N° d'ordre : 740

THESE

présentée à

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

pour l'obtention du titre de

DOCTEUR

En Productique : Automatique et Informatique Industrielle

par

Ali KALAKECH
ingénieur AUB

Conception d'un Logiciel de Programmation Graphique des Robots d'Assemblage

Soutenue publiquement le 7 Juin 1991 devant la Commission d'Examen:

MM.

P. VIDAL	Président
J.P. BOURRIERES	Rapporteur
P. MILLOT	Rapporteur
C. VASSEUR	Directeur de Recherche
E. BIENFAIT	Examinateur

AVANT PROPOS

Le travail exposé dans ce mémoire a été réalisé au Laboratoire d'Automatique de l'Université des Sciences et Techniques de Lille-Flandres-Artois, sous la direction de Monsieur le Professeur Christian VASSEUR. Avant d'en entreprendre l'exposé, je tiens à exprimer ma reconnaissance à toutes les personnes qui m'ont aidé dans ce travail.

J'exprime ma gratitude à Monsieur le Professeur Pierre VIDAL, Directeur du Centre d'Automatique de Lille, pour l'accueil qu'il m'a réservé au sein de son laboratoire et je le remercie d'avoir accepté de présider le jury de ma thèse.

J'adresse de même toute ma reconnaissance à Monsieur Jean Paul BOURRIERES, Professeur à l'Université de Besançon, pour avoir accepté de juger le contenu de ce mémoire et pour sa présence parmi les membres de jury en tant que rapporteur.

Que Monsieur Patrick MILLOT, Professeur à l'Université de Valenciennes, accepte mes plus vifs remerciements pour l'intérêt qu'il a bien voulu porter au jugement de ce travail en acceptant d'être l'un de mes rapporteurs.

J'adresse mes sincères remerciements à Monsieur Christian VASSEUR, Professeur à l'USTLFA et Directeur de l'Ecole Nationale Supérieure des Arts et Industries Textiles de Roubaix, qui m'a confié cette étude.

Je remercie également Monsieur Eric BIENFAIT, Directeur de la société SERPAC pour l'intérêt qu'il a témoigné à ce travail en tant que membre du jury.

J'exprime profondément ma reconnaissance à la *Fondation Hariri* qui a contribué au financement de mes études.

L'ensemble de mes travaux a pu être mené à bien grâce à la coopération et à l'aide de tout le personnel du Laboratoire d'Automatique de Lille, auxquels je tiens à adresser mes plus vifs remerciements.

Table des Matières

INTRODUCTION GENERALE

La Robotique est considérée aujourd'hui comme un des grands axes du progrès scientifique et technique.Toutes les recherches technologiques, calculs économiques et pronostics scientifico-techniques convergent vers une conclusion commune: l'implantation des robots présentent des avantages énormes pour les industries manufacturières.

La compétitivité rend cette implantation primordiale pour les entreprises. L'introduction des robots accélère l'évolution des technologies industrielles de production. Les robots rentrent actuellement dans presque tous les cycles de production: usinage, transport, assemblage, emballage, manutention, ...

Le travail présenté dans cette thèse se situe dans le cadre de l'assemblage robotisé des produits mécaniques. Une grande partie des produits mécaniques sont fabriqués par assemblage de pièces élémentaires. Le travail d'assemblage représente une part importante du coût de production.

L'automatisation des fabrications mécaniques, traditionellement confinée dans des productions de masse, s'étend actuellement aux fabrications en petites séries et même aux applications de fabrication à l'unité.

Pour cela, la flexibilité doit être un facteur essentiel lors de la conception d'une cellule de production automatisée. Une telle conception doit prendre en considération la diminution de la période de vie des produits et l'augmentation du nombre de variantes de ces produits.

Ainsi, la programmabilité et l'adaptabilité se présentent comme des facteurs essentiels qui conditionnent l'utilisation des robots sur des processus d'assemblage.

Le robot a besoin d'une phase de programmation afin de générer les plans d'actions nécessaires pour l'exécution des tâches. Un système de programmation des robots d'assemblage doit assurer trois fonctions principales:

1) Identification de l'environnement du robot:

- cinématique du robot
- volume de travail
- espace atteignable
- type d'effecteurs
- capteurs (vision ou autre)
- commande
- communication
- périrobotique

2) Modélisation du processus d'assemblage et de la base de travail:

- acquisition, modélisation et mémorisation des pièces et composants élémentaires
- modélisation et génération du produit final
- gamme d'assemblage
- ordonnancement des pièces
- enchaînement des étapes et des opérations de montage

3) Apprentissage des opérations de montage des composants. Cette phase d'apprentissage doit permettre au robot d'assemblage d'identifier:

- les positions initiales des composants dans le volume de travail
- les points et les modes de préhension des composants
- les positions finales des composants au sein du produit
- les trajectoires de déplacement que doit effectuer le robot lors de l'exécution des tâches d'insertion des composants.

Les progrès croissants de l'informatique ont permis le développement de systèmes de Conception et de Fabrication Assistées par Ordinateur. Ces systèmes utilisent l'informatique graphique tridimensionnelle.

L'association de l'informatique graphique et de la robotique présente un grand intérêt pour la conception, la programmation et la validation du fonctionnement automatique des robots et des postes d'assemblage robotisés.

La programmation des robots d'assemblage par moyens graphiques se révèle actuellement comme un domaine potentiel de recherche et d'applications. On se propose dans le présent travail, d'étudier une méthodologie de programmation graphique des robots d'assemblage.

L'étude comporte trois parties. Dans la première partie, on présente d'abord, l'état de l'art des robots d'assemblage (chapitre un). Le chapitre deux est consacré à la programmation des robots et à la présentation d'une méthodologie de programmation graphique des robots d'assemblage. Cette méthode nécessite une modélisation informatique tridimensionnelle du processus et de la base de travail du robot.

La deuxième partie détaille la méthode proposée, en deux chapitres. Au chapitre trois, la modélisation du processus d'assemblage et la structuration des données informatiques sont présentées. Ensuite, on introduit dans le chapitre quatre, l'algorithme de montage automatique des pièces et la phase d'apprentissage graphique.

Enfin, une application est présentée au chapitre cinq, qui constitue la troisième partie de l'étude. La méthodologie de modélisation et d'apprentissage graphique est appliquée au processus d'assemblage d'une gamme de moteurs électriques.

On termine par une conclusion générale qui fait le point sur ce qui est déjà réalisé, et qui ouvre des perspectives d'amélioration et de développement dans le cadre de la programmation graphique des robots d'assemblage.

Première Partie

CHAPITRE I

LES ROBOTS D'ASSEMBLAGE

I.1 - Introduction

Ce chapitre présente les robots d'assemblage. On commence par donner une vue générale des robots:

On définit et on décrit le robot. Ensuite, on présente les avantages des robots pour le milieu industriel et les problèmes qui sont encore à résoudre. Par ailleurs, on discute des caractéristiques opératoires des robots, en considérant, en particulier, les deux paramètres les plus importants pour les robots d'assemblage: la répétabilité et la précision de positionnement. Finalement, on aborde les systèmes robotiques et l'intégration des robots dans les systèmes complexes de production.

Dans un deuxième temps, on présente l'assemblage des produits industriels:

on définit et décrit l'assemblage industriel. Ensuite, on analyse les tâches mises en oeuvre lors des opérations de montage. On distingue alors deux types d'actions: les actions positionnelles et les actions fonctionnelles. Par suite, le processus d'assemblage est présenté. On discute les contraintes et les problèmes liés à l'assemblage.

Dans une troisième étape, on présente l'assemblage robotisé et l'intérêt de la flexibilité:

On parle des systèmes d'assemblage flexibles qui utilisent des robots pour l'exécution des opérations de montage des produits. La robotisation de l'assemblage soulève des difficultés qui sont à lever. D'une part, une reconception des produits est nécessaire en vue de robotiser l'assemblage, et d'autre part, la planification de l'assemblage au niveau du robot nécessite une modélisation du processus. La programmation des robots d'assemblage est en effet un domaine de recherche et de développement dans le cadre de l'assemblage robotisé.

En dernière étape, on parle des robots d'assemblage:

La particularité des robots d'assemblage vis à vis des autres robots à usage universel est alors discutée. On présente les types de robots utilisés dans le processus d'assemblage.

I.2- Présentation Générale des Robots

Le mot robot dérive du mot tchèque robota qui veut dire esclavage ou travail forcé [LAM 84]. Cependant on trouve dans la littérature plusieurs définitions données au robot. L'Institut de la Robotique Américaine donne la définition suivante [GRO 84], [CRIT 84]:

"Le robot est un manipulateur programmable et multifonctionnel destiné à transporter du matériel, des pièces, des outils ou des dispositifs spécialisés, par différents déplacements programmés, dans le but d'accomplir une variété de tâches."

L'Association Française de Normalisation (AFNOR) le définit comme un "manipulateur commandé en position, reprogrammable, polyvalent, à plusieurs degrés de liberté, capable de manipuler des matériaux, des pièces ou des dispositifs spécialisés, au cours de mouvements variables et programmés pour l'exécution d'une variété de tâches."

Le dictionnaire parle "d'une machine, qui ressemble à l'homme, et qui a une aptitude à exécuter, par ses propres moyens, des tâches et des commandes à n'importe quelle endroit." Ainsi les robots sont des structures mécaniques qui doivent assister ou remplacer l'homme, tout en étant commandées par des moyens électroniques et informatiques perfectionnés.

La première génération de robot date du milieu des années 1960. Les premiers robots étaient de simples bras mécaniques. L'apparition des microprocesseurs marque le début de la deuxième génération de robots. Les robots étaient alors dotés de langages de programmation. Ces robots entrent véritablement dans l'industrie à partir de 1982 [PAR 83]. Actuellement, une troisième génération de robots, possédant des critères d'adaptabilité et d'intelligence, se développe. Cette génération vise la flexibilité de l'ensemble du système robotique [CAS 87].

I.2.1 - Description du Robot

Physiquement, le robot se compose de trois blocs: le support de base, les bras et la partie terminale. Cependant, pour les robots sophistiqués, des armoires séparées servent de support

pour le calculateur, les automates de commande des actionneurs, l'alimentation, les interfaces, la périrobotique, ...

La partie terminale supporte l'effecteur. Les bras s'articulent autour des joints. Les sections entre les joints s'appellent anneaux. Les actionneurs permettent le mouvement des bras. Un centre de supervision (calculateur) contrôle les actionneurs et pilote l'ensemble du robot. Des capteurs montés sur le robot servent d'une part à contrôler l'état du robot, et d'autre part à détecter l'environnement externe.

*** Les Bras**

Les bras sont formés de membres rigides capable de supporter la charge du robot. Ces membres sont normalement creux et l'espace intérieur contient les moteurs, les équipements électriques, le câblage, ... Le développement actuel tend à réduire le poid du robot en utilisant des bras en fibres composites graphites à la place de l'acier ou l'aluminium.

*** Les Joints**

Les joints sont de type rotatoire ou glissant. Ils doivent être conçus avec précision pour maintenir exactement la position des bras. Des roulements à billes sont utilisés pour permettre aux joints de se déplacer sans friction et avec précision.

*** La Partie Terminale**

C'est la partie qui se trouve à la fin du bras et qui tient l'effecteur. On l'appelle le poignet. La flexibilité est un facteur essentiel dans la conception du poignet. En effet, le robot doit pouvoir orienter son effecteur dans des positions qui sont parfois délicates.

*** Les Effecteurs**

Le but du robot est de réaliser des tâches multiples. Pour cela, le robot est muni d'un effecteur. C'est l'effecteur qui adapte le robot à l'exécution d'une tâche spécifique. Les bras du robot doivent être capables de déplacer l'effecteur suivant une séquence de mouvement. Ainsi le robot est doté de plusieurs degrés de liberté pour pouvoir déplacer l'effecteur à n'importe quelle position. En revanche, l' effecteur a recours à la compliance pour pouvoir corriger les erreurs de positionnement du robot [GRO 84]. La compliance du robot est introduit

ultérieurement dans ce chapitre.

*** Les Actionneurs**

On trouve les moteurs hydrauliques, pneumatiques ou électriques. Actuellement, avec le développement des transistors de puissances, les moteurs électriques se substituent aux actionneurs hydrauliques. Dans ce type de moteurs, on distingue deux choix:

- les moteurs à courant continu, sans commutateurs (Brushless)
- les moteurs à courant alternatif, réversibles et avec asservissement (Servo)

*** Les Capteurs**

Les robots utilisent des capteurs de position, de proximité, de force, de couple et de température. Cependant, l'introduction de la vision augmente la capacité du robot et lui permet de "voir" et de réagir à son environnement. Un robot équipé d'un système de vision sera capable d'exécuter des tâches plus complexes.

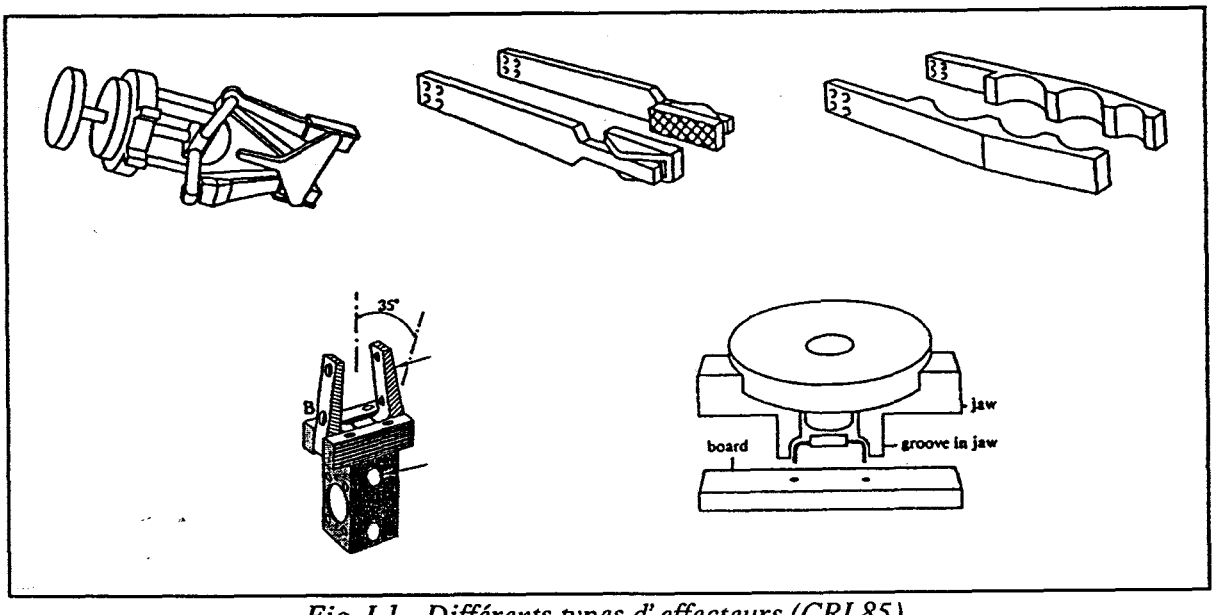


Fig. I.1 - Différents types d' effecteurs (CRI 85)

*** Le Superviseur**

Sur la chaîne de production, le robot se trouve en contact avec d'autres équipements: convoyeurs, chaînes d'alimentation, produits, opérateurs humains, ... On doit pouvoir coordonner toutes les activités dans une station de travail. Ces

activités peuvent être séquentielles ou peuvent se dérouler en parallèle. Un système de coordination et de supervision doit alors piloter l'ensemble des équipements et des activités. Ce système, qu'on appelle le contrôleur de la station [GRO 84], se trouve normalement dans le robot. Il a la responsabilité de contrôler les activités de tous les équipements qui se trouvent dans la station de travail.

I.2.2 - Les Avantages des Robots

Les robots présentent des avantages énormes pour les industriels. En effet le nombre de robots utilisés dans l'industrie augmente de 35 % par an.
Les raisons qui poussent les industriels à utiliser les robots sont les suivantes [CRI 85]:

1) Réduction du coût de production:
   - Le coût du robot amorti est inférieur au coût des travailleurs.
   - Le robot travaille effectivement à 98 % de son temps. L'être humain ne travaille qu'à 80 % de son temps (pause café, temps de déjeuner, fatigue, distraction, ...).
   - Les robots donnent un pourcentage plus élevé de bonne production. L'utilisation des robots minimise en effet la production défectueuse.

   Fréquemment, les systèmes robotiques rentabilisent en entier leurs capital de départ, frais d'installation et de formation, en moins de douze mois.

2) Augmentation de la productivité:
   Les robots exécutent plus vite leurs tâches. D'autre part, le planning des tâches est mieux respecté en robotisant la production. Les problèmes de délais et de retards sont minimisés.
3) Amélioration de la qualité du produit
4) Amélioration de la gestion de production
5) Possibilité d'utilisation de systèmes de supervision intégrés
6) Augmentation du cycle de vie utile du matériel
7) Réduction du temps de réaction et de mise à jour en cas de pannes ou de problèmes
8) Opération dans des milieux dangereux
9) Respect des mesures de sécurité

I.2.3 - Caractéristiques des Robots

Korsakov présente dans [KOR 87], les paramètres caractéristiques des robots. Il cite:

- précision de positionnement
- temps de cycle
- période de vie et de fonctionnement
- répétabilité
- résolution
- vitesse de déplacement
- volume de travail
- capacité de charge
- système de coordonnées
- nombre de degrés de liberté
- méthode d'apprentissage

Cependant, les deux paramètres les plus importants pour les robots d'assemblage sont la répétabilité et la précision de positionnement [CRI 85], [OWE 85], [KOR 87]. Ces paramètres sont affectés par les éléments suivants:

- les forces de gravité sur les bras
- les forces d'accélération
- les engrenages et les courrois des actionneurs
- les effets thermiques
- la portée
- la remontée

I.2.4 - Les Systèmes Robotiques

Dans l'industrie, le robot fonctionne comme partie d'un système complet. Ce système est composé de plusieurs parties ou sous systèmes. Entre autres, on peut citer les convoyeurs, le

système de vision, le système de contrôle, ... Les systèmes robotiques ont quatre fonctions principales qui sont:

- l'action (exécution des tâches)
- la perception de l'univers
- la communication avec l'extérieur
- la décision

La conception d'un système robotique doit tenir compte des paramètres suivants [EVA 77]:

* environnement de l'application:
  - chaleur, humidité, volume de travail, ...
* zone d'accessibilité
* vitesse de déplacement des bras
* type de contrôle:
  - mesure de force sur un ou plusieurs axes
  - contrôle de positionnement
* type de capteurs:
  - capteurs de proximité: détection des objets sans contact
  - capteurs de touche: présence / absence des objets
  - vision simple: contours, trous, dimensions,...
  - vision complexe: reconnaissance des formes,...
* interaction avec les autres équipements dans l'usine

I.3- L'Assemblage des Produits Industriels

On appelle assemblage, l'enchaînement des opérations de montage d'un ensemble de pièces élémentaires pour former un objet final. Ainsi, un processus d'assemblage permet de réunir un certain nombre de composants séparés pour former un produit composite. Ce processus peut être:

- manuel: l'assemblage se fait par des opérateurs humains. C'est le mode traditionnel d'assemblage.

- automatique: des machines spécialisées exécutent des tâches d'assemblage.
- robotisé: par l'utilisation de robots d'assemblage.
- hybride: intégrant des alternatives citées ci dessus.

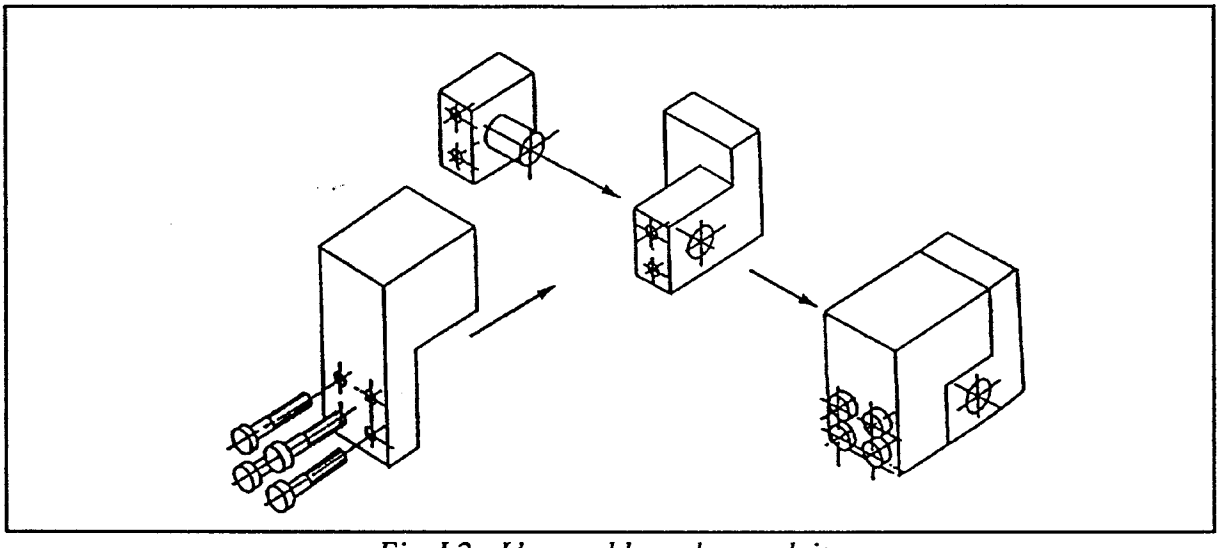


Fig. I.2 - L'assemblage des produits

Les processus d'assemblage tendent actuellement vers la robotisation et cela pour des causes économiques, techniques et sociales [BAY 88]. Cela mène à utiliser les robots et les machines robotiques dans le domaine de l'assemblage.

I.3.1- Description d'un Assemblage

L'assemblage inclut, en plus des techniques de montage, les modes de préhension et les phases de déplacement des éléments au sein du poste de travail [OWE 85]. Le poste de travail est le site sur lequel s'effectue l'assemblage. On peut résumer les phases du processus d'assemblage par:

- préhension des pièces à assembler
- acheminement de ces pièces vers le poste de travail
- réalisation des opérations de montage du produit
- évacuation du produit vers d'autres cycles de production

Les processus d'assemblage doivent donc tenir compte des possibilités et des contraintes de toutes ces phases qui rentrent dans le cycle d'assemblage.

I.3.2- Les Opérations de Montage

On donne souvent le nom d'assemblage, au produit composite résultant d'un processus d'assemblage. Un produit assemblé, ou bien un assemblage, consiste en un ensemble A de pièces rigides, $P_i \varepsilon A$, connectées par une configuration géométrique stable et rigide. On décrit la configuration géométrique de la façon suivante [WON 86]:

Chaque pièce a un système de coordonnées attaché à elle, noté R_i. On appelle M_i la transformation homogène entre le système de coordonnées universel W et le repère R_i de P_i dans son état courant. Dans la configuration finale d'assemblage, chaque pièce doit atteindre un état final $M_i = M_{if}$ spécifié par la géométrie du produit.

D'autre part, une base de données donne une description géométrique complète des pièces élémentaires et de l'assemblage dans ses différents états. Cette description se fait sous la forme de primitives (surface, volume, concavité, quantité de matière, autres paramètres géométriques, ...) pour les composants, et de relations spatiales qui lient ces composants.
Une telle description est considérée comme une représentation de l'état pour un système de planification de l'assemblage.

Ce système utilise souvent une représentation relationnelle de l'assemblage, pour dériver les informations nécessaires sur la position et l'orientation des composants et des sous assemblages lors des opérations de montage.

I.3.3- Analyse des Tâches

Bourjault a analysé dans [BOU 84], les tâches d'assemblage. Cette analyse révèle l'existence de deux types de déplacements mis en oeuvre lors du processus d'assemblage. Le premier type porte sur l'alimentation et l'évacuation des composants et des produits au sein du poste de

travail. On appelle ce type d'actions, les actions positionnelles. Le deuxième type d'actions porte sur les déplacements fin des composants lors des opérations de montage du produit. Ce type d'actions s'appelle actions fonctionnelles

I.3.3.1- Les Actions Positionnelles

Les actions positionnelles sont des actions préparatoires qui assurent les conditions spatiales nécessaires à la réalisation des actions fonctionnelles. Ces actions ont un caractère absolu parce qu'elles sont déterminées, de façon indépendante, par la forme et la nature des composants [JEA 85], [BOU 84]. Les actions positionnelles conditionnent:

- la mise en scène du poste d'assemblage
- la cadence de production
- l'initialisation des actions fonctionnelles

I.3.3.2- Les Actions Fonctionnelles

Ces actions se caractérisent par des mouvements relatifs fins. Elles établissent les liaisons mécaniques entre les différents composants du produit. Les actions fonctionnelles sont déterminées par la composition et l'architecture du produit à assembler. Elles ont un caractère relatif qui se manifeste par l'apparition des contacts entre les composants.

Les actions fonctionnelles peuvent être établies soit par un mouvement simple consistant en une translation, une rotation ou un mouvement hélicoïdal, dans ce cas on parle d'action fonctionnelle simple, soit par un mouvement combiné consistant en une succession de mouvements simples, on parle alors d'action fonctionnelle combinée.

Les actions fonctionnelles intrinsèques à l'assemblage, et qui font appel à la compliance du manipulateur, caractérisent le rôle des robots d'assemblage et les particularisent par rapport aux robots à vocation plus universelle [JEA 85]. On revient plus tard dans ce chapitre, sur la compliance des robots d'assemblage.

I.3.4- Le Processus d'Assemblage

L'assemblage industriel est une tâche compliquée. Une analyse préalable est essentielle en vue de choisir le meilleur processus de production. Korsakov traite dans [KOR 87], ce problème. Le but d'une telle analyse est de:

- établir le degré de spécialisation des opérateurs
- optimiser la division du processus en étapes
- accroître la production
- établir les relations entre les différents cycles industriels

La structuration du processus d'assemblage dépend en majeure partie des éléments suivants:

- la conception du produit
- les dimensions et la masse du produit
- le type de production
- le temps de cycle
- la position des composants et des ateliers dans l'usine

Généralement, le processus d'assemblage se divise en une série d'étapes progressives. Ces étapes sont:

1) **Le pré-assemblage**: appliqué pour faire des stocks de produits ou composants élémentaires.
2) **L'assemblage intermédiaire**: les composants sont montés l'un après l'autre en vue de former le produit final. A chaque intégration d'un composant, un sous assemblage intermédiaire est alors crée. Ce sous assemblage, qui est lui même un assemblage, fait partie de l'assemblage complexe qui est le produit final.
3) **L'assemblage des parties**: cette étape est nécessaire dans le cas où, un des composants du produit est lui même un assemblage. Dans ce cas, il faut assembler ce composant à part avant de l'intégrer dans l'assemblage du produit.
4) **L'assemblage final**: c'est la dernière étape qui aboutit à l'assemblage du produit.

Après l'étape finale, le produit est normalement soumis à des tests d'acceptation et de qualité avant de continuer son cycle dans l'usine.

I.3.4.1 - L'Assemblage Progressif

L'assemblage progressif consiste en une série d'opérations répétitives et similaires. Ces opérations sont exécutées sur plusieurs stations de travail arrangées en chaîne. Si le temps de cycle des opérations sur toutes les stations est bien réglé, l'assemblage entier est fait par un simple transfert continu du travail, d'une station à une autre, le long de la chaîne d'assemblage (flux continu).

Ainsi, le temps de cycle, ou inversement la cadence est un facteur essentiel pour la conception d'une chaîne d'assemblage. Ce facteur influe sur la technologie utilisée dans l'assemblage progressif. Un ajustement de ce facteur nécessite une restructuration et une redivision de la totalité du processus d'assemblage. L'assemblage progressif offre de multiples avantages:

- augmentation de la productivité
- réduction du coût du produit
- réduction des stocks
- optimisation du temps de cycle global
- offre la possibilité d'automatiser ou de robotiser tous les cycles de production
- amélioration de l'harmonie du cycle industriel

Actuellement, la standardisation et l'unification des produits industriels facilite le développement de ce type d'assemblage. Une production en ligne continue nécessitera:

- des technologies de fabrication bien planifiées
- des systèmes d'approvisionnement sans interruption pour les stations de travail
- une chronologie bien définie pour toutes les étapes de l'assemblage

I.3.4.2 - L'Assemblage en Groupe

L'assemblage en groupe consiste à assembler plusieurs produits sur la même station de travail. Ce type d'assemblage assure une utilisation maximale de l'équipement et ne nécessite pas un réajustement. Par contre, il nécessite une station de travail complexe. L'assemblage en groupe est moins usuel dans l'industrie que l'assemblage progressif.

I.3.5 - Les Contraintes et les Problèmes d'Assemblage

Le processus d'assemblage pose plusieurs problèmes [JEA 85]:

- acheminement des composants
- mode de préhension
- contact entre pièces
- collision lors du montage

En pratique, il est impossible de manipuler une pièce solide vers une position finale sans considérer les contraintes imposées par les autres pièces sur les trajectoires de déplacement. On considère deux catégories de contraintes [WON 86]:

1) Les contraintes globales, appelées encore contraintes de trajectoires. La présence des obstacles sur la trajectoire gène l'accès global de la pièce vers sa position finale.

2) Les contraintes locales, appelées encore contraintes de blocage. Dans ce cas, les pièces et les surfaces adjacentes interdisent le mouvement incrémental de la pièce autour de sa position finale. Ces contraintes sont déterminées par le contact de surfaces entre les pièces. La compliance des robots d'assemblage est développée essentiellement pour faire face à ces contraintes.

I.4 - L'Assemblage Robotisé

L'assemblage est un domaine potentiel d'applications pour la robotique. Dans ce domaine on distingue deux catégories d'assemblages: l'assemblage sur une chaîne de production en masse et l'assemblage par lots [GRO 84].

Pour la production en masse, le choix s'oriente plutôt vers l'automatisation. En effet, les équipements sont conçus spécialement pour assembler un produit particulier. Dans ce type de production, un robot s'avère relativement trop lent comparé à une machine spécialisée. En plus, la programmabilité du robot, qui est l'un de ses atouts les plus importants , devient sans intérêt dans une production en masse.

Par contre, Les opérations d'assemblage par lots, sont bien adaptés à l'utilisation des robots. Ceci est dû d'une part à des raisons économiques et d'autre part aux capacités techniques que présente la nouvelle génération des robots. En effet, l'assemblage par lots permet une variation dans les produits à assembler, et par suite, autorise les petites séries.

