a plusieurs conséquences :

- l'image produite est destinée à être étudiée en général pendant un temps très bref. Elle ne doit donc pas être trop complexe.

- l'image doit être produite très rapidement afin d'éviter une atttente trop longue à l'opérateur. Cependant, bien que dépendant beaucoup de l'application traitée, le temps de réponse doit être réduit à son minimum afin de pouvoir envisager des applications temps réels. Il est clair aussi que le mode interactif envisagé impose également un temps de calcul très bref.

- l'image présentée sur l'écran n'est pas en général un produit fini. Ce n'est qu'une ébauche, souvent grossière, permettant de vérifier au plus vite un certain nombre de caractéristiques (type de l'objet, distance relative, distance absolue ...).

Il faudra donc disposer de moyens permettant de préciser les modifications de situations souhaitées, l'opérateur devant entamer un véritable dialogue avec le calculateur. En effet, tout le travail de conception de la scène fera appel aux échanges calculateur/opérateur. Dans ce sens, les commandes opérateur sont assorties de valeurs, implicites ou non, définies par paramétrage. L'écho de l'action et de l'entrée des paramètres doit apparaître sur la console de visualisation pour permettre une auto-vérification aisée par l'opérateur [VAS 82].

De plus, des fonctions purement géométriques seront mises à la disposition de l'opérateur pour lui permettre soit de créer les composants de l'image soit d'élaborer l'image elle-même. Quatre fonctions peuvent être considérées :

- le menu qui permet à l'opérateur de choisir une action à effectuer

- l'introduction et la représentation de valeurs alphanumériques, permettant d'affecter une valeur à un paramètre.

- l'identification d'une partie de l'image permettant d'indiquer la zone où sera effectué le traitement souhaité (zoom).

- mémorisation d'un élément de l'image par validation.

Ces quatre éléments du dialogue pourront être combinés entre eux pour donner des actions plus complexes.

Un autre mécanisme essentiel pour la communication graphique envisagée est celui de la structuration de la représentation [GIL 78]. En effet, l'image doit être décomposée en éléments manipulables séparément c'est-à-dire qu'il doit être possible d'afficher, d'effacer et de déplacer les éléments à tout moment. Il est à remarquer que l'effacement sélectif fait appel à une structure de données identiques à la liste de visualisation.

2.2.STRUCTURE DU LOGICIEL GRAPHIQUE

Le logiciel graphique est constitué d'un ensemble de modules destinés à être insérés dans le programme principal.

La description finale de la scène est composée d'éléments de très bas niveaux (liste de points et de segments de droite). Le programme principal permettant la programmation proprement dite n'a pas vocation à traiter de tels éléments car les données qu'il manipule sont adaptées aux types de calcul qu'il effectue. Il faudra donc développer la structure logicielle nécessaire pour assurer la transcription des données en éléments graphiques. Cette structure peut être décomposée en :

- un logiciel de description [FRE 74] qui permet, à partir des informations fournies par le programme d'application, de coder les éléments pemettant d'obtenir une représentation graphique. Le résultat est alors la scène.

- un logiciel de préparation à la visualisation [NEW 79] qui, à partir de la scène, construit un fichier de paramètres caractéristiques d'une scène bidimensionnelle constituée de points et de segments de droite. A ce stade , le fichier résultant est indépendant des logiciels graphiques utilisés pour produire le dessin.

- un logiciel élémentaire [KRZ 84] qui permet d'obtenir l'image finale à partir des indications fournies par le logiciel de préparation à la visualisation.

Chacun de ces logiciels peut être à son tour divisé en deux parties :

- bibliothèque d'algorithmes
- structure de données

Il est important pour permettre le développement futur des logiciels, de

garder une bonne modularité des différents niveaux de logiciels. En effet il serait tentant de regrouper toutes les données pour éviter parfois certaines redondances ou de mettre en commun des algorithmes. Cela aurait quand même l'avantage de produire un logiciel autonome comportant toutes les simplifications possibles. Néanmoins, il est préférable de développer séparément ces différentes couchès de logiciel car cela permet de ne pas réécrire à chaque développement les programmes élémentaires.

La programmation graphique utilise donc ces logiciels élémentaires pour construire et déterminer la scène représentative de l'objectif final souhaité.

Pour cela, l'opérateur manipule trois types d'entités :

- les objets élémentaires : ce sont les plus petites entités manipulables

- les objets composés, qui sont créés à partir de plusieurs objets élémentaires

- la scène, qui est construite d'objets élémentaires et d'objets composés.

Cette démarche nécessite donc l'élaboration des logiciels pour construire :

- la bibliothèque d'objets

- la scène de façon interactive.

Le logiciel de construction de la bibliothèque permet à partir de la référence des objets élémentaires ou composés de définir la base de travail. Le résultat est alors appelé MENU.

La création des objets élémentaires est la phase cruciale car elle permet la modélisation des objets réels. La définition des objets peut être faite à l'aide d'un programme graphique interactif. Le programmeur doit alors, lors de cette phase, raisonner en tenant compte non seulement de la forme géométrique des objets réels mais également en incluant des informations métriques . Ceci devient rapidement fastidieux et requiert une attention soutenue du programmeur. La démarche adoptée dans le logiciel PRATIC fait appel à la vision artificielle pour acquérir ces informations. Néanmoins, quelle que soit la méthodologie utilisée le résultat devra inclure la structure interne de l'objet, ce qui doit comprendre :

- le nom de l'objet
- le nombre de points
- la suite des points.

A partir de ces informations, le logiciel doit permettre la sélection d'un objet simple ou composé à ajouter dans la scène. Cette sélection peut se faire simplement en précisant le nom de l'objet. Par ailleurs le dialogue peut être amélioré en proposant les graphiques des objets inclus dans le menu. Les propositions peuvent se faire en utilisant la technique du fenêtrage.

Le placement d'un objet simple ou composé dans la scène s'effectue en appliquant à l'objet reconnu une combinaison de transformations géométriques. Ces transformations appliquées au centre de gravité de la représentation graphique des objets en permettent l'animation. Lorsque la position finale désirée est atteinte, celle-ci est validée. La succession de ces validations constitue alors la modélisation de la scène qui sera utilisée lors de l'exécution.

2.3 La modélisation des objets

Ainsi que l'illustre la figure II.1, la modélisation d'un objet intégre deux types de modèle :

- le modèle géométrique [REQ 82],[WES 80] qui ne prend en compte que la forme des objets. Ce modèle est essentiellement utilisé pour l'affichage et l'animation sur l'écran.

- le modèle fonctionnel qui se divise lui-même en deux parties :
  * l'une prend en compte les informations physiques permettant l'élaboration de la stratégie de préhension (point de prise, couple de serrage,..)
  * l'autre prend en compte les références objets définies lors de la conception. Ces références sont utilisées pour la réalisation du modèle d'assemblage.

Le modèle fonctionnel [MEY 82] qui est essentiellement utilisé lors de l'exécution peut également l'être lors de la phase de programmation, à des fins de contrôle et de validation.

contrôle et de validation.

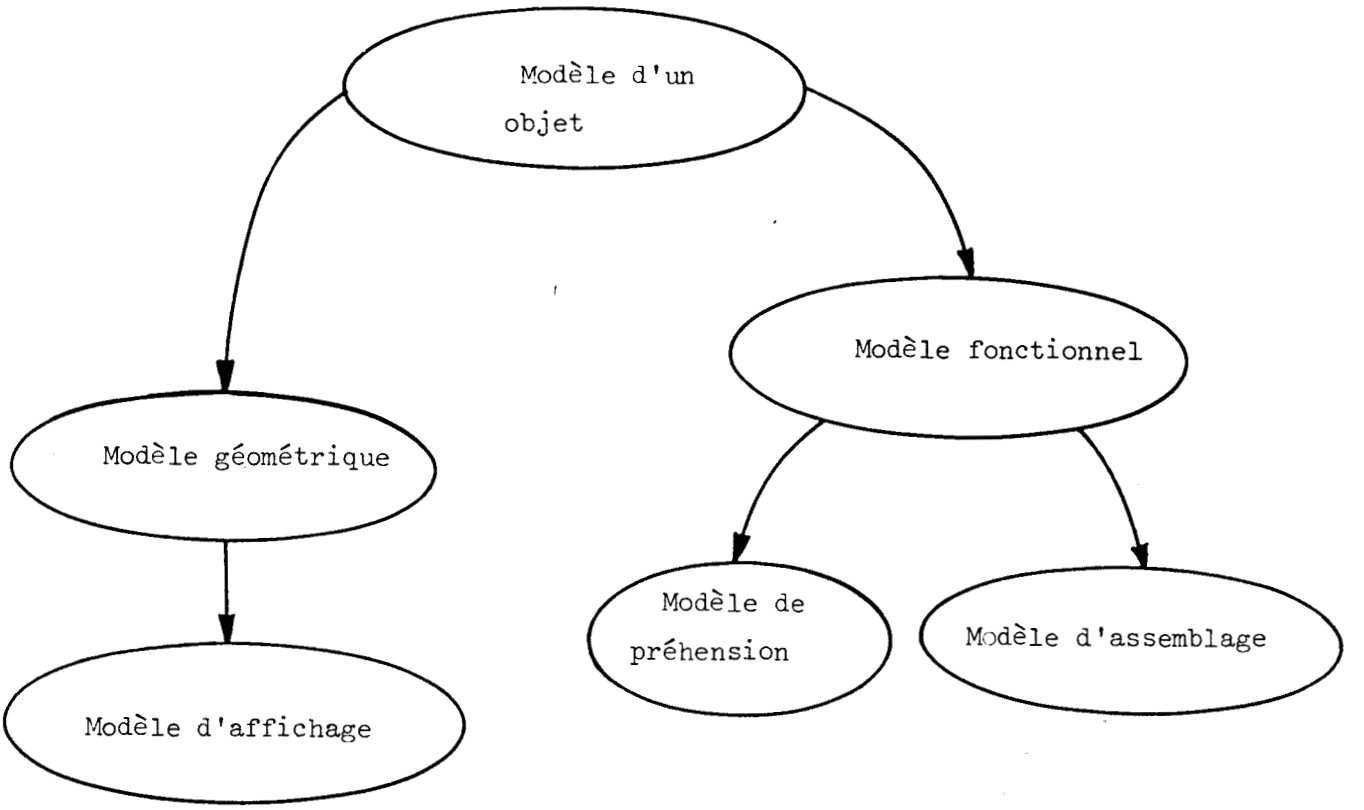


Figure II.1 : Décomposition des modèles

Le modèle géométrique dépend des différentes représentations de l'objet. La figure II.2 montre les représentations d'un objet à un instant donné [GAR 83]. La phase 1 constitue l'interface utilisateur permettant d'acquérir l'objet. La phase 2 permet de modéliser l'objet en fournissant une structure logique de représentation. Dans la phase 3, le modèle logique est traduit dans une structure de données directement utilisable par le calculateur. La phase 4 intégre les logiciels de visualisation utilisant les données précédentes.

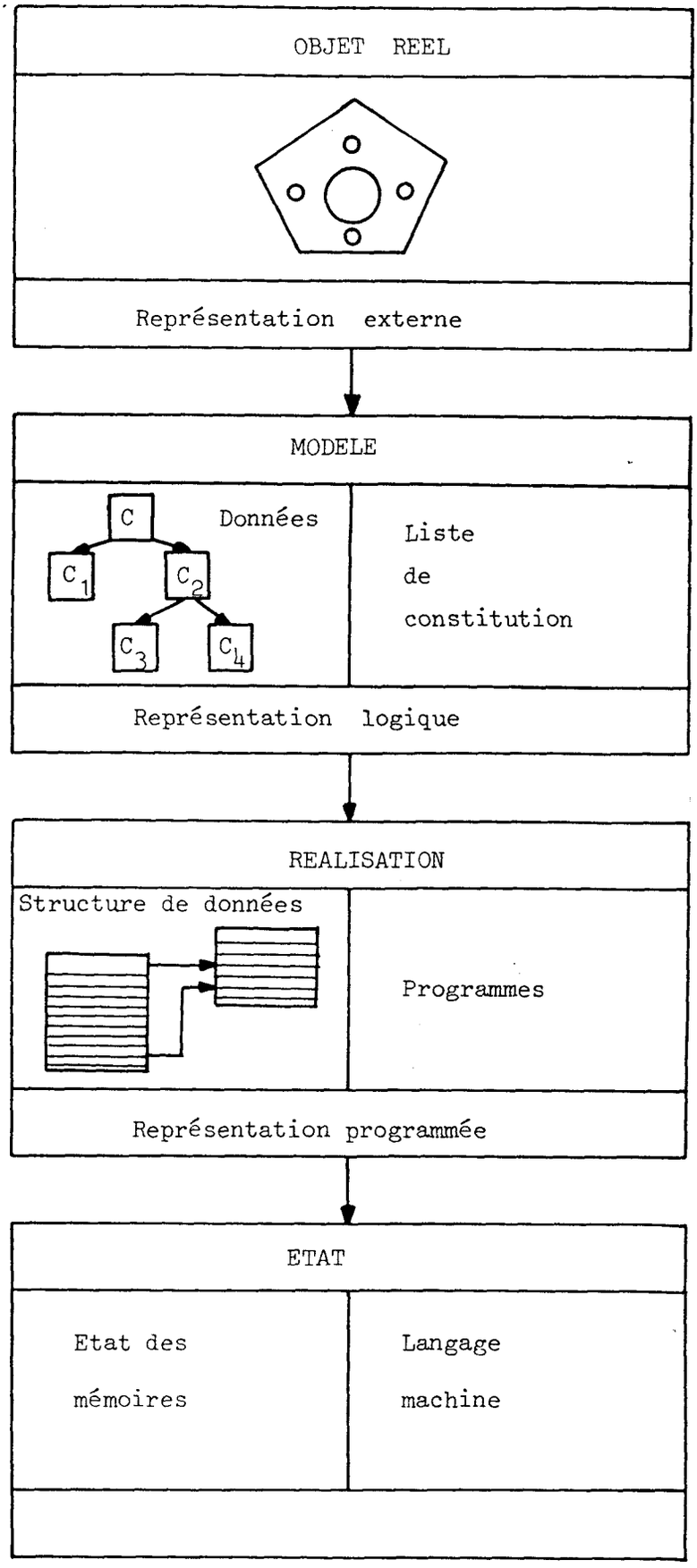


figure II.2: Représentation d'un objet

II.3 La vision artificielle

Dans ce paragraphe, l'objectif essentiel est de mettre en évidence les différents modes de reconnaissance et de positionnement bidimensionnels permettant d'analyser un vrac planaire. L'information résultant de cette analyse est bien sûr primordiale pour assurer la réalisation de l'objectif souhaité. De plus il a semblé intéressant de mettre à profit la phase d'extraction des contours pour acquérir simplement la représentation graphique des objets réels qui sera utilisée lors de la phase de programmation de l'objectif final désiré. Ces contours doivent être codés en tenant compte des contraintes suivantes :

- indépendance de la position sous la caméra
- unités correspondant aux mesures réelles
- module et phase des points de contour codés en coordonnées polaires à partir de l'axe d'inertie maximale.

3.1 Extraction des contours

Les contours de luminance sont définis dans une image comme les zones de variations brutales de l'intensité lumineuse [ROS 82]. Les contours de luminance contiennent souvent une information très riche.

3.1.1 CAS DES IMAGES BINAIRES

Dans une image seuillée, les points de contours sont naturellemnt définis sur chaque ligne et chaque colonne de la matrice de l'image à chaque transition blanc-noir ou noir-blanc.

3.1.2 CAS GENERAL

Dans une image quelconque, on peut détecter les points de contour avec des opérateurs de différenciation spaciale.

a) calcul du gradient

Le gradient de la fonction image F(x,y) est défini en tout point par :

$$\vec{\nabla} F(x,y) = \begin{pmatrix} gx \\ gy \end{pmatrix}$$

où $\quad gx = \dfrac{\partial F(x,y)}{\partial x} \quad$ et $\quad gy = \dfrac{\partial F(x,y)}{\partial y}$

Lorsque la fonction est représentée sous sa forme discrète F(i,j). Les composantes gx et gy sont estimées par une convolution discrète à l'aide d' un filtre dont la réponse impulsionnelle est donnée par hx,hy (opérateur de Roberts) [PAV 78]

$$hx = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \qquad hy = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \qquad \begin{pmatrix} i,j & i+1,j \\ i,j+1 & i+1,j+1 \end{pmatrix}$$

On calcule donc en tout point de l'image (i,j)

$$gx = F(i+1,j+1) - F(i,j)$$

$$gy = F(i+1,j) - F(i,j+1)$$

L'approximation du module du gradient est

$$\| \vec{\nabla} F(i,j) \| \simeq (gx^2 + gy^2)^{\frac{1}{2}}$$

l'orientation est donnée par

$$\Theta g \simeq Arctg (gy / gx)$$

D'autres opérateurs effectuent une estimation du gradient (Nobel, Kirsh, Prewitt). La determination des points de contour sont ceux pour lesquels le module du gradient est supérieur à un certain seuil.

b) utilisation du laplacien

Le laplacien de F(x,y) est un scalaire défini par

$$L(F(x,y)) = \frac{\partial^2 F(x,y)}{\partial x^2} + \frac{\partial^2 F(x,y)}{\partial y^2}$$

Le passage en discret requiert la convolution de l'image avec des noyaux permettant l'approximation discrète de la somme des dérivées secondes.

La détermination des points de contours sont obtenus lorsque le laplacien change de signe.

Ces méthodes sont largement utilisées. D'autres existent cependant :

- la méthode de recherche de coincidence de contours [HUE 71], [PAV 77], [DUD 72]

- la méthode de recherche par régions de propriétés similaires [MAR 76],[BRI 70]

- les méthodes statistiques, l'analyse de texture [BHA 81], [ROS 81]

3.1.3 CODAGE DE CONTOUR

Lorsque les points de contour ont été extraits de l'image, puis affinés, il faut alors les considérer comme la représentation discrète de courbes fermées de R . Plusieurs techniques sont utilisées pour coder les lignes de contour :

a) Codage par coordonnées

On stocke en mémoire les coordonnées cartésiennes de chaque point de contour.

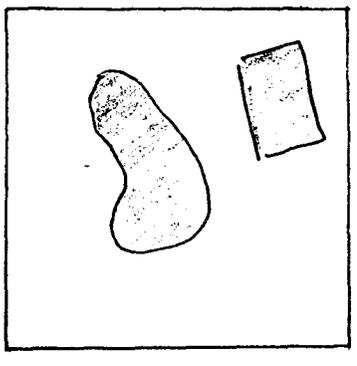
b) codage par marqueur

On associe à l'image une matrice binaire, l'état de chaque cellule indique la présence ou l'absence de point de contour.

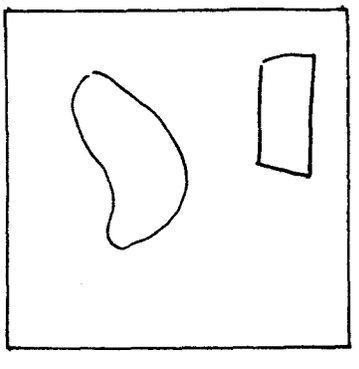
c) codage par vecteur de Freeman

La méthode consiste à coder les déplacements élémentaires que l'on effectue en suivant la ligne de contour.