La chaîne d'assemblage, dans ce cas, doit donc être capable de répondre à la variation de production. Ce type de production nécessite un système d'assemblage flexible. Les industriels appellent ce type de systèmes: "Systèmes d'Assemblage Adaptables et Programmables (APAS)" [GRO 84]. Les robots constituent le noyau de tels systèmes. Ils présentent en effet, deux caractéristiques importantes: la programmabilité et l'adaptabilité.

I.4.1 - Les Systèmes APAS

Un système APAS est formé de un ou plusieurs robots et de l'ensemble des équipements qui forment un poste de travail: convoyeurs, systèmes de contrôle et de supervision, opérateurs humains, ...

Les robots se chargent d'exécuter les opérations de montage du produit. La programmabilité et l'adaptabilité sont les deux critères qui font du robot le noyau d'un système APAS. La programmabilité du robot est nécessaire pour inclure dans l'assemblage des cycles complexes

et variés de déplacements. En plus, un système APAS doit être capable de mémoriser plusieurs programmes correspondant aux différents produits à assembler.

D'autre part, le système doit s'adapter à tout changement dans le mode de production. L'adaptabilité est nécessaire pour permettre au système de compenser les changements d'environnement. Les variations d'environnement incluent:

- la variation de position et d'orientation des pièces à assembler
- la présence de pièces défectueuses
- le changement de l'état du sous assemblage
- la détection d'opérateurs humains ou d'autres obstacles dans le volume de travail

I.4.2- Les Problèmes de la Robotisation

Les deux grandes catégories de travaux effectués dans le cadre de l'assemblage industriel sont: la conception du produit et la réalisation du produit [BAY 88].

I.4.2.1- La Conception du Produit

L'assemblage manuel des produits existants repose sur la dextérité humaine, le sens visuel et l'intelligence des opérateurs: qualités que le robot ne possède pas. Pour cela, il est nécessaire de reconcevoir le produit en vue de robotiser ou automatiser son assemblage. Une telle reconception doit tenir compte des facteurs suivants [TOD 86]:

- réduction du nombre et du type de composants
- symétrie dans la conception des composants
- prise en compte du processus d'assemblage: mode d'alimentation de la chaîne, stockage du produit, progression dans l'usine, ...

D'autre part, l'efficacité de la robotisation de l'assemblage est directement liée à la conception des produits à assembler et à la manière dont le processus d'assemblage est appliqué. Les

critères de conception d'un nouveau produit sont les suivants:

- optimisation du coût de production
- fonctionnalité
- sûreté et la fiabilité
- méthode d'assemblage du produit
- apparence du produit

I.4.2.2- La Réalisation de l'Assemblage

Pour réaliser un produit, il faut pouvoir programmer les tâches d'assemblage au niveau du robot et du système d'assemblage. Cela nécessite une modélisation de la base de travail et une planification du processus d'assemblage. La planification de l'assemblage doit:

1 - fournir le mode d'alimentation des composants
2 - établir l'ordre de montage des pièces
3 - programmer les opérations de montage:
    a - définir la position initiale de chaque pièce
    b - définir le mode de préhension des pièces
    c - définir les trajectoires de parcours lors du montage
    d - définir la position finale de chaque pièce au sein de l'assemblage
    e - définir la position finale du produit
4 - contrôler l'assemblage
5 - évacuer le produit et recommencer le cycle

Actuellement, on trouve différentes approches de planification d'assemblage. Les principales approches sont [BAY 88]:

- élaboration des gammes de montage à partir d'une base de données [TEM 85]
- décomposition en assemblages élémentaires [MAG 82]
- utilisation des liaisons fonctionnelles entre les composants [BOU 82]
- utilisation de l'Intelligence Artificielle [KEM 85]

Cependant, la programmation des opérations de montage reste un domaine ouvert de recherche en Robotique. Plusieurs projets sont en cours de développement pour la programmation et l'apprentissage des robots d'assemblage. On présente dans cette thèse, une méthode d'apprentissage et de génération des trajectoires par l'intermédiaire de la programmation graphique. La programmation graphique sera introduite dans le chapitre suivant.

I.4.2.3 - La Compliance

La compliance est la faculté de modification en temps réel de la trajectoire de parcours imposée préalablement au robot, sous l'effet des forces de contacts apparaissant lors du déplacement. Les robots d'assemblage utilisent cette compliance pour créer des déplacements correctifs en fonction des efforts rencontrés. Elle est nécessaire pour pouvoir compenser les imprécisions et les erreurs de positionnement. Ces erreurs sont dues aux facteurs suivants [OWE 85] :

- imperfection du système de commande
- changement d'environnement
- dispersion de la production
- variation de position et de la configuration de saisie

La correction d'erreurs de positionnement nécessite une détection d'efforts apparaissant entre les pièces en contact. L'action du robot pourra donc être modifiée par la gestion des efforts décelés. On distingue deux types de compliance:

- la compliance active
- la compliance passive

I.4.2.3.1 - la Compliance Active

Dans la compliance active, les efforts de contact sont mesurés par des capteurs, ce qui permet

l'asservissement des actionneurs pour réaliser la correction de position et d'orientation nécessaire [COT 74], [VAN 79], [REB 84]. Ce mécanisme nécessite donc [ROU 83]:

* Un captage d'efforts; ce captage est réalisé

  - soit par des capteurs extérieurs multicomposante force - couple, c'est le cas de la "Table Compliante Active" proposée par HITACHI dans [KAS 81], du "Poignet Actif à Compliance Adaptable" de VAN BRUSSEL [BRU 81], ou du préhenseur à doight munis de capteurs d'efforts proposé par J.C. GUINOT dans [GUI 83].

  - soit par répercusion sur les variables de commande de robot, des efforts résistants rencontrés par les actionneurs.

* Un traitement reliant les déplacements à effectuer aux forces détectées.

* Une commande permettant d'effectuer les mouvements correctifs. Pour la commande on distingue deux approches:

  - l'utilisation explicite de retour d'effort en spécifiant des relations linéaires entre force et couple d'une part et position et orientation d'autre part. On note dans ce sens les travaux de SALISBURY dans [SAL 80].

  - l'usage de la commande hybride, qui consiste à commander certains degrés de liberté en force et commander indépendamment les autres en position [BRA 82]. La commande hybride dote les robots d'assemblage de la faculté d'adaptation aux erreurs de positionnement. Elle introduit deux boucles d'asservissement différentes pour les forces et pour les variables de mouvement (position et vitesse). On note dans ce cas les travaux de A. ROBERT dans [ROB 86-b], de J. IRIGOYEN dans [IRI 86], ..

I.4.2.3.2 - la compliance passive

La compliance passive consiste à l'utilisation d'éléments élastiques qui, sollicités par des

efforts d'interactions, subissent une déformation entraînant de petits déplacements qui autorisent la réalisation correcte du montage. Un dispositif flexible permet par exemple à l'effecteur, de faire des petits déplacements pour annuler les forces et couples engendrés au cours de l'exécution de la tâche.

Un robot doté de cette faculté est nommé robot auto-adaptatif. Dans ce sens, L. LEMARCHAND définit dans [LEM 86] l'auto-adaptativité comme étant "la faculté qu'a un dispositif de procurer aux objets à assembler un mouvement relatif compatible avec la géométrie de leur contact mutuel et qui n'est pas forcément identique à celui imposé par le générateur du mouvement absolu". On distingue deux approches:

- l'usage d'organes compliants extérieurs, comme le "Remote Compliance Center" (RCC) décrit dans [NEV 79], le "Passive Compliance Device" (PCD) [CAL 79], le préhenseur pneumatique auto-adaptatif (PPA) proposé par L. LEMARCHAND dans [LEM 86], ..

- l'approche proposée par J.P. BOURRIERES dans [BOU 84-b] et reprise par P. JEANNIER dans [JEA 85] qui consiste à utiliser les efforts d'interactions de façon implicite pour établir une adaptabilité intrinsèque au robot. L'idée de base est d'exploiter l'élasticité des articulations en programmant les gains d'asservissement. P. JEANNIER propose de réaliser l'autoadaptation par la maîtrise des souplesses articulaires grâce au pilotage des gains d'asservissement. Cette approche nécessite une coopération avec les constructeur des robots.

I.5- Les Robots d'Assemblage

L'utilisation des robots industriels dans l'assemblage a un grand potentiel de croissance. Actuellement, plusieurs compagnies perçoivent que l'assemblage robotisé va réduire le coût de production. Pour cela, elles sont en train de développer de nouvelles générations d'usines.

I.5.1- Les Types de Robots d'Assemblage

L'assemblage robotisé est dominé par quelques types de robots. Ainsi, pour les applications simples de type "Pick and Place", les robots à configuration cylindrique sont largement utilisés. Les raisons de ce choix sont la rapidité, le faible coût et la précision que présente ce type de robots.

Pour les applications plus complexes, on retrouve en dominance, les robots du type SCARA. Ce type de robot possède une bonne définition de son axe vertical, ce qui lui permet une grande souplesse dans l'insertion verticale des composants. L'insertion verticale est une opération prépondérante dans la plupart des tâches d'assemblage. Les robots cylindriques possèdent cette même propriété mais ils sont moins utilisés dans l'assemblage.

On constate aussi l'utilisation des robots PUMA dans l'assemblage. Cela est du au fait que ce robot était le premier disponible sur le marché, et qu'il possède la rapidité et la précision nécessaires pour des tâches d'assemblage.

Dans la nouvelle gamme des robots d'assemblage, la dominance reste toujours pour les robots SCARA [TOD 86]. En exemple, on peut citer le robot ADEPT THREE (rapidité 11 m/s), le robot IBM 7545 (répétabilité 0.05 mm), ...

Les robots cartésiens ont aussi leur part sur le marché. On note le robot IBM 7565 à puissance hydraulique, le robot PRAGMA A3000, ...

Dans la famille des robots polaires, on trouve le robot ASEA IRb1000 qui possède une rapidité considérable (jusqu'à 12 m/s).

I.5.2- Caractéristiques des Robots d'Assemblage

Un robot d'assemblage doit posséder en général:

- un mécanisme positionnel unique pour déplacer et orienter son effecteur à n'importe quelle position
- un mécanisme fonctionnel propre pour l'exécution des tâches (pousser, tourner des vis, ...)

Une première caractéristique d'un robot d'assemblage est alors qu'il doit disposer d'une articulation terminale qui lui permette d'atteindre chaque point de l'espace atteignable, avec une attitude et une orientation quelconque de son effecteur.

Une deuxième caractéristique est la flexibilité. L'augmentation du nombre d'articulations d'un robot accroît sa flexibilité. Mais cela entraîne des problèmes techniques supplémentaires et augmente le coût propre du robot.

D'autres caractéristiques opératoires des robots d'assemblage sont les suivantes [SEP 85], [USI 90]:

- la répétabilité: c'est la caractéristique la plus importante pour les robots d'assemblage. On distingue entre la répétabilité de la saisie d'une pièce par l'effecteur, et la répétabilité de la trajectoire. Actuellement, les robots de type SCARA atteignent une précision de répétabilité de l'ordre de 0.025 mm.

- la précision de positionnement: c'est une caractéristique importante pour les robots d'assemblage. Il ne faut pas la confondre avec la répétabilité. Les robots SCARA ont une précision de positionnement de l'ordre de 0.1 mm.

- la vitesse: elle est inversement proportionnelle à la durée de déplacement. Elle détermine la cadence du poste.

- la capacité de charge.

I.6 - Conclusion

On a présenté dans ce chapitre l'état de l'art de la Robotique.

Une description détaillée du robot révèle qu'il possède, en plus de ses organes mécaniques, un centre de supervision et des capteurs qui l'aident à contrôler son état interne et à détecter l'environnement externe.

L'étude des caractéristiques des robots révèle que la répétabilité et la précision de positionnement sont les paramètres les plus importants pour les robots d'assemblage.

L'assemblage, qui est une phase importante de la production industrielle, présente, en effet, un vaste domaine d'application pour la robotique.

On a également discuté dans ce chapitre le processus d'assemblage: on a abordé l'analyse des tâches, les contraintes et les problèmes spécifiques de l'assemblage, la division du processus en cycles élémentaires, ...

La flexibilité devient actuellement un critère essentiel pour les systèmes d'assemblage automatisés. Cependant, l'utilisation des robots sur les chaînes d'assemblage mène à deux nécessités de natures différentes: la reconception des produits et la modélisation informatique du processus d'assemblage.

Une étude de marché des robots d'assemblage révèle une prédominance traditionnelle des robots PUMA de UNIMATION, et une tendance actuelle vers le robots de type SCARA (IBM 7545, ADEPT THREE, ...).

Cependant, la programmation de ces robots reste un souci pour les constructeurs et les industriels. Le chapitre suivant introduit la programmation des robots d'assemblage.

CHAPITRE II

LA PROGRAMMATION GRAPHIQUE DES ROBOTS D'ASSEMBLAGE

II.1 - Introduction

Ce chapitre présente une méthode de programmation hors ligne des robots d'assemblage, par l'utilisation de moyens graphiques.

Dans un premier temps, on présente les méthodes de programmation des robots d'assemblage. On distingue entre les méthodes de programmation en ligne et les méthodes de programmation hors ligne. La programmation hors ligne présente des avantages et des promesses pour les robots d'assemblage.

Actuellement, on envisage de programmer les robots d'assemblage par simulation graphique. La programmation graphique des robots d'assemblage se développe grâce à l'usage et au perfectionnement actuel des systèmes de CAO.

Dans un deuxième temps, on présente la méthode de programmation graphique développée dans cette étude. On définit le processus d'assemblage sous son aspect informatique et on décrit la réalisation des opérations d'apprentissage graphique des robots d'assemblage.

L'implantation logicielle de cette méthode nécessite la conception d'un système interactif. On évoque dans une troisième étape, l'aspect interface homme - machine, et on présente les méthodes et les modèles de conception de systèmes interactifs.

II.2- Les Méthodes de Programmation des Robots

La programmation doit permettre de définir une séquence d'actions à exécuter par le robot [CAS 87]. Les modes de programmation des robots varient suivant le type de robot et le genre d'applications envisagées. On parle de programmation en ligne et de programmation hors ligne.

II.2.1- La Programmation En Ligne

Le moyen traditionnel de programmer les robots est l'apprentissage qui se fait directement sur le robot. Le robot subit en effet une phase d'apprentissage avant d'exécuter tout seul les tâches. Plusieurs méthodes d'apprentissage en ligne sont utilisées. On cite ci dessous les principales méthodes.

II.2.1.1- L'Apprentissage par Déplacement Manuel

Dans ce type d'apprentissage, un opérateur guide le robot en déplaçant manuellement les bras du robot sur les trajectoires de parcours. Le robot mémorise les déplacements dans l'ordre appris.
En phase d'exécution, le robot répète exactement toutes les séquences d'actions mémorisées en phase d'apprentissage.

II.2.1.2- L'Apprentissage par Télécommande

L'opérateur a recours à une boite à bouton pour commander les déplacements des bras du robot. La commande du déplacement est possible dans toutes les directions (axes X, Y, Z et angles alpha, bêta, gamma) et dans les deux sens (positif et négatif). L'opérateur n'a qu'à

appuyer sur un bouton pour faire déplacer le bras voulu. Une fois le déplacement voulu réalisé, l'opérateur appuie sur un bouton de validation, et l'étape est mémorisée par le robot.

II.2.1.3- L'Apprentissage par Entrée Directe des Coordonnées

Les coordonnées des points à atteindre sont rentrées directement via un clavier ou une boite de commande. Le robot optimise seul les trajectoires à parcourir entre les points à atteindre. Ces trajectoires peuvent être altérées par des points de passage obligatoires.

La programmation par apprentissage en ligne est largement utilisée dans l'industrie. Cela est dû au fait que ce mode de programmation est simple, rapide, et ne nécessite pas l'utilisation d'outils sophistiqués, ni l'intervention de spécialistes [BIE 87].
Par contre, ce mode de programmation nécessite l'arrêt de production pendant la phase d'apprentissage. En plus, ce mode demande la présence d'opérateurs sur le site de travail, ce qui présente un risque non négligeable [CAS 87].

II.2.2- La Programmation Hors Ligne

La programmation hors ligne des robots se fait sans avoir recours à l'outil qui peut donc continuer à produire. On passe en effet par une étape de simulation. Cette programmation s'accomplit sur un terminal d'ordinateur. La programmation hors ligne présente plusieurs avantages [GRO 84], [CAS 87].
Un premier avantage est l'augmentation du temps d'engagement des moyens de production. En effet, il n'y a pas une perte dans le temps de production du robot. L'apprentissage d'une nouvelle tâche se fait en même temps que le robot continue à exécuter l'ancienne tâche sur la chaîne de production. Cela veut dire un temps d'utilisation maximum du robot et des équipements de la station de travail.
Un deuxième avantage est la prise en compte des effecteurs et la possibilité d'effectuer des modifications sans arrêter l'outil productif.
Un autre avantage de la programmation hors ligne est la possibilité d'intégrer le robot dans le

système d'information de l'usine. Dans les futurs systèmes de production, la programmation des robots se fera au niveau du système de CFAO de l'usine.

La programmation hors ligne nécessite une phase de validation. L'utilisation d'un modèle informatique est alors indispensable pour simuler le robot, la station de travail, le mécanisme de production, les tâches, ... Toutes les informations relatives à la programmation du robot, doivent être disponibles dans une base de données. La connaissance de la forme géométrique des objets à manipuler est nécessaire à l'établissement d'une telle base de données utilisable pour simuler et analyser le fonctionnement du système robotisé. Elle est nécessaire en plus, pour détecter d'éventuelles collisions. La détermination des relations entre les éléments géométriques du robot est également indispensable.

On peut regrouper les différentes méthodes de programmation hors ligne des robots, en trois classes:

- La programmation par langage traditionnel
- la programmation automatique à l'aide de l'intelligence artificielle
- la programmation graphique.

II.2.2.1- La Programmation par Langage Traditionnel

La programmation textuelle, par langage informatique, a été développée à partir de 1970. Les langages de programmation présentent les caractéristiques suivantes [SOR 83], [CAS 87]:

- facilité d'utilisation
- interaction avec l'environnement
- aptitude au parallélisme
- niveau de modélisation
- mode de déplacement

Actuellement, plusieurs langages de robot existent sur le marché. Parmi les systèmes les plus récents, on peut citer:

* AML [TAY 82],

permettant d'interfacer le robot avec d'autres systèmes (vision ou autre).

* LMAC [BHP 87],

qui est un langage modulaire utilisant la notion d'abstraction et du typage.

* AROWS [TAY 82].

Ce système propose son propre interface utilisateur.

* il faut noter les travaux de RADHAKRISHAN. Il propose dans [RAD 87] un système de programmation interactif qui permet à l'utilisateur de développer ses programmes en mode interactif en utilisant un simple PC.

II.2.2.2- La Programmation Automatique par Intelligence Artificielle

L'Intelligence Artificielle est le développement des techniques permettant de simuler dans un système informatique, un comportement assez complexe, réputé intelligent et proche de celui d'un être humain [GUY 85]. Actuellement, on dispose de systèmes artificiels qui assistent les robots dans [PAR 83]:

- l'appréhension de l'univers
- l'élaboration de modèles informatiques
- la représentation graphique
- la prise de décisions
- l'accomplissement des actions et l'exécution des tâches

La programmation automatique, nommée encore programmation implicite, utilise les notions de l'intelligence artificielle pour tenter d'offrir au programmeur la possibilité d'exprimer les tâches en termes d'objectifs à atteindre plutôt qu'en termes d'actions. Un tel système, dit intelligent, doit être doté de capacités de raisonnement et d'apprentissage [BRA 85]. Cependant, cette notion de système intelligent n'est pas très formelle: on préfère parler de système de programmation automatique. Les systèmes qu'on connaît sont:

* HANDEY [LJM 87],

implanté pour des systèmes robotiques simples de type "Pick and Place".

* LOLA [BRU 87],

qui utilise la notion de "FRAME" pour définir l'univers.

* TURBO-CAPP [WAN 87],

qui permet d'interpréter des informations provenant d'une base de données CAO et d'entreprendre un raisonnement géométrique et relationnel sur les objets de l'univers.

* ELMARAGHY [ELM 87],

développé à l'université de Mc Master - CANADA. Ce système prend en compte la modélisation de l'univers, la planification des mouvements fins et la mise en route d'un système de vision expert. Le domaine d'application de ce système est l'assemblage mécanique.

* SHARP,

qui est en cours de développement au LIFIA [LAU 87]. L'idée de base de ce projet est de concevoir un environnement complet de programmation des robots d'assemblage. SHARP suggère un interface homme - machine convivial permettant de communiquer au système une modélisation de l'univers et une description du mécanisme d'assemblage en termes de relations géométriques.

II.2.2.3- La Programmation Graphique

Des nouvelles méthodes de programmation des robots d'assemblage, à l'aide de moyens graphiques, sont en cours de développement. L'expression graphique est en effet un excellent mode de communication homme-machine [BIE 87]. La représentation visuelle facilite la compréhension et accélère le processus de perception de l'univers [PAR 83].

Les modèles de programmation graphique des robots sont souvent confondus avec les systèmes de CAO qui sont à vocation plus générale. L'approche qui consiste à s'appuyer sur

des systèmes de CAO pour développer des applications dédiées à la robotique, a donné naissance à plusieurs produits. On peut citer:

* CATIA ROBOTIQUE, développé par le LAMM et qui est une version orientée robotique du système de CAO CATIA de DASSAULT [HAL 85].

* CIMSTATION de la société SILMA [CIM 89]

* ROBCAD de TECNOMATIX [ROB 89]

*IGRIP de DENEB [IGR 89]

ROBOTICS de Mc DONNEL DOUGLAS [MCD 89].

D'autres systèmes proposent des solutions propres à la robotique. Ces systèmes sortent en majorité des laboratoires de recherche. On présente ci dessous les principaux systèmes:

* LMAD-G [GUY 85]

C'est un système de conception de trajectoires qui propose à l'utilisateur un certain nombre de primitives géométriques, graphiques et robotiques. Au niveau graphique, le système est limité par des primitives simples de type afficher un point, tracer une droite ou tracer un cercle.

* PRATIC [BIE 87]

PRATIC est un système de programmation de robots d'assemblage basé sur l'acquisition et le traitement d'images. Son utilisation est limitée à l'espace 2.5D (un seul plan ou des plans parallèles).

* PAMELA [THE 88]

C'est un système de planification des mouvements fins liés aux tâches d'assemblage. Il est développé à l'INPG dans le cadre du projet SHARP présenté ci dessus. La modélisation des objets de l'univers se fait par sommets, arètes et faces. Pour cela, la base de travail ne peut supporter que des objets polyédriques.

* ARES [DCC 88]

Développé par le CEA en tant qu'atelier destiné aux robots mobiles, le système propose dans sa partie modélisation graphique, une double représentation interne, en utilisant la construction arborescente de solides (CSG) pour définir les objets de l'univers et la représentation polyédrique pour visualiser ces objets [MOU 90].

* SYNAPSE [RAM 90]

Il présente une méthode de modélisation de l'espace d'un robot mobile. Une triangularisation de l'espace est effectuée pour former des polygones convexes. Un objet est décrit dans cet espace par ses surfaces extrêmes frontières. La modélisation des objets se fait par des sommets et des segments.

La programmation graphique des robots nécessite l'utilisation d'un modèle géométrique qui comporte une composante fonctionnelle mathématique et une composante graphique de visualisation (Fig. II.1).

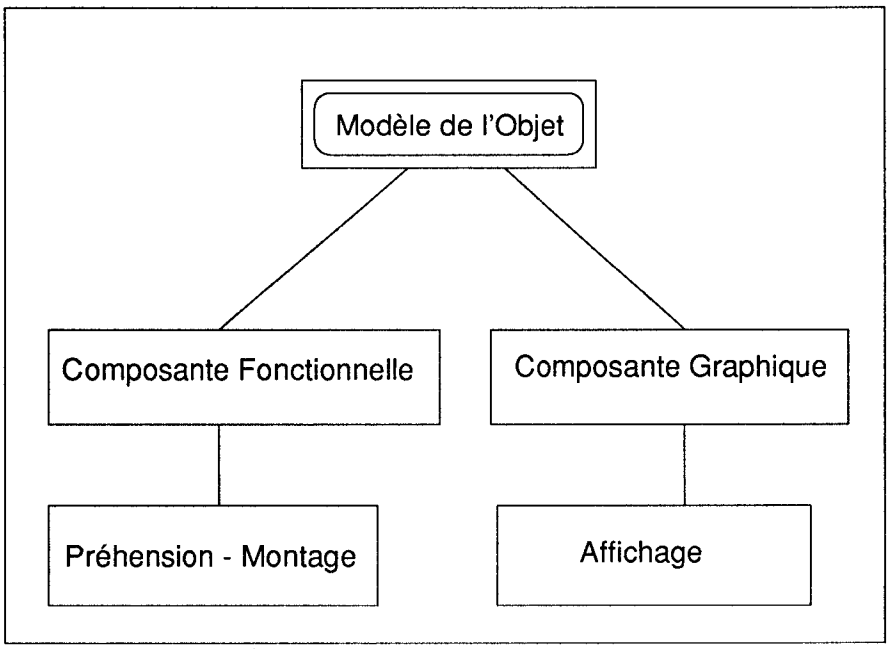


Fig. II.1 - Le modèle de l'objet

L'expression d'une tâche se fait alors suivant deux niveaux. D'une part, une représentation graphique de la base de travail sur l'écran d'un ordinateur, facilite la compréhension du processus d'assemblage et par suite l'intervention du programmeur; d'autre part, des relations mathématiques gèrent les liaisons entre les composants et assurent le contrôle du calculateur

en phase d'apprentissage, lors du déroulement des tâches. On utilise dans cette étude, une méthode de programmation graphique des robot d'assemblage. La méthode utilisée, est présentée ci dessous.

II.3- La Programmation Graphique d'un Assemblage

II.3.1- Présentation

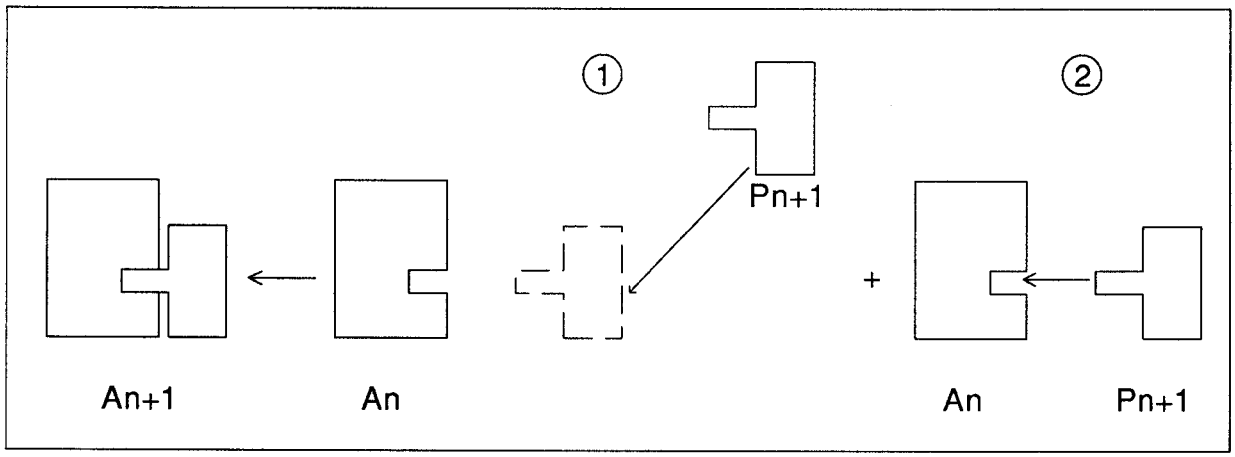


Fig. II.2 - Les opérations de montage

Soit A_M un assemblage formé de M pièces: P_1, .., P_M.

- A_n est le sous assemblage d'ordre n formé par les n premières pièces: P_1, .., P_n ; $n < M$.

- An+1 est le sous assemblage d'ordre n+1 formé par les n+1 premières pièces: P_1, .., P_{n+1} ; $(n+1) \leq M$.

- P_{n+1} est la (n+1) ème pièce.

Pour un robot d'assemblage, le montage de A_M nécessite deux types d'approches (fig. II.2):

* l'approche rapide, qui se déroule sans contraintes physiques ou collisions (étape 1). Ce sont les actions positionnelles.

* l'approche terminale qui prend en compte les collisions et les contraintes mécaniques et géométriques de l'assemblage (étape 2). Ce sont les actions fonctionnelles.

Nous nous sommes intéressés dans cette étude, à l' approche terminale, qui définit les trajectoires que doivent parcourir les P_{n+1}, lors du montage.

En effet, pour le montage de P_{n+1}, les contraintes et les caractéristiques de l'approche terminale sont, dans l'ordre de leur importance:

- la position finale de P_{n+1}
- la trajectoire de parcours entre la position finale et initiale de P_{n+1}
- le point et le mode de préhension du robot
- la position initiale de P_{n+1}

Il paraît donc plus intéressant que la phase d'apprentissage, dans la programmation du robot, procède par retour en arrière, c'est à dire, en commençant par la position finale de P_{n+1}, pour arriver à sa position initiale. Ainsi, le démontage de l'objet est utilisé pour définir les trajectoires de parcours et les points et modes de préhension du robot.

En effet, la planification de l'assemblage d'un objet composé de plusieurs pièces peut se voir comme une recherche de trajectoires dans l'espace de toutes les configurations progressives du sous assemblage.

L'état initial correspond à la configuration dans laquelle toutes les pièces sont détachées. L'état final correspond à la configuration dans laquelle toutes les pièces sont montées proprement pour former l'objet désiré.Les déplacements qui changent la configuration d'un état à un autre, forment les opérations d'assemblage.

Il peut y avoir plusieurs trajectoires entre l'état initial et l'état final de l'assemblage. Pour cela, plusieurs configurations peuvent être faites à partir d'un même ensemble de pièces. Ainsi, les

facteurs de branchement d'un état initial à un état final sont supérieurs aux facteurs de branchement de l'état final à l'état initial. Dans de telles conditions, une recherche inverse est plus efficace qu'une recherche directe pour résoudre le problème de séquencement de l'assemblage.

Ainsi le problème de l'assemblage est ramené à un problème de désassemblage comportant moins de confusions et donc plus simple à résoudre. L'équivalence entre les deux problèmes n'est pas toujours possible. En effet, puisque les opérations d'assemblage ne sont pas nécessairement réversibles, l'équivalence des deux problèmes ne tient que si les opérations utilisées en désassemblage, sont inverses d'opérations d'assemblage réalisables.