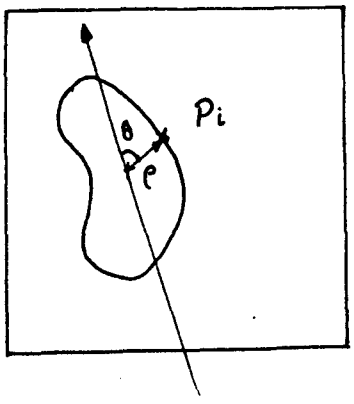
L'utilisation des coordonnées, moyennant les tranformations nécessaires à l'obtention d'un codage intrinsèque invariant, permet un passage aisé du repère caméra au repére utilisateur et fournit ainsi une représentation graphique efficace des objets réels. La figure II.3 résume les traitements impartis à la vision artificielle pour permettre l'acquisition des informations relatives aux contours des objets. Ces informations sont ensuite traitées par le superviseur pour déterminer la modélisation de ces objets .



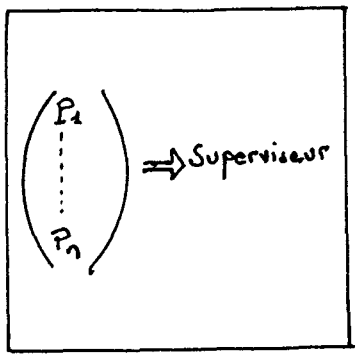
a) acquisition de l'image



b) extraction des contours



c) codage du contour en coordonnées polaires par rapport au centre de gravité de l'objet dans le repère camera



d)transfert de ces informations au superviseur

figure II.3 : Traitements impartis à la vision

3.2 RECONNAISSANCE ET POSITIONNEMENT BIDIMENSIONNELS

La plupart des systèmes de vision industriels existant à l'heure actuelle ne savent réaliser que la localisation d'un objet connu, plan et dans de bonnes conditions d'éclairage [ZIM 83], [MIR 84],[FAU 84]. On peut dire que ces systèmes ne fonctionnent qu'en présence d'un contraste important entre le fond et l'objet qui permet par simple seuillage de se ramener à une image binaire facilement exploitable. Par ailleurs ils imposent la contrainte supplémentaire d'objets isolés. La caractérisation de ces objets (type, position, orientation) est alors effectuée par des attributs globaux tels que périmètre, surface, inertie... La reconnaissance s'effectue ensuite soit à partir d'arbres de décisions binaires soit à partir de tests bayesiens.

L'évolution de ces systèmes vers une plus grande généralité doit permettre l'analyse de scènes comprenant :

- un morceau d'objet
- plusieurs objets se touchant
- plusieurs objets pouvant se chevaucher partiellement

- plusieurs objets pouvant avoir des défauts.

Ces systèmes doivent se caractériser par une capacité accrue de segmentation d'images. Ils doivent également être capables d'isoler des objets ou des portions d'objets dans des conditions de contraste et d'éclairage assez mauvaises en utilisant des opérateurs d'extraction de contours performants.

De plus, les programmes qu'ils mettent en oeuvre manipulent des descriptions symboliques de modèles d'objets. En conséquence les procédures de reconnaissance et de positionnement font de plus en plus appel à l'intelligence artificielle plutôt qu'à la reconnaissance de formes statistiques ou à l'analyse de données.

Plusieurs systèmes ont été récemment développés dont ceux de PERKINS [PER 78], JUVIN [JUV 83], ou BOLLES et CAIN [BOL 82]. Ces systèmes utilisent, pour reconnaitre les objets, des modèles sous formes d'attributs locaux tels que des petits trous circulaires, des coins rectanglaires ou des segments de droites.

Le système PVV (ITMI) [LUX 84] comporte deux modules permettant une reconnaissance accrue. Le premier module extrait de l'image des attributs à base de segments de droites. Le deuxième module basé sur la prédiction et la vérification d'hypothèses interprète les attributs extraits par la segmentation en termes de modèles des objets. L'avantage de cette reconnaissance est que le système est indépendant de la position de la caméra.

Un autre système est HYPER (INRIA, HYpothèse, Prédiction et Evalation Récursive) [AYA 83]. La partie extraction de contours est indépendante de la partie reconnaissance. Elle utilise la combinaison d'opérateurs de dérivation du premier et du second ordre. Les attributs images et modèles sont donc des segments de droites et le modèle privilégie un certain nombre de segments. La reconnaissance et le positionnement s'évaluent par prédiction et évaluation récursive d'hypothèses. L'hypothèse étant l'appartenance d'un segment privilégié du modèle à la scène, l'étape de vérification met à jour récursivement l'estimation transformant le modèle dans la scène à l'aide d'un filtre de KALMANN.

3.3 .SEPARATION DES OBJETS SE TOUCHANT

3.3.1 POSITION DU PROBLEME

L'analyse du vrac bidimensionnel conduit souvent à l'analyse de scène dans laquelle des objets se touchent (cf figure II.4).

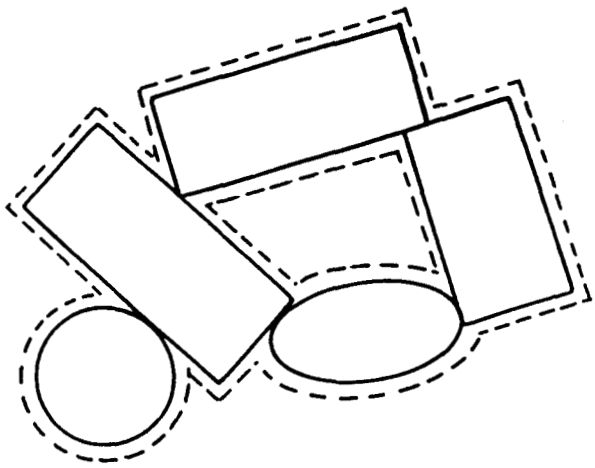


figure II.4: Exemple de vrac

L'extraction directe après seuillage du contour met en évidence :

- des points de rebroussement aux zones de contact des objets
- un segment de contact d plus ou moins long selon les objets.

Le calcul des paramètres globaux des objets à reconnaître devra donc s'effectuer en reconstituant le contour, à partir d'un contour initial non connu enveloppant tous les objets.

3.3.2 RECONSTRUCTION DU CONTOUR

Plusieurs démarches peuvent être envisagées pour reconstituer le contour.

a) Recherche des points de rebroussement des contours par analyse de l'annulation des dérivées secondes des contours et changement des signes de la dérivée première, puis appariement des points trouvés susceptibles d'appartenir à un segment de contact. Pour cette méthode les calculs nécessaires à la détection des points de rebroussement ne posent pas de problèmes mais les temps de calcul sont importants . Par contre l'appariement de ces points est difficile car la distance de contact dépend notamment :

- de la forme de l'objet
- de l'orientation de celui-ci
- de la zone où s'effectue le contact

(cf figure II.5)

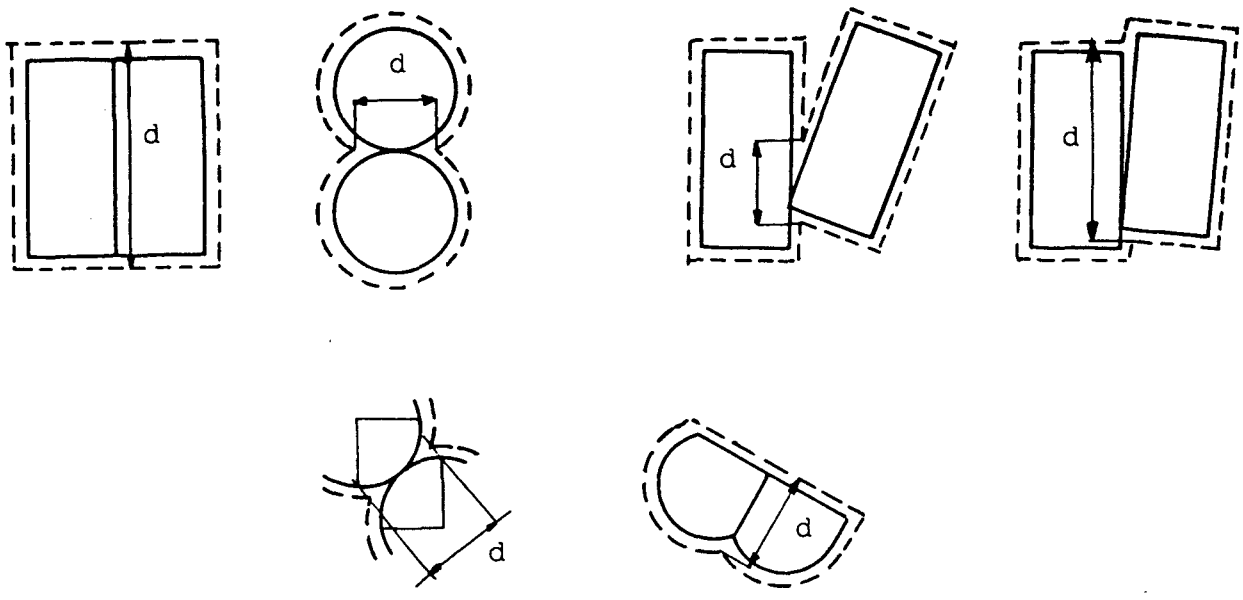


Figure II.5 : Détermination de la distance d

On voit donc qu'il est pratiquemment impossible d'élaborer un algorithme permettant l'appariement des points de rebroussement pour des objets quelconques. Il

est clair également que ce type d'algorithme devient extrèmement performant pour les objets possédant une description de contour invariante par rapport à l'orientation de l'objet et une distance d de contact constante. C'est le cas par exemple des objets circulaires (cf figure II.6).

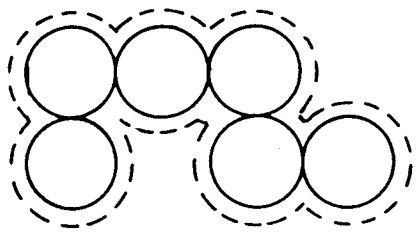


Figure II.6 : Distances de contact d identiques

b) Une autre méthode consisterait à déterminer :

- soit des segments spécifiques des objets et de les rechercher dans l'image [PER 78], [BOL 82].

- soit de travailler sur la signature des objets [JUV 83].

Ces méthodes possèdent l'avantage de pouvoir prendre leurs décisions de reconnaissance même si l'objet est partiellement occulté. Néanmoins elles ne peuvent aboutir que si ces segments particuliers peuvent être extraits, ce qui n'est pas toujours le cas dans le vrac industriel où le nombre d'objets à reconnaître peut atteindre plusieurs dizaines comme dans le cas de la photo II.7.

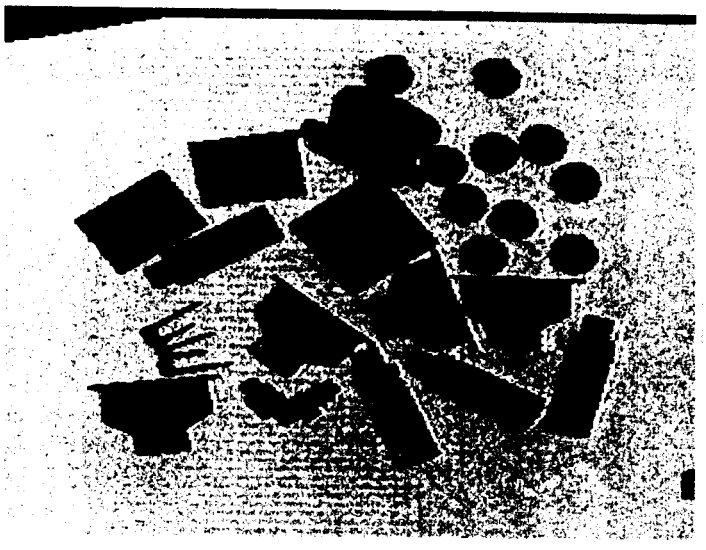


Photo II.7 : Vrac industriel

c) La démarche adoptée dans le système utilisé dans le cadre de cette étude est différente. Elle se réduit à l'analyse de scènes.

Les cas du chevauchement et de la vision partielle sont donc ici exclus. L'analyse de l'image s'effectue alors en deux étapes principales :

- la première étape développe une technique de segmentation plus élaborée pour permettre la séparation des objets se touchant.

- la deuxième étape détermine la reconnaissance et la position des objets une fois l'extraction des contours effectuée par le calcul des attributs globaux tels que surface, périmètre ...

La morphologie mathématique a été employée lors de la première étape. L'emploi de ses opérateurs a permis, dans le contexte du vrac planaire, d'améliorer les temps de calcul et d'obtenir des temps typiques de reconnaissance par objet de l'ordre de 100 millisecondes compatibles avec une exécution temps réel.

III.4 La morphologie mathématique

Les outils de la morphologie mathématique développés par Matheron [MAT 67] et Serra [SER 82] sont principalement adaptés à l'analyse et la caractérisation des textures [MEY 79]. Néanmoins leur utilisation s'est avérée très utile pour la séparation des objets. Les opérations les plus couramment utilisées de la morphologie mathématique sont l'érosion, la dilatation, la squelettisation, l'ouverture et la fermeture.

4.1 DILATATION ET EROSION D'UNE IMAGE

Notation : soit X l'ensemble des points constituant l'image et y un point de cette image, B un ensemble de points appelé élèment structurant. Un de ces points, y0 est particularisé et nommé centre de B. On dit alors que B est positionné en y lorsque y=y0 et on désigne par By l'ensemble ainsi positionné.

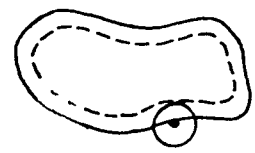
- Dilatation de X par B

On note Y=X+B l'ensemble des points

$$Y = \{ y : By \quad X = 0 \}$$

L'opération revient donc à inclure dans Y les points interceptés par B lorsque y balaye l'ensemble X.

contour original



contour aprés dilatation

figure II.8 : Dilation

- Erosion de X par B

On note Y = Y - B l'ensemble des points

$$Y = \{ y : By \subset X \}$$

On remarque donc que Y est formé des points y tels que l'élèment structurant centré en y, By, soit inclus dans X.

figure II.9: Erosion

- dualité

Erosion et dilatation sont duales par rapport à la complémentarité soit

$$(X + B)^C = X^C - B$$

autrement dit il revient au même de dilater les points blancs que d'éroder les points noirs de l'image.

4.2 OUVERTURE ET FERMETURE

L'ouverture de X par B, notée XB est définie par

$$XB = (X - B) + B.$$

Réciproquement, la fermeture de X par B (notée XB) est définie par

$$XB = (X + B) - B.$$

L'ouverture consiste à effectuer une érosion puis une dilatation. L'ensemble XB est alors le domaine balayé par tous les translatés de B inclus dans X. Cette opération tend à lisser les contours de X et à supprimer les petites irrégularités dans l'image (Serra page 52).

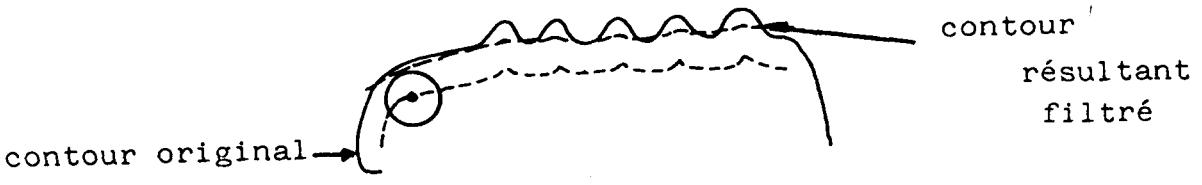


Figure II.10 : Ouverture

La fermeture est duale de l'ouverture par complémentarité.

$$(XB)^C = (X^C)B \text{ et } (X^C)B = (XB)^C$$

La fermeture des points noirs s'interprète donc comme l'ouverture des points blancs de l'image.

4.3 SQUELETTISATION D'UNE IMAGE BINAIRE

Le squelette d'une forme X quelconque est défini de façon imagée en considérant que les contours de la forme soient érodés de l'extérieur jusqu'à ce que les fronts d'érosion se rejoignent en des points qui constituent le squelette.

La définition précise du squelette nécessite la définition de la distance d(x,A) d'un point x à un ensemble A. Cette distance est définie par

$$d(x,A) = \inf (d(x,y))$$
$$\text{quand y appartient à A}$$

où d(x,y) désigne la distance euclidienne entre deux points x et y . Pour chaque point x appartenant à X, on calcule d(x,Fx), distance de x à la frontière Fx de la forme X. L'ensemble des points du squelette est constitué des points z pour lesquels la distance d(z,Fx) est réalisée par au moins deux points distincts de la frontière Fx. Si on associe à chaque point z la distance q(z) = d(z,FX) on peut alors reconstruire la forme X entièrement, à partir du squelette.

Le squelette d'une région quelconque est généralement très sensible au bruit. Par contre, calculé sur des régions plutôt allongées comme les régions de contour, le squelette fournit une représentation robuste et affinée des contours. Cette représentation affinée des contours rend ceux-ci mieux adaptés à des traitements ultérieurs.

En pratique, une approximation du squelette sur une image numérique binaire peut être calculée par des algorithmes séquentiels ou bien par des opérateurs morphologiques locaux et exécutables en parallèle. Le lecteur trouvera le fondement mathémathique et l'analyse des propriétés du squelette dans Serra, chapitre XI et dans Lantuejoul [LAN 78]. Seuls quelques exemples de squelette sont donnés par la figure II.11.

Figure II.11: Exemples de squelettes

4.4 METHODOLOGIE EMPLOYEE

L'application du squelette permet donc de trouver les zones frontières séparant les objets (cf figure II.12).

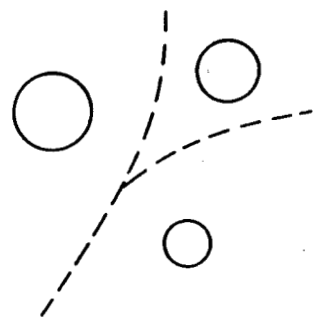


Figure II.12 : Séparation des objets par le squelette.

Cette propriété fondamentale dans notre application permet de séparer les objets. En effet l'action conjointe d'une érosion puis d'une squelettisation nous donne le résultat souhaité (cf figure II.13).

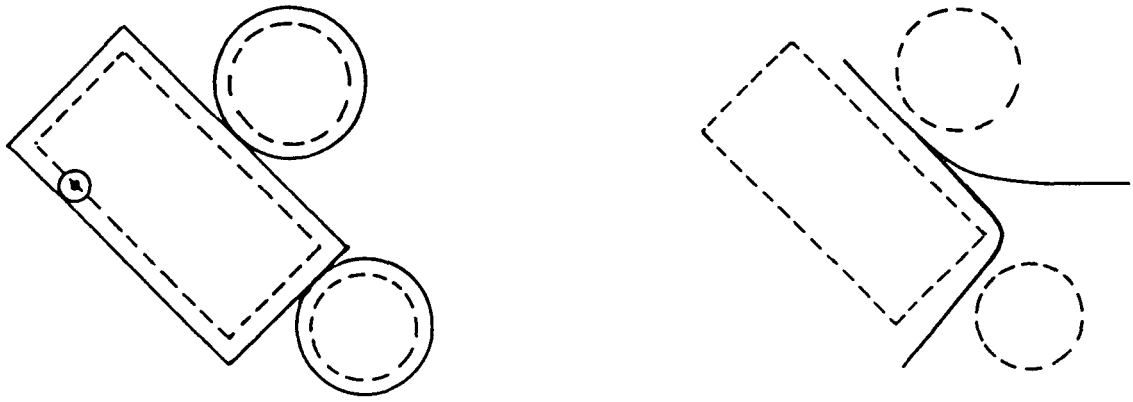


figure II.13: Séparation des objets

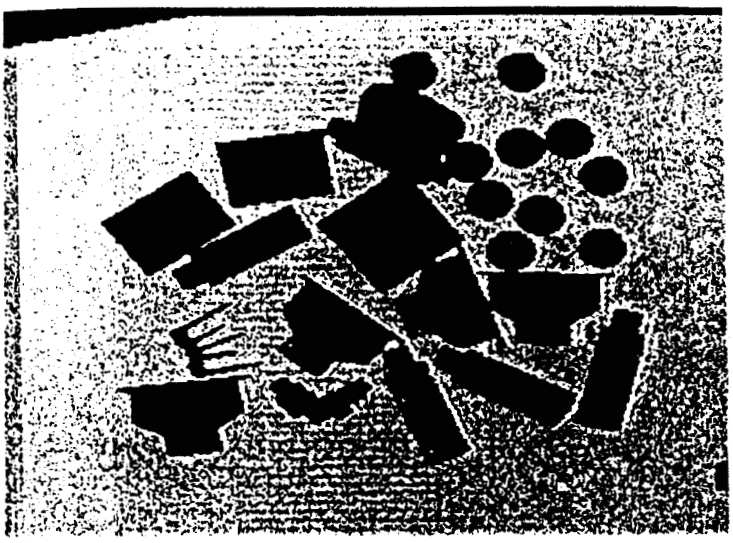
La reconstitution des objets par les opérateurs morphologiques permet par la suite de minimiser la perte d'informations engendrée sur les objets.

4.5 REGLES D'ELABORATION DE L'ALGORITHME UTILISE

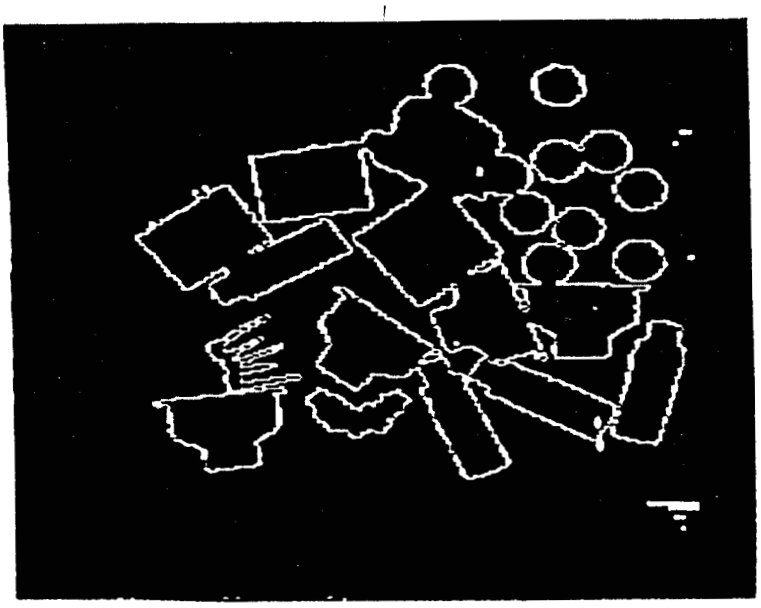
Pour utiliser pleinement la méthodologie ci dessus, il est nécessaire que l'algorithme développé puisse définir les paramètres d'action des opérateurs employés. On peut définir cinq règles qui président à l'élaboration des séquences morphologiques.

1) passage en revue des trois modes de réduction de l'information.

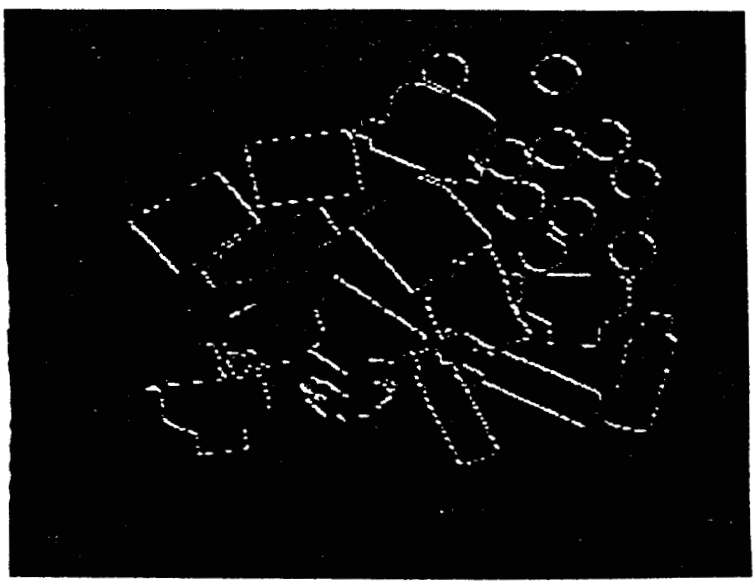
* réduction spatiale : limiter les zones d'intérêt à des sous ensembles du champs initial.



a; binarisation de l'acquisition du vrac



b: extraction des contours avant traitements



c: extraction des contours aprés utilisation des algorithmes développés.

* réduction des définitions : quand a t-on intérêt à passer en binaire ?

* réduction local-global : isoler une partie de l'image.

2) lorsque deux opérations n'interagissent pas l'une sur l'autre, commencer par la plus grossière. Celle qui simplifie le plus est souvent la plus rapide à mettre en oeuvre.

3) connaître l'inventaire des séquences élémentaires de traitement qui s'enchaînent de manière exclusive.

4) quand deux portions de traitement interagissent, commencer par celle qui modifie le moins l'autre.

5) ne pas hésiter à perdre une information qu'on peut récupérer par la suite.

La connaissance de ces règles permet de définir les paramètres d'action des opérateurs . Dans le cas où le raisonnnement est fait en fonction de l'objet, hypothèse vérifiable et fondée sur la structure et la forme de celui-ci, ces paramètres permettent d'assurer à coup sûr une séparation des objets et par la suite leur reconnaissance . Ces paramètres d'action des opérateurs morphologiques deviennent donc des paramètres de reconnaissance propres à l'objet.

L'algorithme développé, qui intègre non seulement la prise en considération des règles qui viennent d'être exposées mais également une segmentation automatique par évaluation de la perte d'informations, donne de très bons résultats. Le pourcentage d'échecs, évalué sur plusieurs centaines de pièces est inférieur à 5%.

II.5 Conclusions

La première partie de ce chapitre a permis de définir les bases d'un langage de programmation graphique. La seconde partie montre l'utilisation de la vision artificielle pour acquérir les informations nécessaires à la modélisation des objets, puis dans une seconde phase, l'étude s'est portée sur la résolution du vrac planaire.

En conclusion, on peut donc définir le cahier des charges du système de vision artificielle . Il doit être capable d'après les ordres du superviseur:

- d'extraire les informations nécessaires (codage du contour) pour la détermination de la modélisation des objets par le superviseur.

- de séparer les objets se touchant pour permettre le calcul des attributs globaux et par conséquent la reconnaissance des objets constituant la scène lors de l'exécution.

Ces deux fonctions, totalement imparties au système de vision artificielle, sont précédées par la phase d'acquisition de l'image. Il est donc indispensable de posséder une fonction permettant l'acquisition. La souplesse de commande est acquise par le paramétrage de ces trois fonctions qui permettent réellement la commande de la vision artificielle par le superviseur.

On constate donc que tous les systèmes de vision artificielle possédant ces trois fonctions seront utilisables par le langage PRATIC.

CHAPITRE III

PRATIC,

Programmation de Robot Assistée par Traitement d'Images et Caméra.

III.1 Présentation

PRATIC (Programmation de Robots Assistée par Traitement d'Images et Caméra) est un système de programmation et d'exécution évolutif essentiellement destiné à des applications de robotique mettant en oeuvre des tâches qui nécessitent des déplacements soit dans un même plan, soit dans des plans parallèles.

Alors que les applications des systèmes de vision artificielle portent exclusivement sur leur intégration dans le contexte de tâches robotisées telles que : positionnement, reconnaissance et inspection, le langage PRATIC, exposé dans ce chapitre, vise à permettre l'utilisation du système de vision dès la phase de développement des programmmes. Dans ce sens, PRATIC simplifie le processus de modélisation géométrique des objets manipulés et de l'univers de travail.

La conception de PRATIC est modulaire, c'est-à-dire que l'opérateur peut développer et compléter le noyau de base fourni sans reconception importante du système global. Il peut donc envisager facilement l'adaptation et la reconfiguration du système à ses propres matériels.

Cette structure permet également à PRATIC, une fois la base de connaissance acquise, d'être autonome dans la phase de programmation. La programmation hors-ligne ne requiert alors que l'ordinateur principal (superviseur).

Les développements poursuivis sur cette structure permettent de dégrader le système logiciel pour n'utiliser que le module exécutif. Cet aspect est très important car, économiquement il conduit à une réalisation de faible coût lors de l'installation de plusieurs robots sur site industriel.

III.2 Architecture du système

2.1 ARCHITECTURE MATERIELLE

Elle est illustrée figure III.1. Elle se compose d'un ordinateur MICRAL 9050 (Microprocesseur 8086 avec coprocesseur arithmétique 8087 et Système d'exploitation CP/M 86) configuré selon une architecture temps réel dans laquelle il joue le rôle de superviseur décisionnel vis à vis du robot, du système de traitement d'images et de la console graphique de visualisation. Le robot utilisé pour l'illustration est un robot SCEMI 6 axes commandé par un calculateur DEC LSI 11/23. Le système de vision utilisé est le VISIOMAT de chez ROBOTRONICS utilisant une caméra C.C.D. Plusieurs versions du logiciel ont été élaborées en fonction des ressources graphiques disponibles qui peuvent être soit la console graphique du Micral (480 x 1024 points) soit une console couleur SECAPA 550 (600 x 800 points) ou TEKTRONICS 4107 (640 x 480 points). Les photos apparaissant dans ce chapitre et montrant les différentes phases de la programmation ont été tirées sur la console SECAPA 550. Les différents couplages physiques entre le calculateur de supervision et les éléments commandés sont réalisés à partir de liaisons séries RS 232, selon un réseau en étoile.

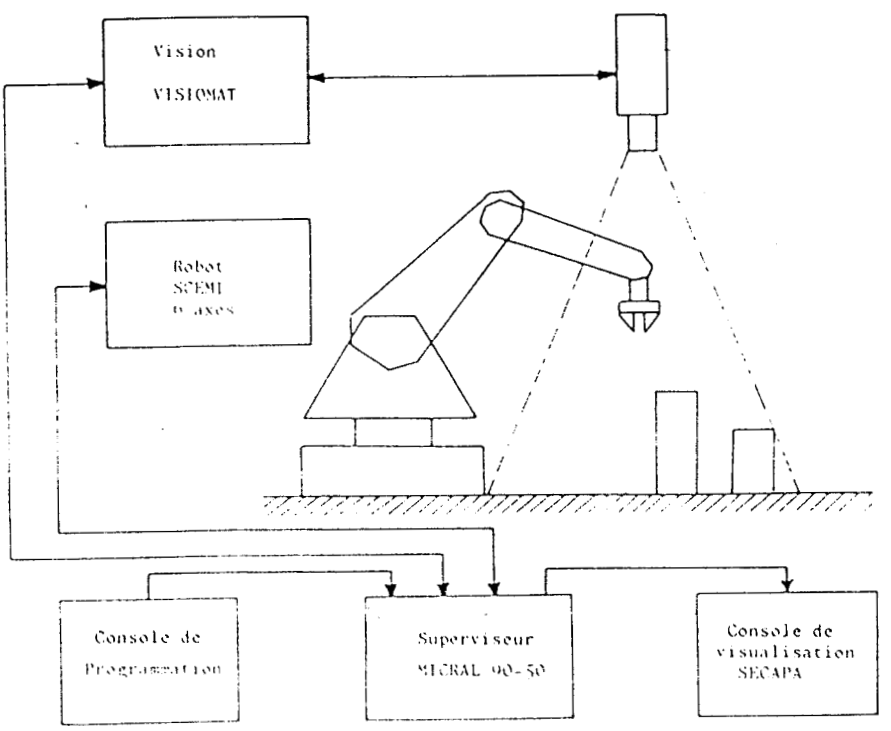


Figure III.1 : Architecture matérielle du système

2.2 ARCHITECTURE LOGICIELLE

2.2.1 Présentation

Le fonctionnement temps réel de cette structure hiérarchisée est assuré de la manière suivante (figure III.2) :

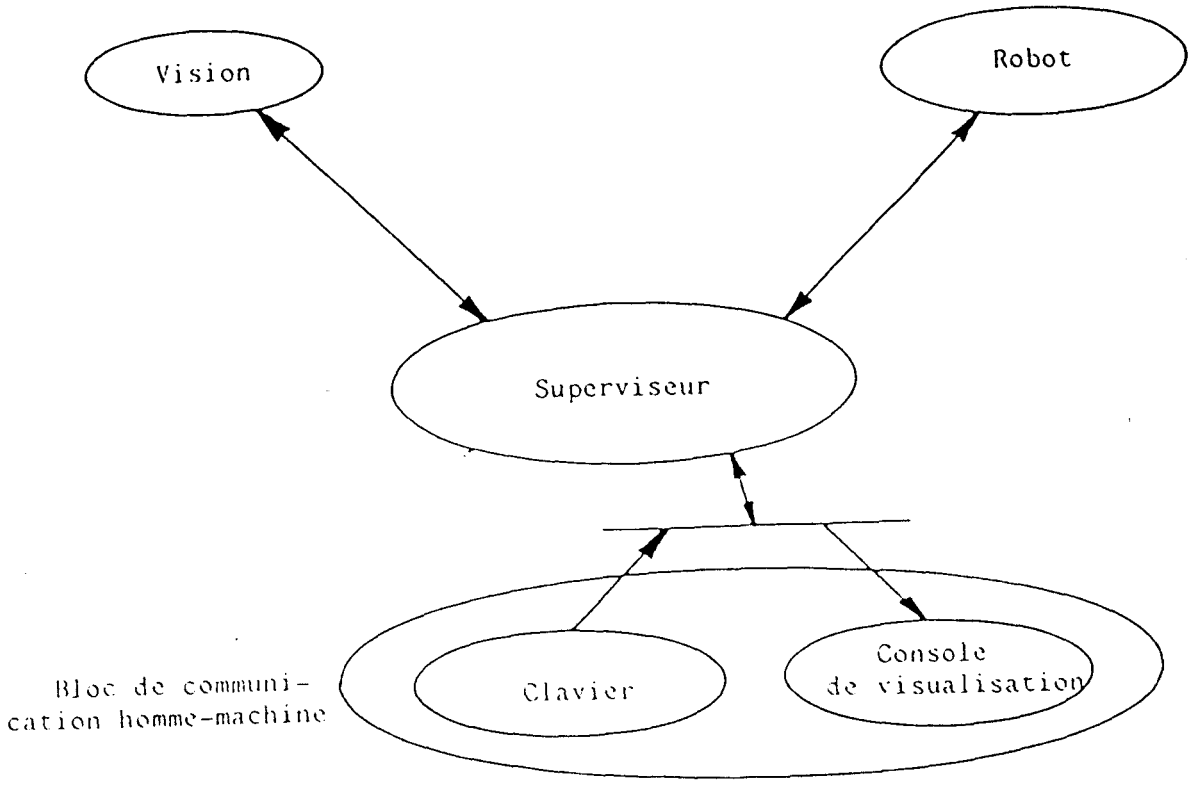


figure III.2 : Architecture logicielle

- le calculateur superviseur prend l'initiative de lancer les opérations sur chacun des éléments à commander.

- en retour, par un protocole adapté, chacun des éléments interrompt le calculateur superviseur, soit pour rendre compte d'une opération terminée, soit pour transmettre des

données résultant d'une opération (traitement d'images notamment).

- les éléments commandés peuvent également générer des interruptions vers le calculateur superviseur en cas, soit d'anomalie, soit d'appel de l'opérateur à travers le clavier.

Cette structure a été développée en PASCAL MT+86 qui fournit la possibilité de gérer les interruptions dans un langage de haut niveau.

2.2.2 Organisation logicielle

Par l'utilisation du système de vision artificielle et par le dialogue homme-machine, PRATIC dispose d'une base de données exploitée pour la programmation et l'exécution. Dés lors le logiciel se décompose en trois modules principaux :

- module de définition
- module de programmation
- module d'exécution

Ces trois modules, ainsi que l'illustre la figure III.3, constituent le noyau de PRATIC. Ce noyau utilise, par ailleurs, des logiciels intégrés dans les éléments périphériques (robot, vision artificielle, visu, clavier). Une telle structuration permet de répartir l'intelligence de traitement de manière aussi efficace que possible. Il en résulte une agrégation de l'information qui permet de minimiser les échanges entre le calculateur superviseur et les éléments commandés.