L'expression, "opération de désassemblage", se réfère dans notre cas, à l'inverse d'une opération d'assemblage faisable. Il est évident que les opérations non réversibles ne sont pas traitées dans ce cas.

II.3.2- Réalisation

La méthode de programmation graphique des robots d'assemblage proposée ici, comporte six étapes successives définies ci dessous:

1) Modélisation du processus d'assemblage

2) Représentation graphique de la base de travail

3) Montage automatique de l'assemblage

4) Désassemblage graphique des composants

5) Apprentissage des tâches et génération des plans

6) Validation

II.3.2.1 - 1ère Etape: Modélisation du Processus d'Assemblage

Pour pouvoir effectuer un apprentissage programmé, on doit modéliser le processus d'assemblage au niveau informatique. On modélise la base de travail: les robots, les composants, les produits, ... A chaque objet ou pièce réelle, correspond un modèle approprié et facile à manipuler par ordinateur.

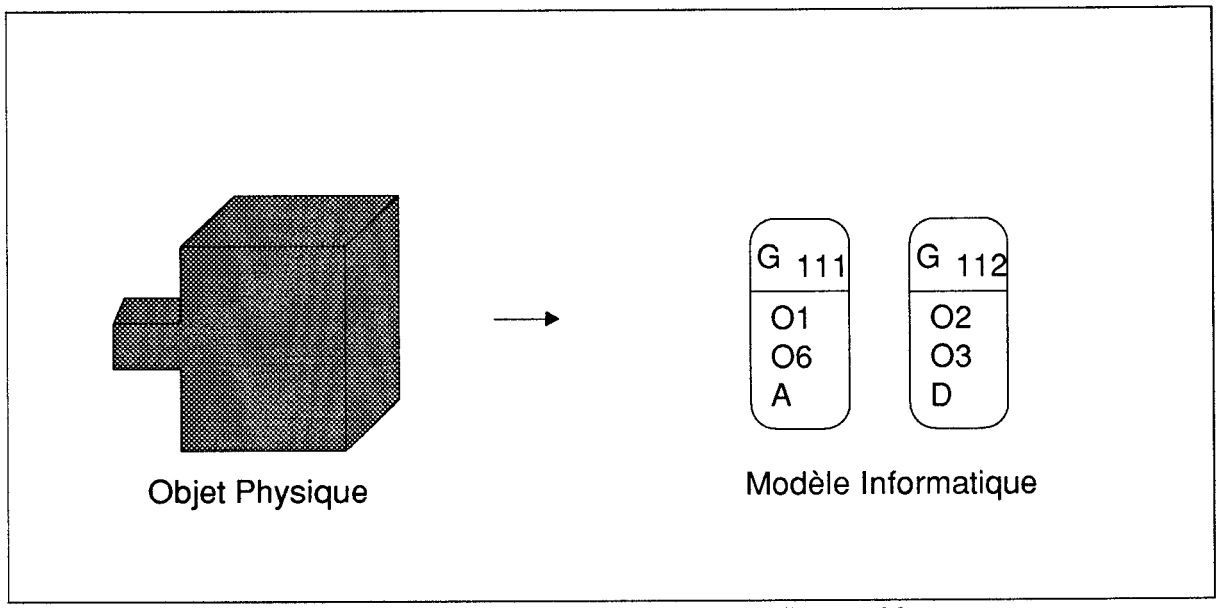


Fig II.3 - Modélisation de l'univers de l'assemblage

De plus, on modélise les tâches d'assemblage. Les tâches sont traduites par des expressions logiques et mathématiques qui peuvent être exécutées et contrôlées par calculateur. L'univers physique de l'application est alors transformé en un univers informatique simulé.

II.3.2.2 - 2ème Etape: Représentation Graphique de la Base de Travail

L'apprentissage graphique des robots nécessite la représentation de la base de travail sur l'écran d'un calculateur superviseur. Dans le cadre de l'assemblage, on doit mémoriser et représenter graphiquement les pièces ainsi que l'objet à assembler.

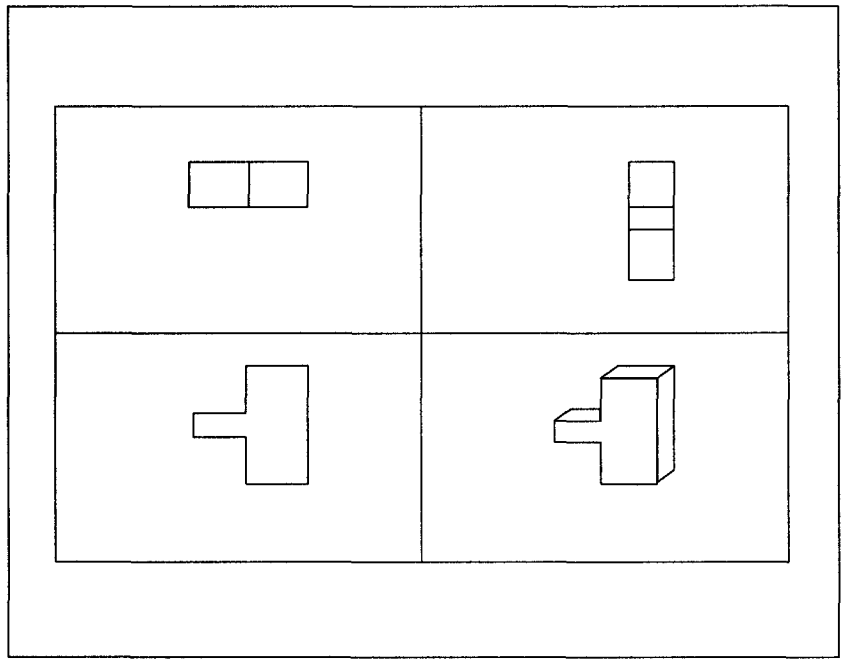


Fig. II.4 - La représentation graphique des objets

Chaque composant est présenté sur un écran, par les trois vues classiques du dessin industriel 2D, et par une quatrième vue perspective globale 3D. A cet effet, on découpe l'écran en quatre quadrants dans lesquels on visualise les quatre vues du composant simulé.

II.3.2.3 - 3ème Etape: Montage Automatique de l'Objet

Dans un troisième temps, on effectue le montage automatique des pièces sans se soucier des contraintes physiques. Les graphes représentants les pièces, seront alors assemblés sur l'écran, et l'objet final assemblé, est totalement défini.

II.3.2.4 - 4ème Etape: Désassemblage Graphique des Composants

On procède ensuite à l'apprentissage des trajectoires terminales par démontage graphique, en tenant compte des collisions. L'opérateur humain procède au désassemblage, en démontant les pièces une après l'autre, dans l'ordre convenable du démontage. En fait, les pièces assemblées

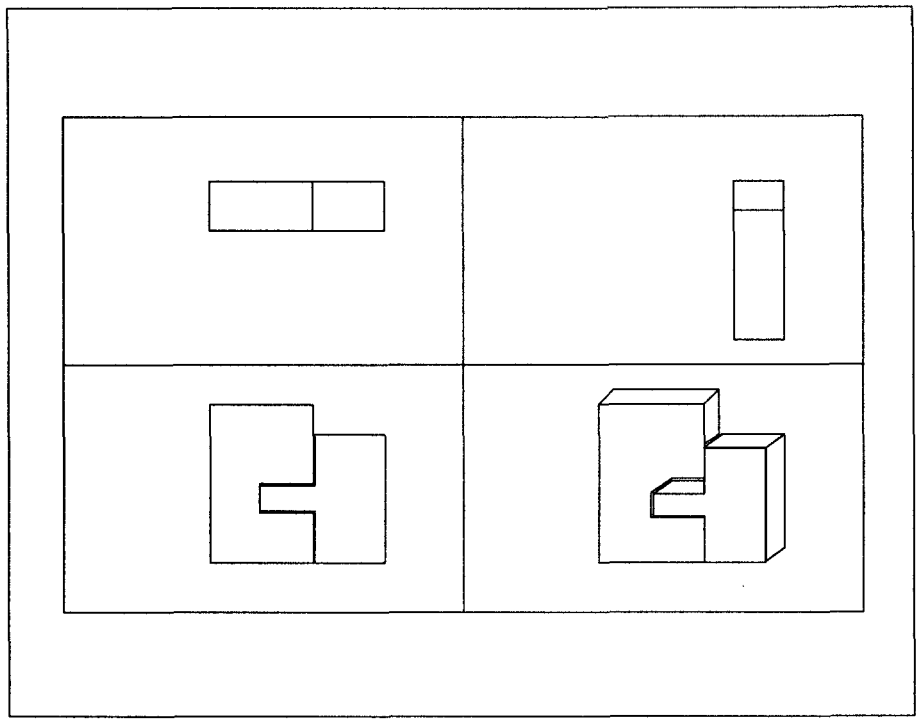


Fig II.5 - Le montage automatique

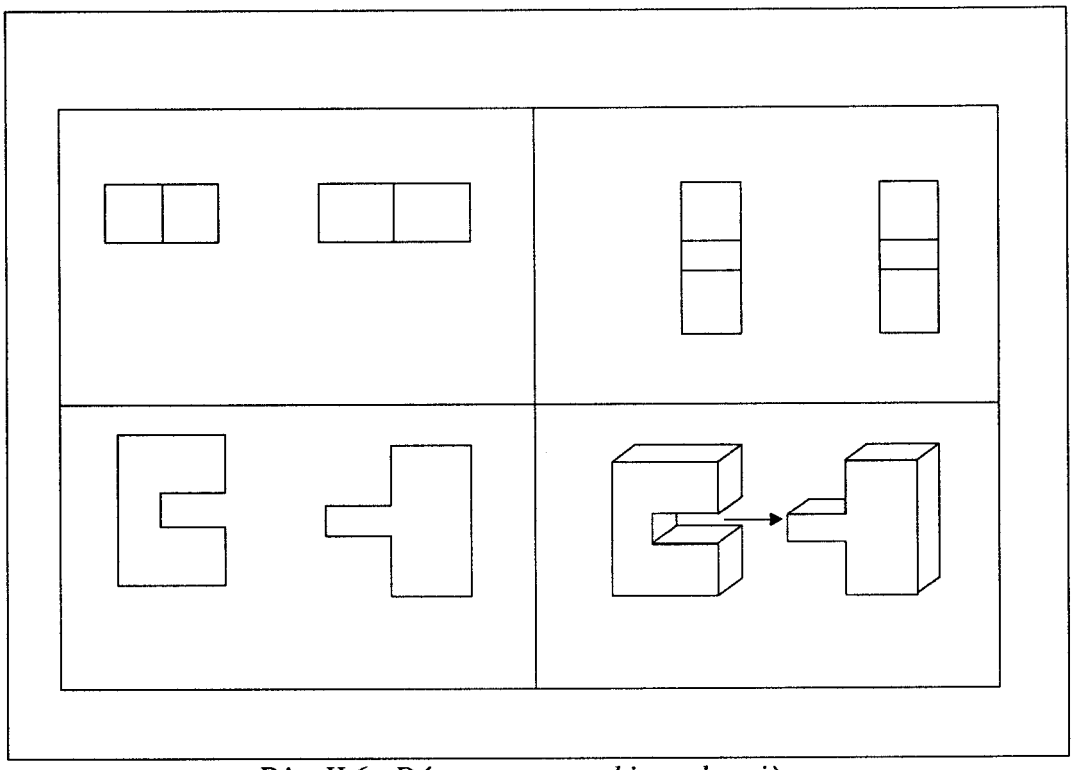
étant visualisées sur l'écran, l'opérateur lance des commandes de déplacement de la pièce à démonter. Ces commandes sont transmises et affectées graphiquement, sur les quatre quadrants de l'écran.

Dans cette phase, le calculateur contrôle toutes les commandes de déplacement lancées par l'opérateur. Il exécute les commandes qui sont valides, et rejette les commandes de déplacement ou d'exécution de tâches qui seront physiquement impossibles (collisions, mauvais positionnement, instabilité , ...). Ce démontage se fait de manière semi-automatique, révélant ainsi, une interactivité opérateur - superviseur.

Une fois le démontage final validé, le calculateur mémorise et sauvegarde les trajectoires de parcours, les points de passage, les points et les modes de préhension, ...

II.3.2.5 - 5ème Etape: Apprentissage des Tâches et Génération des Plans

On envisage alors un apprentissage graphique pour le robot. En effet, une simple inversion du cycle et des trajectoires de la phase d'apprentissage va permettre au générateur de plan, de



Dig. II.6 - Démontage graphique des pièces

programmer les tâches et les actions qui sont à exécuter par le robot.

II.3.2.6 - 6ème Etape: Validation

Une phase de validation, appliquée sur le modèle graphique, est finalement nécessaire avant de passer à l'exécution réelle du programme généré, sur les objets physiques.
Il est à noter enfin que, une bonne modélisation des pièces et des objets est nécessaire et exigée, pour s'assurer de la validité du transport des programmes d'exécution, du monde simulé au monde réel de l'application.

II.3.3 - Implantation Logicielle

La méthodologie de programmation graphique des robots d'assemblage proposée dans le présent travail est profondément basée sur l'interactivité opérateur - machine. En effet, et en se référant à la description de la méthode présentée ci dessus, le principe de cette méthode

consiste à permettre au calculateur "d'emprunter" l'intelligence humaine de l'opérateur, pour pouvoir exécuter l'application (programmation des tâches d'assemblage).

Pour cela, l'interactivité se présente non seulement comme un simple moyen d'échange d'informations, mais c'est un moyen de "coopération" à part entière entre deux systèmes de natures différentes (l'homme et la machine), et qui possèdent, chacun pour sa part, une certaine intelligence.

La réussite de notre méthodologie nécessite une coopération parfaite entre le système informatique et l'opérateur. L'élaboration d'un dialogue bidirectionnel aisé est donc primordial pour le bon fonctionnement du système:

* Dans le sens homme - machine, l'opérateur doit pouvoir transmettre à l'ordinateur son savoir faire humain (intelligence). L'ordinateur, pour sa part, doit pouvoir traduire, interpréter, contrôler et exécuter la pensée de l'opérateur. On dit qu'il se "synchronise" au rythme de l'intelligence de l'opérateur, pour faire évoluer l'état du système. La communication dans ce sens, doit être naturelle et orientée application.

* Dans l'autre sens, le dialogue machine - homme doit permettre à l'opérateur, d'une part de réagir et de valider l'état du système (étape 3: phase de montage automatique), et d'autre part d'expérimenter et de corriger ses actions (étape 4: phase d'apprentissage graphique). L'usage de moyens graphiques auto-explicatifs est le plus adapté à ce type de dialogue.

Ainsi, l'implantation logicielle de cette méthode de programmation graphique des robots d'assemblage nous amène à concevoir un logiciel interactif destiné à l'environnement industriel. La conception d'un logiciel interactif doit prendre en considération le dialogue homme - machine.

Les modèles d'architecture des systèmes informatiques interactifs séparent l'application propre, de son interface utilisateur. Cet interface devient l'image du système vue par l'utilisateur. Il assure la communication et l'échange des données entre l'application et l'utilisateur. Cette séparation se traduit par la modularité au niveau du développement logiciel.

La réalisation logiciel de systèmes interactifs, au niveau de l'interface utilisateur, reste en

grande majorité dominée par l'intuition du réalisateur. Cela est du à l'absence d'une méthodologie de conception ou d'un modèle de référence. Cependant, l'aspect interface homme - machine devient de plus en plus méthodique et cela avec l'apport des sciences cognitives et du génie logiciel.

Les travaux de recherche actuels dans le domaine des relations homme - machine essayent de formuler des méthodes et des modèles utiles pour la conception des systèmes interactifs. L'implantation du noyau de l'application étant traitée dans la deuxième partie du présent travail, on va présenter ci dessous l'aspect interface homme - machine.

II.4 - L'Interface Homme - Machine

la conception des systèmes interactifs doit exploiter les connaissances dans le domaine de l'interaction homme - machine. Cependant, deux sources de difficultés s'apparentent: la diversité de l'environnement utilisateur et la dispersion de l'expression des échanges.

Les travaux de recherche sur l'interface homme - machine sont traités parallèlement par les ergonomes, sous l'aspect des sciences cognitives, et par les informaticiens du génie logiciel.

II.4.1 - La Contribution des Sciences Cognitives

Les sciences cognitives abordent le concept d'interface homme - machine selon deux démarches : théorique et expérimentale. D'une part, les ergonomes essayent de formuler des modèles théoriques qui servent à prédire et à expliquer le comportement général cognitif de l'être humain face à l'utilisation d'un système informatique. D'autre part, ils suggèrent des solutions pratiques pour des cas bien cernés, et s'intéressent à l'évaluation prédictive des performances. Parmi les études théoriques qui ont sensibilisé l'informatique aux concepts de l'interface homme- machine, on peut citer:

- Le modèle du Processeur Humain [NEW 86]; ce modèle tente de formaliser les performances du sujet humain à l'aide de paramètres et de lois mathématiques.

- Le modèle GOMS [COU 88] qui essaye de décrire les mécanismes cognitifs de l'opérateur lors de l'accomplissement des tâches routinières. Il introduit la notion de décomposition d'une tâche en une hiérarchie de buts. L'inconvénient de ce modèle est qu'il ne traite pas le cas d'interruption et n'évoque pas la possibilité d'erreur de la part de l'opérateur.

- La Théorie de L'Action de D. NORMAN [NOR 86] qui distingue entre l'état effectif et l'état perçu du système. L'état effectif est une fonction portant sur des variables physiques, caractéristiques du modèle conceptuel du système. L'état perçu est la traduction de l'état effectif sous forme de variables psychologiques caractéristiques de la représentation mentale de l'utilisateur. La différence entre le monde physique et le monde mental met en évidence la nécessité pour l'opérateur d'effectuer des traductions. La Théorie de L'Action est un modèle explicatif du comportement.

- ACT [AND 83], qui est une théorie formelle qui s'appuie sur le formalisme et le principe des systèmes de production. Cette théorie s'attache à la représentation de la connaissance , la reconstitution mnésique, la résolution des problèmes et à l'apprentissage. ACT permet également de construire des modèles de simulation exécutables.

Les théories des sciences cognitives retiennent une approche descendante pour la conception des systèmes interactifs. Elles suggèrent de procéder en trois étapes:

- analyses des tâches
- définition du fonctionnement sémantique
- présentation du système.

J. COUTAZ a traité dans [COU 88] l'apport des sciences cognitives dans le domaine de la conception de l'interface homme - machine. Elle considère comme centrales à la méthodologie de conception:

- la notion de modèle mental
- la nécessité de combler la distance entre l'évaluation et l'exécution

- la décomposition des tâches en sous tâches dotées chacune d'un ensemble de variables psychologiques utiles à la conduite du travail
- elle propose d'ajouter le paramètre d'observation expérimentale.

SCHNEIDERMAN pour sa part, propose dans [SCH 86] une méthode de conception qui consiste à:

- définir un profil type utilisateur
- identifier les tâches réalisables les plus fréquentes
- choisir un type d'interaction qui satisfait les deux priorités établies ci dessus.

II.4.2 - Les Apports du Génie Logiciel

Parallèlement aux sciences cognitives, un ensemble de techniques informatiques s'est développé pour faire face aux besoins en matière d'interface homme - machine. Les travaux de recherche en génie logiciel et informatique humaine tentent de proposer des méthodes et de formuler des modèles de conception appropriés. Ainsi on trouve d'une part, un ensemble de méthodes et de modèles conceptuels pour traiter le problème, et d'autre part un ensemble d'outils logiciels qui en sont dérivés.

Les méthodes de conception des interfaces utilisateur décomposent souvent le problème en quatre niveaux:

- niveau conceptuel
- niveau sémantique
- niveau syntaxique
- niveau lexical.

Cette décomposition n'est pas toujours apparente, mais elle sert de base pour décrire et évaluer les différents modèles de conception. Les travaux de recherche sur l'interface homme - machine peuvent être vus sous trois axes:

- les architectures des systèmes interactifs

- les modèles de dialogue
- les outils de spécification des interfaces.

II.4.3 - Les Architectures des Systèmes Interactifs

Toutes les architectures des systèmes interactifs partent d'un modèle de base qui consiste à séparer l'application de son interface utilisateur. Les architectures qui dominent les interfaces développés actuellement s'inspirent du modèle de SEEHEIM [KAR 90]. Ce modèle divise l'interface en trois composantes qui communiquent entre elles successivement:

- la présentation
- le contrôleur de dialogue
- l'interface d'application.

Cependant, on peut regrouper les modèles d'architecture en deux catégories:

- les modèles Langages
- les modèles Multiagents.

II.4.3.1 - L'Architecture Langage

L'architecture Langage s'inspire du mode de dialogue entre individus. Ce mode s'appuie sur un langage qui permet d'externaliser la pensée. Le langage de dialogue entre le système et l'utilisateur est défini selon trois composantes:

- sémantique, définissant la signification des phrases et qui représente les concepts et le savoir faire à échanger.
- syntaxique, définissant la construction des phrases du langage à partir des éléments syntaxiques.
- lexicale, définissant la production des unités syntaxiques à partir d'un vocabulaire.

Cette modélisation sous forme d'un langage conduit à structurer un logiciel interactif en trois composantes logiques. On retrouve le modèle de SEEHEIM (fig. II.7):

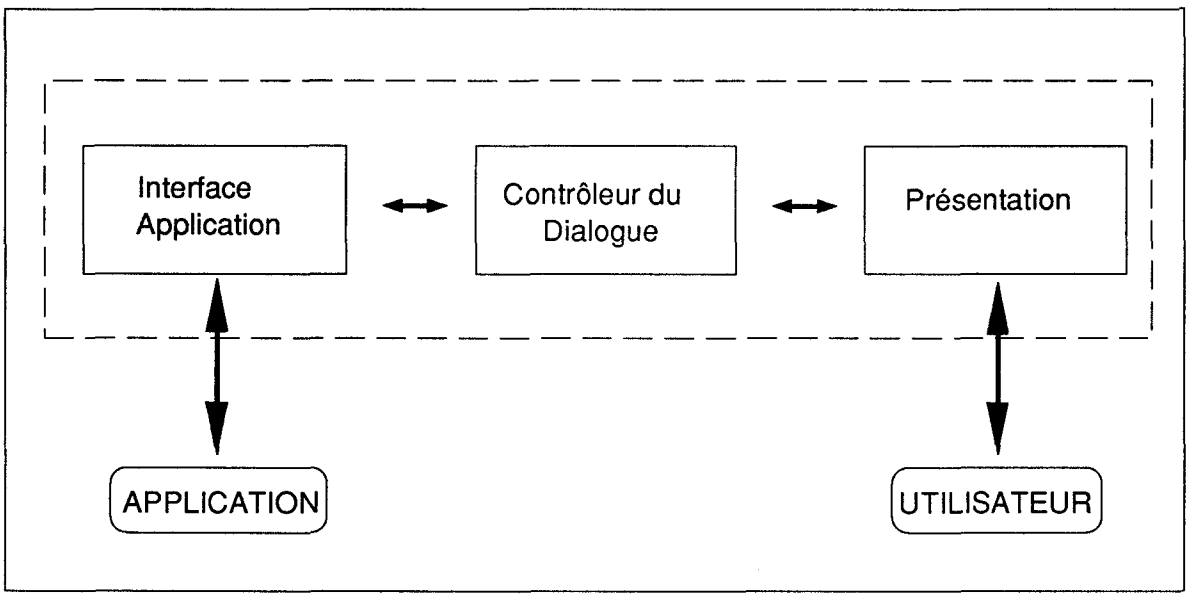


Fig. II.7 - Le modèle de SEEHEIM

* **L'Interface Application**: il définit la sémantique du langage d'interaction et le protocole de communication entre l'application et le Contrôleur de Dialogue.

* **Le Contrôleur de Dialogue**: c'est le médiateur entre la présentation et l'Interface Application. Il effectue l'analyse syntaxique du langage d'interaction.

* **La Présentation**: responsable de la mise en oeuvre de l'image du système, elle effectue l'analyse lexical et convertit les informations entre le format du monde physique externe et celui du monde informatique interne.

Le modèle Langage présente un cadre de pensée simple. Il nécessite un fonctionnement séquentiel. Cependant, ses principaux inconvénients sont [COT 90]:

- l'ordre de priorité, qui est donné à la forme de l'information
- la centralisation du traitement
- le découpage du langage d'interaction en deux sous-ensembles distincts.

II.4.3.2 - L'Architecture Multiagent

Le modèle multiagent s'inspire du fonctionnement des systèmes de type stimuli-réponses. Ce modèle structure le système interactif en un ensemble d'agents spécialisés qui réagissent à des événements et qui produisent des événements. Un agent est une entité physique ou abstraite qui est capable d'agir sur elle même et sur son environnement en communiquant avec d'autres agents, et dont le comportement est la conséquence de ses observations, de sa connaissance et des interactions avec les autres agents [FEG 88].

L'agent possède des récepteurs et des émetteurs et dispose d'une mémoire à deux niveaux: l'une sert de tableau d'informations pour mémoriser son état, et l'autre sert à enregistrer les événements détectés, dans un tableau d'offres.

Le modèle multiagent se caractérise par une organisation fortement modulaire, un parallélisme dans le traitement et une communication par événements. Il permet un dialogue multiple à plusieurs fils et assure la possibilité d'effectuer des traitements distribués. L'abstraction de ce modèle le rapproche des modèles de programmation orientée objet.

Parmi les modèles de conception d'interfaces, possédant une architecture multiagent, on peut citer le modèle PAC [COU 88] qualifié de modèle visuel de conception d'interfaces [WAN 91], MVC [GOL 84], et GWUIMS [SIB 86].

II.4.4 - Les Modèles de Dialogue

Le modèle de dialogue homme - machine doit définir les propriétés et les caractéristiques des méthodes de conception d'interfaces. Il vise à rendre l'expertise ergonomique disponible et utilisable par les concepteurs des systèmes interactifs.
Un modèle de dialogue est utilisé pour la génération d'une interface à partir des spécifications de haut niveau. Cependant, d'autres fonctionnalités théoriques se dégagent d'une telle modélisation, comme la formalisation des recommandations ergonomiques et de la notion de style d'interface, ou même l'évaluation d'une interface existante [MIC 90].

Le premier modèle du dialogue était CLG développé par T. MORAN [MOR 81]. Le modèle définit une représentation des étapes de conception d'un système interactif et décrit la représentation mentale que l'utilisateur a du système. Il est basé sur une structure linguistique qui permet de représenter un système informatique à plusieurs niveaux d'abstraction:

* La partie Conception englobe les niveaux Tâche et Sémantique
* La partie Communication contient les niveaux Syntaxe et Interaction
* La partie Physique décrit les niveaux Présentation et Unités E/S.

Les principaux inconvénients de ce modèle sont:

- impossibilité de spécifier le parallélisme entre activités
- manque de modélisation des constituants graphiques de l'interface
- manque de représentation des caractéristiques de l'opérateur.

Ce modèle est pratiquement peu utilisé. Les travaux de recherche s'orientent actuellement vers des modèles dits compréhensifs. Un tel modèle peut se voir comme une classe générique abstraite dont les instances permettent de bâtir des systèmes interactifs donnés. L'idée générale est que tout interface doit pouvoir être décrit comme une instance particulière d'un modèle compréhensif.

Les modèles compréhensifs incluent la modélisation des composantes graphiques du dialogue. Dans un tel modèle, la représentation des tâches de l'opérateur se fait de manière hiérarchique. Les tâches sont décomposées en sous tâches et doivent pouvoir être temporellement ordonnées. Le modèle compréhensif supporte la possibilité d'exprimer des tâches parallèles ou séquentielles. Il représente des systèmes interactifs à contrôle mixte: les tâches à manipuler proviennent également de l'opérateur ou du système.

II.4.5 - Les Spécifications des Interfaces

Les outils de spécification portent essentiellement sur les aspects syntaxiques et lexicaux de l'interface. On distingue trois orientations [KAR 90]:

- la syntaxe du dialogue
- la géométrie de l'interface
- le déroulement du dialogue.

Les langage de spécification d'interfaces utilisent diverses techniques:

* **Les grammaires**: elles permettent de spécifier la syntaxe du dialogue. Les grammaires sont efficaces pour décrire des interfaces de type textuel, par contre, elles sont moins adaptées pour des interfaces graphiques. DCG présenté dans [WEI 88] est un exemple de ce type.

* **Les automates**: de tailles grandes, ils présentent l'avantage de visualiser l'enchaînement d'actions autorisées. On peut citer dans cette catégorie l'utilisation des réseaux de transition appelés RTN (Recursive Transition Network) ou des Réseaux De Pétri (RDP) [WAN 91].

* **Les langages spécifiques**: ils décrivent des interfaces de type formulaire et ont l'avantage de proposer des schémas d'interaction standard.

* **Les modèles à événements**: ils sont basés sur le concept des événements. Les événements peuvent être générés par l'utilisateur ou même par le dialogue lui même. L'intérêt de ce formalisme est la possibilité de décrire le parallélisme et de gérer les dialogues multiples [KAR 90] [WAN 91]. X-WINDOW, ESTEREL présenté dans [GER 88] et ALGEA [FLE 87] utilisent des modèles à événements.

* **Les spécifications interactives**: elles consistent à construire interactivement et graphiquement des interfaces graphiques. C'est l'approche la plus populaire. On trouve plusieurs outils de spécification d'interfaces de ce genre, ces outils sont connus sous le nom d'éditeurs d'interface. On peut citer: GRAFFITI développé par S. KARSENTLY [KAR 87], SOS INTERFACE [HUL 86], MASAI [MAS 89], l'éditeur de VITAMIN [WAN 91] [SOE 88], EGERIE développé par BULL en collaboration avec l'INRIA, ..

II.4.6 - Synthèse

La réalisation de systèmes interactifs dispose aujourd'hui de modèles d'architecture et d'outils. Les modèles proposés ne sont pas encore assez formels pour être facilement appliqués. Il reste encore une distance à franchir entre les concepts théoriques de l'interface et leur intégration dans un outil. Les outils les plus prometteurs sont les éditeurs d'interfaces, mais leur utilisation n'est pas encore banalisée.