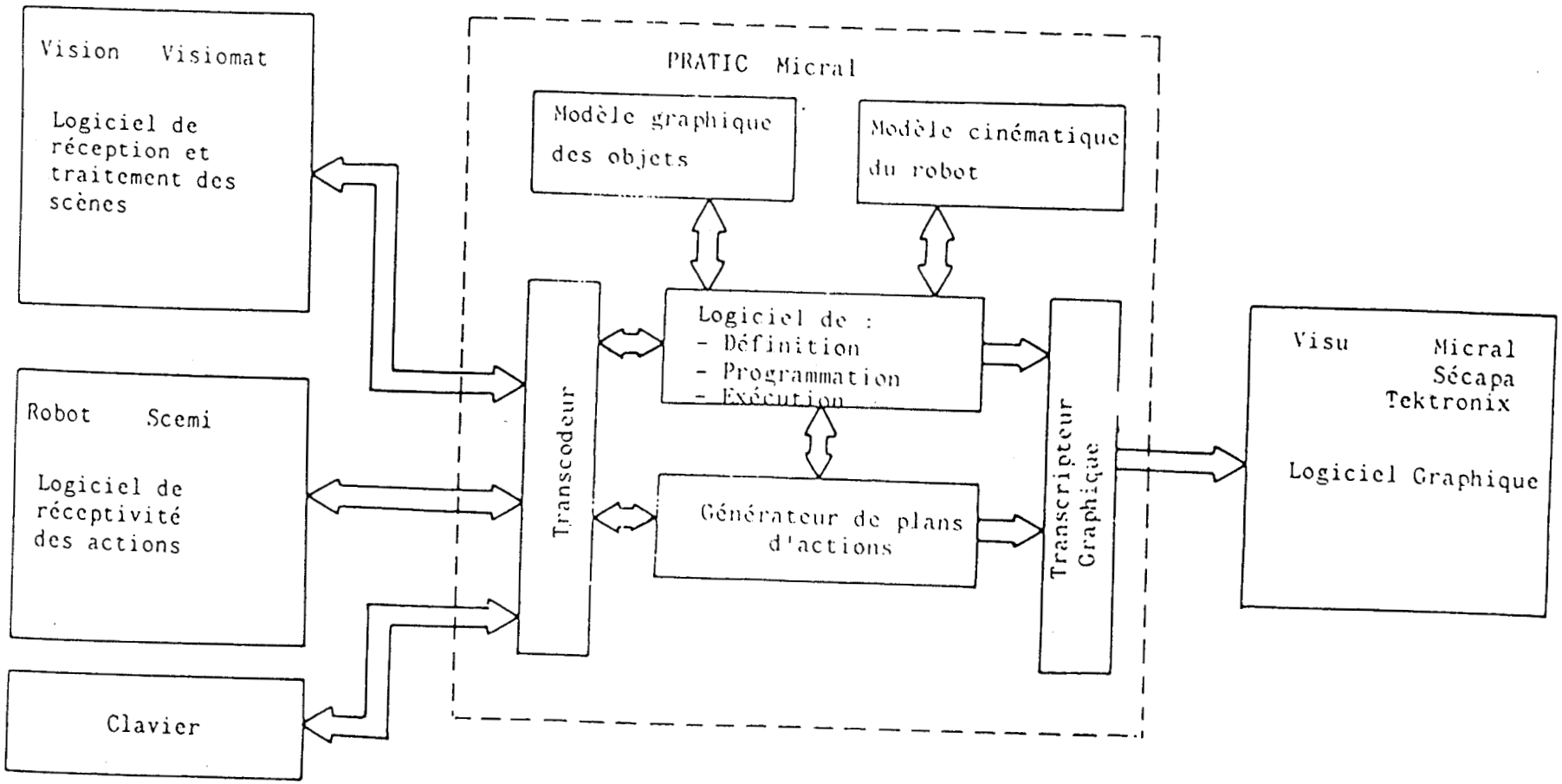


figure III.3 : Répartition des traitements

III.3 Les trois modules fondamentaux de PRATIC.

Le noyau de PRATIC est décomposé en trois modules principaux auxquels on peut accéder par l'intermédiaire d'un menu interactif. Ces trois modules, décrits dans cette section, sont :

- le module de définition
- le module de programmation
- le module d'exécution

La structure fonctionnelle générale de ce noyau est illustrée figure III.4. Le lecteur pourra se référer à cette figure pour une meilleure compréhension lors de la présentation des différents modules, dans la suite du développement.

3.1. LE MODULE DE DEFINITION

Ce module, qui est un prérequis de la phase de programmation, permet en premier lieu de créer la base de connaissances nécessaire à l'adaptation aux différentes ressources utilisées pour réaliser un ensemble de tâches données. Ces ressources intègrent deux types d'entités physiques :

- les objets à manipuler eux-mêmes.
- tous les éléments permettant de manipuler les objets : robots, préhenseurs, entrées-sorties, éléments de périrobotique.

L'accès à la définition de ces entités physiques est réalisé par le choix de différentes options proposées, ainsi que l'illustre la photo de la figure III.5.

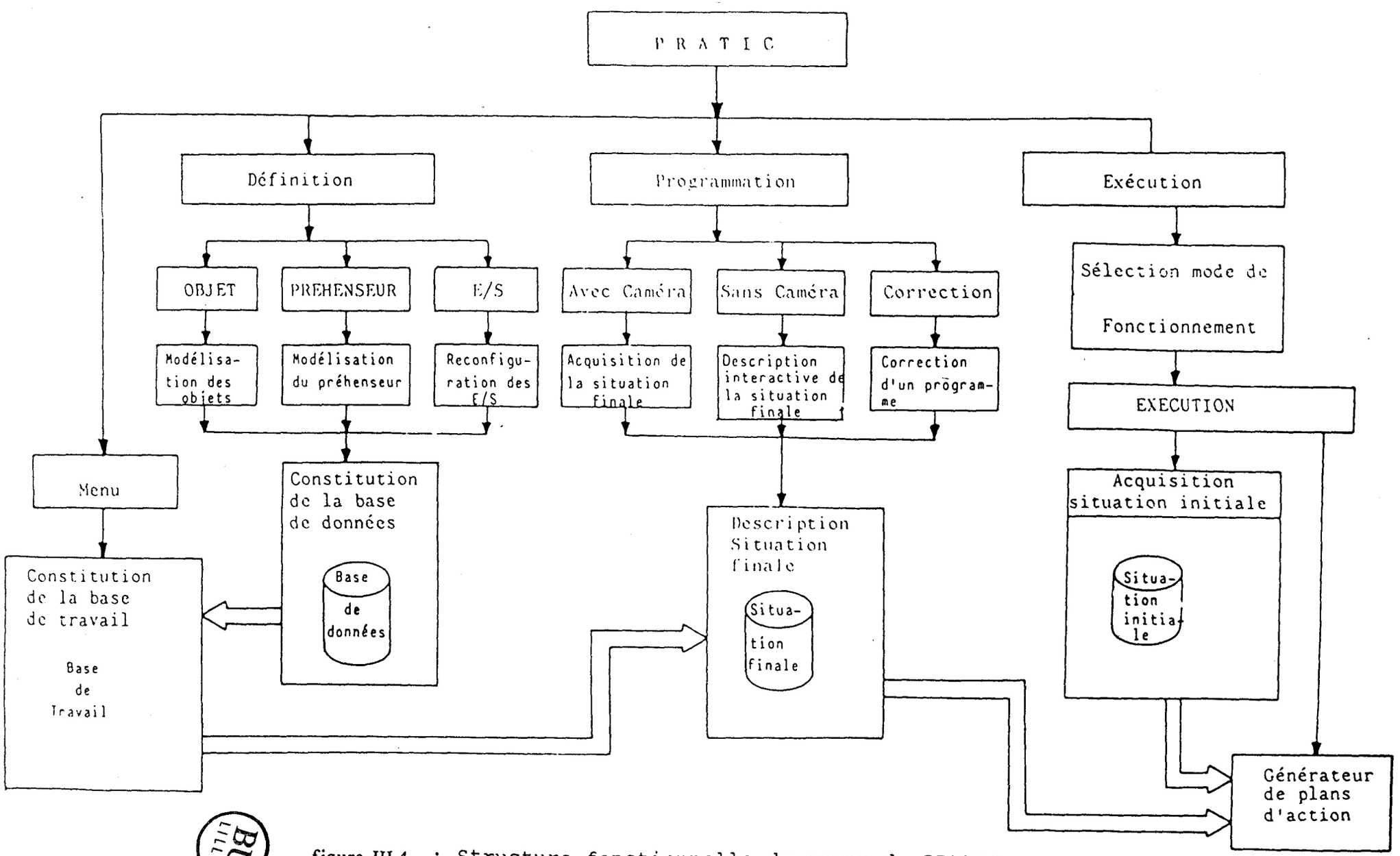


figure III.4 : Structure fonctionnelle du noyau de PRATIC

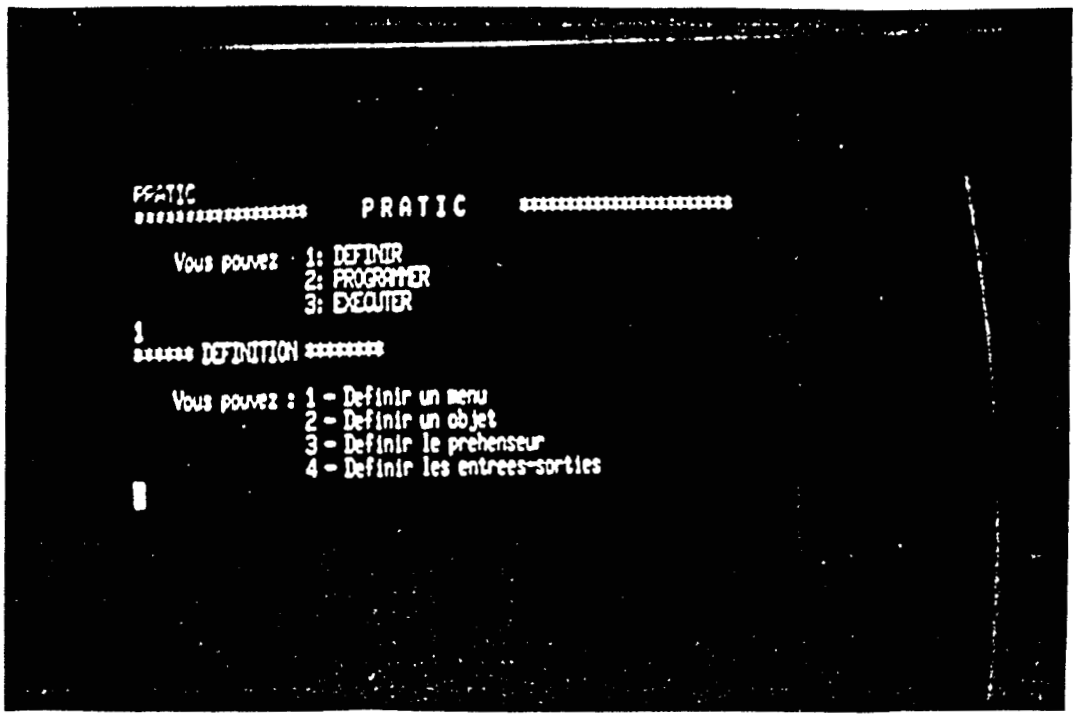


Figure III.5 : Menu de définition.

Actuellement 4 champs de définition sont opérationnels :

1 - Définition d'un menu

2 - Définition des objets

3 - Définition des préhenseurs

4 - Définition des entrées-sorties.

Les champs 2 et 3 permettent de réaliser la modélisation à proprement parler des entités physiques et donc de construire la base de connaissance du système . C'est ainsi que la modélisation géométrique des objets et des préhenseurs est basée sur l'utilisation des informations fournies par le système de traitement d'images.

Le champ 4 permet notamment une configuration des entrées-sorties du système et apporte ainsi une certaine souplesse au niveau matériel: adaptation des vitesses de transmission sur les lignes séries, définition des protocoles de transmission, ...

Le champ 1 permet de définir un menu de travail, c'est à dire de construire, à partir de la base de connaissance, une base de travail contenant tous les éléments nécessaires à la réalisation du programme envisagé (voir figure III.6).

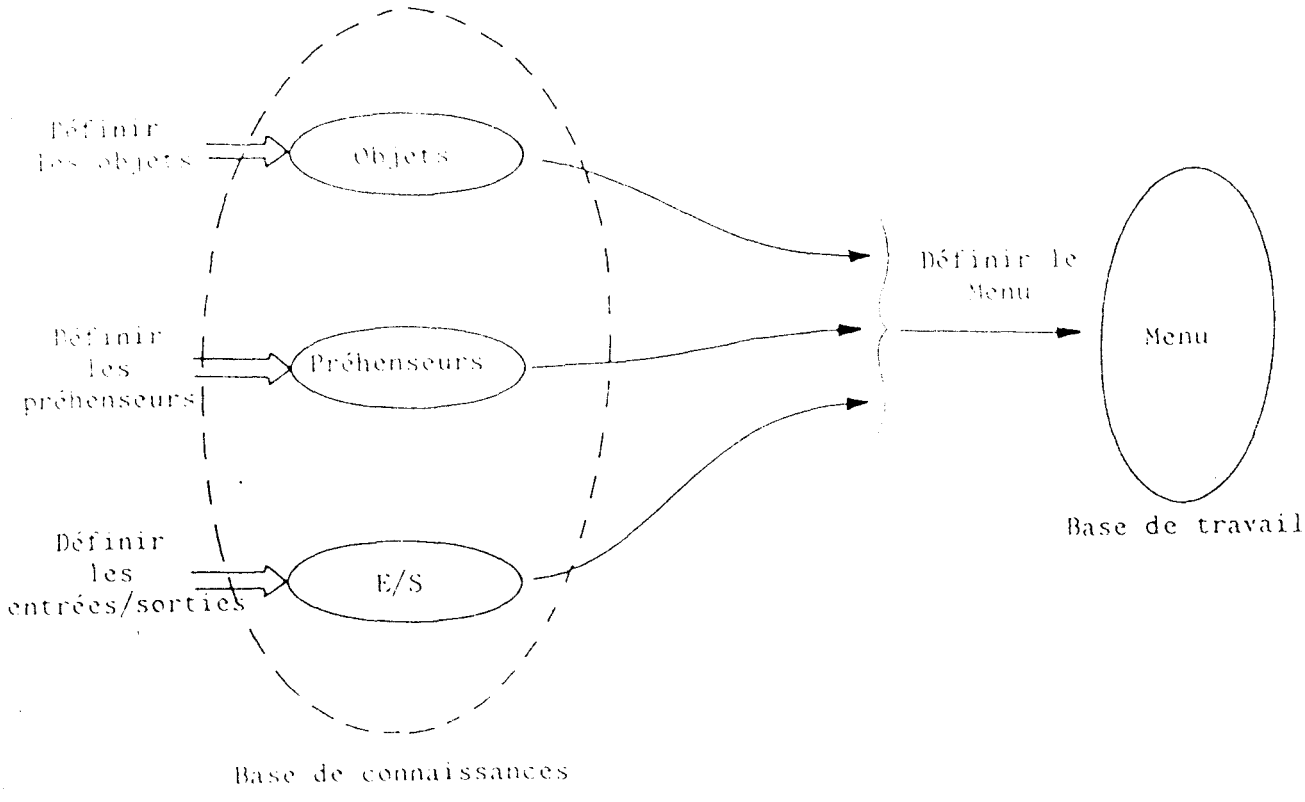


Figure III.6 :Fonctionnalité du module de définition

La réactualisation de la base de connaissance se fait selon les mêmes procédures avec toutefois un certain nombre de verrous logiciels empêchant notamment :

- de modéliser un objet déjà modélisé

- d'écraser un modèle existant

- d'écraser un menu existant

3.2 LE MODULE DE PROGRAMMATION

C'est l'élément central de PRATIC. Par accès aux modèles d'objets contenus dans la base de travail définie précédemment, il permet de décrire la situation finale à obtenir. Cette description peut être effectuée de 2 manières possibles:

- soit interactivement au moyen d'un éditeur graphique permettant de sélectionner un objet dans la base de travail puis de lui donner la position finale souhaitée dans le volume d'évolution du robot (figure III.7) (mode de programmation sans caméra).

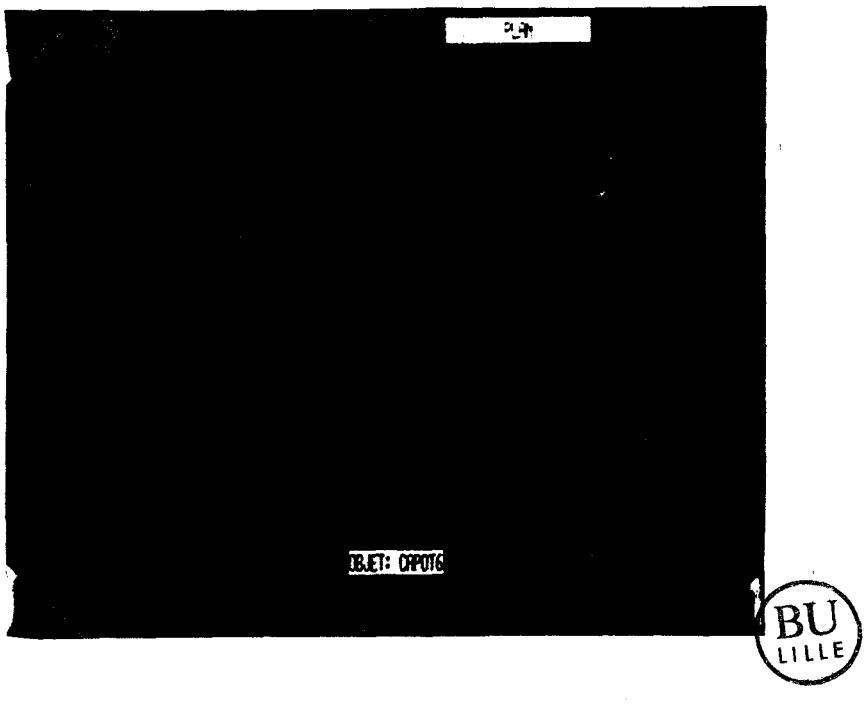


Figure III.7 :Edition interactive de la situation finale.

- soit globalement à l'aide d'une caméra et d'un système de traitement d'images (mode de programmation avec caméra).

Dans le premier cas l'opérateur raisonne au niveau de l'objet tandis que dans le deuxième cas la programmation s'effectue au niveau de l'objectif.

Par ailleurs, la correction des programmes s'effectue dans tous les cas dans le mode interactif (Figure III.8).

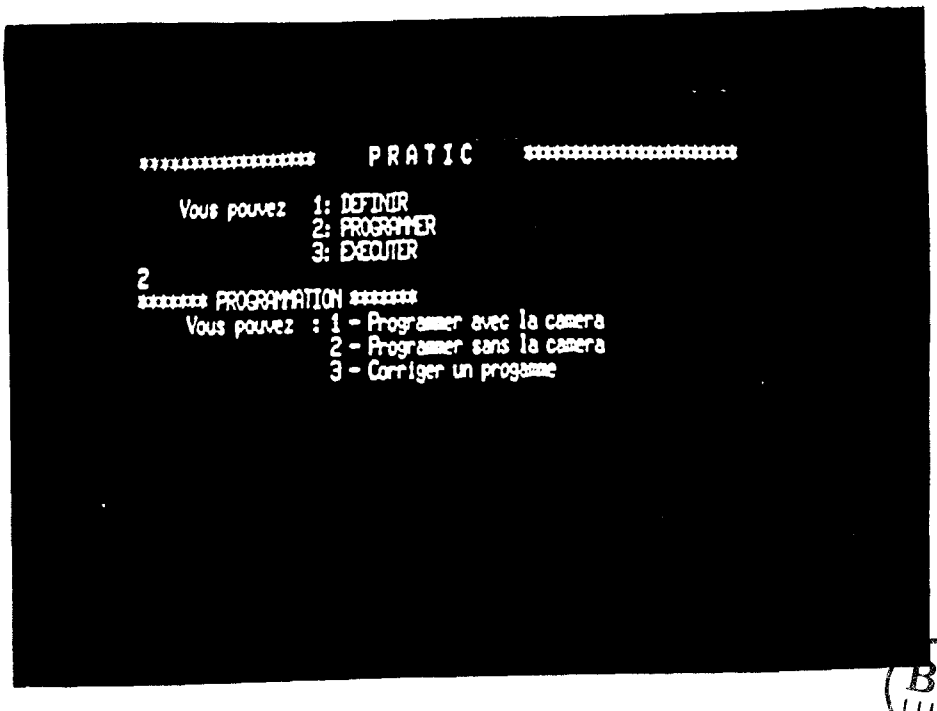


Figure III.8: Le menu du module de programmation.