Dans ce contexte, définir son propre interface utilisateur, ou même remettre en cause le principe de séparation entre l'application propre et son interface (comme le suggère S. KARSENTLY dans [KAR 90]) reste une alternative envisageable. Cependant, la formalisation de démarches ergonomiques de conception d'interfaces prouve son intérêt et apporte un grand appui aux réalisateurs de systèmes interactifs. On note en particulier les suggestions de C. KOLSKI dans [KOL 89], D. FASSOTTE dans [FAS 86] et P. MILLOT dans [MIL 87] qui portent sur l'utilités des interfaces graphiques pour le dialogue homme - machine.

Dans le cadre du présent travail, une solution réaliste et pratique consiste à développer notre propre interface utilisateur, tout en s'inspirant des modèles visuels de conception, et en pratiquant une démarche ergonomique de conception d'une interface graphique de communication homme - machine. Une approche orientée objet est la plus appropriée pour l'implantation logicielle de notre méthodologie de programmation graphique des robots d'assemblage.

II.5 - Conclusion

Les robots d'assemblage nécessitent une phase d'apprentissage avant de pouvoir exécuter les tâches de montage des pièces.
Cette phase doit permettre au robot de définir:

- les positions initiales des pièces
- les modes de préhension
- les trajectoires de parcours
- les positions finales des pièces au sein de l'assemblage.

Cette phase d'apprentissage peut se faire En-ligne (directement sur le robot), ou Hors-ligne (sans immobilisation du robot). Cependant, la programmation Hors-ligne présente des avantages économiques et techniques.

La programmation par simulation graphique est un domaine de programmation Hors-ligne des robots d'assemblage. On a présenté dans ce chapitre le schéma de la méthode proposée de programmation graphique des robots d'assemblage. Cette méthode s'appuie fortement sur la coopération opérateur - machine.

La conception de l'interface homme - machine devient de plus en plus méthodique. Les travaux de recherche tentent de formuler des modèles et de proposer des outils d'aide à la conception de ces interfaces.

Nous présentons dans la seconde partie de cette étude, le détail de la méthodologie de programmation graphique des robots d'assemblage, en deux chapitres:

- chapitre trois: modélisation, structuration des données et représentation graphique
- chapitre quatre: montage automatique et apprentissage

2ème Partie

CHAPITRE III

MODELISATION ET STRUCTURATION DES DONNEES

III.1 - Introduction

Ce chapitre présente la modélisation et la structuration des données, pour un système d'apprentissage graphique des robots d'assemblage.

Dans un premier temps, on définit la modélisation informatique et on décrit les trois niveaux de modélisation: le niveau physique, le niveau mathématique et le niveau représentation. On aborde ensuite la modélisation solide, qui est une branche de la modélisation informatique, en présentant les différentes approches utilisées. Un tel type de modélisation est en effet choisi pour modéliser l'univers de l'assemblage.

On introduit alors la méthode proposée de modélisation du processus d'assemblage. Cette modélisation est basée sur l'approche objets. On formule un modèle mathématique pour les produits et les composants élémentaires. Un assemblage est décrit en effet, comme un ensemble ordonné de plusieurs pièces élémentaires, qui sont connectées par des règles de positionnement. On modélise la pièce par un ensemble de formes géométriques. Les formes sont modélisées par des paramètres quantitatifs et qualitatifs, qui tiennent compte de la géométrie, du volume et de la matière solide de la forme.

Finalement, on définit des structures de données arborescentes et indicielles pour manipuler et sauvegarder d'une manière uniforme, le processus d'assemblage. Un exemple de modélisation est donné en fin de chapitre pour illustrer la méthode utilisée.

III.2 - La Modélisation Informatique

Un modèle est un objet artificiel construit dans le but de simplifier l'observation d'un objet réel. L'intérêt de la modélisation réside dans le fait que certaines caractéristiques peuvent mieux être examinées sur le modèle que sur l'objet physique.

Un modèle informatique consiste en un ensemble de données emmagasinées en mémoire ou dans un fichier de l'ordinateur. Le but de ce modèle est d'examiner et de manipuler l'objet réel à l'aide du calculateur. La modélisation est vue comme une transformation du monde réel physique vers le monde informatique. Cette vue de la modélisation présente trois niveaux [REQ 80], [HOP 83], [GUE 86] (Fig. III.I):

- le niveau physique
- le niveau mathématique
- le niveau représentation

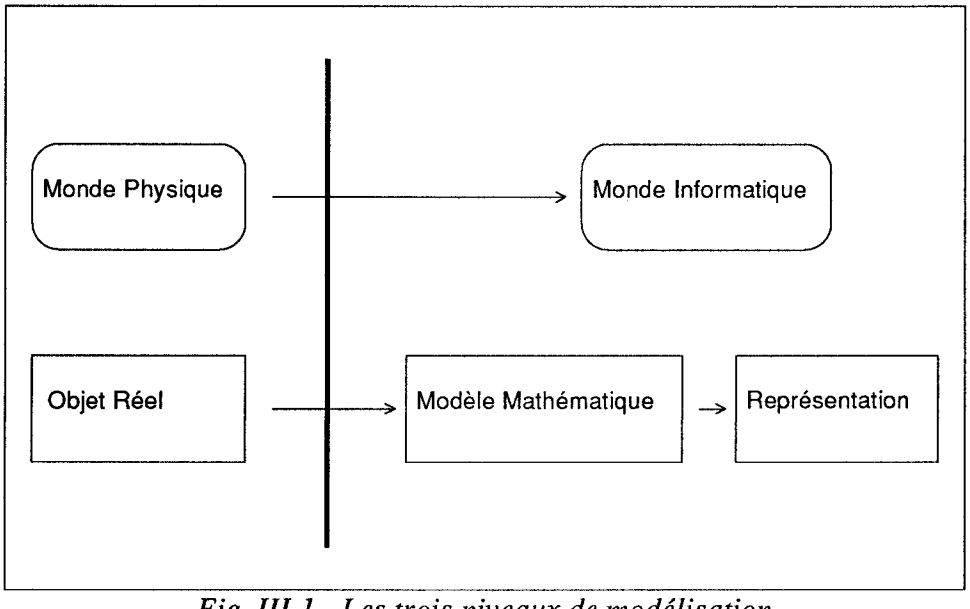


Fig. III.1 - Les trois niveaux de modélisation

III.2.1 - Le Niveau Physique

Le but du modèle est de matérialiser des objets réels tridimensionnels. Malheureusement, on ne peut jamais percevoir un objet réel dans sa totale complexité et dans tous ses détails.

III.2.2 - Le Niveau Mathématique

Pour pouvoir créer un modèle informatique, on doit adopter à une idéalisation de l'objet physique tridimensionnel. Cette idéalisation de l'objet doit d'une part, avoir une connexion avec le monde réel de l'objet et d'autre part, elle doit permettre d'assigner une représentation de cet objet au niveau d'un ordinateur.

Pour cela, on doit caractériser rigoureusement l'espace des objets à modéliser. Une telle caractérisation peut se faire par l'utilisation des concepts mathématiques de la théorie des points et de la topologie algébrique.

III.2.3 - Le Niveau Représentation

Un autre aspect de la modélisation consiste à assigner à l'objet mathématique construit, une représentation convenable pour la manipulation informatique. Cette vue tridimensionnelle permet de caractériser les différentes méthodes de modélisation en analysant les relations entre l'objet mathématique crée et la représentation par ordinateur.

III.3 - La Modélisation Solide

La modélisation solide est une branche de la modélisation informatique, qui fournit tous les éléments nécessaires à une représentation complète des objets physiques solides (fig. III.3). Cette représentation permet de répondre algorithmiquement et sans intervention d'un

opérateur humain, à toutes les questions sur la manipulation de l'objet [MAN 88].

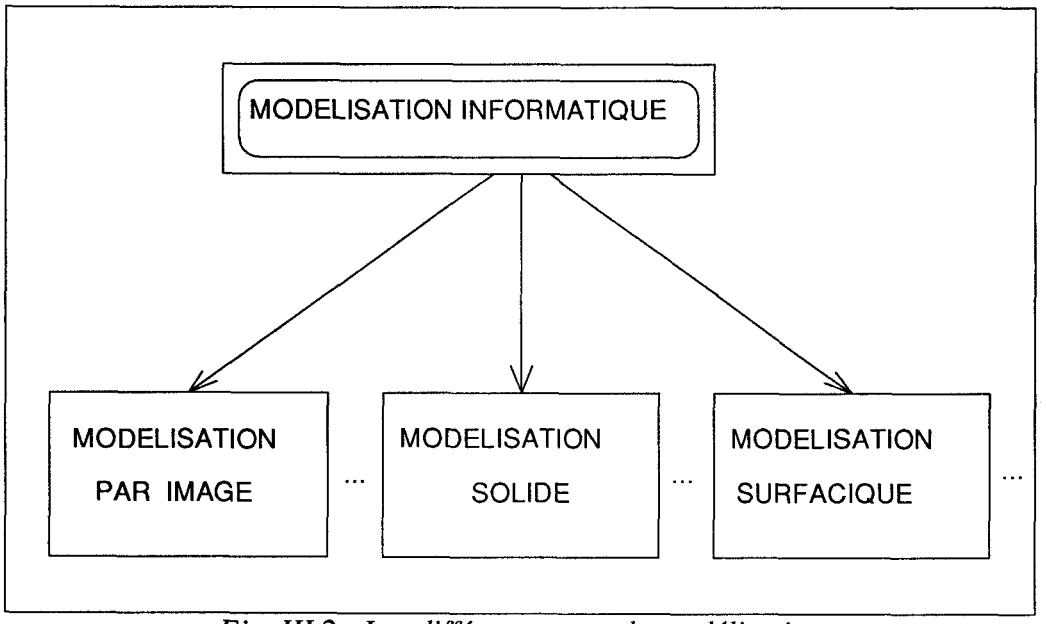


Fig. III.2 - Les différents types de modélisation

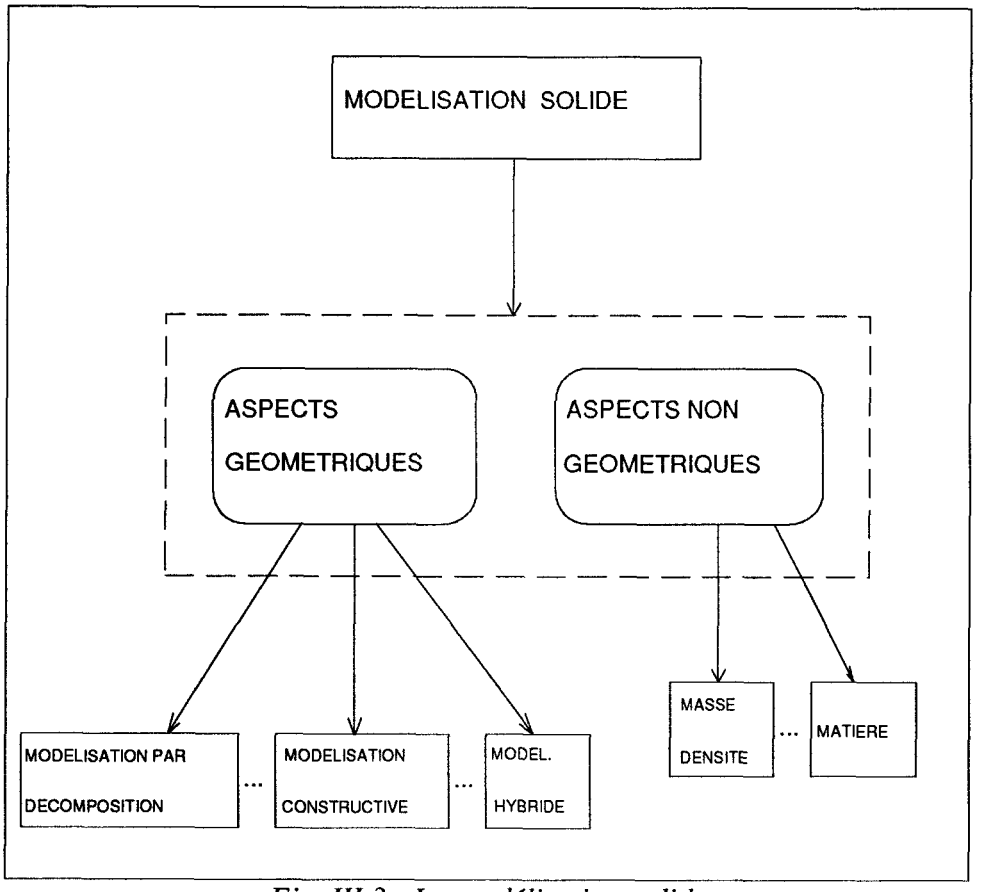


Fig. III.3 - La modélisation solide

Il existe d'autres types de modélisation qui sont plus ou moins orientés vers des applications spéciales. Parmi les autres types de modélisation, on peut citer:

* la modélisation par image

  Ce type de modélisation est utilisé dans le système PRATIC [BIE 87] présenté dans le chapitre deux (sect. II.2.2.3). Il est également adopté par A. ROBERT qui propose dans [ROB 86] une approche de modélisation tridimensionnelle de l'environnement d'un robot mobile par stéréovision. Il faut noter encore que le système HANDEY (introduit en chapitre deux, sect. II.2.2.2) utilise des capteurs Laser pour mesurer les arètes des objets.

* la modélisation surfacique

  Largement utilisé dans les systèmes actuels de programmation graphique des robots. Le système SYNAPSE (sect. II.2.2.3) utilise la modélisation surfacique pour décrire l'univers du robot.

La modélisation solide révère un caractère beaucoup plus général et fournit donc un champs d'applications plus large.

III.4 - La Modélisation Géométrique

La géométrie d'un objet contient une part importante d'information sur cet objet [REQ 77], [MAN 88]. En plus, les techniques de mémorisation et de manipulation des données géométriques ont un caractère général et ne dépendent pas de l'application envisagée. Dans ce sens, il est utile de distinguer entre les données géométriques d'un objet et les données qui modélisent des aspects non géométriques de l'objet.

On appelle modèle de l'objet, la totalité des données qui le caractérisent. En revanche, la partie purement géométrique des données constituent le modèle géométrique. Evidemment, un modèle géométrique est un sous ensemble ou sous modèle du modèle de l'objet. On trouve dans la littérature plusieurs approches pour la modélisation solide géométrique. On présente ci dessous les approches majeures [REQ 77], [TIL 81], [ANS 85], [MAN 88].

III.4.1 - La Modélisation par Décomposition

Un solide est décrit par une combinaison de blocs élémentaires collés ensemble.

III.4.2 - La Modélisation Constructive

Dans ce genre de modélisation, on utilise, des sous objets ou des formes élémentaires, stockés en mémoire. Les primitives utilisées pour modéliser un objet, sont uniquement des points.

III.4.3 - La Modélisation par Frontière

Dans ce type de modélisation, un solide est représenté indirectement par ses surfaces de bord. Ce type de modélisation s'approche de la modélisation surfacique présentée ci dessus. LMAD-G (chap. II, sect. II.2.2.3) utilise ce type de modélisation pour décrire les trajectoires du robot.

III.4.4 - La Modélisation Hybride

Puisqu'aucune des approches citées ci dessus, ne présente en elle seule une solution complète de la modélisation solide, on a recours parfois à des représentations multiples pour des systèmes de modélisation complexes. Un modèle hybride est capable de supporter plusieurs représentations solides coexistentes. Il essaye d'utiliser la portion la plus convenable pour chaque tâche. Ces différentes représentations doivent être consistantes. Cette consistance est matérialisée par des algorithmes de conversion entre les représentations. Le système ARES (présenté en chapitre deux, sect. II.2.2.3) procède à ce type de modélisation.

III.5 - La Modélisation du Processus D'Assemblage

La modélisation du processus d'assemblage est nécessaire pour l'implantation informatique. Cette modélisation doit fournir les informations suivantes:

- description géométrique et physique des objets et du robot [BRA 78], [BAE 79]
- description cinématique des liaisons fonctionnelles [MAS 81]
- description des caractéristiques et des contraintes du robot [WHI 82]
- description de l'environnement périrobotique [BIE 87]
- description des tâches en termes de relations géométriques.

La modélisation tridimensionnelle de l'univers est l'une des limitations majeures des systèmes de programmation des robots. En effet, presque la majorité des systèmes actuels se contentent d'une modélisation bidimensionnelle de l'espace (SYNAPSE, PRATIC, ..). Par la force des choses, ces systèmes ne sont pas capable de présenter des solutions "précises" pour des applications réelles de robots qui opèrent dans le monde physique tridimensionnel. Cette contrainte s'accentue de plus en plus lorsque la précision devient un critère essentiel à prendre en considération. C'est en effet le cas pour la programmation de processus d'assemblage fin de produits.

Une deuxième catégorie de systèmes proposent des modèles tridimensionnels, mais sont limités aux objets polyédriques. C'est le cas du modèle SIMONS utilisé par les systèmes SHARP et PAMELA pour modéliser l'univers (sect. II.2.2.2 et II.2.2.3). SIMONS en effet modélise les objets de l'univers par des sommets, arètes et faces. Si cette méthode marche pour les objets polyédriques, elle n'est pas cependant capable de modéliser correctement des surfaces non bornées (une sphère par exemple). Le système propose dans ce cas de compléter les données par les surfaces de bord de l'objet considéré [THE 88] (une sphère est donc modélisée par un cube!). Le même problème est rencontré dans le système SMGR proposé par J. TROCCAZ dans [TRO 85], qui modélise un objet par une liste de faces polygonales [LAU 87].

Pour faire face à ce problème de modélisation, une autre alternative a été adoptée par les chercheurs de l'Université de Lisbonne et ceux de l'Université de Karlsruhe - RFA. Ils ont

conçus des systèmes de programmation de robots d'assemblage qui ne disposent pas de modèle propre de l'univers, mais qui s'appuient sur les modèles de conception CAO des produits de la base de travail [MOU 90].

Pour notre part, nous nous proposons dans le présent travail, de formuler une modélisation tridimensionnelle propre à l'univers des robots d'assemblage. Un modèle mathématique pour définir et mémoriser la base de travail du robot d'assemblage est présenté ci dessous. La méthodologie d'implantation informatique de ce modèle est basée sur l'Approche Objets. Cette modélisation est donc exprimée en terminologie compatible avec un langage orienté objets.

Rappelons que l'Approche Objets permet de réaliser l'abstraction des données en introduisant la notion d'encapsulation. La stratégie d'une telle approche amène à structurer l'univers d'une application en terme d'objets plutôt qu'en terme traditionnel de procédures. L'univers de l'application est par conséquent composé d'un ensemble d'objets qui détiennent, chacun, les clés de leur comportement.

Ces objets sont regroupés, selon leur type, en classes. Une classe s'apparente à un type abstrait de données et décrit une famille d'objets ayant la même structure et le même comportement. On peut dire que la classe est l'entité conceptuelle qui décrit l'objet et qui sert de modèle pour construire des représentants physiques appelés "instances". Une instance est donc un objet particulier qui est créé en respectant les plans de constructions de sa classe.

Dans ce sens, chaque objet de l'univers est généré à partir d'une classe d'appartenance qu'il connait grâce à une relation d'instanciation. Informatiquement, chaque classe possède deux composantes:

- une composante statique: les "Attributs" qui caractérisent l'état des objets de la classe.
- une composante dynamique: les "Méthodes" qui caractérisent les actions pouvant être effectuées par les objets de la classe.

A l'instar de ce modèle, la définition d'une classe comprend deux parties: la définition des attributs d'une part, et la définition des méthodes d'autre part. On shématise une classe, par le graphe de la figure III.4.

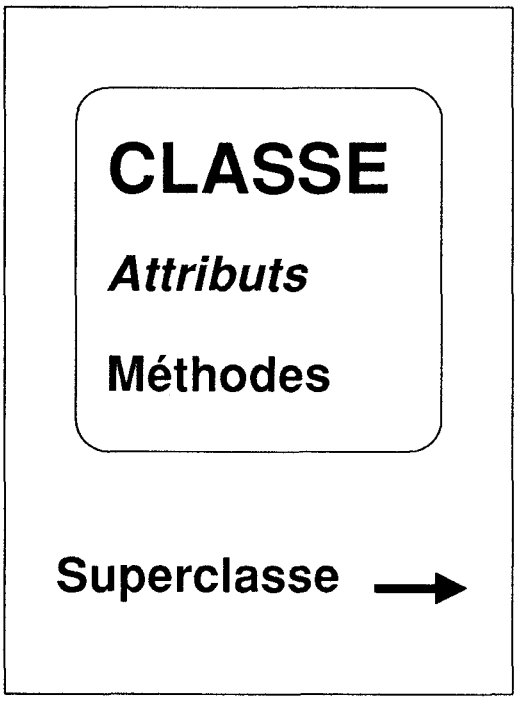


Fig. III.4 - La notion de " Classe " en langage objets

Les langages à objets modélisent l'univers par des classes organisées hiérarchiquement. L'ensemble des classes de l'univers forme en effet une hiérarchie dans laquelle chaque classe "hérite" des attributs et des méthodes de ses classes mères appelées "Superclasse". La relation d'héritage est transitive: les caractéristiques des classes supérieures sont transmises à toutes les classes hiérarchiquement inférieures.

Ainsi, pour modéliser un univers par approche objets, il suffit de définir les classes de cet univers et les relations d'héritage qui lient l'ensemble de ces classes entre elles. En revenant au processus d'assemblage, on va modéliser l'univers par des classes. L'espace de travail (l'assemblage, les pièces élémentaires, ...) est en effet représenté par des entités informatiques mémorisables. Une structure de données arborescente est ensuite envisagée.

III.5.1 - Définition de l'Espace de Travail

L'espace de travail E, est un espace géométrique tridimensionnel. Cet espace est repéré par un

repère de base T_0, orthonormé. Ce repère est désigné par la terminologie "repère absolu". Dans cet espace de travail, on peut définir plusieurs repères de référence.

Un repère T est défini dans E par une matrice M (4*4) homogène. Cette matrice est la matrice de passage du repère T au repère absolu T_0. Un tel repère peut être mobile: il peut se déplacer dans l'espace E. Dans ce cas, sa matrice M = M (X, Y, Z) évolue en fonction de la position instantanée de ce repère, par rapport au repère absolu. La classe Repère sera définie ultérieurement.

III.5.2 - Représentation de l'Assemblage

L'assemblage A_M est défini par un ensemble ordonné de pièces muni de règles de positionnement.

$A_M = \{ P_i \ ; \ i = 1, .., M \} \cup \{$ règles de positionnement $\}$

- M: nombre de pièces élémentaires constituant l'assemblage.
- i: indice des pièces, qui prend sa valeur sur l'ensemble des entiers naturels inférieurs ou égaux à M.
- P_i: pièce élémentaire d'ordre i. Les pièces étant ordonnées dans l'ordre du montage de l'assemblage. Cet ordre est imposé par l'application envisagée de l'assemblage. La classe "Pièce" sera définie ultérieurement.

Les règles de positionnement sont définies comme suit:

- l'assemblage A_M est affecté d'un repère Γ dont le centre se trouve à la position finale de l'assemblage.

- chaque pièce P_i est affectée d'un repère Γi. Ce repère est défini comme un repère mobile relié à la pièce correspondante et dont le centre est positionné relativement à Γ.

- les règles de positionnement se traduisent alors par des matrices de passage Hi , i = 1,..,M définissant les matrices de transfert de Γi (en position finale), par rapport à Γ.

Ces matrices sont nécessaires pour trouver les positions relatives finales des P_i par rapport à Γ.

Ainsi, on peut formuler A_M mathématiquement, de la façon suivante:

$$A_M = \{ (P_i, H_i) \ ; \ i = 1, .., M \} \qquad (1)$$

Cette formulation se traduit en terminologie "Objets" par la génération d'une classe appelée classe "Assemblage" (fig. III.5). Cette classe modélise et définit un assemblage réel quelconque.

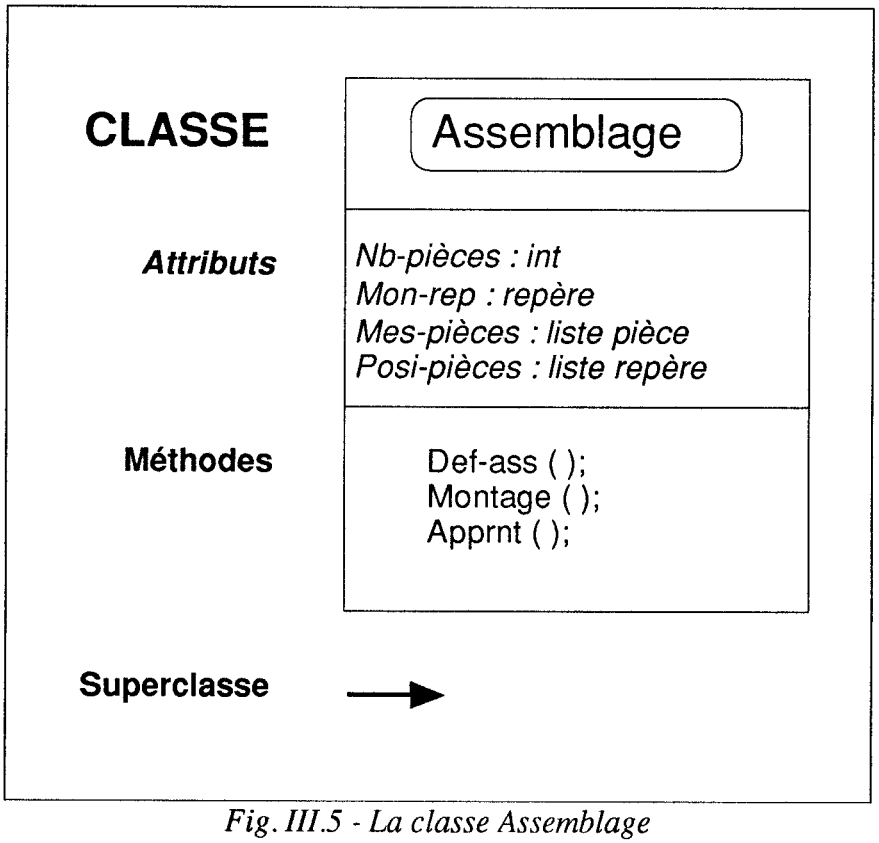


Fig. III.5 - La classe Assemblage

Les attributs de cette classe sont:

- Nb-pièces = M, le nombre de pièces dans l'assemblage.
- Mon-rep = Γ, le repère associé à l'assemblage. La classe "Repère" est définie ci après.
- Mes-pièces: liste énumérant toutes les pièces P_i de l'assemblage.
- Posi-pièces: liste énumérant les matrices de positionnement H_i, associés aux pièces P_i de l'assemblage.

Les méthodes associées à cette classe sont:

- Def-ass (): cette méthode définit la gamme d'assemblage choisie en indiquant les pièces correspondantes, l'ordre du montage des pièces, et en implémentant les règles de positionnement définies et mentionnées ci dessus.

- Montage (): effectue le montage des pièces, de manière semi automatique, en vue de préparer la phase d'apprentissage graphique. Cette méthode est détaillée dans le chapitre suivant.

- Apprnt (): c'est la phase d'apprentissage graphique utilisant le démontage interactif des pièces, présenté dans le chapitre suivant.

III.5.3 - Les Pièces

Une pièce est composée d'un ensemble de formes géométriques élémentaires, chacune de ces formes étant dotée d'une signature:

$$P_i = \{ G_{ij} \ ; \ j = 1, .., g_i \} \cup \{ \text{les signatures des } G_{ij} \}$$

- g_i : nombre de formes élémentaires dans la pièce P_i
- j: indice des formes, son domaine est [1 .. g_i]
- G_{ij}: la jème forme élémentaire de P_i. La classe "Forme" est définie ci après.

La signature associée à une forme permet de modéliser des aspects non géométriques de cette forme. Ainsi on peut associer un signe algébrique (+ ou -) à chaque forme, pour différencier les formes pleines des cavités et par suite modéliser la notion de présence/absence de matière

dans une forme.

Il est évident que d'autres aspects peuvent être modéliser et pris en compte par l'intermédiaire de cette signature, comme par exemple, la masse de la forme, la densité de matière, l'élasticité,...

L'utilisation de cette signature est étroitement liée à l'application spécifique envisagée. Dans le cadre du présent travail, on se contente d'utiliser cette notion de signature pour modéliser la présence ou l'absence de matière, en associant à chaque pièce un signe algébrique.

En général, une pièce comporte deux types de zones (fig. III.6):

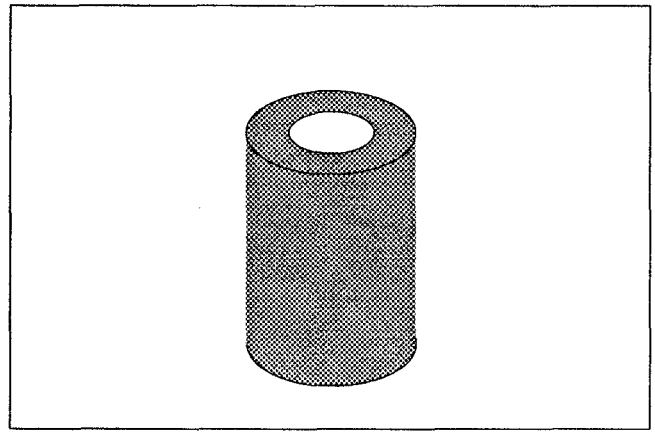


Fig. III.6 - Une forme peut présenter des cavités

- une zone de matière solide (présence de matière)
- une zone vide représentée par les cavités (absence de matière). Il est évident qu'une zone de ce type n'a de sens que si elle se trouve à l'intérieur d'une zone du premier type.

Ces deux zones sont modélisées géométriquement par les formes élémentaires G_{ij}, déjà citées. De plus, pour mettre en évidence cette notion de matière, on procède comme suit:

* on modélise la présence de matière dans une forme par l'affectation d'un signe + à cette forme.

* on introduit la notion de matières négatives pour les cavités, qui sont des zones auquelles on a retranché de la matière, partant du constat que:

A - B = A + (-B).

* on modélise donc la forme contenant de la matière négative, par l'affectation d'un signe négatif (-).

* une cavité est alors modélisée par la somme de deux formes (fig. III.7):

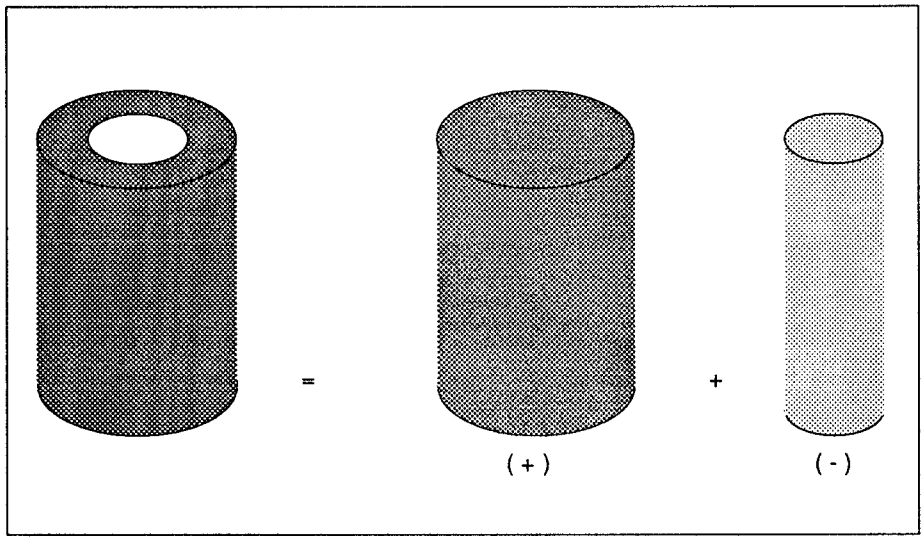


Fig. III.7 - La notion de matière négative

- une forme + qui modélise l'enveloppe de cette cavité
- une forme - qui modélise une zone imaginaire contenant de la matière négative.

La signature d'une forme G_{ij}, se traduit par un indice algébrique n_{ij} associé à cette forme. On peut donc, définir mathématiquement une pièce, par la notation suivante:

$$P_i = \{ (G_{ij}, n_{ij}) \ ; \ j = 1, .., g_i \} \qquad (2)$$

- n_{ij}: signature de la forme G_{ij}, appartenant au domaine { +, - }

Selon la terminologie "Objets" , une pièce est définie et générée à partir d'une classe "Pièce" ,

qui est elle même une sous classe de la classe "Assemblage" (fig. III.8)

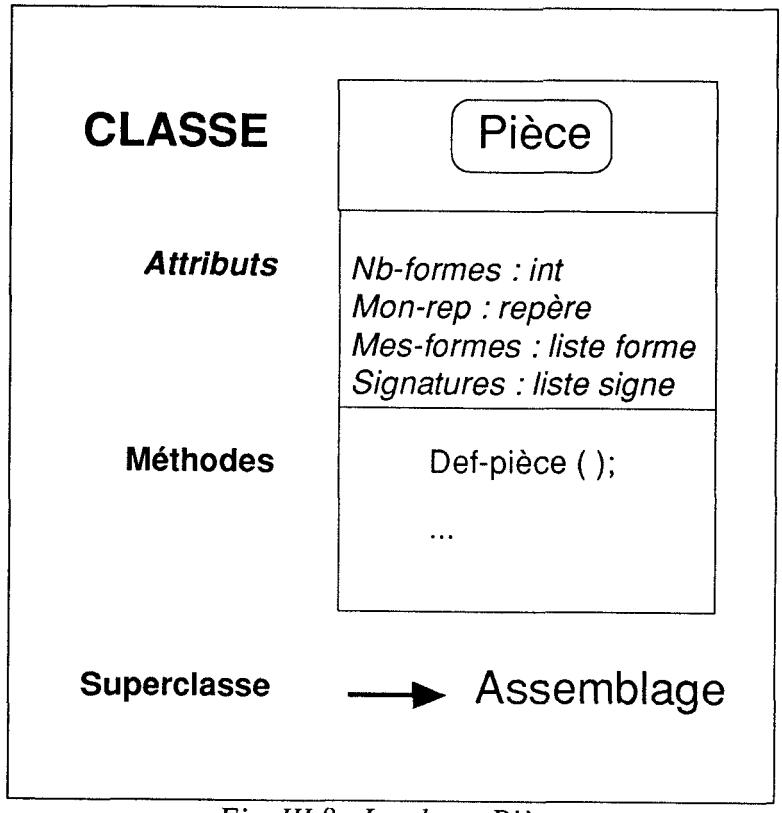


Fig. III.8 - La classe Pièce

Les attributs de cette classe sont:

- Nb-formes = g_i, nombre de formes existant dans la pièce
- Mon-rep: le repère associé à la pièce
- Mes-formes: liste énumérant toutes les formes G_{ij} de la pièce. La classe "Forme" est définie ultérieurement.
- Signatures: liste énumérant toutes les signatures n_{ij} associées aux formes G_{ij}.

La méthode Def-pièce () de cette classe, permet de définir la pièce comme mentionné ci dessus, c'est à dire associer l'ensemble des couples (G_{ij}, n_{ij}) comme défini dans l'equation (2). La classe "Pièce" admet comme superclasse, la classe "Assemblage" définie précedemment.

III.5.4 - Les Formes

Dans le domaine industriel, les principales formes géométriques constituant la quasi totalité des pièces mécaniques sont: les polyèdres, les cylindres, les cônes, les sphères et les formes à sections cylindriques, coniques ou sphériques. Il est évident que ces formes peuvent être pleines ou creuses représentant ainsi des cavités.

Dans cette étude, on limite le choix sur ces formes, tout en envisageant une structure de données qui pourra s'étendre pour englober d'autres formes élémentaires possibles.

Ainsi par exemple, une vis, qu'on peut définir à partir des formes déjà mentionnées, pourra se présenter comme une nouvelle forme élémentaire.

Dans ces conditions, une forme est définie par un ensemble de paramètres géométriques muni de règles de génération:

$$G_{ij} = \{ \text{paramètres géométriques} \} \cup \{ \text{règles de génération} \}$$

Par exemple:

* Un cylindre régulier est défini par trois éléments (Fig. III.9):

- un point C, qui est le centre de sa surface de base
- un vecteur V, qui indique la direction de la génératrice, et la hauteur
- un rayon r, rayon de la base.

Cependant, la normalisation du travail nécessite l'adoption d'une structure unique de représentation. Le choix de cette structure doit assurer la compatibilité entre les différentes représentations des formes ainsi que la possibilité de représenter de nouvelles formes.

La structure proposée consiste à adopter un seul type d'éléments pour la représentation des formes: c'est l'élément point, qu'on appelle par défaut "sommet". Ainsi, à partir d'un certain

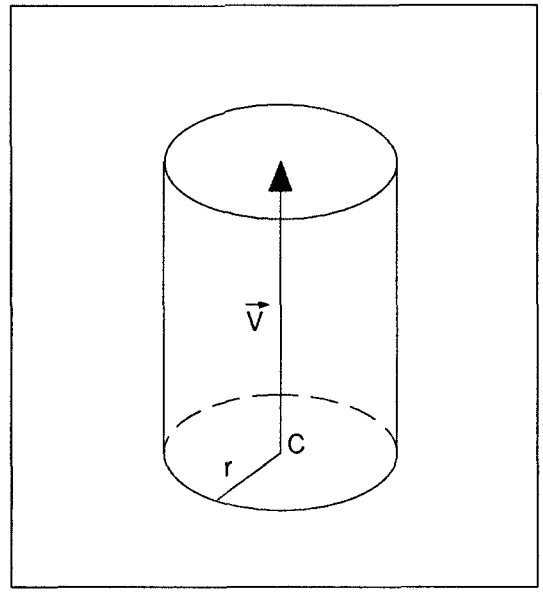


Fig. III.9 - Les paramètres géométriques du cylindre

nombre de sommets, on pourra trouver les paramètres géométriques nécessaires à l'identification d'une forme. Si on reprend l'exemple du cylindre, sa définition normalisée revient à l'énumération des trois sommets A, B et C de la figure III.10, à partir desquels on peut reconstituer les éléments r, C et V de la figure III.9.

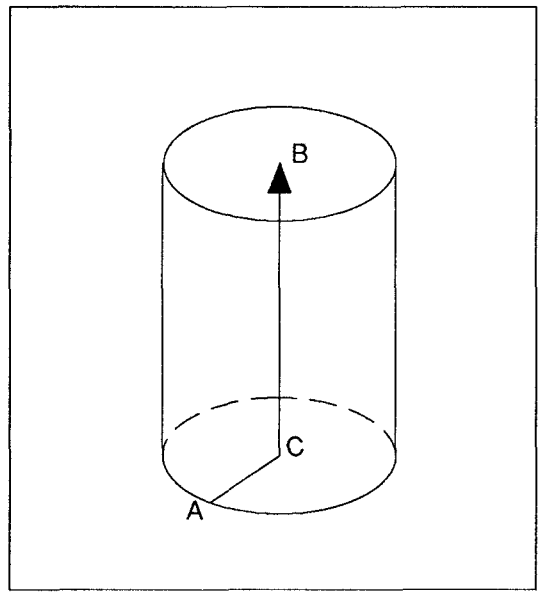


Fig. III.10 - Les paramètres normalisés du cylindre

Le passage d'une représentation solide à une autre, est assuré par de simples opérations mathématiques de conversion. Cette structure permet d'étendre le choix jusqu'à des formes

quelconques (approximées par les courbes de Béziers ou des B-splines).

Une forme est alors définie par un ensemble ordonné de sommets, et de règles de génération qui s'appliquent à ces sommets:

$$G_{ij} = \{ S_{ijk} , k = 1,.., s_{ij} \} \cup \{ \text{règles de génération} \} \quad (3)$$

- s_{ij}: nombre total de paramètres (sommets), nécessaire pour modéliser la forme G_{ij}.
- k: indice des sommets. Son domaine est l'ensemble des entiers naturels inférieurs ou égaux à s_{ij}.
- S_{ijk}: kième sommet de la forme G_{ij}. La classe "Sommet" est définie ultérieurement.

Les règles de génération de formes géométriques sont des algorithmes (méthodes de traitement) qui sont associées à des ensembles ordonnés de sommets (arguments), et qui ont pour but, de générer à partir de ces sommets, des formes géométriques élémentaires.

A chaque type de formes géométriques élémentaires (cube, cylindre, sphère, ...), correspond une règle de génération spéciale. Ainsi, une forme géométrique est définie par l'association d'un ensemble déterminé de sommets ordonnés et d'une règle spécifique de génération.

En terminologie "Objets" , on définit une forme par la classe "Forme" (fig. III.11). Cette classe admet pour attributs, les sommets S_{ijk} et pour méthodes, les règles de génération définies ci dessus. Cependant, puisque chaque type élémentaire est défini par une règle de génération spéciale, il parait utile de classer hiérarchiquement les formes élémentaires. Chaque forme élémentaire est donc modélisée par une sous classe propre. Ces sous classes, correspondant aux différentes formes élémentaires, admettent comme superclasse, la classe "Forme".

Dans ces conditions, la classe mère "Forme" devient donc une classe abstraite qui regroupe les différentes classes associées aux différentes formes élémentaires. A titre d'exemples, on présente ci dessous, les classes associées aux différentes formes élémentaires les plus courantes, ainsi que les règles de génération correspondantes:

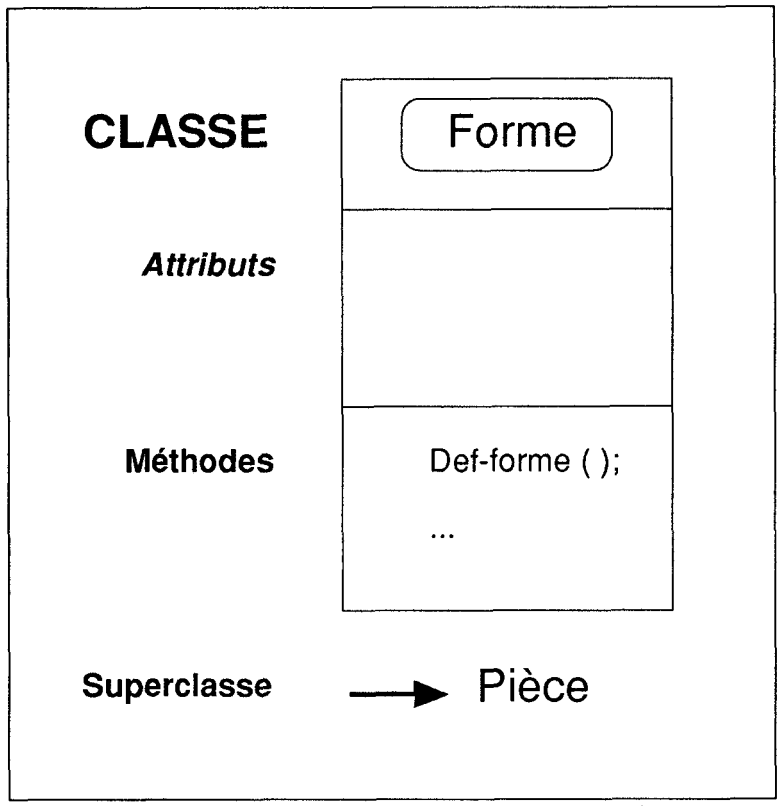


Fig. III.11 - La classe Forme est une classe abstraite

- les cubes
- les cylindres
- les sections cylindriques
- les cônes
- les sections coniques (tronc coniques)
- les sphères

III.5.4.1 - Les Cubes

Les cubes sont modélisés par la classe "Cube" (fig. III.12). Les attributs de cette classe sont:

- S1: sommet
- S2: sommet
- S3: sommet

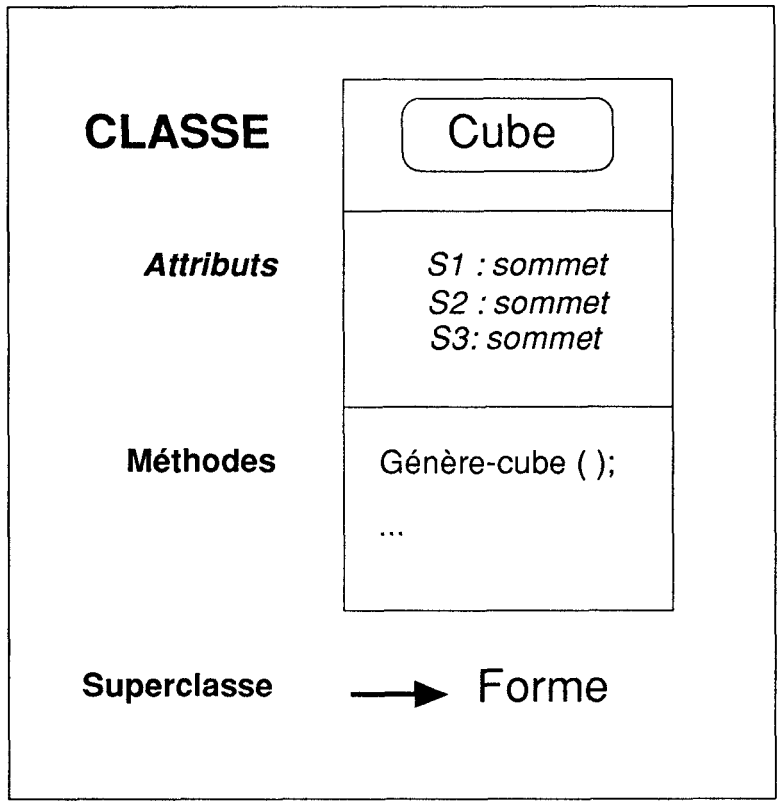


Fig. III.12 - La classe Cube

La méthode Génère-cube (), associée à cette classe, implémente la règle de génération présentée ci dessous:

* arguments:

- { S1, S2, S3} ensemble ordonné formé par trois sommets S1, S2 et S3

* tâche:

générer le cube qui admet (fig. III.13):

- S1 comme centre d'une de ses faces,
- S2 comme centre de la face parallèlement opposée,
- S3 comme un sommet de la face définie par S2.

On démontre géométriquement qu'on peut générer un cube qui satisfait à de telles règles et

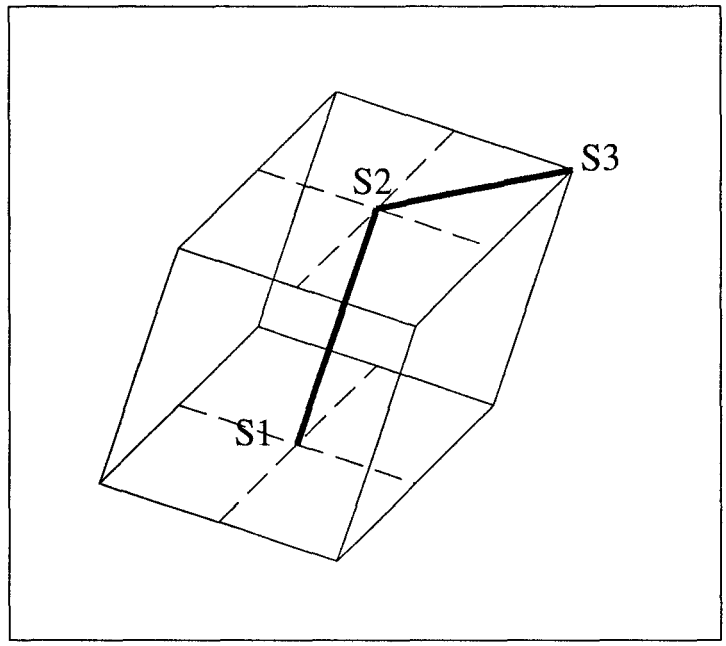


Fig. III.13 - La génération du cube

que ce cube est unique. L'algorithme de génération de ce cube est le suivant:

- calcul du vecteur (S1 S2)
- détermination du plan perpendiculaire à ce vecteur et qui passe par S2
- tracé dans ce plan d'une surface carrée de centre S2 et qui admet S3 comme sommet
- faire glisser cette surface, le long du segment (S1 S2)

III.5.4.2 - Les Cylindres

Les cylindres sont modélisés par la classe "Cylindre" (fig. III.14). Les attributs de cette classe sont:

- S1: sommet
- S2: sommet
- S3: sommet

La méthode Génère-cyl (), associée à cette classe, implémente la règle de génération suivante:

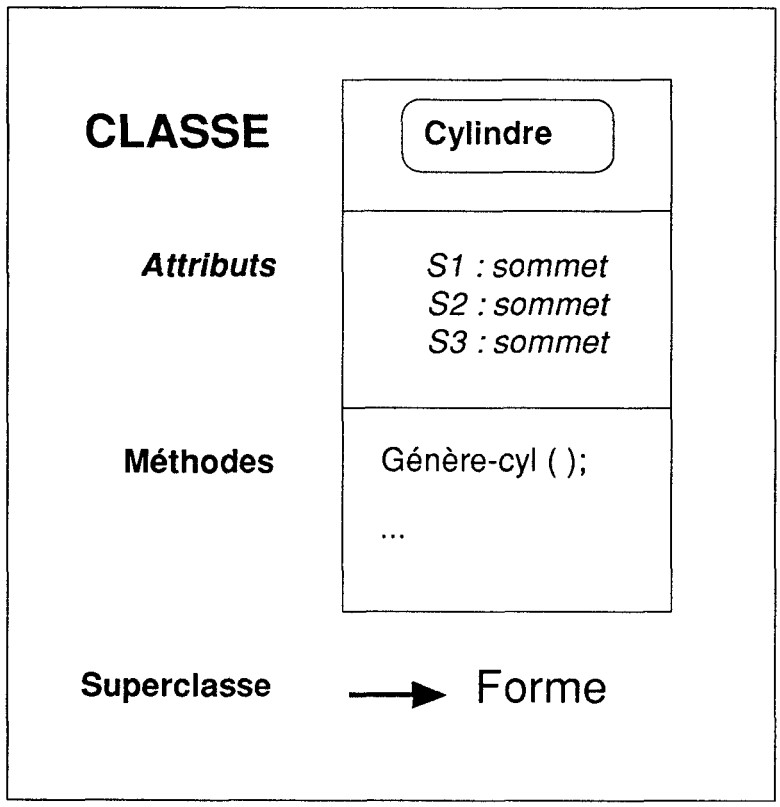


Fig. III.14 - La classe Cylindre

* arguments:

- { S1, S2, S3} ensemble ordonné formé par trois sommets S1, S2 et S3

* tâche:

générer le cylindre régulier qui admet (fig. III.15):

- S1 comme centre d'une de ses bases,
- S2 comme centre de sa deuxième base,
- S3 comme un point de bord de la deuxième base.

On démontre géométriquement qu'on peut générer un cylindre unique qui satisfait à de telles règles. L'algorithme de génération de ce cylindre est le suivant:

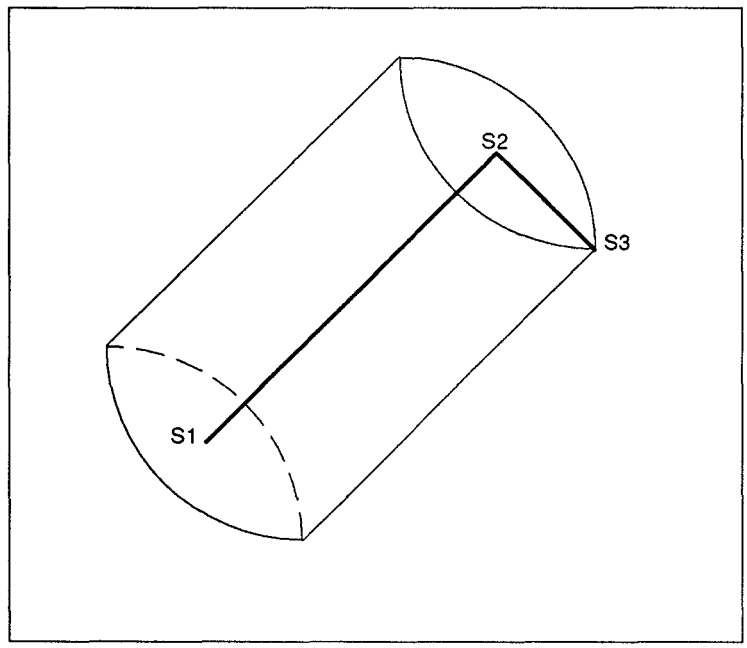


Fig. III.15 - La génération du cylindre

- calcul du vecteur (S1 S2)
- détermination du plan perpendiculaire à ce vecteur et qui passe par S2
- tracé dans ce plan d'une surface circulaire complète, de centre S2 et qui passe par S3
- faire glisser cette surface, le long du segment (S1 S2)

III.5.4.3 - Les Sections Cylindriques

Les sections cylindriques sont modélisées par la classe "Sec-cyl" (fig. III.16). Cette classe admet la classe "Cylindre" comme superclasse. Ses attributs sont:

- S1: sommet
- S2: sommet
- S3: sommet
- S4: sommet

La méthode Génère-cyl (), associée à cette classe, implémente la règle de génération d'une section cylindrique. On présente ci dessous cette règle:

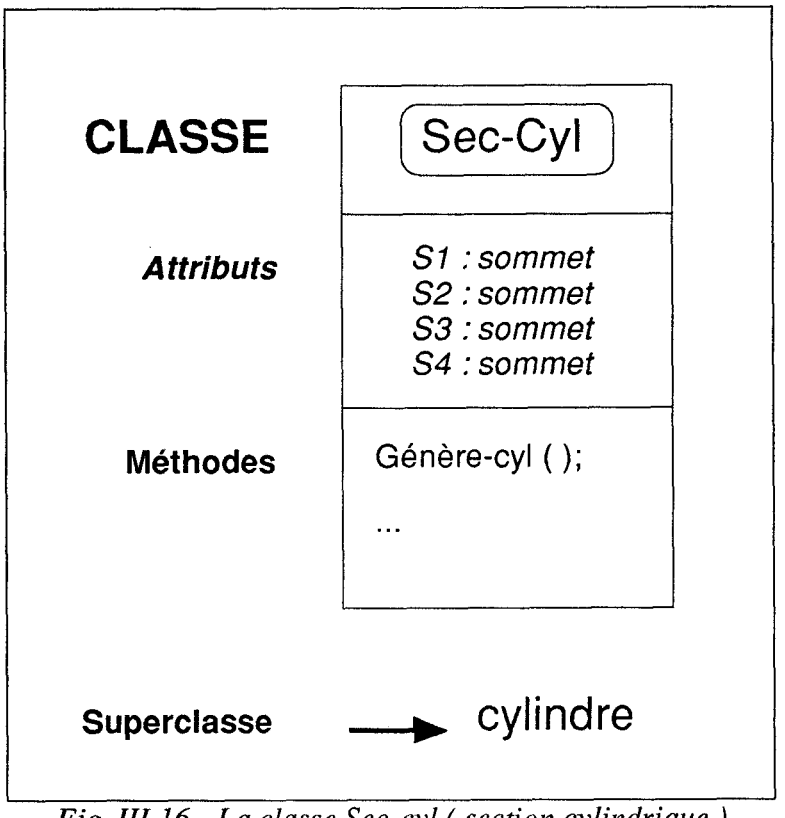


Fig. III.16 - La classe Sec-cyl (section cylindrique)

* arguments:

- { S1, S2, S3, S4 } ensemble ordonné formé par quatre sommets S1, S2, S3 et S4

* tâche:

générer la section cylindrique suivante (fig. III.17):

- S1: centre d'une de ses surfaces de bases,
- S2: centre de sa deuxième surface de base,
- S3: premier point de bord de la deuxième surface de base,
- S4: second point de bord de la deuxième surface de base. L'ordre des points S3 et S4 doit être toujours donné en respectant un parcours anti-horaire de la surface de base.

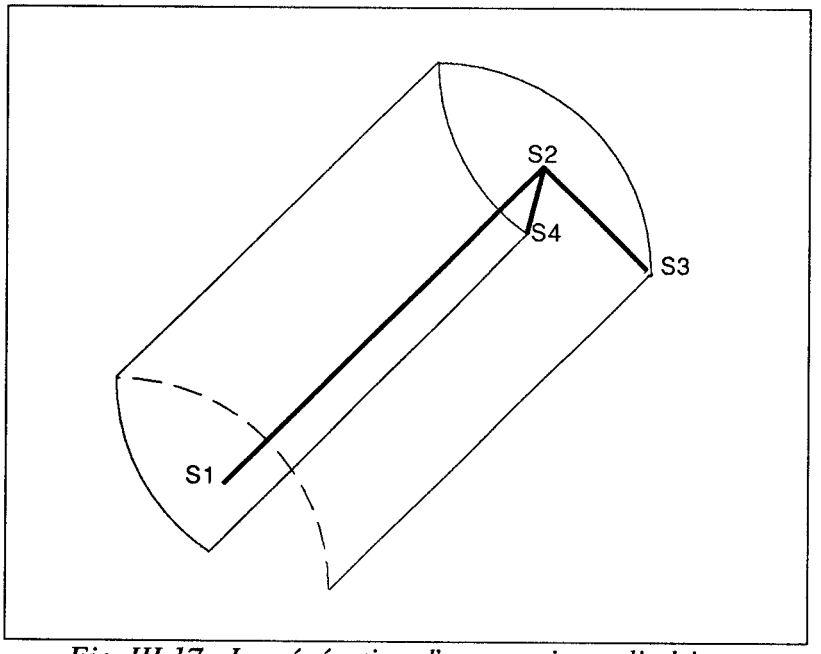


Fig. III.17 - La génération d'une section cylindrique

On démontre géométriquement qu'on peut générer une telle section cylindrique et qu'elle est unique. L'algorithme de génération est le suivant:

- calcul du vecteur (S1 S2)
- détermination du plan perpendiculaire à ce vecteur et qui passe par S2
- tracé dans ce plan d'une section de surface circulaire, de centre S2 et qui soit limitée par S3 et S4 dans un parcours anti-horaire
- faire glisser cette surface, le long du segment (S1 S2)

III.5.4.4 - Les Cônes

Les cônes sont modélisés par la classe "Cône" (fig. III.18). Les attributs de cette classe sont:

- S1: sommet,
- S2: sommet,
- S3: sommet,

La méthode Génère-cône (), associée à cette classe, implémente la règle de génération d'un

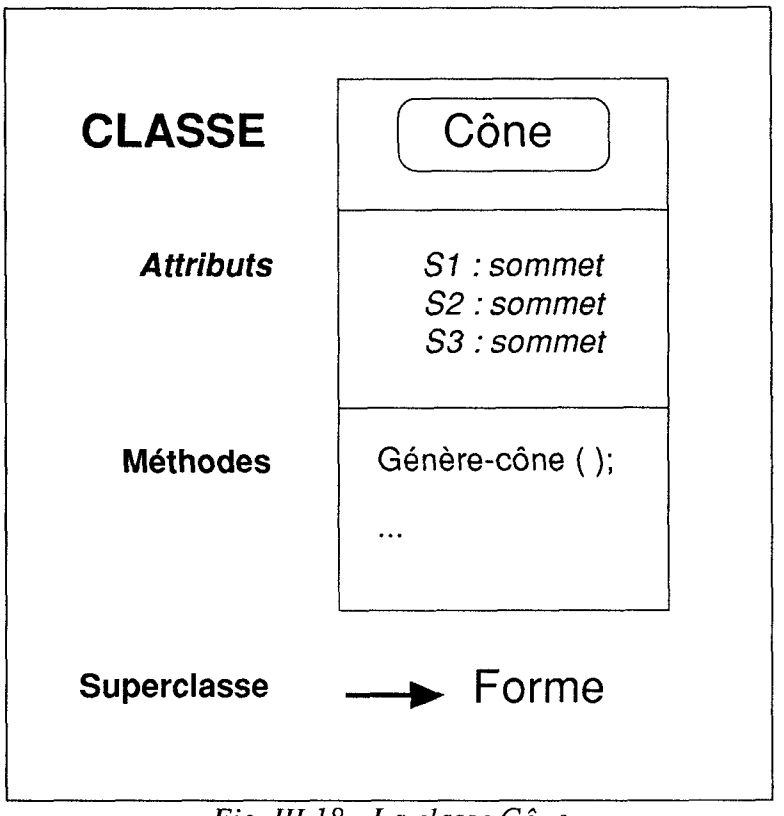


Fig. III.18 - La classe Cône

cône. On présente ci dessous cette règle:

* arguments:

- { S1, S2, S3} ensemble ordonné formé par trois sommets S1, S2 et S3

* tâche:

générer le cône régulier qui admet (fig. III.19):

- S1: sommet du cône,
- S2: centre de la base,
- S3: point de bord de la base.