Dans le cas de la programmation avec caméra, l'analyse de la situation finale est réalisée par le système de traitement d'images qui appréhende la scène et transmet les résultats de son analyse au superviseur. Celui-ci, utilisant alors la base de travail élaborée lors de la phase de définition, construit un modèle géométrique de la scène analysée.

Par ailleurs, afin de faciliter une programmation hors ligne, indispensable dans un contexte industriel, il est fondamental de pouvoir s'affranchir des contraintes qu'apporterait l'utilisation soit du repère caméra, soit du repère robot. Pour cela le module de programmation élabore, à partir du modèle géométrique de la scène analysée, un modèle géométrique de la

situation finale qui apparaît alors comme une entité manipulable accessible à l'opérateur. Celui-ci peut alors accéder au positionnement de la scène dans l'espace robot sans être lié à l'espace caméra.

Enfin, le module de programmation de PRATIC fournit d'autres facilités telles que, zoom, repérage d'objets par la couleur, sélection de vitesses de déplacement, ... qui seront décrites plus en détail dans l'exemple d'application présenté section IV.

3.3 LE MODULE D'EXECUTION

Le module d'exécution propose différents choix de déroulement des mouvements nécessaires pour réaliser la situation finale précédemment enregistrée :

- mode pas à pas
- mode cycle par cycle
- mode automatique

Ces modes d'exécution sont classiques. Leur description qui sort du cadre de cette étude ne sera donc pas effectuée ici.

L'opérateur a également le choix d'accéder ou non à l'illustration graphique de la tâche sur la console de visualisation (figure III.9).

Quel que soit le mode d'exécution choisi, celle-ci commence par l'acquisition de la situation initiale présente sous la caméra. Les informations de cette acquisition sont ensuite transmises au calculateur superviseur qui possède alors tous les éléments nécessaires pour modéliser et éventuellement représenter sur écran la situation initiale.

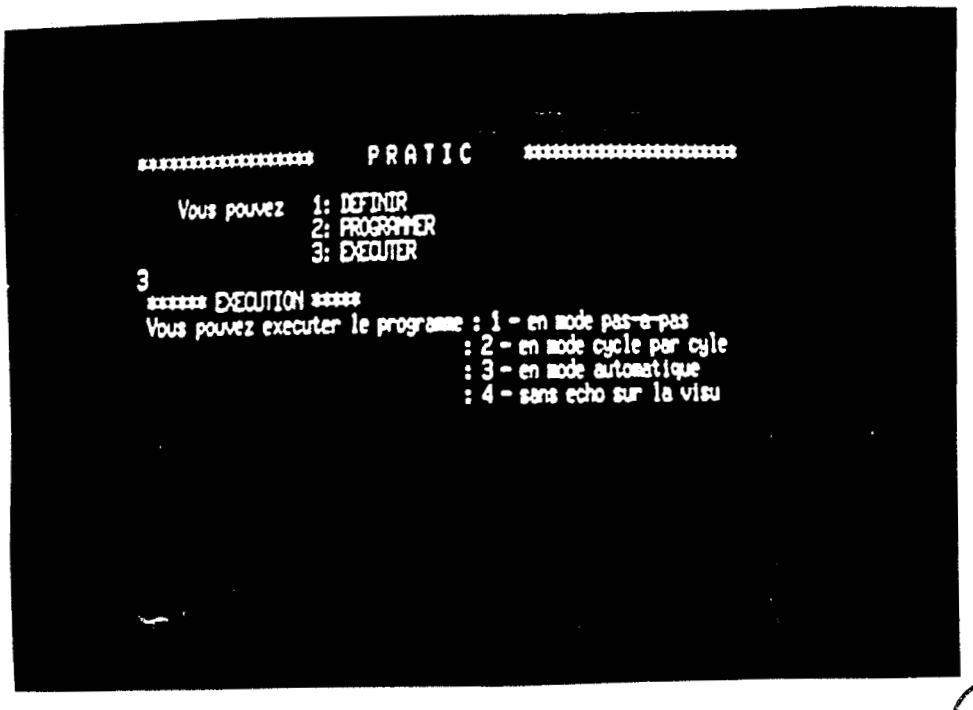


Figure III.9 : Choix proposés par le module d'exécution.

Toutes les descriptions ainsi collectées, tant au niveau de la situation finale (objectif) qu'à celui de la situation initiale, sont alors transmises au générateur de plans d'actions qui détermine les chemins et mouvements adéquats, puis les commandes permettant d'atteindre la situation finale.

Ce générateur de plans d'actions est également écrit en PASCAL MT+. Par ailleurs, de par sa modularité il est accessible à l'opérateur qui peut le modifier pour y inclure de nouvelles règles d'accès au but.

Les informations fournies par le générateur sont ensuite utilisées par le module de commande qui pilote le robot. Il est à noter que tous les traitements nécessaires lors de l'exécution requièrent actuellement environ 500 msec.

III.4 EXEMPLE D'APPLICATION

4.1 INTRODUCTION

Dans sa phase actuelle de développement PRATIC est essentiellement dédié à la réalisation de tâches nécessitant des déplacements soit dans un même plan, soit dans des plans parallèles. Dans ce sens il est particulièrement bien adapté à la programmation des robots cartésiens ou SCARA.

L'exemple développé dans cette section utilise un robot SCEMI 6 axes (seul disponible au moment de l'expérimentation) couplé à un système de vision ROBOTRONICS VISIOMAT, par l'intermédiaire d'un calculateur de supervision du type MICRAL 9050. La console graphique utilisée est une console SECAPA 550 (figure III.10).

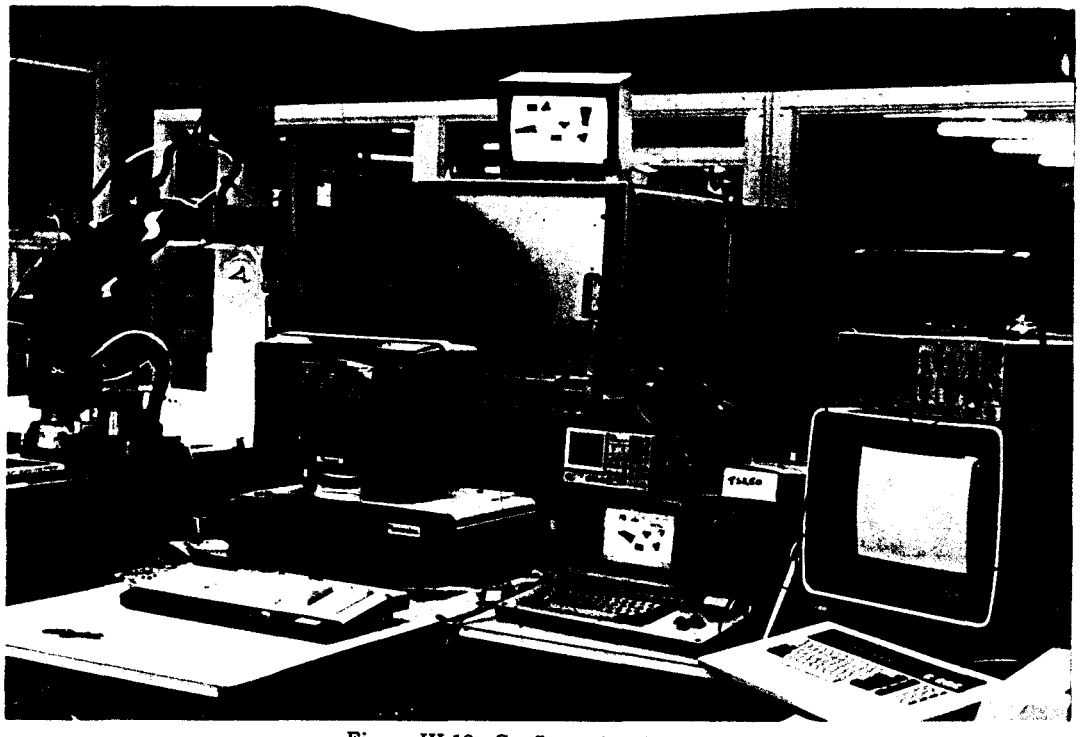


Figure III.10 : Configuration d'expérimentation.

Le but de la manipulation est de montrer les différentes étapes de la programmation et de l'exécution d'un programme.
Ainsi que l'illustre la figure III.11, la collection d'objets à manipuler est constituée de trois types:

- capot
- cellule

- tube

Par ailleurs le préhenseur est une ventouse munie d'un dispositif de détection de présence.

L'objectif à atteindre est de réaliser un arrangement type de ces objets, à partir d'un vrac planaire placé sous la caméra.

Figure III.11 :La collection d'objets à manipuler.

4.2 PROGRAMMATION

La modélisation des objets est réalisée, dans une première étape, en les plaçant un à un sous la caméra, ce qui permet au module de définition de générer la base de connaissance, puis par sélection de l'opérateur, la base de travail.

En mode de programmation interactive, la deuxième étape consiste à sélectionner dans la base de travail les objets qui feront partie de l'arrangement désiré. Pour cela, on utilise les représentations des modèles géométriques sur la console graphique. Plusieurs commandes permettent l'accès

à la description finale des positions des objets à manipuler.

L'opérateur accède en premier niveau à la représentation planaire du domaine accessible par le robot en fonction de l'altitude du préhenseur (Figure III.12).

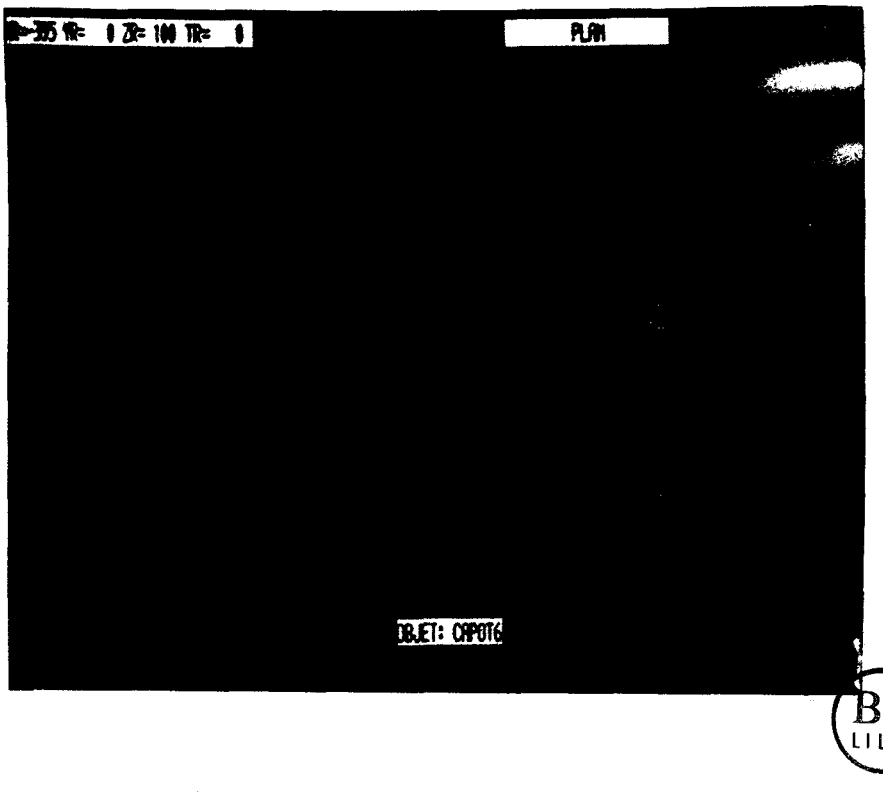


Figure III.12 : Domaine accessible au robot

Une première commande permet ensuite de faire défiler devant une fenêtre les objets de la base de travail et d'en sélectionner un. A partir de cet instant la représentation graphique de l'objet est liée au déplacement d'un curseur. L'opérateur déplace alors dans le plan l'objet choisi. Des commandes spécifiques permettent enfin d'accèder à la programmation de la rotation (Figure III.13) et à la programmation de l'altitude de dépose de l'objet manipulé.

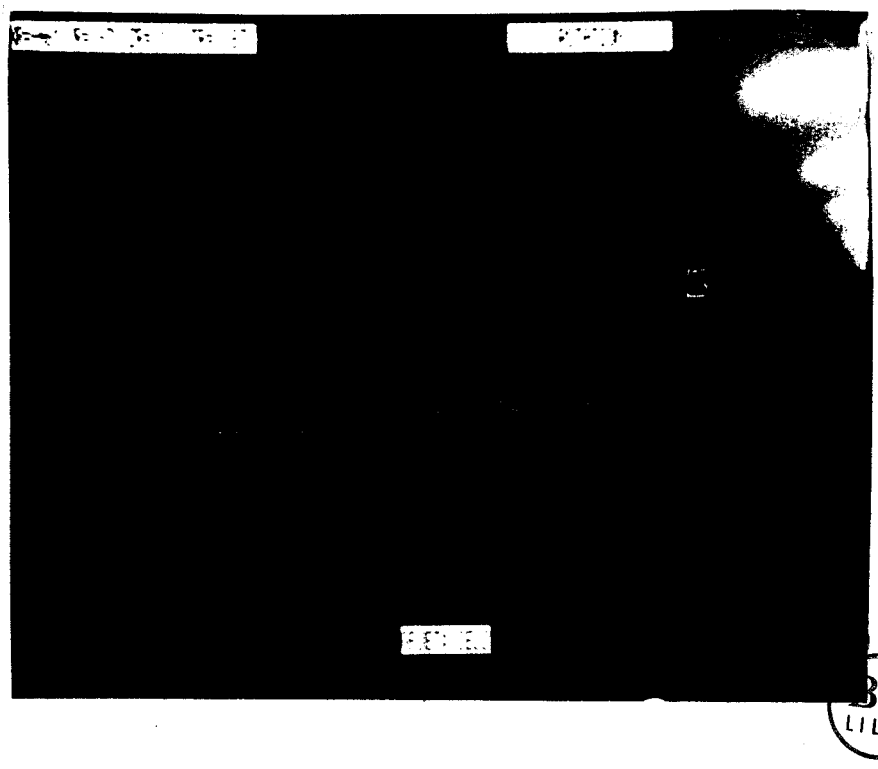


Figure III.13 : Rotation de l'objet à déposer.

Cette procédure est répétée pour chacun des objets appartenant à l'arrangement désiré, de manière à décrire complètement l'objectif final.

Dans le cas d'une programmation avec caméra, les étapes précédentes sont supprimées. En effet il suffit ici de présenter simplement sous la caméra l'arrangement final désiré et de le déplacer globalement dans le domaine atteignable du robot jusqu'à la position (X,Y,Z,Θ) désirée. Il apparait donc que la programmation avec caméra a pour effet de lier entre eux les objets utilisés de la base de travail. Dans ce sens ce type de programmation met directement en oeuvre le concept de liaison forte entre les repères objets.

4.3 AMELIORATION DU DIALOGUE HOMME-MACHINE

Les facilités offertes pour améliorer le dialogue Homme-Machine sont les suivantes:

- indication sur l'écran, à chaque instant, de la position exacte de l'objet manipulé dans le repère robot : affichage X,Y,Z et Θ.

- possibilité de zoom sur la zone en cours de définition. Ce zoom visualise les objets à l'échelle 1 et donc ne fausse pas la perception qu'a l'opérateur de la scène représentée (Figure III.14).

- possibilité de règler la vitesse de déplacement des objets (deux vitesses actuellement disponibles).

- identification de l'objet manipulé par utilisation de couleurs différentes.

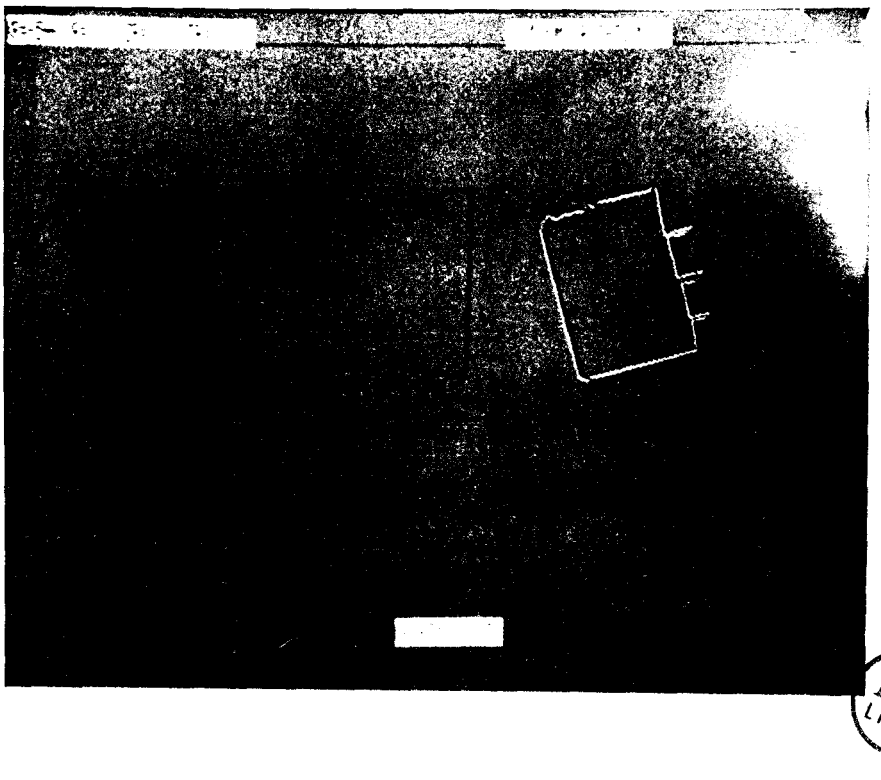


Figure III.14 : Accès au zoom et à la rotation.

III.3 Exécution du programme

Indépendamment du mode d'exécution choisi (pas à pas, cycle par cycle, automatique) le déroulement du programme comporte les étapes suivantes :

- représentation en pointillé de l'arrangement final désiré (figure III.15a, partie droite).
- acquisition et analyse de la situation initiale présente sous la caméra (figure III.15b).
- modélisation et représentation en trait plein de cette situation initiale (figure III.15a, partie gauche).
- réalisation physique de l'arrangement final (figure III.15c).

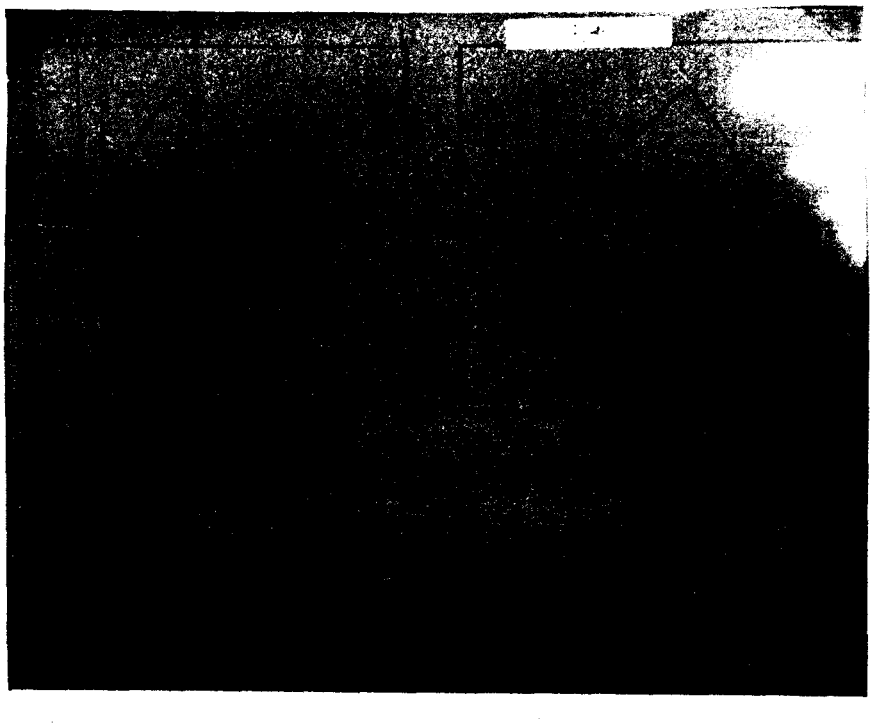


Figure III.15a : Echo sur la console, durant la phase d'exécution.

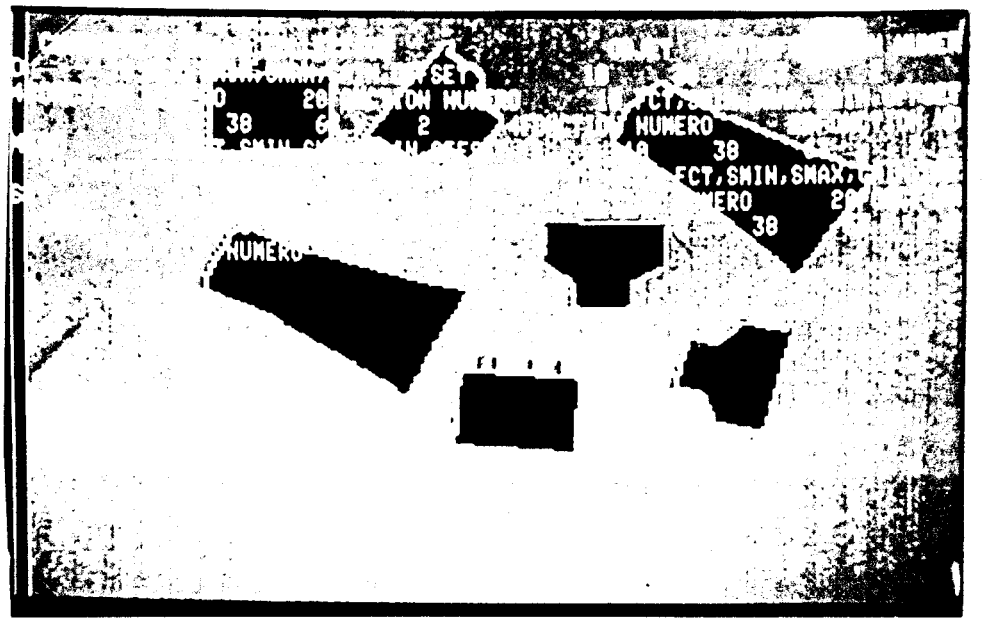


Figure III.15b : acquisition et analyse de la situation initiale.

Figure III.15c : réalisation physique de l'arrangement final.

III.5 CONCLUSIONS ET DISCUSSION

Le langage de programmation de robot proposé dans cet article permet un gain de temps important, notamment dans la phase de développement des programmes. Ce gain de temps est dû aux propriétés suivantes :

- en premier lieu, l'utilisation de représentations visuelles accélère considérablement, chez l'opérateur, le processus de compréhension de l'univers perçu, ce qui permet une description aisée d'objets physiques sans faire appel à l'esprit d'abstraction.

- en second lieu, la syntaxe utilisée est ramenée à sa plus simple expression, ce qui permet un apprentissage rapide du langage.

Toutefois, dans l'état actuel d'avancement des travaux il ne serait pas raisonnable d'envisager la réalisation d'un langage complètement polyvalent résolvant tout type d'application. C'est la raison pour laquelle PRATIC a été développé, dans un premier temps, pour une classe d'applications précise : réalisation de tâches nécessitant des déplacements soit dans un même plan, soit dans des plans parallèles. Cette classe d'applications peut évidemment être mise en oeuvre sur tout type de robot. Cependant, il est clair que les robots les mieux adaptés sont ici les robots de types cartésiens ou SCARA.

En ce qui concerne la précision obtenue, outre la précision dûe au robot lui-même, un bon étalonnage du système de vision a permis de travailler en dessous du millimètre.

Sur la base des concepts développés dans PRATIC, la grande modularité de celui-ci, alliée à l'utilisation de l'excellent moyen de communication

universel que constitue la vision, permet d'envisager une bonne adaptabilité à d'autres classes d'applications.

Enfin il est envisagé, sur le plan matériel, d'acquérir une meilleure transportabilité, par la transposition des logiciels sur calculateurs compatibles IBM-PC. Une telle démarche vise à diminuer les investissements nécessités par une programmation hors ligne et donc de réduire les immobilisations de matériels industriels sur le site, pendant la phase de développement des applications.

CHAPITRE IV

METHODOLOGIE ET DEVELOPPEMENT

IV.1 Introduction

Le système PRATIC présenté au chapitre précédent continue d'être développé. Deux axes principaux ont été approfondis : la structure temps réel, le générateur de plans d'actions.

Ces développements n'intéressent donc qu'en premier lieu le module exécutif. Les améliorations de la programmation seront examinés à la fin de ce chapitre.

L'extension du module exécutif est nécessaire pour permettre l'exécution :

- mono robot, multi préhenseurs

- multi robots, multi préhenseurs

alors que l'exècution mono-robot, mono-préhenseur est réalisée par le logiciel présenté au chapître III.

Avant d'examiner la structure de ce nouvel exécutif, il faut revenir sur le générateur de plans d'actions inclus dans PRATIC.

IV.2 Le générateur de plans d'actions

Le générateur de plans d'actions a pour objet de donner à un robot les moyens de décider lui-même des actions à accomplir en vue de remplir une tâche globale [FAH 74], [SAC 75]. Les générateurs de plans ont l'ambition de permettre la commande du robot en lui spécifiant uniquemment l'objectif à atteindre. Pour donner au robot des tâches à exécuter, l'opérateur n'a plus à se préoccuper de décomposer chaque action élémentaire et de calculer les déplacements et trajectoires, il se contente de déterminer soit graphiquement et de manière interactive la situation finale, soit de présenter l'objectif final souhaité sous la caméra, le générateur de plans d'actions se chargeant du reste.

Pour déterminer l'enchaînement d'actions élémentaires que devra réaliser le robot (le plan d'actions), le générateur de plans a besoin:

- d'un modèle de l'univers à partir duquel l'objectif devra être atteint

- d'un modèle des actions élémentaires qu'il est capable d'effectuer

- d'une définition de l'objectif à atteindre.

La construction du plan s'effectue à partir des modèles acquis dans les phases précédentes :

- modélisation des objets permettant le raisonnement sur les trajectoires et la mise à jour du modèle de l'univers. Pour réduire les temps de calcul les objets sont approximés par un modèle parallélipèdique (L x l x M), l'information L x l étant facilement extraite par le traitement d'images. Tous les objets présents dans la scène sont donc considérés comme autant d'obstacles parallélipèdiques [BRA 82], [BRA 83].

- modélisation du préhenseur permettant d'accéder à la détermination de la prise et dépose automatiques [LAU 81], [MAT 74]. Deux critères ont simplifié cette détermination :

  * les objets se trouvent dans des plans parallèles et le préhenseur ne possède pas les rotations Oy,Ox (robot de type Scara).

  * le préhenseur est une ventouse, ce qui réduit les possibilités de prise. En général c'est le centre de gravité de l'objet qui correspond au point de prise.

Le synoptique de mise en oeuvre du générateur de plans d'actions est donné figure IV.1.

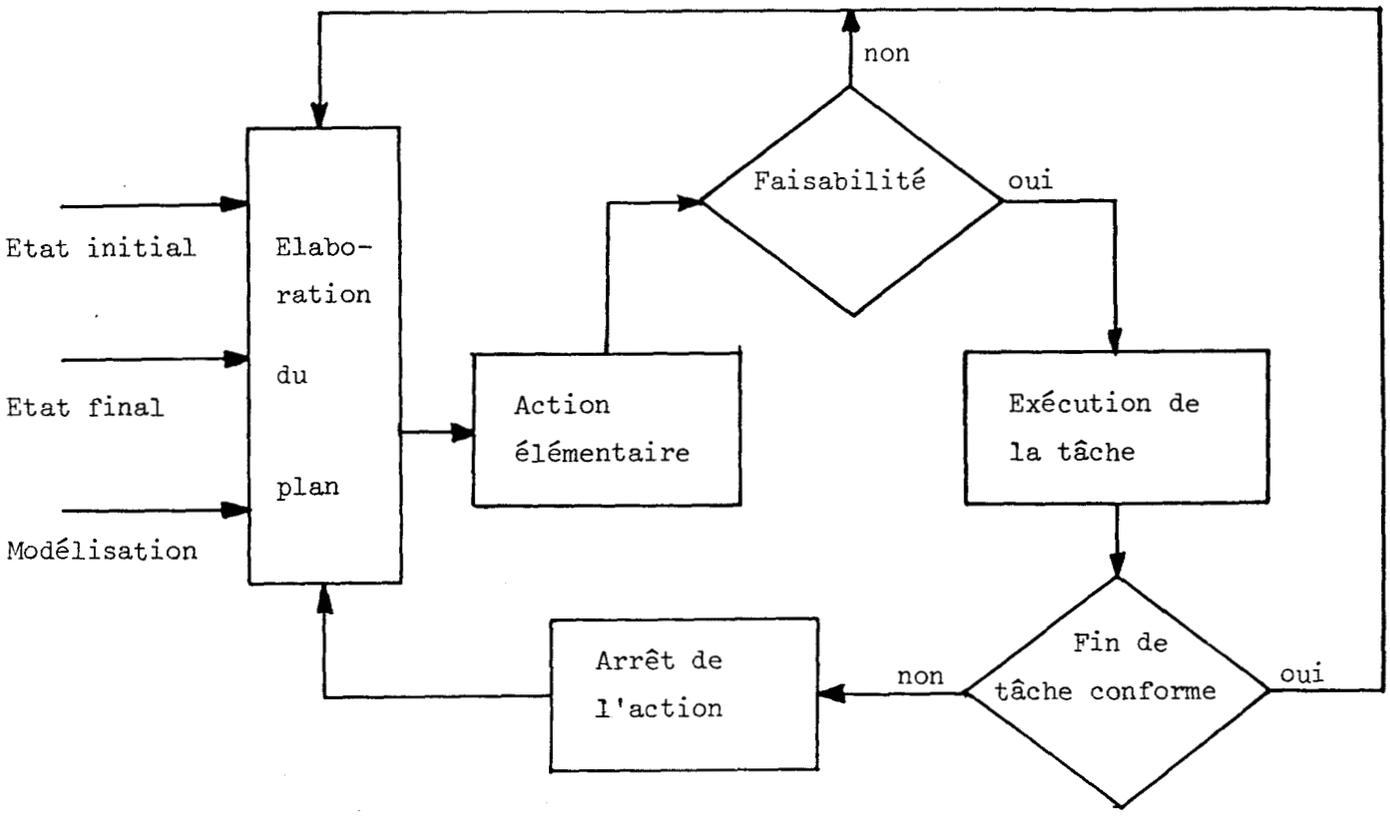


Figure IV.1 : Structure du générateur de plans d'actions.

Le générateur d'actions développé est basé sur une stratégie de mise en relation des fins et des moyens (EPS). Il met en évidence une différence entre la situation initiale et l'objectif à atteindre. Il sélectionne les actions adéquates pour réduire la différence [ITM 83], [FAR 85]. Puis en fonction de la conformité des exécutions il itère le processus pour obtenir la correspondance entre l'objectif et le modèle de l'univers.

La première phase du générateur détermine l'enchaînement des actions à réaliser en s'appuyant sur les critères de progression : vérification que l'action va dans le sens de l'amélioration et que cette action conduit à une solution optimale (cela nécessite d'inclure les critères d'optimalisation tels que chemin le plus court, action la plus rapide). Ensuite cette phase inclut la faisabilité de l'action envisagée en prenant en compte les contraintes de sécurité, les limitations dues aux caractéristiques du robot puis les obstacles dans la trajectoire [KUN 82].

La deuxième phase du générateur s'apparente plus à un contrôleur d'exécution car elle s'assure que les actions effectuées sont conformes au déroulement du plan [BAN 84], [POR 85]. Lorsque des incidents se produisent ou que l'état initial ne permet pas d'atteindre l'objectif final désiré, il est nécessaire de faire appel dans ce cas soit à un logiciel capable d'appréhender la situation et de déterminer les phases qui pourraient mener au but, ce sont les systèmes experts, soit de requérir l'intervention de l'opérateur, c'est cette démarche que PRATIC permet.

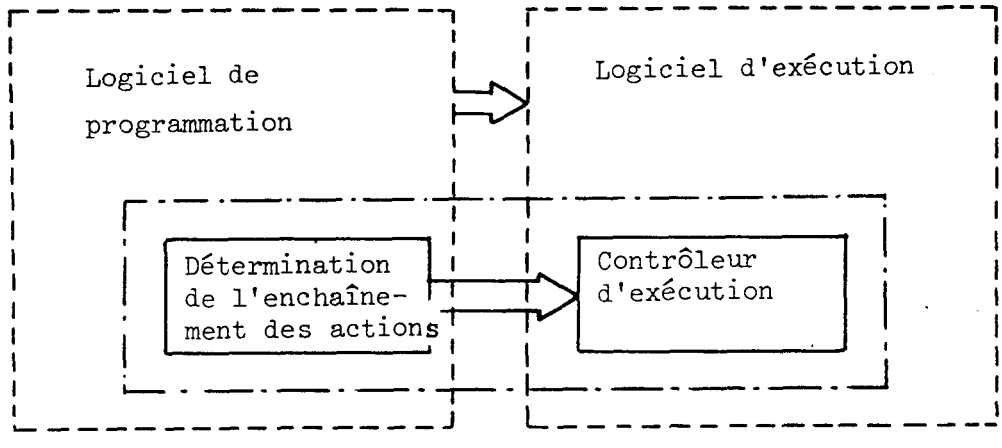


Figure IV.2 : Décomposition du générateur

Il est alors possible de séparer les deux phases (figure IV.2) précédemment décrites : phase de détermination, phase d'exécution. Ainsi on peut inclure la phase de détermination dans le logiciel de programmation tandis que la deuxième phase peut être incluse dans le logiciel d'exécution. Cette méthodologie a été employée dans PRATIC car elle possède plusieurs avantages :

Tout d'abord, l'inclusion de la phase de détermination des actions au niveau programmation permet de prendre en compte l'enchaînement des opérations que l'opérateur effectue pour décrire l'objectif final (description graphique). La connaissance de ces informations simplifie le générateur car il peut alors s'appuyer sur le mode opératoire décrit par l'opérateur. Certes le générateur ne peut à ce moment déterminer exactement le plan car il ne connaît pas l'état initial mais il fournit l'enchaînement des actions à effectuer ce qui est l'essentiel à ce moment.

Ensuite, cette inclusion réduit la taille de l'exécutif et par conséquent contribue à l'amélioration des temps d'exécution.

Par ailleurs, la partie du générateur incluse dans l'exécutif détermine les critères de faisabilité pour l'état initial et contrôle l'exécution. Il est évident que lors d'incidents l'appel au générateur total est alors requis, puis suivant ses conclusions, l'appel de l'opérateur est demandé ou non.

Enfin, cette séparation permet d'envisager simplement la programmation des exécutions multi-robots, multi-préhenseurs dans le cas de déplacements d'objets (cf figure IV.3). La modularité de PRATIC est encore ici essentielle car seul le module du générateur déterminant l'enchaînement des tâches inclus dans le logiciel de programmation a été étendu (cf figure IV.2), les autres modules (en particulier le module d'exécution) restent inchangés.

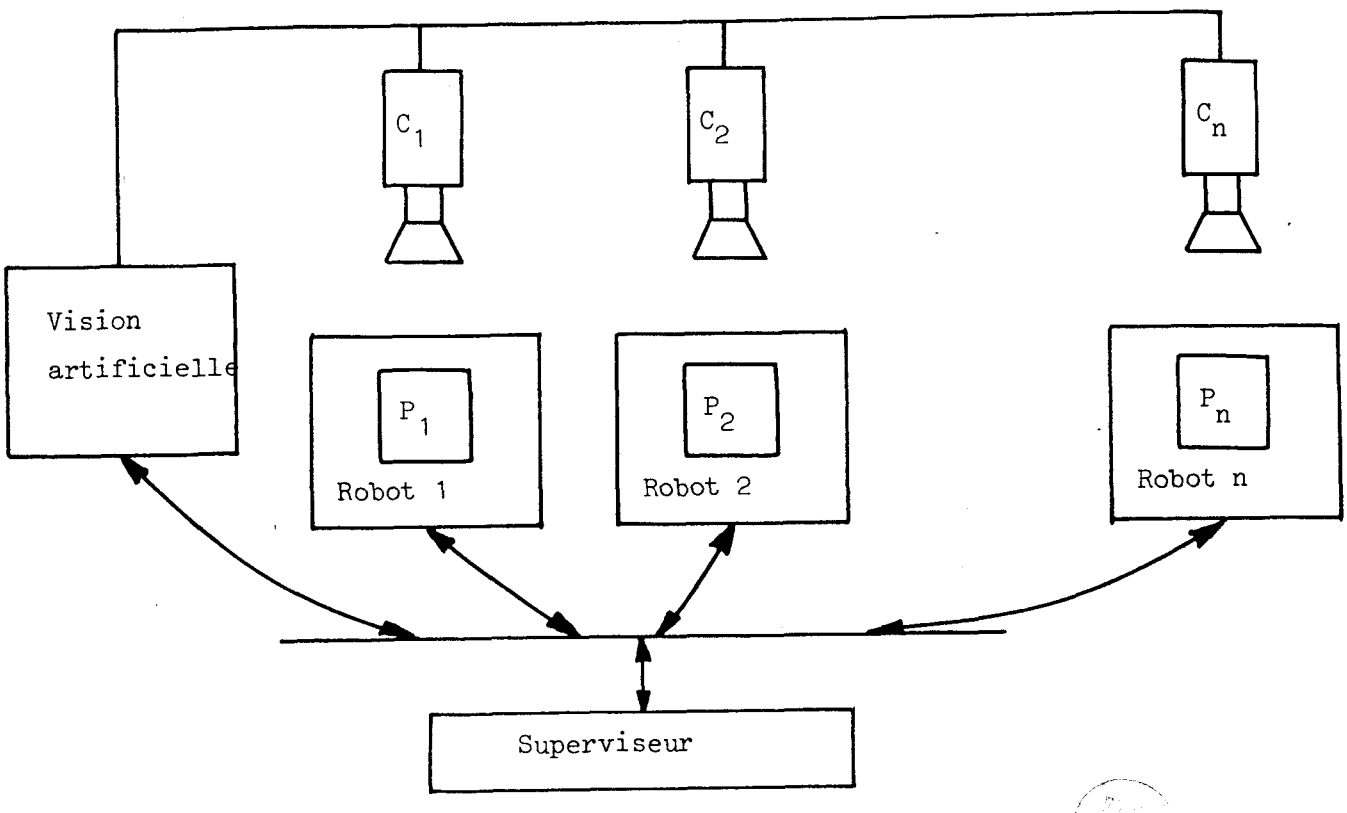


Figure IV.3 : Exècution multi-robot : architecture materielle

Lors de la phase de programmation la connaissance des ressources (robots, préhenseurs) permet à partir de la détermination de l'enchaînement des actions de répartir les différentes tâches en fonction des différentes possibilités d'exécution. Les critères d'affectation des tâches s'apparentent à ceux de la gestion de production. PRATIC prend en compte pour cette affectation la répartition équitable

du nombre d'objets devant être manipulés par robot. Ce critère complété par l'évaluation du temps d'exécution de chaque action élémentaire permet d'optimiser les temps d'occupation de chaque robot et de déteminer ainsi le temps de cycle minimum pour réaliser l'objectif final.