On démontre géométriquement qu'on peut générer un cône unique qui satisfait à de telles règles. L'algorithme de génération est le suivant:

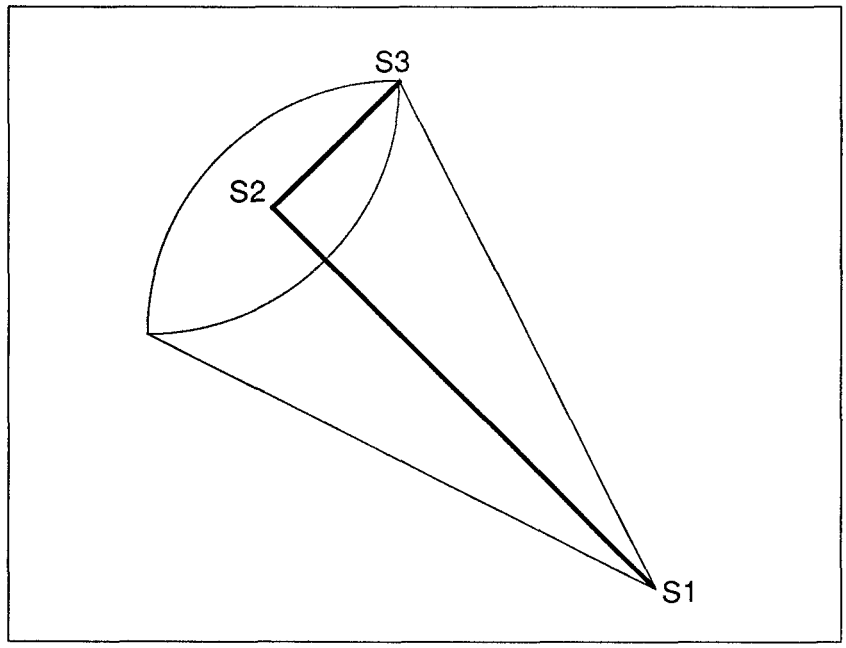


Fig. III.19 - La génération d'un cône

- calcul du vecteur (S1 S2)
- détermination de la surface triangulaire formée par (S1, S2, S3)
- faire pivoter cette surface, autour du vecteur (S1 S2) en un tour complet

III.5.4.5 - Les Sections Coniques

Les sections coniques sont modélisées par la classe "Sec-con" (fig. III.20), sous classe de la classe "Cône". Les attributs de cette classe sont:

- S1: sommet
- S2: sommet
- S3: sommet
- S4: sommet

La méthode Génère-cône (), associée à cette classe, implémente la règle de génération d'une section conique. On présente ci dessous cette règle:

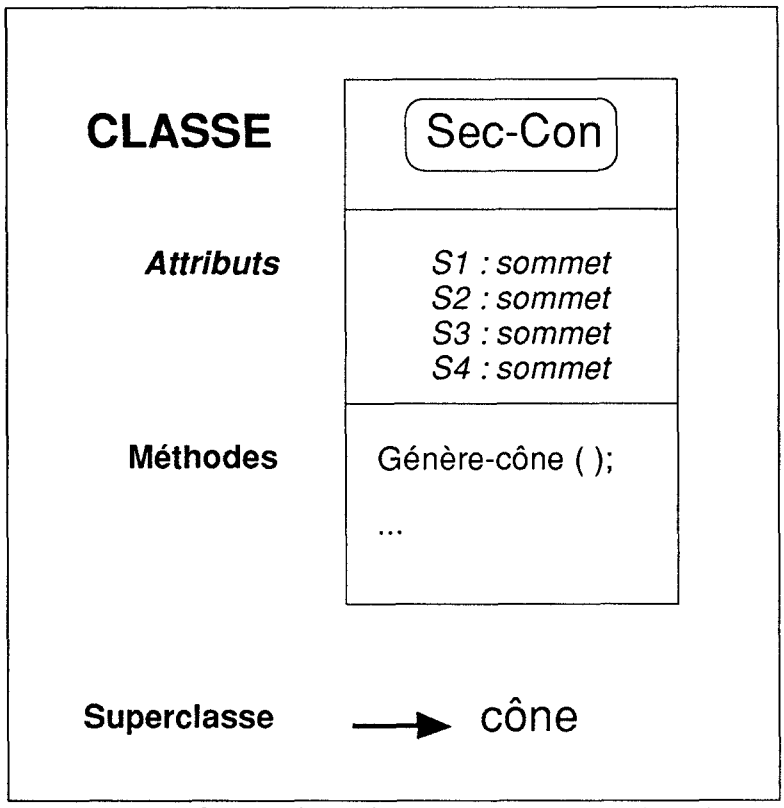


Fig. III.20 - La classe Sec-con (section conique)

* arguments:

- { S1, S2, S3, S4} ensemble ordonné formé par quatre sommets S1, S2, S3 et S4

* tâche:

générer la section conique suivante (fig. III.21):

- S1: centre de la petite surface de base,
- S2: centre de la grande surface de base,
- S3: un point de bord de la grande surface de base,
- S4: un point de bord de la petite surface de base, choisi de sorte que les quatre points S1, S2, S3 et S4 soient coplanaires.

Une telle section conique est géométriquement réalisable. L'algorithme de génération est le suivant:

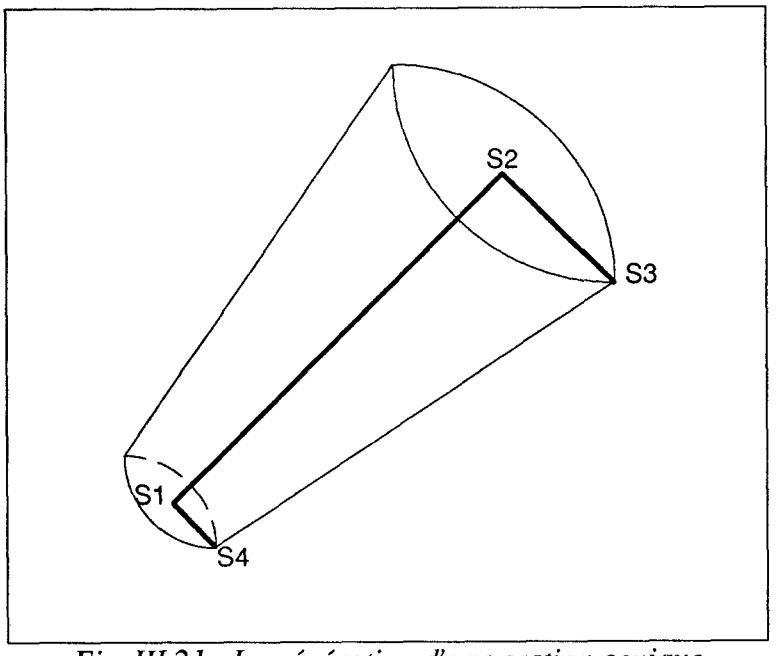


Fig. III.21 - La génération d'une section conique

- calcul du vecteur (S1 S2)
- détermination du quadrilataire formée par (S1, S2, S3, S4)
- faire pivoter cette surface, autour du vecteur (S1 S2) en un tour complet

III.5.4.6 - Les Sphères

Les sphères sont modélisées par la classe "Sphère" (fig. III.22). Les attributs de cette classe sont:

- S1: sommet
- S2: sommet

La méthode Génère-sph (), associée à cette classe, implémente la règle de génération d'une sphère. On présente ci dessous cette règle:

* arguments:

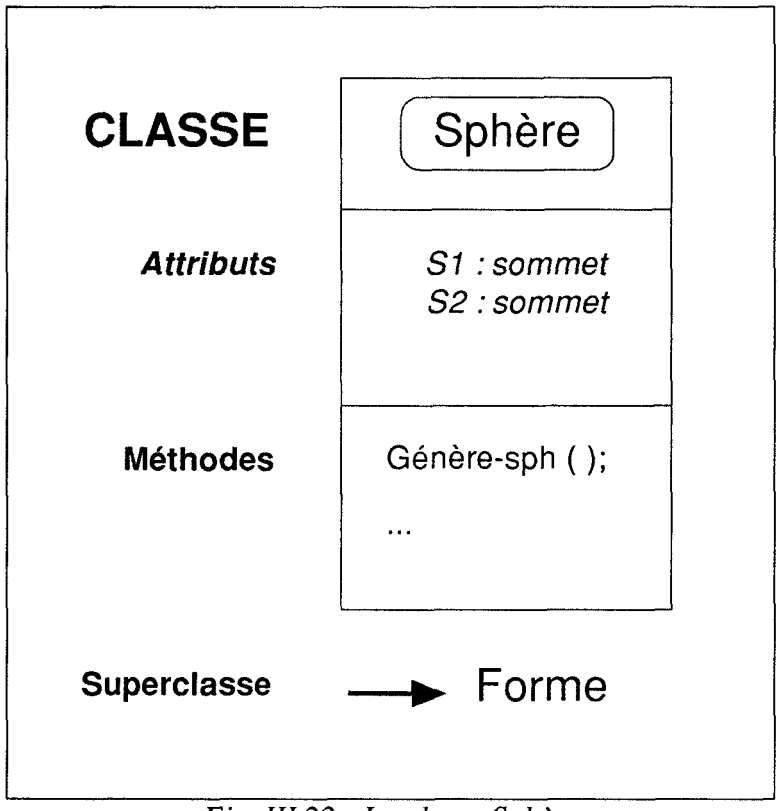


Fig. III.22 - La classe Sphère

- { S1, S2} ensemble ordonné formé par deux sommets S1 et S2

* tâche:

générer la sphère qui admet (fig. III.23):

- S1: centre de la sphère,
- S2: un point de la surface de bord.

L'algorithme de génération est le suivant:

- calcul du vecteur (S1 S2)
- détermination d'une surface circulaire de centre S1, et qui passe par S2
- faire pivoter cette surface, autour du vecteur (S1 S2) en un tour complet

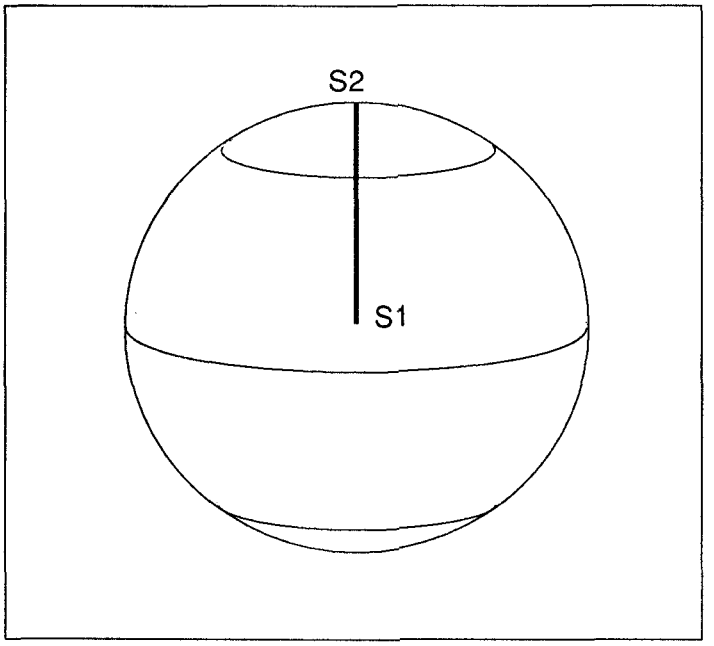


Fig. III.23 - La génération de la sphère

III.5.5 - Les Sommets

Un sommet est un point de l'espace E, décrit dans un repère T. Il est défini par ses coordonnées dans le repère correspondant:

$$S_{ijk} = C_{ijk} \qquad (5)$$

- C_{ijk}: coordonnées de S_{ijk}.

Cependant, il est possible de représenter en coordonnées cartésiennes, cylindriques ou sphériques, suivant la forme envisagée. On peut aussi représenter dans un système de coordonnées référentiel qu'on doit définir préalablement. Dans un système cartésien, un sommet est défini par une matrice unitaire qui représente ses coordonnées homogènes:

$$S = [\, x\ y\ z\ 1\,]^T \qquad ([\, x\ y\ z\ 1\,]^T = \text{matrice transpose de } [\, x\ y\ z\ 1\,])$$

x,y,z: coordonnées cartésiennes de S.

Les sommets sont générés à partir de la classe "Sommet" (fig. III.24). L'attribut de cette classe est:

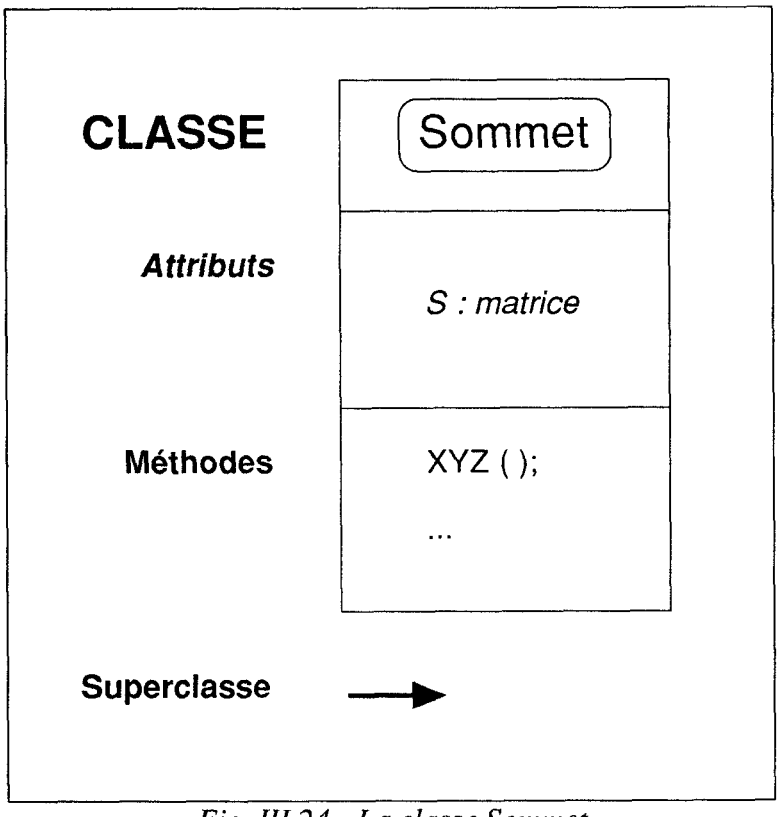


Fig. III.24 - La classe Sommet

- S: matrice.

On peut associer à cette classe des méthodes classiques de calcul des coordonnées: XYZ (). Ces méthodes, triviales, ne sont pas présentées.

III.5.6 - Les Repères

Un repère Γ est défini dans E par sa matrice de passage par rapport au repère absolu T_0. Pour représenter l'effet de translation et de rotation en une même matrice de passage, on définit le repère par une matrice homogène (4*4). Cette matrice est composée de la façon suivante (Fig. III.25):

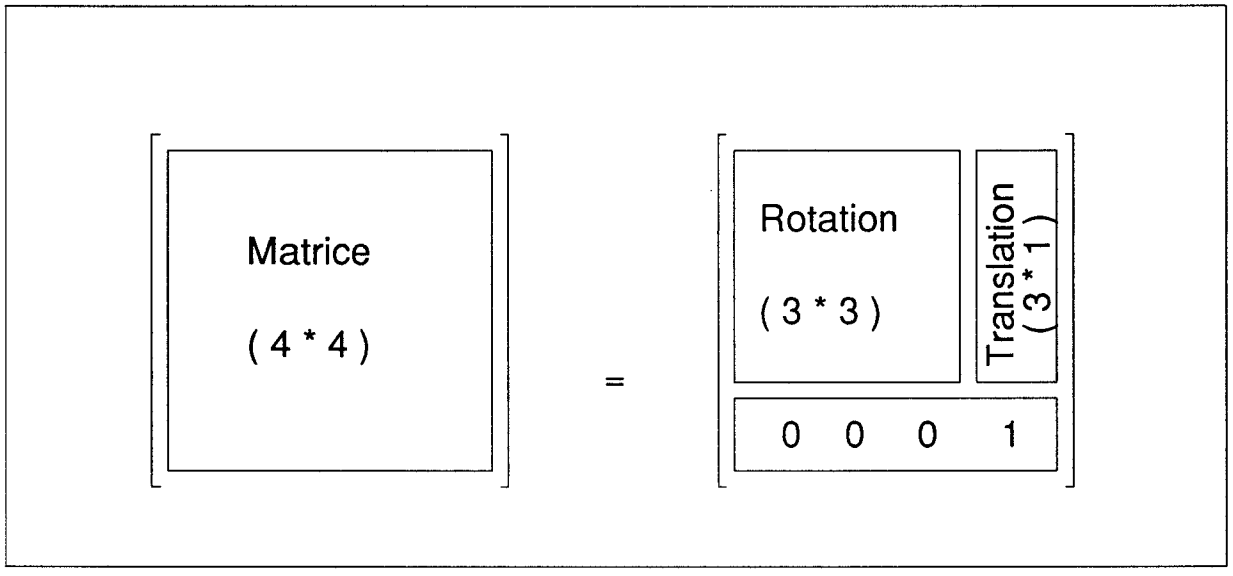


Fig. III.25 - La composition de la matrice homogène

- les trois premières lignes et les trois premières colonnes de la matrice définissent la rotation par rapports aux trois axes, par une sous matrice (3*3)

- dans la quatrième colonne, on représente la translation par rapport aux trois axes, par une matrice colonne (3*1)

- sur la quatrième ligne de la matrice, on insère la matrice (1*4): [0 0 0 1]

Pour modéliser les repères, on crée la classe "Repère" (fig. III.26). L'attribut de cette classe est:

- M: matrice.

Les méthodes Cal-coord () associées à cette classe, sont des méthodes classiques de calcul de changement de repère, de translation, de rotation,...

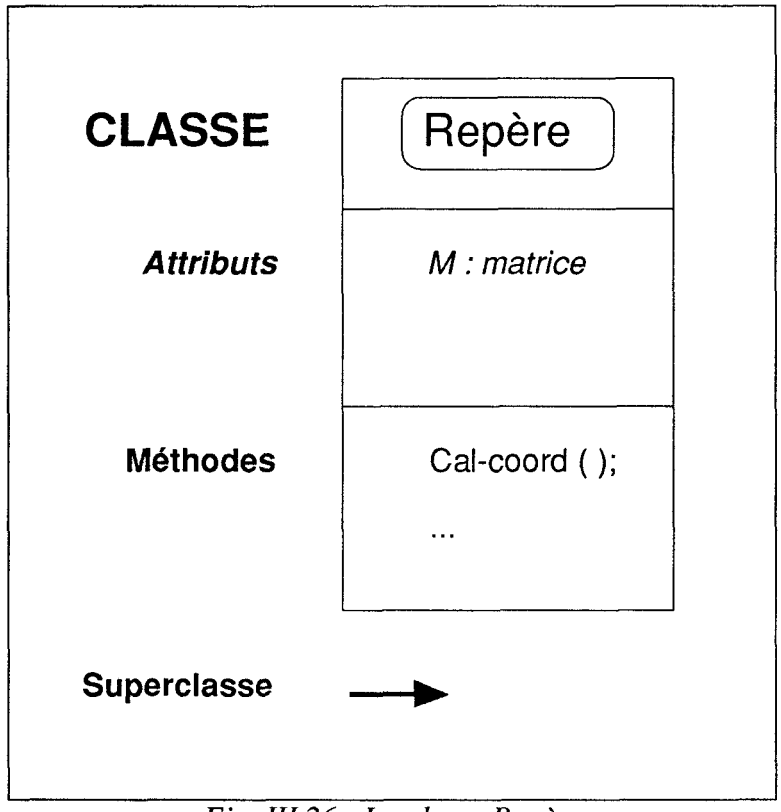


Fig. III.26 - La classe Repère

III.5.7 - Le Graphe d'Héritage

La figure III.27 montre le graphe d'héritage des classes de l'univers définies ci dessus.

III.6 - La Structuration des Données

En tenant compte de ce qui précède on peut construire une structure arborescente pour mémoriser et manipuler le processus d'assemblage.

III.6.1 - Structure des Formes

La structure des formes géométriques G_{ij} est présentée par la figure III.28. Elle comporte les

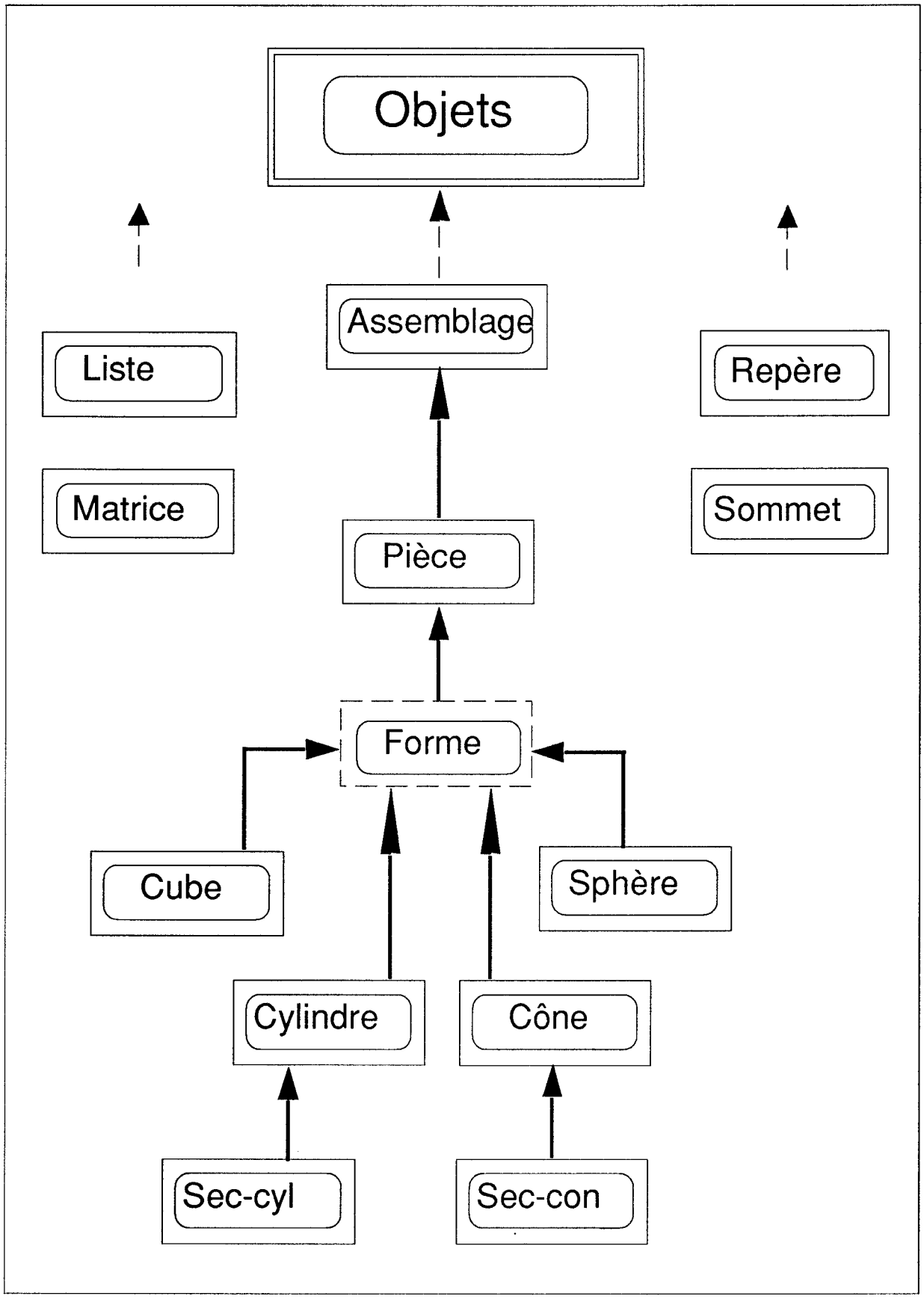


Fig. III.27 - Le graphe d'héritage des classes

entitées suivantes:

- s_{ij}: le nombre de sommets associés à la forme.
- $C_{ij}(k)$, $k = 1, .., s_{ij}$: les coordonnées des sommets de la forme.

en notation indicielle,

$$G_{ij} = \{ s_{ij}, C_{ijk} \} \qquad (7)$$

III.6.2 - Structure des Pièces

D'après ce qui précède, on peut présenter la structure de la pièce P_i par le graphe de la figure III.28. Cette structure comporte les entitées suivantes:

- Γi : repère associé à la pièce Pi
- g_i : nombre de formes élémentaires dans la pièce P_i
- n_{ij} : signature de la jème forme de P_i. $n_{ij} = n_i(j)$, $j = 1,..,g_i$
- F_{ij}: classe d'appartenance de la jème forme G_{ij}
- s_{ij} : nombre de sommets dans la jème forme de P_i
- C_{ijk} : kième sommet de la jème forme de P_i.

en notation indicielle,

$$P_i = \{ \Gamma i, gi, nij, Fij, sij, Cijk \} \qquad (8)$$

III.6.3 - Structure de L'Assemblage

La structure de l'assemblage A_M , montrée en figure III.29 devient alors (en notation

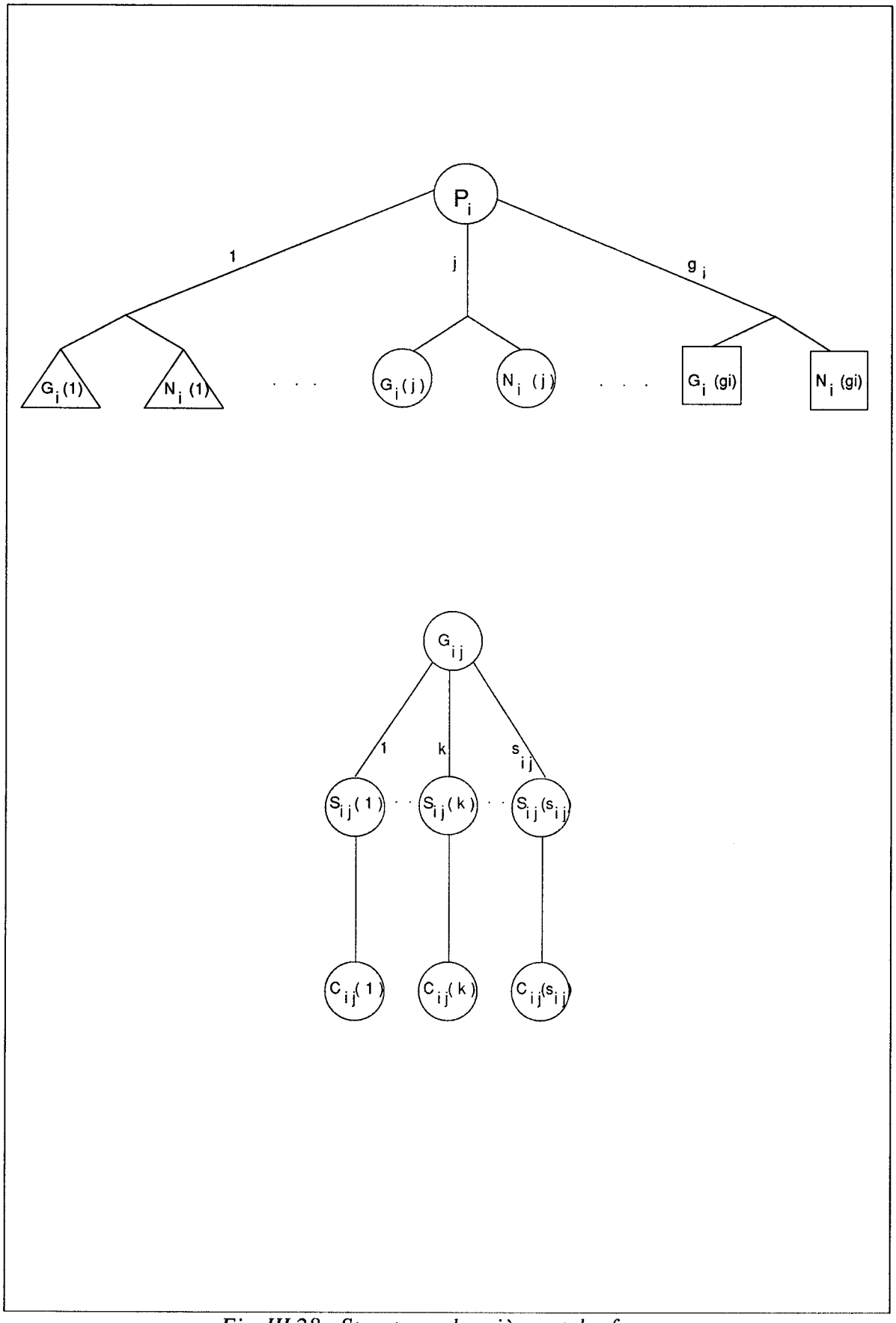


Fig. III.28 - Structures des pièces et des formes

indicielle):

$$A_M = \{M, \Gamma, g_i, \Gamma i, Hi, nij, Fij, sij, Cijk, \} \quad (9)$$

M: nombre de pièces dans l'assemblage A_M

Γ: repère de A_M

H_i: matrice de positionnement de la pièce P_i par rapport à A_M

$i = 1,..,M$

$j = 1,..,g_i$

$k = 1,..,s_{ij}$

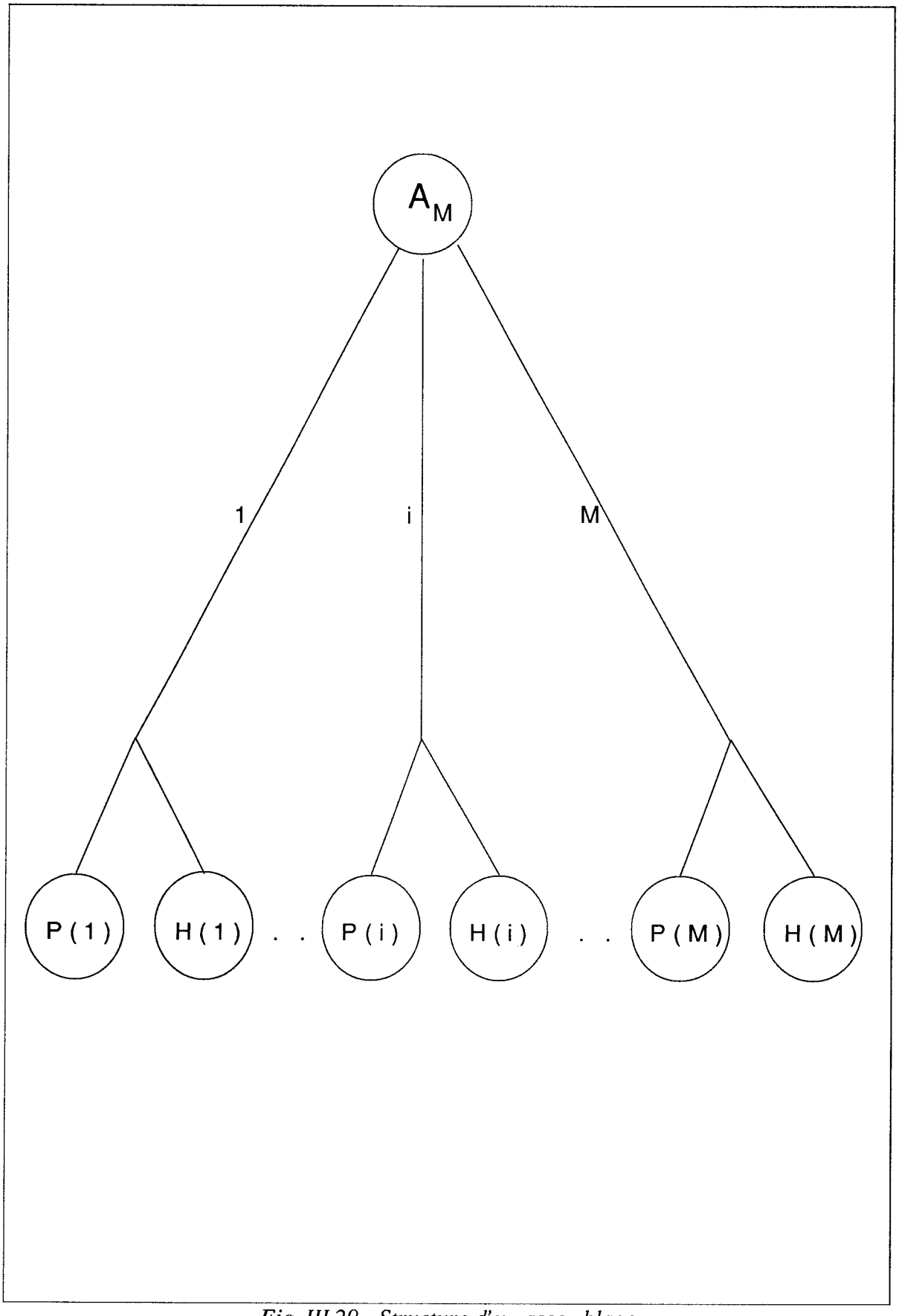


Fig. III.29 - Structure d'un assemblage

III.7 - Exemple

Soit à modéliser la pièce suivante (fig. III.30):