Il est à noter que durant cette phase, si les ressources sont multi-préhenseurs, la minimisation des temps de cycle inclut la détermination des actions de dépose conduisant à un parcours minimum. L'utilisation de robots multi-préhenseurs s'avère extrèmement performante dans les opérations de vissages multiples lors d'un assemblage (figure IV.4).

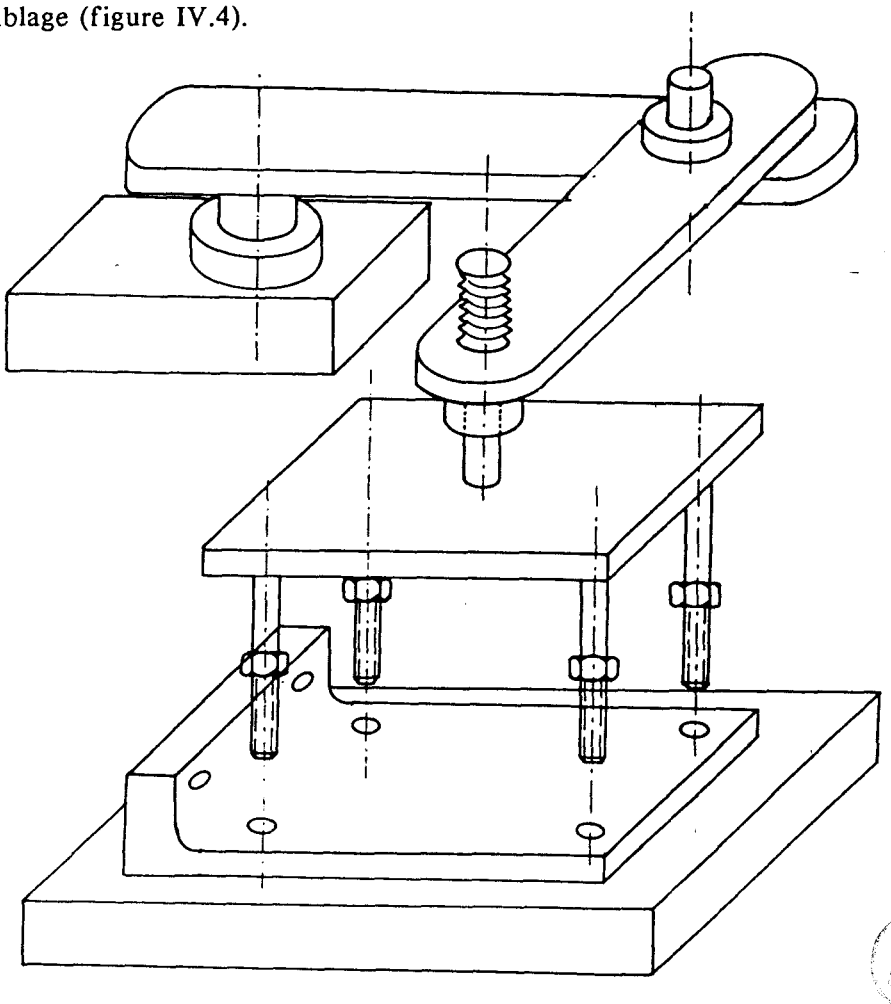


figure IV.4 : Exemple d'application multi-préhenseur

De même, on remarque que ce type de préhenseur réduit non seulement les trajectoires nécessaires pour effectuer la prise des différentes pièces mais également réduit les temps d'exécution par la possibilité d'opérations simultanées.

La méthodologie développée ne considère plus la phase de programmation en tant que telle mais lui adjoint tout le logiciel nécessaire à la préparation de l'exécutif. Il est dès lors possible de contrôler les actions de l'opérateur durant la

phase de programmation. Cette possibilité n'a pas été développée car elle oblige le programmeur à spécifier avant la programmation les ressources qui seront utilisées et donc la programmation résultante dépend dans ce cas du mode d'exécution envisagé. Il est plus judicieux de décrire l'objectif final sans références aux ressources car la modélisation ainsi acquise n'est plus à refaire lors d'un changement de machine. La figure IV.5 montre l'organigramme développé.

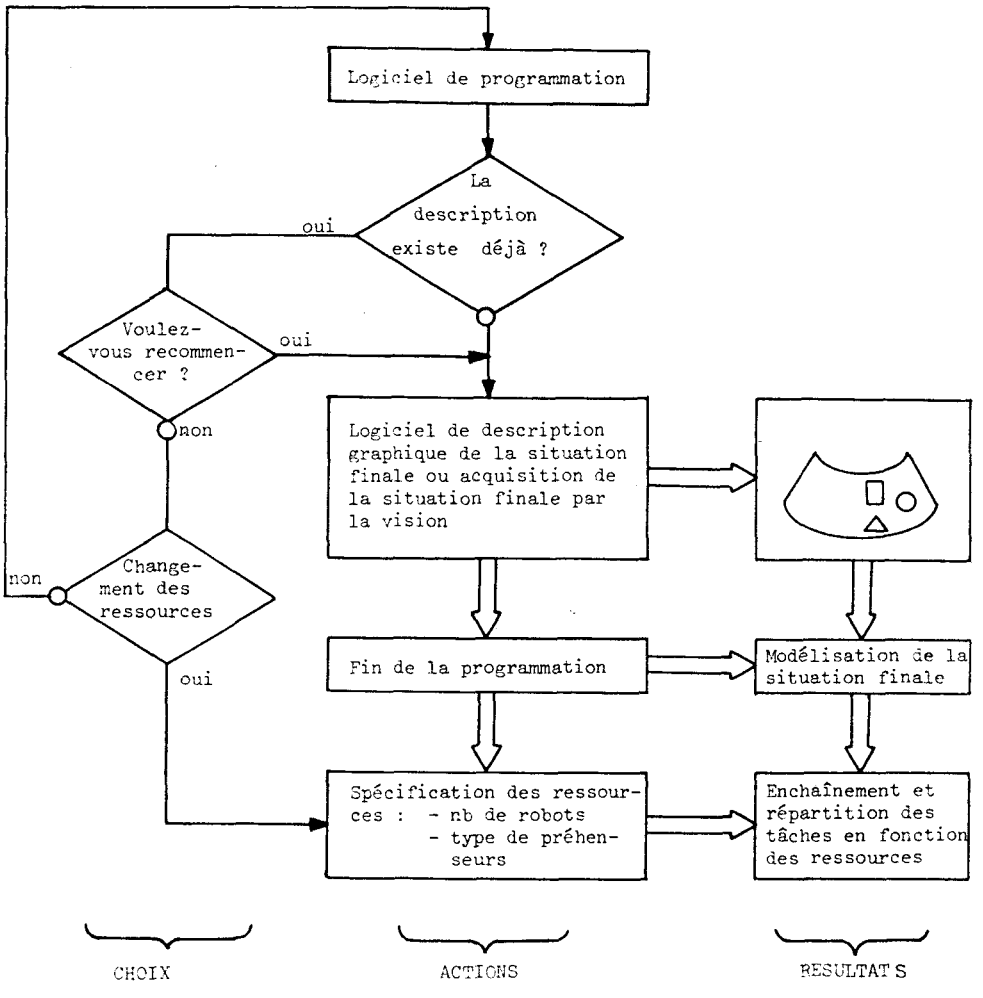


Figure IV.5 : Extension du module de programmation

On constate donc que la phase de programmation fournit outre une modélisation de l'objectif à atteindre, la modélisation de l'exécutif proprement dit. Cette modélisation de l'exécutif se situe au niveau effecteur dont les primitives de commandes (prendre, déplacer, déposer, activer capteurs) sont arrangées dans l'ordre d'exécution des différentes tâches et incluent la synchronisation entre les différentes ressources dans le cas multi-robots.

IV.3 Structure de l'exécutif

Pour permettre l'exécution des programmes enregistrés il est nécessaire d'avoir une organisation temps réel. Cette structure permet non seulement au contrôleur d'exécution de connaître l'état actuel du monde mais également :

- de définir un chien de garde pour la surveillance des tâches

- de gérer les interruptions de priorités égales ou différentes

- d'utiliser au maximum les différentes ressources disponibles.

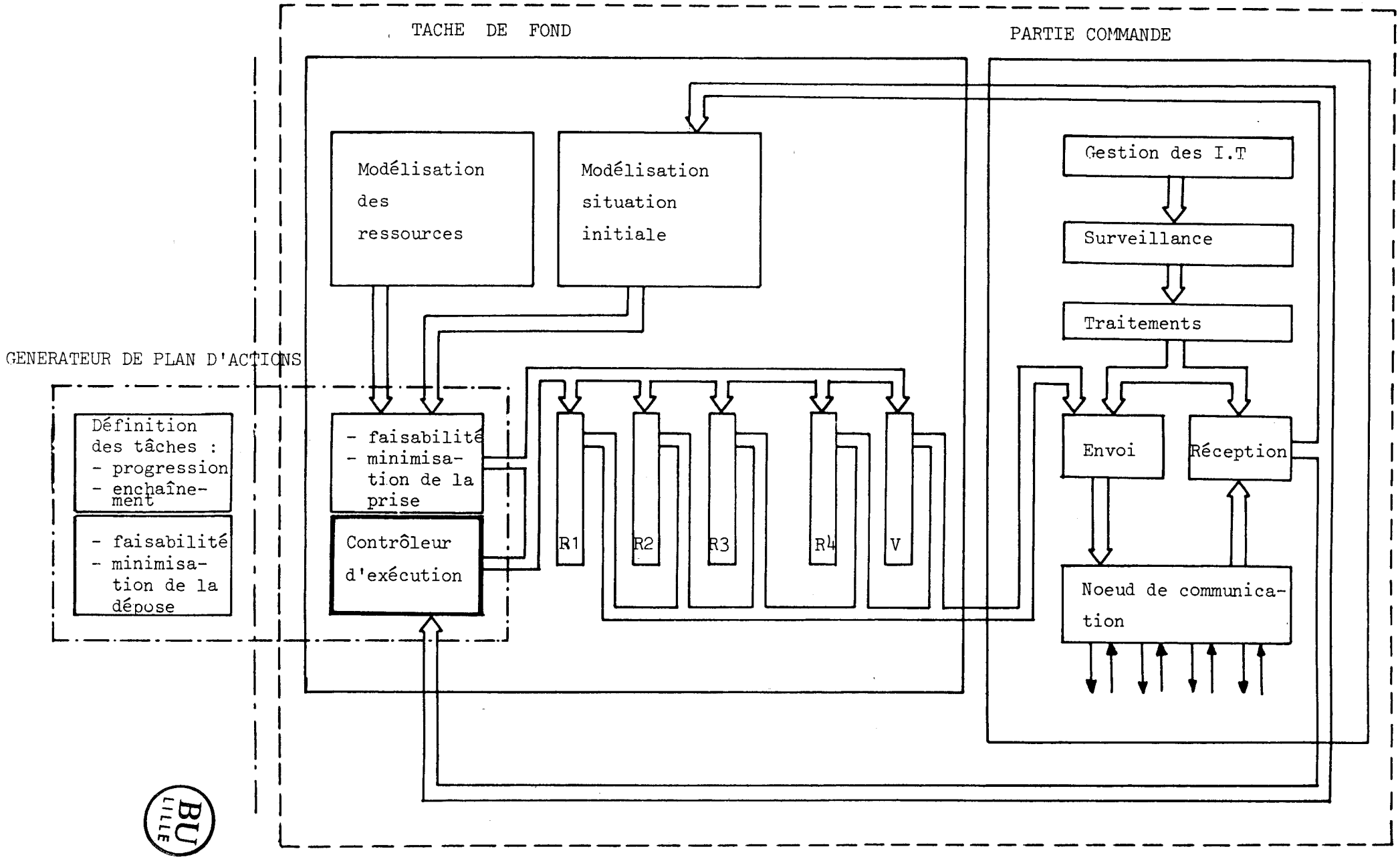
De plus, l'utilisation du système de vision lors de l'exécutif nécessite à elle seule une architecture temps réel [BON 83], [GIL 77], [CAL 82]. En effet, lors d'une réalisation industrielle il ne serait pas pensable d'analyser la scène puis d'effectuer ensuite la commande. L'exécutif requiert donc l'exécution parallèle des deux fonctions bien distinctes que sont l'acquisition de la situation initiale et la commande des différentes ressources. On voit donc apparaître la nécessité de synchronisation entre ces deux modules et la hiérarchisation de la commande des ressources. Il est évident que le module d'acquisition de la situation initiale doit être en tache de fond du superviseur, alors que la partie commande est activée lors de la gestion des interruptions (cf figure IV.6).

PROGRAMMATION

EXECUTIF

TACHE DE FOND

PARTIE COMMANDE



IV.4 Conclusions

Après avoir défini au chapitre III le cahier des charges du système de vision artificielle, on peut définir celui du ou des robots. Le langage PRATIC détermine en phase finale les actions élémentaires des robots. Celles-ci sont principalement :

- le déplacement en x,y,z,Θz
- la prise
- la dépose.

Ces actions sont paramètrables pour transmettre les informations de position (x,y,z,Θz) mais également le type de l'effecteur concerné par ces actions. Il a été également nécessaire de développer plusieurs autres fonctions pour que le robot soit véritablement commandable et pour simplifier au maximum l'interaction du superviseur dans le déroulement des actions. C'est alors dans ce but qu'ont été développées les fonctions de gestion des capteurs et actionneurs incluant :

- l'affection d'une sortie
- le test d'une entrée
- l'affection d'une sortie conditionnée à l'état d'une entrée.

Des fonctions spécifiques de contrôle sont également nécessaires telles que :

- l'arrêt d'un mouvement
- la reprise du mouvement
- l'arrêt total du fonctionnement du robot en cas d'anomalies graves.

En conclusion tous les robots étant capables d'assurer l'exécution des fonctions précédemment décrites peuvent être commandés par le langage PRATIC. On remarque donc que les robots SCARA ne sont pas les seuls commandables.

IV.5 Développement envisagé

Le logiciel présenté procure un gain de temps très important et une facilité accrue lors de la programmation mais est néanmoins limité dans la manipulation et la reconnaissance des objets dans des plans. Ces limitations sont principalement dues à l'absence d'information de hauteur donnée par la vision artificielle.

IV.5.1 Modélisation

Des développements peuvent être envisagés à tous les niveaux, mais le niveau le plus important est celui de la modélisation car il doit posséder toutes les informations nécessaires à la manipulation [KIR 85].

Or, si l'on prend, à titre d'exemple, une base de données d'un logiciel de modélisation plane, il nous apparaît qu'il s'agit d'un modèle de visualisation utilisé uniquement parce qu'il permet un certain nombre d'opérations au niveau de la visualisation et de l'interaction. Il ne peut être considéré que comme un bon modèle pour des objets polyèdriques. Un modèle plus riche est celui de l'arbre de construction, bien qu'il ne s'agissent encore que d'un modèle fondé sur la géométrie des objets. Il nous semble comme nous l'avons développé dans PRATIC, que l'un des aspects essentiels à conserver est la logique employée par le concepteur pour le construire.

Prenons l'exemple simple où intervient un assemblage (cf figure IV.7).

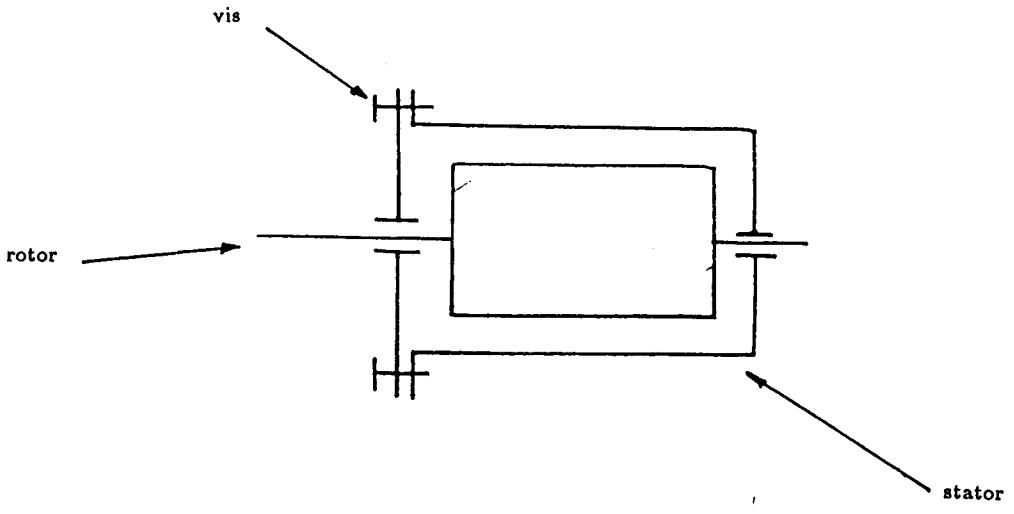


figure IV.7 : Exemple d'assemblage

Cette représentation qui indique bien la géométrie de l'ensemble ne permet pas l'assemblage lui-même, puisqu'il sera impossible d'insérer le rotor une fois le couvercle posé et les vis mises. La modélisation de l'assemblage doit être aussi partie intégrante du modèle. Le fait important est que le concepteur, lors de la création de l'assemblage, a effectué la chronologie des opérations. Il est donc essentiel que dès ce moment les modèles des objets s'enrichissent de la façon dont ils ont été créés. De plus le concepteur faisant obligatoirement référence aux cotations métriques, il est important de pouvoir en disposer lors de la manipulation. La primitive INSERER en sera grandement améliorée. On peut donc dire que la phase de modélisation sera encore plus riche si elle peut être prise en compte dès la phase de conception du produit (par exemple sur un système de CAO) [GAR 83].