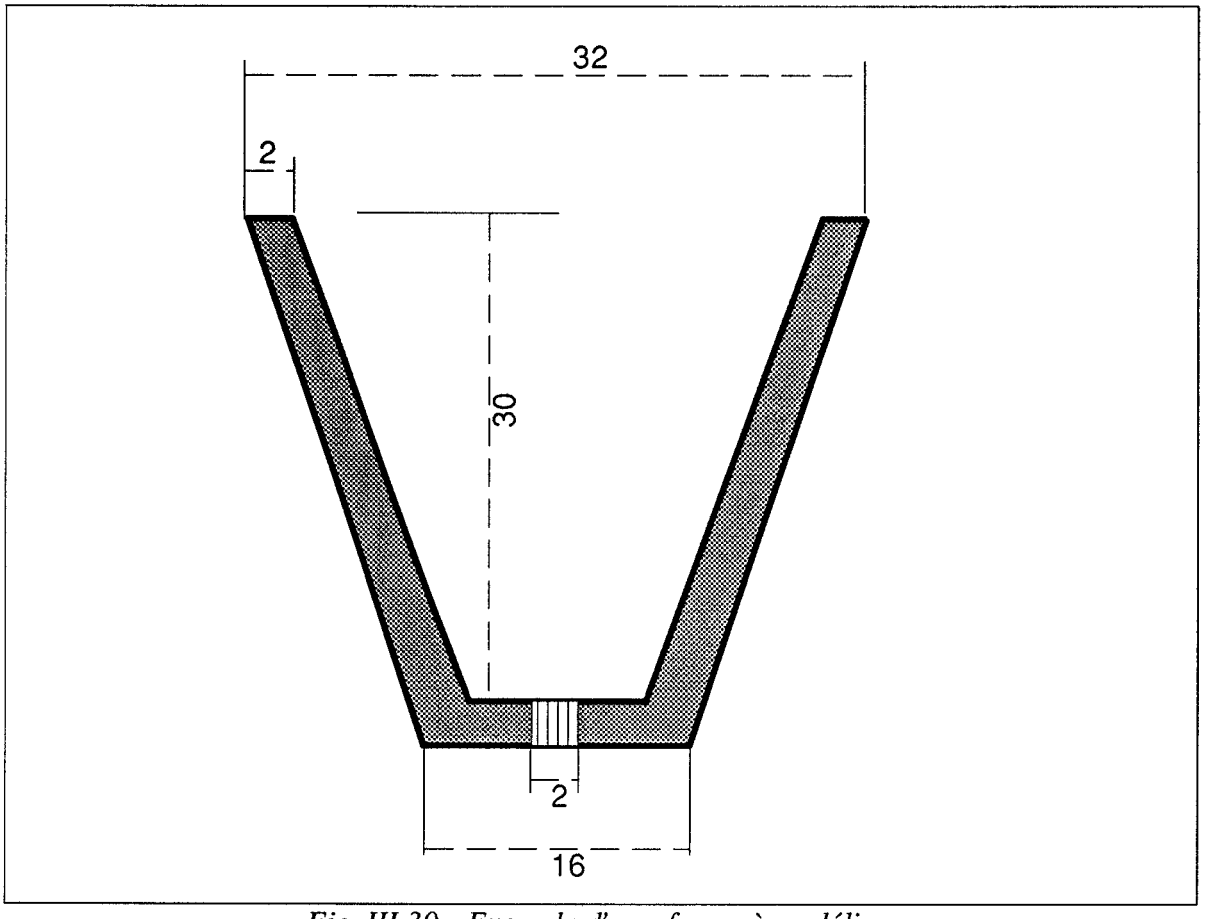


Fig. III.30 - Exemple d'une forme à modéliser

III.7.1 Description Physique de la Pièce

La pièce à modéliser est de forme conique. Elle est notée P_1. Elle a une profondeur de 30 cm et une épaisseur de bord uniforme de 2 cm. Elle possède au milieu de son fond, un trou de 2 cm de diamètre. Le diamètre supérieur externe est de 32 cm et le diamètre inférieur externe est de 16 cm.

III.7.2 - Modélisation Mathématique

On définit un repère de coordonnées mobile T1, cartésien et normé en cm. Ce repère est attaché à la pièce. Son centre se trouve au centre du fond de la pièce. Il est direct et son axe Z coïncide avec l'axe de la pièce (fig. III.31).

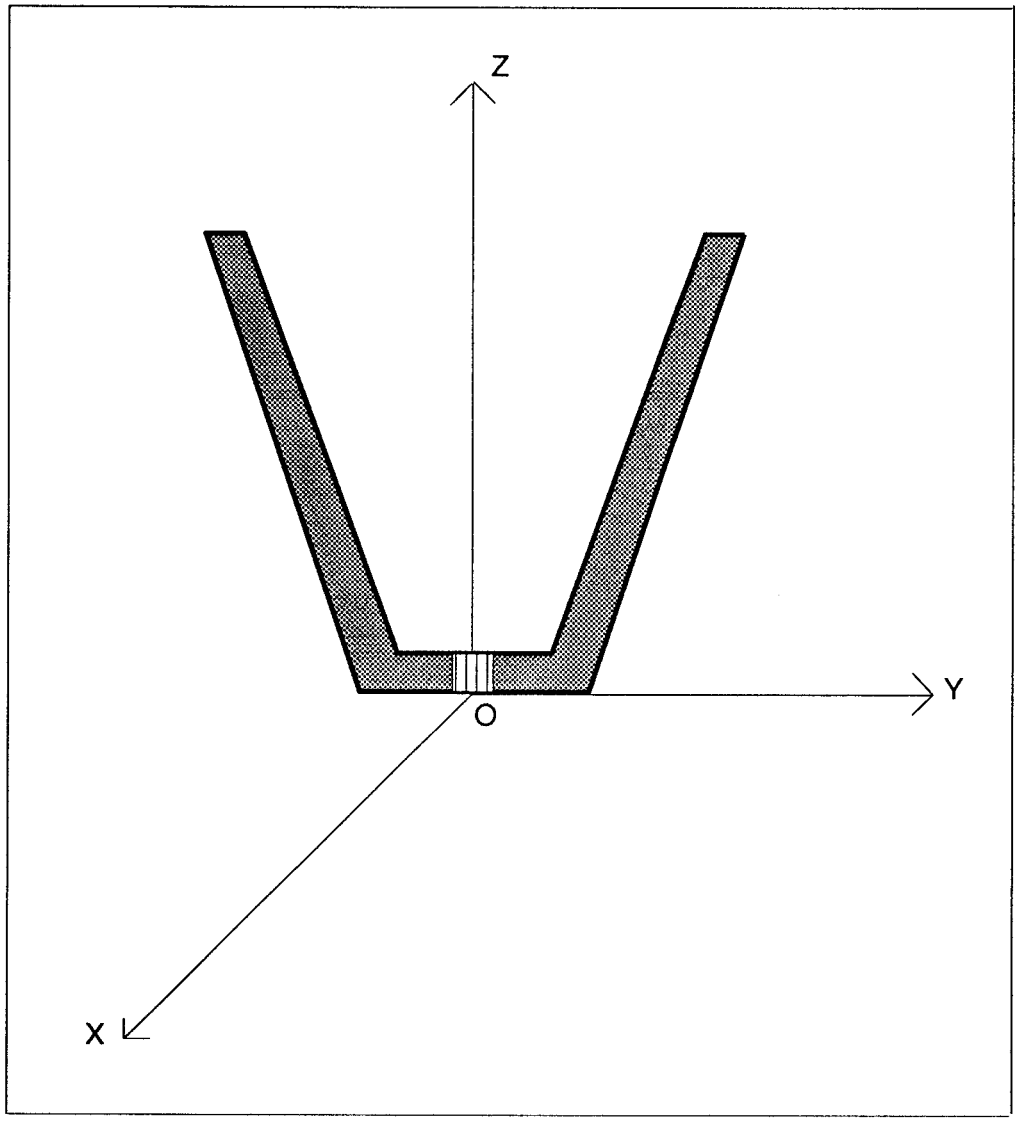


Fig. III.31 - La modélisation mathématique de la forme

La pièce P_1 est définie par trois formes élémentaires:

- une forme conique pleine, G_{11}, limitée par les surfaces externes du bord,
- une forme conique creuse, G_{12}, limitée par les surfaces internes du bord,
- une forme géométrique creuse, G_{13}, qui définit le trou.

D'après (2), on obtient:

$$P_1 = \{ (G_{11}, +) , (G_{12}, -) , (G_{13}, -) \}$$

P_1 est donc générée par instanciation de la classe "Pièce" (fig. III.32).

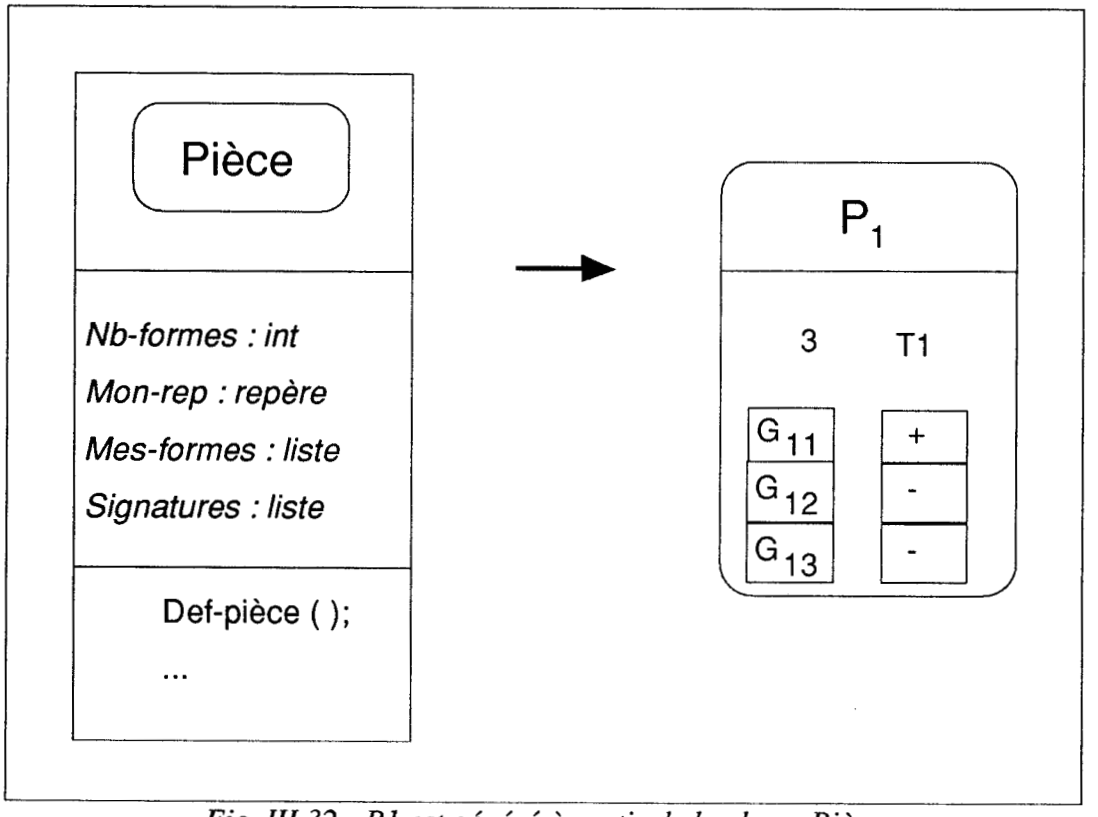


Fig. III.32 - P1 est généré à partir de la classe Pièce

G_{11} définit la première forme. C'est une forme conique non complète (section conique). On définit G_{11} à partir de la classe "Sec-con" (fig. III.34). Les paramètres de G_{11} sont:

$$G_{11} = \{ O, O2, B, A \}$$

Les sommet O, O2, B et A sont générés par instanciation de la classe "Sommet" (fig. III.35), compte tenu des cotations de la figure III.30.

$$O = [\,00\ \ 00\ \ 00\ \ 1\,]^T$$
$$O2 = [\,00\ \ 00\ \ 32\ \ 1\,]^T$$
$$B = [\,00\ \ 16\ \ 32\ \ 1\,]^T$$
$$A = [\,00\ \ 08\ \ 00\ \ 1\,]^T$$

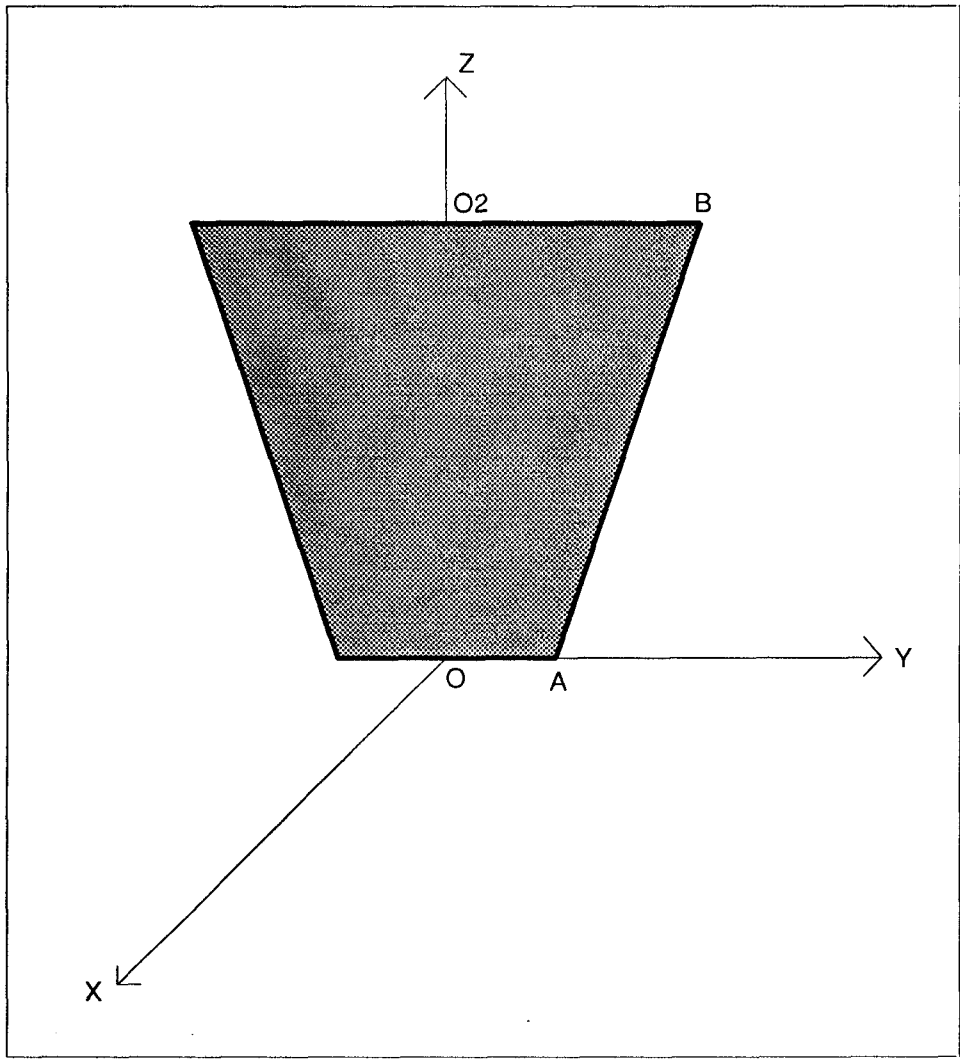


Fig. III.33 - La première forme de la pièce

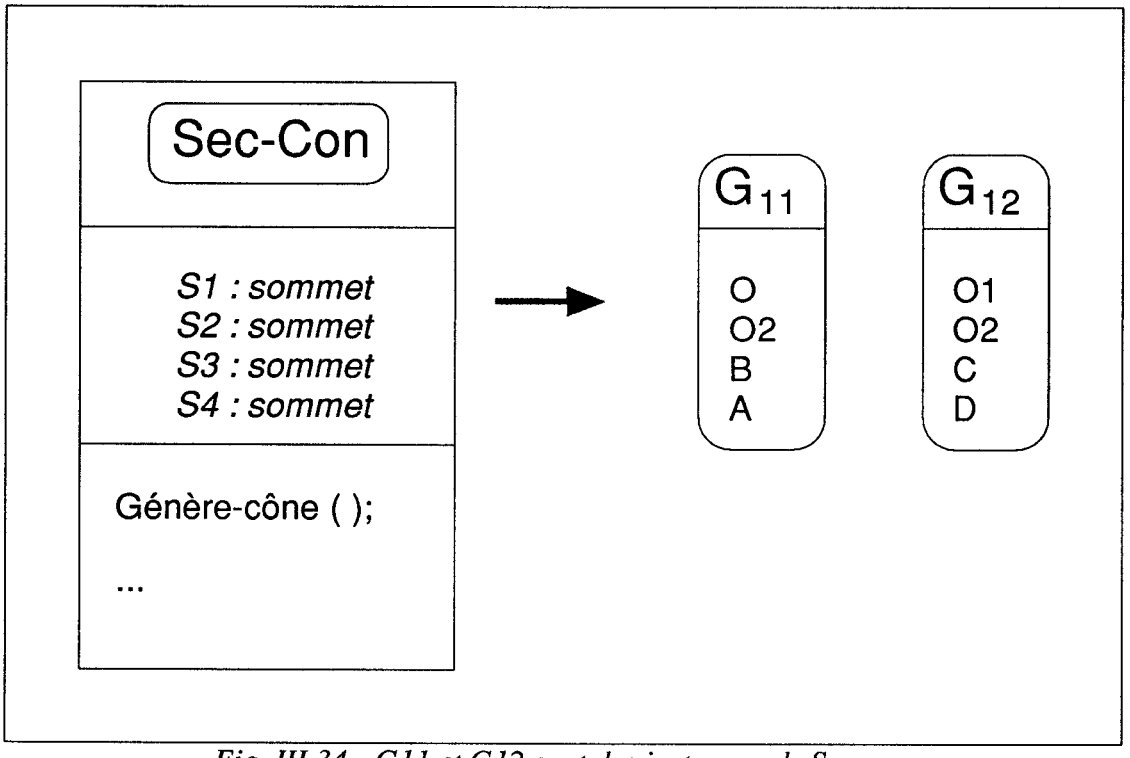


Fig. III.34 - G11 et G12 sont des instances de Sec-con

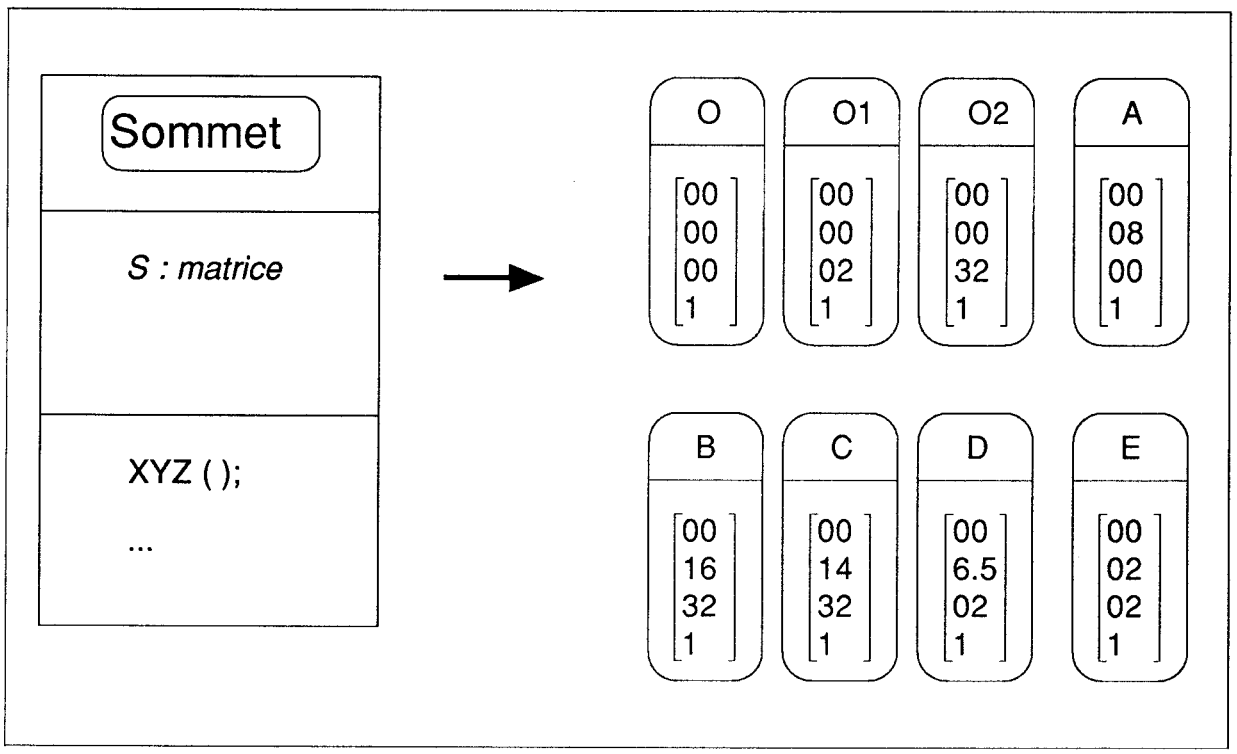


Fig. III.35 - La génération des sommets par instanciation

Similairement,

G_{12} = { O1, O2, C, D }

avec:

O1 = [00 00 02 1]^T
O2 = [00 00 32 1]^T
C = [00 14 32 1]^T
D = [00 6.5 02 1]^T

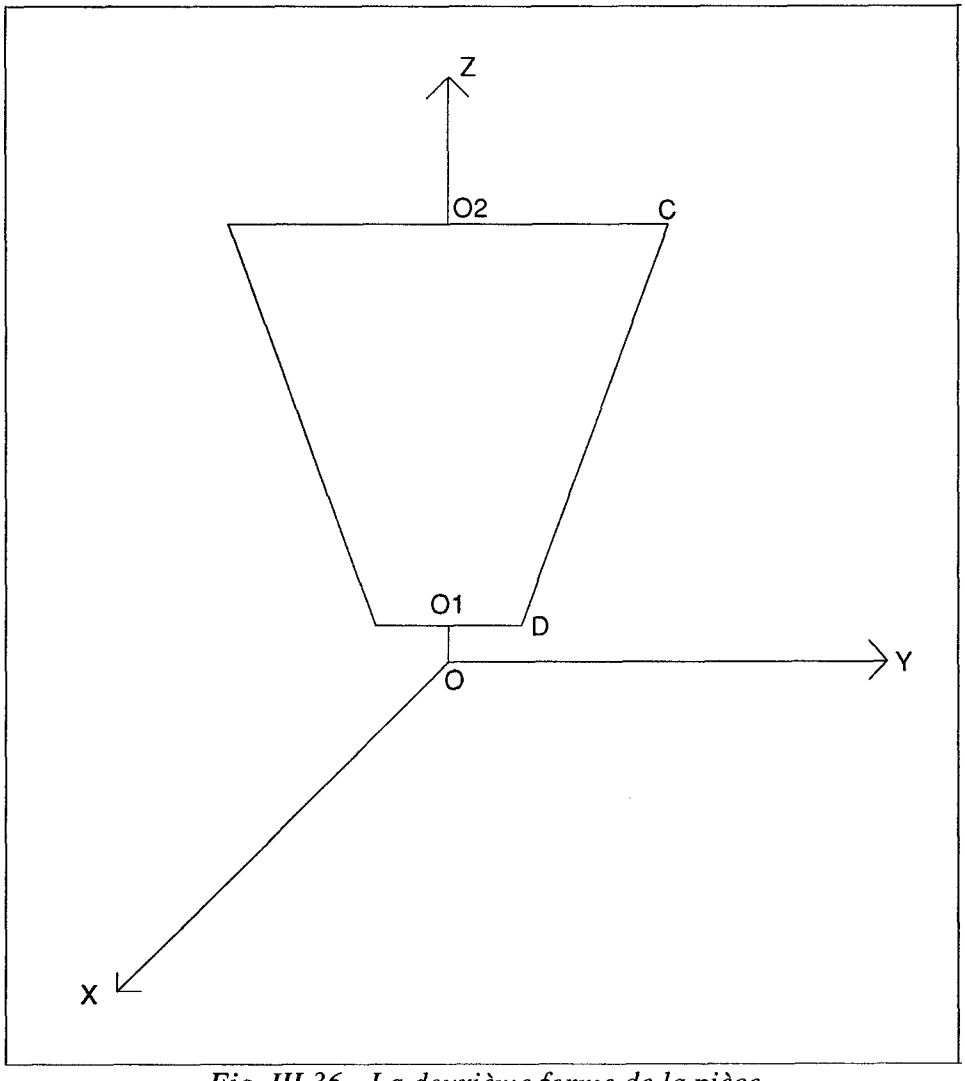


Fig. III.36 - La deuxième forme de la pièce

G_{13} est généré à partir de la classe "Cylindre" (fig. III.38):

$G_{13} = \{ O, O1, E \}$

avec:

$O = [\ 00\ \ 00\ \ 00\ \ 1\ ]^T$
$O1 = [\ 00\ \ 00\ \ 02\ \ 1\ ]^T$
$E = [\ 00\ \ 02\ \ 02\ \ 1\ ]^T$

Les points O, O1 et E sont générés par la classe Sommet (Fig. III.35).