IV.5.2 .Programmation

Les développements envisagés au niveau de la programmation s'attachent surtout à la programmation de l'objectif. Les développements des

systèmes de vision artificielle ne sont pas prêts de pouvoir déterminer l'inclusion des pièces. Il n'est donc pas raisonnable dans l'état actuel d'envisager l'analyse d'un moteur assemblé par la vision artificielle pour en déduire les éléments le constituant. Les développements pouvant améliorer la programmation doivent donc se situer au niveau de la facilité de description, PRATIC a entamé cette facilité.

Mais dès à présent, un système de vision artificielle donnant l'altitude de l'objet serait très apprécié, l'opérateur ne raisonnant plus que sur la position X,Y et Θ, la côte Z pouvant être déterminée automatiquement par le logiciel soit pour la préhension soit pour la dépose. En effet losqu'il y a superposition d'objets ou même assemblage, on peut calculer assez facilement la côte Z de dépose de la face de contact. C'est sous cet aspect que PRATIC peut actuellement apporter une contribution trés importante.

IV.5.3 Exécutif

Les champs d'amélioration de l'exécutif sont bien sûr le générateur de plans d'actions et l'exécutif temps réel. Le problème le plus important que doit résoudre le générateur est le problème de la préhension. Comment prendre un objet pour le déposer dans telle position et avec tel préhenseur ? Il semble intéressant dans ce but de développer un système expert spécialisé dans la préhension. Celui-ci devra fournir au générateur, par exemple l'orientation de l'outil terminal, l'ouverture des mors et les points de préhension possibles.

CONCLUSIONS GENERALES

Nous avons créé le logiciel PRATIC afin d'aider l'opérateur à mettre au point l'ensemble des opérations que devra suivre le robot. En effet on mesure chaque jour combien il est difficile, même en disposant d'un langage évolué de mettre en place une manipulation à partir d'un robot, cette opération devient encore plus difficile lorsqu'il s'agit de plusieurs robots. Cette difficulté vient en grande partie de l'insuffisance de modélisation du robot que nous cherchons à pallier en utilisant des logiciels faisant appel en particulier aux techniques les plus élaborées que sont la visualisation graphique et les générateurs de plans d'actions.

Ces deux techniques permettent une programmation hors ligne avec l'élaboration d'une solution optimale des déplacements dans l'espace robot. L'apparition d'évènements imprévus, lors de l'exécution ou de divergences dans la réalisation de ce qui a été programmé, est prise en compte par le générateur de plan d'actions qui apporte une amélioration de la solution en rapport avec la conformité du chemin initialement donné.

Les fonctions graphiques de PRATIC sont spécifiques à la désignation de l'objectif désiré. Les atouts sont :

- sa portabilité

- son faible encombrement lui permettant de fonctionner sur micro-ordinateur.

La fonction génération de programme de commande, de programmation et de définition des objets, s'appuie sur des entités modulaires qui constituent un ensemble de primitives organisées en un système ouvert et pour l'instant suffisant.

La fonction générateur de plans d'actions est très prometteuse à travers les tests que nous avons réalisés. Néanmoins, elle peut être aisément complétée par une fonction d'intelligence artificielle et reste, à cet état, du domaine de la recherche.

PRATIC n'est déja plus un logiciel de laboratoire car il a été implanté sur un robot industriel de conditionnement.

Les développements peuvent se poursuivre vers trois directions :

- développement de logiciel graphique en complétant la description de la tâche (extension vers le 3D)

- incorporation d'un logiciel d'intelligence artificielle

- définition d'un module résolvant tous les problèmes de préhensions.

Cependant, en dépit de toutes les évolutions possibles, notre objectif majeur reste que PRATIC continue à fonctionner sur micro-ordinateur.

BIBLIOGRAPHIE CHAPITRE 1

[ASE 79]
ASEA Inc. :
" Industrial robot system ",
ASEA AB Sweden, Rep YB 110-310 E, 1979.

[BAE 79]
A.BAER, C. EASTMAN, M. HENRION :
" Geometric modeling : a survey ",
Computer Aided Design, Vol 11, n 5, pp 253-273, sept 1979.

[BAT 81]
M. BATHOR, A. SIEGLER :
" Graphic simulation for robot programming ",
Computer and Automation Institute, Budapest HUNGARY, 1980.

[BON 82]
S. BONNER, KG. SHIN :
" A comparative study of robot languages ",
I.E.E.E Computer pp 82-96 dec. 1982.

[BRA 83]
M. BRADY :
" Task planning in robot motion : planning and control",
MIT PRESS Cambridge, 1983.

[BRA 78]
I. BRAID :
" New directions in geometric modeling ",
CAM Workshop on geometric modeling, Arlington TEXAS, 1978.

[DAR 75]
J.A. DARRINGER, M.W. BLASSEN :
" MAPPLE, a hight level language for research in mechanical assembly ",
IBM, Tech. rep. RC 5606, sept 1975.

[DAV 79]
J. DAVID :
" Analyse des procédés d'apprentissage d'un robot à peindre ",
Séminaire international sur les langages et méthodes de programmation des robots industriels ", INRIA, Rocquencourt FRANCE, 1979.

[DON 80]
Mc DONNEL DOUGLAS Inc. :
" Robotic system for aerospace batch manufacturing ",
Mc DONNEL DOUGLAS Inc, fev. 1980.

[ERN 61]
H.A. ERNST :
" A computer controlled mechanical hand ",
Thesis of Massachusett institute of technologie, Cambridge 1961.

[FAL 80]
D.FALEK, M. PARENT :
" An evolutive language for an intelligent robot ",
Industrial Robot, sept 1980.

[FEL 71]
A. FELHMAN
" The Stanford Hand Eye project ",
IJCIA, London ENGLAND, sep 1971.

[FIN 74]
R. FINKEL :
" AL, a programming system for automation ",
Artifical Intelligence, Stanford, Rep AIM 243, nov 1974.

[FRA 82]
J.W. FRANKLIN, G.J. VANDERBRUG :
" Programming vision and robotic system with RAIL ",
SME Robot VI, pp 392-406, march 1982.

[GIN 79]
G. GINI, D. GUISE
" Introducing software system in industrial robot ",
Symposium on industrial robot, Washington, march 1979.

[HEG 83]
W.B. HEGINBOTHAN, G. GATEHOUSE :
" Interactive computer-aided design for robot implementation ",
Rev. Tijdschrift, Vom 26, n 2, 1983.

[HOL 77]
H.R. HOLT :
" Robot decision making ",
CINCINATTI-MILACRON Inc., Rep MS 77-751, 1977.

[LAT 79]
J.C. LATOMBE :
" Une analyse structurée d'outils de programmation pour la robotique industrielle ",
cf [DAV 79].

[LAT 81]
J.C. LATOMBE, E. MAZER :
" LM, a hight level language for controlling assembly robot ",
Symposium on industrial robot, Tokyo JAPAN, oct. 1981.

[LAT 82]
J.C. LATOMBE :
"Survey of advance general purpose sofware for robot manipulators ",
IMAG Grenoble, RR n 330 ,1982.

[LAU 82]
C. LAUGIER :
" LISP 3D, logiciel pour la manipulation et la visualisation de scènes tridimensionnelles ",
IMAG Grenoble, RR n 328, 1982.

[LAU 83]
C. LAUGIER :
" La programmation des robots : expression textuelle et expression graphique ",
IMAG Grenoble, RR n 387, mars 1983.

[LIB 77]
L.I. LIBERMAN, M.A. WESLEY :
"AUTOPASS, an automatic programming system for computer mechanical assembly ",
IBM, Journal of research, Vol 21, pp 321-333, 1974.

[LIE 82]
A. LIEGEOIS :
" Développement d'un système de C.A.O et de simulation de robots manipulateurs ",
Première journée ARA, Poitiers 1982.

[LOZ 76]
T. LOZANO PEREZ :
" The design of a mechanical assembly system ",
Artificial Intelligence , MIT, tech rep n 397, 1976.

[LOZ 79]
T. LOZANO PEREZ,M.A. WESLEY :
" An algorithm for planning collision free path among polyhedral obstacles ",
ACM, Vol 22, n 10, pp 560-570, 1979.

[LOZ 83]
T. LOZANO PEREZ :
" Robot programming ",
Proceedings of I.E.E.E, Vol.71, n 7, pp 821-840, july 1983.

[MAS 81]
M.T. MASON :
" Compliance and force control for computer controlled manipulators ",
I.E.E.E Trans Sys .,Vol 11, n 6, pp 418-433, june 1981.

[MAZ 82]
E. MAZER :
" LM-GEO, geometric programming of assembly robot ",
IMAG Grenoble, RR n 296, mars 1982.

[MIR 82]
J.F. MIRIBEL, E. MAZER :
" Manuel de réfèrence LM ",
IMAG Grenoble, 1982.

[PAU 77]
R.P. PAUL :
" WAVE : A model based language for manipulator control ",
Industrial Robot, March 1977.

[PAR 72]
W.I. PARK :
" Minicomputer software organisation for control of industrial robot ",
Joint Automatic Control Conference, San Fransisco, nov 1972.

[POP 78]
R.J. POPPLESTONE :
" RAPT, a language for describing assemblies ",
The Industrial Robot, july 1978.

[RUO 79]
C.F. RUOFF :
" TEACH, a concurrent robot control language ",
Proc. I.E.E.E Compsac, Chicago, pp 442-445, 1979.

[SAL 78]
M. SALMON :
" SIGLA, the Olivetti SIGMA robot programming language",
Symposium of industrial robot, Stuttgart, West-Germany, june 1978.

[SIL 73]
D. SILVER :
" The litter robot system ",
M.I.T. Artificial intelligence, Stanford, Rep AIM 243, Nov. 1974.

[TAK 79]
K. TAKASE, R.P. PAUL, E.J. BERG :
" A structured approach to robot programming and teaching ",
Proc. I.E.E.E. Compsac, Chicago, nov. 1979.

[TAY 76]
R.H. TAYLOR :
" The synthesis of manipulator control programs from task level specifications ",
Artificial Intelligence Stanford, Rep AIM 282, july 1976.

[TAY 82]
R.H. TAYLOR, P.D. SUMMER, J.M. MEYER :
" AML, a manufacturing language ",
Robotics res, Vol 1, n 3, 1982.

[UNI 79]
UNIMATION Inc. :
" VAL, an industrial robot programming and control system ",
cf [DAV 79].

[WAT 70]
G.S. WATKINS
" A real time visible surface algorithm ",
Uni. of UTAH, UTECCS 70101, 1970.

[WHI 82]
D.E. WITHNEY :
" The mathematics of coordinated control of arms and manipulators ",
Jour. of dynamics systems, measurement and control, pp 303-309, dec 1982.

[WIL 75]
P.M. WILL, D.D. GROSSMAN :
" An experimental system for controlled mechanical assembly ",
I.E.E.E. Trans Computer, Vol C-24, n 9, pp 879-888, 1975.

BIBLIOGRAPHIE CHAPITRE 2

[AYA 83]
N. AYACHE :
" un système de vision bidimensionnelle en robotique industrielle ",
Thèse de l'Univ. PARIS-SUD, 1983.

[BHA 81]
B. BHANU :
" Shape matching and image segmentation using stochastic labelling ",
Thesis of Univ. OF SOUTH CALIFORNIA, 1981.

[BIE 84]
E. BIENFAIT :
" Vision assistée par ordinateur des trajectoires d'un robot ",
Centre d'automatique, Lille, juin 1984.

[BOL 82]
R.C. BOLLES, R.A. CAIN :
" Recognising and locating partially visible objects ",
Robot Vision, Springer Verlag, pp 43-82, 1982.

[BOU 82]
P.BOUTIN :
" Ergonomie et systèmes complexes : la place de l'opérateur humain ",
Microtechnique, n 11, pp 11-13, jan 1982.

[BRI 70]
C.C. BRICE, C.L. FENNEMA :
" Scene analysis using regions ",
SRI Artificial intelligence, Vol 1, 1970.

[DUD 72]
R.O. DUDDA :
" Use of hough transformation to detect lines and curves pictures ",
ASS computer, Vol 15, pp 11-15, jan 1972.

[FAU 84]
O.D. FAUGERAS :
" Vision par ordinateur en robotique ",
1er Colloque image, Biarritz FRANCE, mai 1984.

[FRE 74]
H.FREEMAN :
" Computer processing of line drawing images ",
ACM Computing Surveys, Vol 6, n 1, march 1974.

[GAR 83]
Y. GARDAN :
" Systèmes de CFAO ",
Edition Hermes, mars 1983.

[GIL 78]
W.K. GILOI :
" Interactive computer graphic : data, structures, algorithms, languages ",
Edition Prentice Hall, 1978.

[JUV 83]
D. JUVIN :
" Contribution à la reconnaissance automatique des images appliquées à la robotique ",
Thèse de l'Univ. Paris-Sud, 1983.

[KRZ 84]
T. KRZEWINA :
" La normalisation graphique et les implications de GKS ",
Electronique Idustrielle, n 65,pp 33-37, fev 1984.

[LAN 78]
C. LANTEJOUL :
" La squeletisation et son application aux mesures topologiques des mosaïques polycristallines ",
Thèse de l'Ecole des Mines, Paris, 1978.

[LUX 84]
A. LUX, V. SOUVIGNIER :
" PVV: un système de vision appliquant une stratégie de prédiction vérification ",
4ème Congrès de recon. des formes et I.A., Paris, pp 223-234, 1984.

[MAR 76]
A. MARTELLI :
" An application of heuristic search method to edge and contour detection ",
ASS. computer, Vol 19, march 1976.

[MAT 67]
G. MATHERON :
" Eléments pour une théorie des milieux poreux ",
Edition Masson, Paris, 1967.

[MEY 79]
F. MEYER :
" Cythologie quantitative et morphologie mathématique ",
Thèse de l'Ecole des Mines, Paris, 1979.

[MIR 84]
MINISTERE DE L'INDUSTRIE ET DE LA RECHERCHE :
" Système de vision et reconnaissance de formes ",
14 th Symposium of industrial Robot, Detroit, 1984.

[MOR 76]
P. MORVAN, M. LUCAS :
" Introduction à l'infographie interactive ",
Edition Larrousse , 1976.

[MYE 82]
W. MYERS :
" An industrial perspective on solid modeling ",
I.E.E.E, CG&A, pp 86-97,1982.

[NEW 79]
W. NEWMAN, R.F. SPROUL :
" Principe of interactive computer graphic ",
Mc Graw Hill, New York, 1979.

[PAV 77]
T. PAVLIDIS :
" *Structural pattern recognition* ",
Springer Verlag, Berlin, 1977.

[PAV 78]
T.PAVLIDIS :
" *A review of algorithms for shape analysis* ",
Computer graphics and image processing, pp 243-258, 1978.

[PER 78]
W.A. PERKINS :
" *A model-based vision systemfor industrial parts* ",
I.E.E.E Trans. on Computer, Vol C.27, n 2, pp 126-143, 1978.

[REQ 82]
A. REQUICHA, H.B. VOELCKER :
" *Solid modeling : an historical summary and contempory assessment* ",
I.E.E.E, CG&A, march 1982.

[ROS 81]
A. ROSENDFELD, R.C. SMITH:
" *Tresholding using relaxation* ",
I.E.E.E, Vol PAMI n 5, pp 598-606, 1981.

[ROS 82]
A. ROSENFELD, A. KAK :
" *Digital picture processing* ",
Academic Press, 1982.

[SER 82]
J. SERRA :
" *Mathematical morphology and image analysis* ",
Academic Press, 1982.

[VAS 82]
C. VASSEUR :
" *La notion d'évènement dans les systèmes dynamiques* ",
USTL Lille France, 1982.

[VUL 83]
J.L. VULDY :
" *Graphisme 3D* ",
Edition Eyrolles, 1983.

[WES 80]
M.A. WESLEY :
" *Construction and use of geometric models* ",
Springer-Verlag, New York, pp 79-136, 1980.

[ZIM 83]
N.J. ZIMMERMAN :
" *Overview of industrial vision system* ",
D.E.B. Publishers, Netherlands, 1983.

BIBLIOGRAPHIE CHAPITRE 4

[BAN 84]
J. BANGRATZ, C. PORQUET, M. DESVIGNES, F. CUOZZO :
" Manipulation d'assemblage exècutée par un robot de troisième génération ",
GERSIC, Caen France, 1984.

[BON 83]
J.C. BONIN, A. HAURAT, M.C. THOMAS :
" LMAC, un système de programmation modulaire pour la robotique ",
Congrès AFCET, Besançon France, 1982.

[BRA 82]
J.M. BRADY :
" Part description and acquisition using vision ",
Proc SPIE, may 1982.

[BRA 83]
J.M. BRADY :
" Trajectory planning ",
Eds Mitpress, Cambridge, 1983.

[CAL 82]
J.P. CALVEZ :
" Une méthodologie de conception des systèmes multi-micro-ordinateurs pour les applications en temps réel ",
Thèse , Nantes France, 1982.

[FAH 74]
S.E. FAHLMAN :
" A planning system for robot construction tasks ",
Artificial Intelligence, Vol 5, pp 1-49, 1974.

[FAR 85]
H. FARRENY :
" Les systèmes experts : principes et exemples ",
Editions Cepadues, 1985.

[GIL 77]
W.K. GILOI, H. BERG :
" Data structures architectures, a new approach to parallel processing ",
Proc of IMAS symp, Munich, pp 327-331, 1977.

[HAS 82]
T. HASEGAWA :
" A new approach to teaching object description for a manipulation environment ",
12 symp on Industrial Robot, Paris, pp 87-97, june 1982.

[ITM 83]
ITMI SA :
" Cours d'intelligence artificielle : méthodes, outils, applications ",
ITMI SA, Meylan FRANCE, 1983.

[KIR 85]
R.H. KIRSCHBROWN, R.C. DORF :
" KARMA, a knowledge based robot manipulation system ",
Robotic 1, Nort-Holland, pp 3-12, 1985.

[KUN 82]
H.B. KUNTZE, W. SCHILL :
" Methods for collision avoidance in computer controlled industrial robot ",
12 symp on Industrial Robot, Paris, pp 519-530, june 1982.

[LAU 81]
C. LAUGIER :
" A program for automatic grasping of objects with a robot arm ",
11 symp on Industrial Robot, Tokyo, oct 1981.

[MAT 74]
D. MATHUR :
" The grasp planner ",
Dep Artificial Intelligence, Edimbourg, DAI working, 1974.

[POR 85]
C. PORQUET :
" *La génèration de plans d'action pour un robot* ",
GERSIC, Caen France, 1985.

[SAC 75]
E.D. SACERDOTI :
" *A stucture for plans and behavior* ",
Artificial center, Tech. n 109, SRI, aug. 1975.