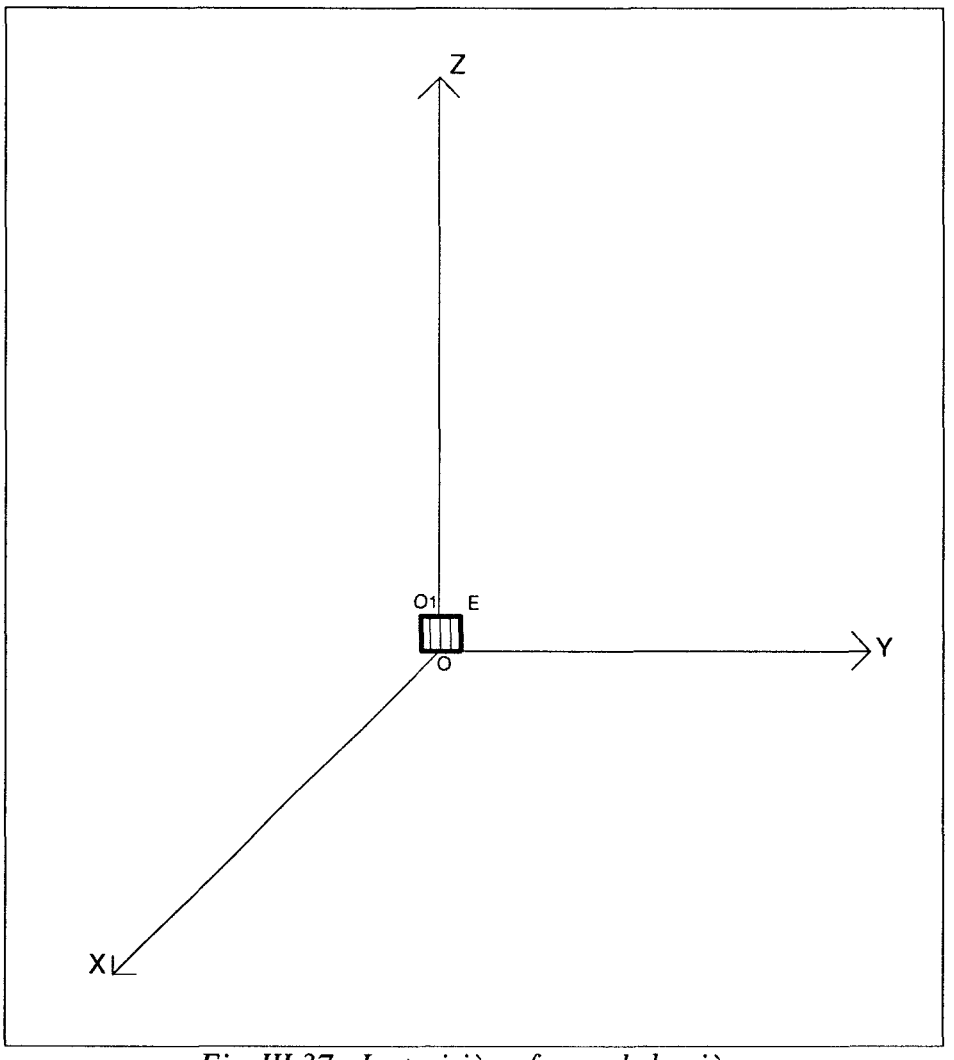


Fig. III.37 - La troisième forme de la pièce

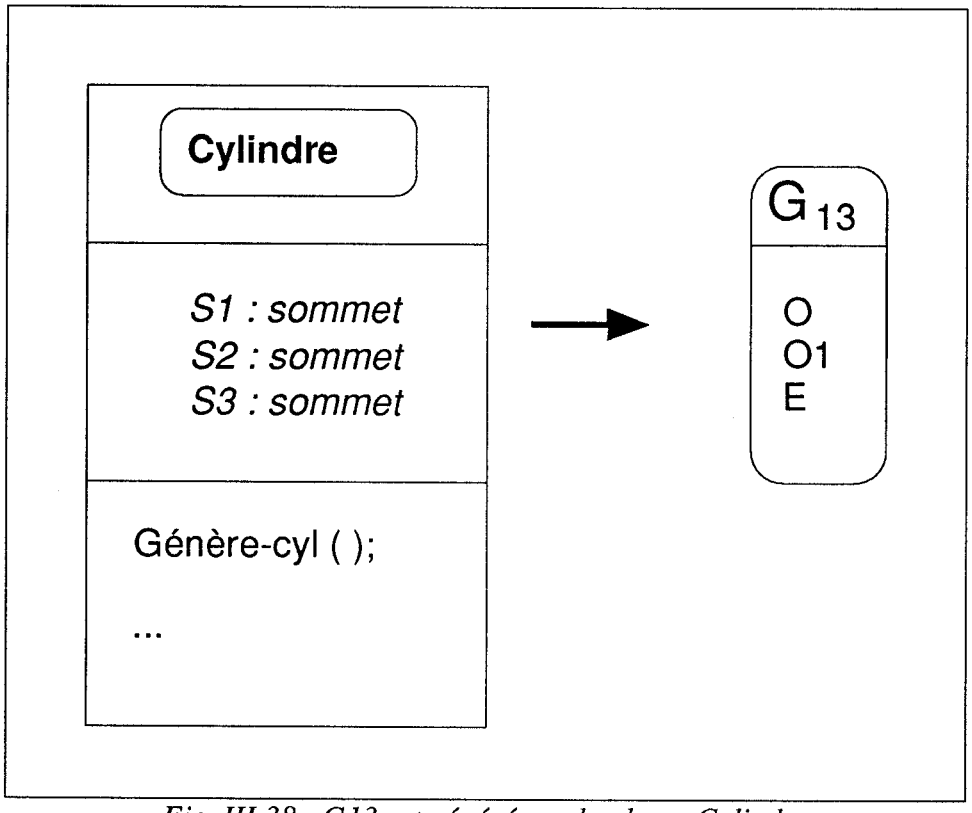


Fig. III.38 - G13 est généré par la classe Cylindre

III.8 - Conclusion

La modélisation informatique d'un processus passe par trois niveaux: le niveau physique, le niveau mathématique et le niveau représentation.

On a présenté dans ce chapitre, la modélisation du processus d'assemblage. Cette modélisation est nécessaire pour le système d'apprentissage graphique des robots d'assemblage. Le modèle proposé se classe parmi les modèles solides hybrides. L'univers de l'assemblage est modélisé par des classes selon la méthodologie orientée "Objets".

Chaque composant (pièce) est décrit par des paramètres géométriques, formés à partir d'un seul élément de base: l'élément "sommet". La structure proposée peut être extensible pour intégrer d'autres paramètres susceptibles de modéliser des aspects non géométriques du composant (matière, masse, densité, élasticité, ...). Des équations et des formules mathématiques de conversion gèrent la base de travail.

L'assemblage, quant à lui, est modélisé par un ordonnancement de composants et de matrices de positionnement. L'exemple cité en fin de chapitre, illustre la méthodologie utilisée.

Le chapitre suivant utilise cette modélisation pour réaliser la programmation du robot par apprentissage graphique.

CHAPITRE IV

LE MONTAGE AUTOMATIQUE ET LA PHASE D'APPRENTISSAGE

IV.1 - Introduction

L'apprentissage graphique des opérations d'assemblage est réalisé par démontage interactif des pièces. Pour cela, on effectue dans un premier temps le montage automatique des pièces dans l'ordre convenable, sans tenir compte des collisions: l'assemblage complet est ainsi présenté sur l'écran.

On procède ensuite au démontage graphique des pièces, l'une après l'autre, dans l'ordre inverse de leur montage, et cela en tenant compte des collisions. Ce démontage se déroule d'une manière interactive. En effet, l'opérateur, à partir des informations graphiques disponibles sur l'écran, lance des commandes de déplacement. Le calculateur traite ces commandes et valide celles qui sont physiquement possibles, c'est à dire celles qui ne créent pas d'effets de collisions.

L'interface de visualisation joue donc un rôle important lors de cette phase d'apprentissage. On présente dans ce chapitre, le montage automatique des pièces et la phase d'apprentissage graphique. Ensuite, on décrit l'interface de visualisation et l'aspect interactif homme-calculateur.

Finalement, le schéma type complet d'une session d'apprentissage graphique est présenté.

IV.2 - Définition

Soient:

- A_R, le sous assemblage formé par les R premières pièces à assembler: P_1,.., P_R ; $R \leq M$.

A_R est défini par,

A_R = { P_u ; u = 1,..,R } U { règles de positionnement dans A_R }

- A_{R+1}, le sous assemblage formé par les (R+1) premières pièces P_1,.., P_R, P_{R+1}.

De la même façon, A_{R+1} est défini par,

A_{R+1} = { P_v ; v = 1,.., (R+1) } U { règles de positionnement dans A_{R+1} }

- A_M est alors l'assemblage complet désiré.

L'ensemble des pièces de A_R et des règles de positionnement associées sont inclus dans A_{R+1} et par suite dans A_M. De ce fait, on dit que la structure A_R est incluse dans la structure A_{R+1}, ou inversement, A_{R+1} est une extension de A_R. En effet, le sous assemblage A_{R+1} est formé à partir du sous assemblage A_R par l'insertion de la pièce P_{r+1}.

Formellement, on considère les structures A_R comme les termes d'une suite récurrente dont le premier terme est A_0, et qui converge vers l'assemblage final A_M. Cette suite est croissante. La règle de récurrence ou de production est donnée par:

$A_{R+1} = A_R$ (+) P_{R+1}

- (+): Opérateur qui traduit l'insertion de P_{R+1} dans A_R. Cet opérateur sera défini ultérieurement.

On se propose alors d'effectuer un montage automatique des pièces: on construit A_M de manière récurrente à partir de A_0. A_0 est théoriquement le sous assemblage vide défini par une règle de positionnement unique indiquant la position souhaitée de l'assemblage.

IV.3 - Méthodes D'Assemblage

L'algorithme de montage se limite donc à trouver les méthodes d'assemblage de A_{R+1} à partir de A_R, ce qui revient à définir l'opérateur (+). Une simple itération sur R nous permet alors de trouver A_M. Cet algorithme, qu'on va présenter ci dessous est implanté par la méthode Montage ( ) associée à la classe Assemblage (sect. III.5.2).

En utilisant l'annotation majuscule pour représenter l'assemblage et l'annotation minuscule pour les pièces, Les structures respectives de l'assemblage et des pièces peuvent s'écrire de la façon suivante (sect. III.6.2, eq. 8 et III.6.3 eq. 9):

$P_r = \{ \Gamma r, g_r, n_{rj}, f_{rj}, s_{rj}, c_{rjk} \}$

- Γr : repère associé à la pièce Pr
- g_r : nombre de formes élémentaires dans la pièce P_r
- n_{rj} : signature de la jème forme de P_r
- f_{rj}: classe d'appartenance de la jème forme
- s_{rj} : nombre de sommets dans la jème forme de P_r
- c_{rjk} : kième sommet de la jème forme de P_r.

De façon similaire,

$A_R = \{ R, \Gamma, G_r, \Gamma r, H_r, N_{rj}, F_{rj}, S_{rj}, C_{rjk}, \}$

R: nombre de pièces dans l'assemblage A_R

Γ: repère de A_r = repère de A_M

H_r: matrice de positionnement de la pièce P_r par rapport à A_r

$r = 1, .., R$

$j = 1, .., G_r$

$k = 1,..,S_{rj}$

Le problème revient donc à trouver,

$$\{ G_{r+1}, \Gamma r+1, Hr+1, N(r+1)j, F(r+1)j, S(r+1)j, C(r+1)jk, \}$$

en fonction de:

$$\{ G_r, \Gamma r, Hr, Nrj, Frj, Srj, Crjk, \}$$

et

$$\{ \Gamma r+1, gr+1, n(r+1)j, f(r+1)j, s(r+1)j, c(r+1)jk \}$$

En effet, la structure de $A_{(r+1)}$ permet sa décomposition en deux sous structures:

* $A_{r+1} = A1_{r+1} + A2_{r+1}$

- $A1_{r+1} = A1_{r+1}(j, k)$: sous structure de A_{r+1} rapportée au domaine $j \leq G_r$

- $A2_{r+1}$: sous structure complémentaire rapportée au domaine $G_r < j \leq G_{r+1}$

Avec cette décomposition, on peut démontrer, qu'en général:

* $A1_{r+1} = A_r$

Cette démonstration découle directement du fait que les structures P_i sont supposées indépendantes et par suite, la somme des structures est égale à la structure des sommes.

* $A2_{r+1} = H_{r+1}$ (*) $P_{r+1}((j-G_r), k)$

- H_{r+1}: matrice de passage du repère de P_{r+1} à celui de A_M. (noter que les A_i ont tous le même repère.)
- (*): opérateur de multiplication qui transforme c_{ijk} en c'_{ijk} tel que, $c'_{ijk} = H_i * c_{ijk}$.

Explicitement, on construit A_{r+1} en utilisant les formules de passage suivantes:

$G_{r+1} = G_r + g_{r+1}$;

Pour:

- $j \leq G_r$,

$N_{(r+1)j} = N_{rj}$

$F_{(r+1)j} = F_{rj}$

$S_{(r+1)j} = S_{rj}$

$C_{(r+1)jk} = C_{rjk}$

- $G_r < j \leq G_{r+1}$,

$N_{(r+1)j} = n_{r+1}(j\text{-}Gr)$

$F_{(r+1)j} = f_{r+1}(j\text{-}Gr)$

$S_{(r+1)j} = s_{r+1}(j\text{-}Gr)$

$C_{(r+1)jk} = H_{r+1} * c_{r+1}((j\text{-}Gr), k)$

Ayant défini les formules de passage, on peut alors énoncer l'algorithme de montage.

IV.4 - Algorithme de Montage

IV.4.1 - Principe

L'algorithme de montage comporte deux boucles principales imbriquées. La première boucle, calcule les A_r. Le point de départ est A_0 et la condition d'arrêt est limitée à A_M. Les formules de passage sont celles définies ci dessus.

La seconde boucle, comporte des tests de réduction. En effet, dans des cas très courants, l'insertion de P_{r+1} modifie la structure de A_R. Une telle situation se produit par exemple lors de l'insertion d'un cylindre dans une pièce comportant une cavité cylindrique. Dans ce cas $A1_{r+1}$ ne sera plus égal à A_R. Une boucle de calcul est donc nécessaire afin de réduire la structure de A_R.

IV.4.2 - Structure Générale

L'algorithme de montage devient alors (fig. IV.1):

1) Initialisation

2) Boucle Principale:

* calcul de A_r (structure primaire)

    * boucle des tests:

    - test I

    - test II

    - réduction de A_r (structure définitive)

    * fin de la boucle des tests

* visualisation de A_r

* fin de la boucle principale

3) Acquisition et Visualisation de A_M

4) Sortie.

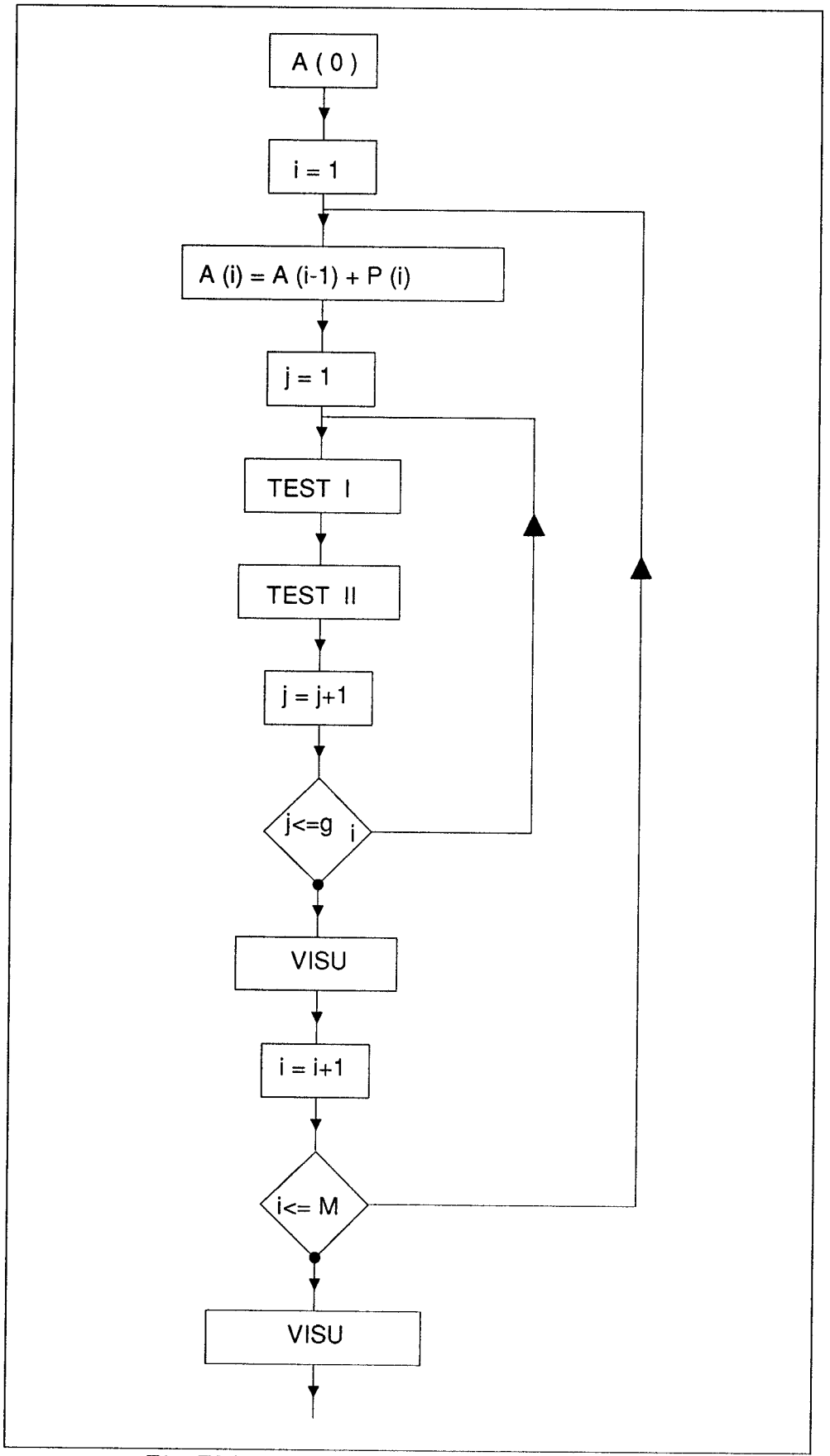


Fig. IV.1 - L'algorithme de montage automatique

IV.5 - Tests de Réduction

Ces tests ont pour but principalement de trouver les variations de $A1_r$ en fonction de $A2_r$. Pour cela on représente A_r sous la forme suivante:

$$A_r = A1_r + A2_r$$

Avec:

$$-A1_r = \{ R1, \Gamma, G1_r, \Gamma r, H1r, N1rj, F1rj, S1rj, C1rjk \}$$

$$A2_r = \{ R2, \Gamma, G2_r, \Gamma r, H2r, N2rj, F2rj, S2rj, C2rjk \}$$

Dans ce cas, la structure de A_r s'écrit (en tenant compte des termes qui nous intéressent):

$$A_r = \{ (G1_r, G2_r), (N1_r, N2_r), (S1_{rj}, S2_{rj}), (F1_{rj}, F2_{rj}), (C1_{rjk}, C2_{rjk}), .. \}$$

Les tests visent alors à comparer les formes élémentaires de $A1_r$ et $A2_r$. Pour cela, on utilise le langage suivant:

* Deux formes sont dites réductibles si elles ont (fig. IV.2):

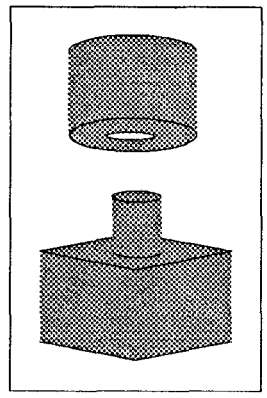


Fig. IV.2 - Deux formes réductibles

- la même classe d'appartenance,
- des signatures opposées ((+) et (-)),
- les mêmes paramètres géométriques.

Dans ce cas, la cavité sera absorbée par la forme pleine lors de l'assemblage.

* Deux formes sont dites complémentaires si elles ont (fig. IV.3):

- La même classe d'appartenance,
- la même signature,
- des faces en commun.

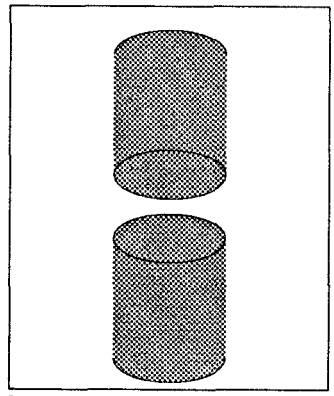


Fig. IV.3 - Deux formes complémentaires

Dans ce cas, les deux formes se concatènent lors de l'assemblage, pour donner une nouvelle forme.

IV.5.1 - Test I

Le premier test porte sur les couples (j1, j2) pour lesquels,

$$F1(j1) = F2(j2)$$

et

$$N1(j1) = - N2(j2)$$

Pour ces couples, on compare si les deux formes en question ont les même paramètres géométriques. Par exemple, pour deux cylindres, on teste si les deux formes ont le même centre, le même rayon, et le même vecteur directeur (sect. III.5.4).

Dans ce cas, les deux formes d'indice j1 et j2 se réduisent. Cela se fait en annulant tous les paramètres relatifs à j1 et j2. Les deux formes disparaissent de la structure, et par suite, G_r est diminué de deux.

IV.5.2 - Test II

Si ce n'est pas le cas, on teste si les couples des formes relatifs à (j1, j2) ont des faces en commun. Pour cela, on forme les couples (j1, j2) pour lesquels,

$$F1(j1) = F2(j2)$$

et

$$N1(j1) = N2(j2)$$

Sur ces couples, on cherche à trouver si les deux formes en question ont des faces communes. Dans ce cas les deux formes sont complémentaires. Cela se traduit par la fusion des deux formes en une seule. On enlève ces deux formes de la structure de l'assemblage et on insère une nouvelle forme qui est créée par concaténation des deux anciennes formes. G_r est alors diminué de 1.

IV.6 - La Phase d'Apprentissage

Le montage automatique des pièces étant effectué, on peut procéder à la phase d'apprentissage par démontage. En effet, lors du montage automatique, la trajectoire de parcours de la pièce est directe: la pièce P_i se déplace de sa position initiale à sa position finale sans tenir compte des collisions. Ce déplacement direct est donné par la matrice de transfert H_i définie par (fig. IV.4):

- une translation
- une rotation

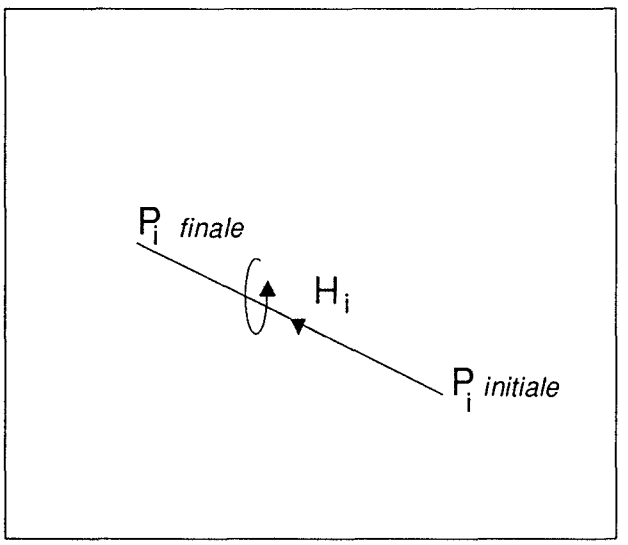


Fig. IV.4 - La trajectoire du montage automatique

Par contre, en phase de démontage, la pièce doit parcourir le chemin inverse, de sa position finale à sa position initiale, en tenant compte des collisions. Pour cela, le déplacement effectué n'est pas direct. La pièce P_i doit parcourir au total un déplacement caractérisé par $R_i = H_i^{-1}$ qui n'est pas directe, comme le montre le schéma de la figure IV.5 .

On peut décomposer la matrice de déplacement R_i, en une suite de parcours élémentaires directes. Si on nomme R_{il} la lième étape intermédiaire de parcours, on peut poser l'équation suivante:

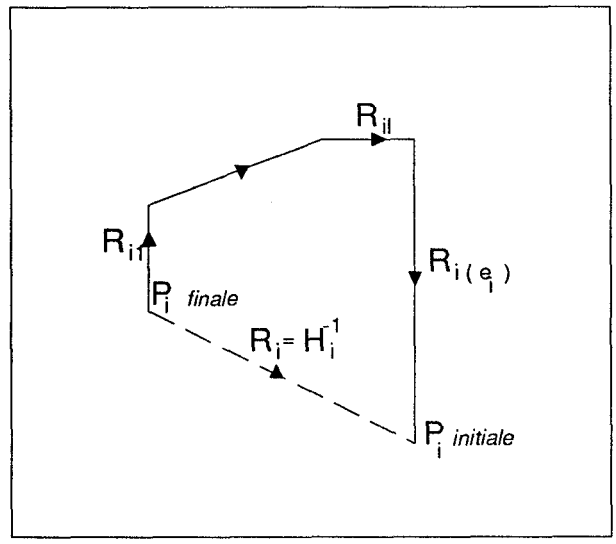


Fig. IV.5 - Trajectoire de parcours lors du démontage

$$R_i = \prod_{l=1}^{ei} R_{il}$$

- R_i: matrice résultante totale du déplacement de la pièce P_i lors du démontage
- e_i: nombre d'étapes intermédiaires de déplacement
- l: indice des déplacements élémentaires intermédiaires. Son domaine est [1 e_i]
- R_{il}: matrice du lième déplacement élémentaire de la pièce P_i

Le rôle de l'opérateur lors de la phase d'apprentissage est de déterminer, de manière interactive, pour chaque pièce P_i de l'assemblage, ses étapes de parcours intermédiaires: c'est à dire déterminer e_i et les R_{il}. En effet, si

$$\{ R_{i1}, R_{i2}, .., R_{i(ei)} \}$$

est la liste ordonnée des matrices R_{il} de P_i, définissant les trajectoires de parcours de la pièce lors de son démontage, on peut calculer par simple inversion la trajectoire réelle de montage de la pièce. Si on nomme:

$$H_{il} = R_{il}^{-1}$$

la liste inverse

$$\{ H_{i(ei)}, H_{i(ei-1)}, .., H_{i1} \} \qquad (10)$$

donne la trajectoire de montage de la pièce. C'est cette trajectoire qui est nécessaire au robot pour exécuter la tâche d'assemblage.

Tout le but de la phase d'apprentissage revient donc à déterminer interactivement les trajectoires R_{il}. Pour cela on procède de la façon suivante:

1) On effectue le montage graphique de toutes les pièces dans l'ordre convenable du montage, et cela sans tenir compte des collisions.

2) On présente l'assemblage total sur l'écran suivant quatre vues: les trois vue du dessin industriel et une quatrième vue perspective 3D simplifiée.

3) Pour chaque pièce présente dans l'assemblage, et en suivant l'ordre inverse de celui défini par la gamme d'assemblage, on exécute le cycle suivant:

3.1) Le système indique sur l'écran, la position de la pièce en question au sein de l'assemblage, et sa position initiale de départ (avant le montage).

3.2) A partir de ces informations graphiques, l'opérateur commence par régler le mode de préhension de la pièce. Pour cela, il sélectionne sur l'écran:

- le point de préhension de la pièce

- une configuration géométrique indiquant le mode de préhension. Une telle configuration est étroitement liée au préhenseur (effecteur) du robot. Le choix de cette configuration est donc réglé en tenant compte du type de l'effecteur réel connecté au robot.

La sélection graphique d'un point sur l'écran peut se faire par les touches flèches du clavier ou par utilisation d'une souris.

3.3) Ensuite, l'opérateur procède en lançant des commandes de déplacements élémentaires de type translation ou rotation suivant un axe. Pour se faire, il peut

choisir d'une manière interactive (par menu):

- le type de déplacement: translation, rotation, .., ou combiné
- l'axe de déplacement: X, Y, Z, ou autre
- le pas de déplacement

Ensuite, il fait exécuter son choix par l'ordinateur (une commande de déplacement comporte donc trois arguments: le type, l'axe et le pas).

3.4) Une fois la commande de déplacement élémentaire lancée, le calculateur interprète cette commande et calcule l'effet de collision d'une telle commande. Ceci s'effectue en testant géométriquement si le volume engendré par la pièce chevauche d'autres volumes déjà occupés dans la scène.

Si le déplacement demandé n'entraîne pas de collision, le calculateur valide le choix de l'opérateur et exécute la commande lancée: il déplace la pièce à sa nouvelle position en recalculant le nouveau sous assemblage résultant du démontage de la pièce. Le système sauvegarde alors la matrice de déplacement dans une liste chaînée.

Dans le cas contraire, si le déplacement demandé est physiquement impossible, le calculateur rejette la commande et signale à l'opérateur que son choix n'est pas valide. L'opérateur doit donc modifier les arguments de sa commande.

3.5) Les étapes 3.3 et 3.4 sont répétée jusqu'à ce que la pièce ait retrouvée avec succès sa position initiale. Dans ce cas l'opérateur valide le démontage de cette pièce et demande au calculateur de sauvegarder cette phase d'apprentissage.

3.6) Le système interprète la liste des matrices de transfert trouvée en 3.4. Il inverse les matrices et les reclassent dans une nouvelle liste suivant l'ordre inverse (conformément à l'équation 10). Il sauvegarde finalement cette liste.

4) Le cycle est répété jusqu'au démontage de toutes les pièces. Le système sauvegarde les phases d'apprentissage de toutes les pièces.

Ce cycle d'apprentissage graphique est implanté par la méthode Apprnt ( ) associée à la classe Assemblage (Sect. III.5.2).

IV.7 - L'Interface de Visualisation

La phase d'apprentissage nécessite une interactivité entre l'opérateur et la machine. L'ergonomie au niveau de l'interface de visualisation est un facteur essentiel. On présente ci dessous les écrans et les menus interactifs.

IV.7.1 - Le Menu Général

Le menu général de l'application est présenté par la figure IV.6 . Les options de ce menu sont:

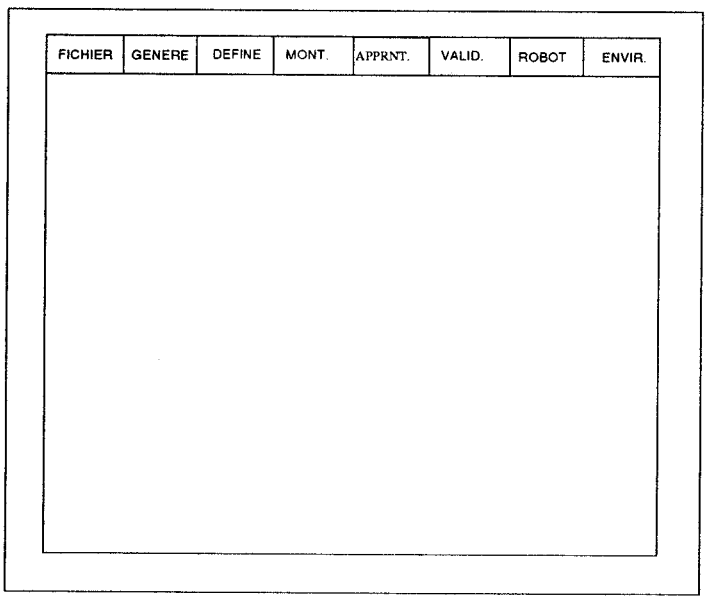


Fig. IV.6 - Le menu général

- **FICHIER**: gestion des fichiers et du système

- **GENERE** : génération des formes géométriques élémentaires issues des classes:

Cylindre, Sphère, Cube, ... Cette option est détaillée ci dessous.

- **DEFINE**: définit les instances objets de l'application, issues des classes: Pièce, Assemblage, ... Cette option est détaillée ci dessous.

- **MONT.** : écran du montage automatique des pièces. Cette option est détaillée ci dessous.

- **APPRNT.**: écran d'apprentissage graphique du processus d'assemblage. Cette option est détaillée ci dessous.

- **VALID.**: validation graphique (par simulation) de la phase d'apprentissage. En effet, une phase de validation du cycle d'assemblage est nécessaire avant de passer à l'exécution réelle (par le robot) du processus.

- **ROBOT** : choisit le modèle de robot de l'application: configuration cinématique, effecteur, ...

- **ENVIR.**: définit l'environnement de l'application: échelle, communication avec l'extérieur, synchronisation, ...

IV.7.2 - Le Menu GENERE

Le menu GENERE se charge de la génération des formes géométriques élémentaires: méthodes Génère- ( ) associées aux classes Cube, Sphère, Cylindre, Cône, Sec-cyl et Sec-cône. La figure IV.7 montre l'écran de ce menu. Il comporte les options suivantes:

- **CUBE**: génération d'une instance de la classe Cube. L'entrée des attributs se fait par un menu-page (fig. IV.7).

- **CYLINDRE**: génération d'une instance cylindre

- **SPHERE**: génération d'une sphère

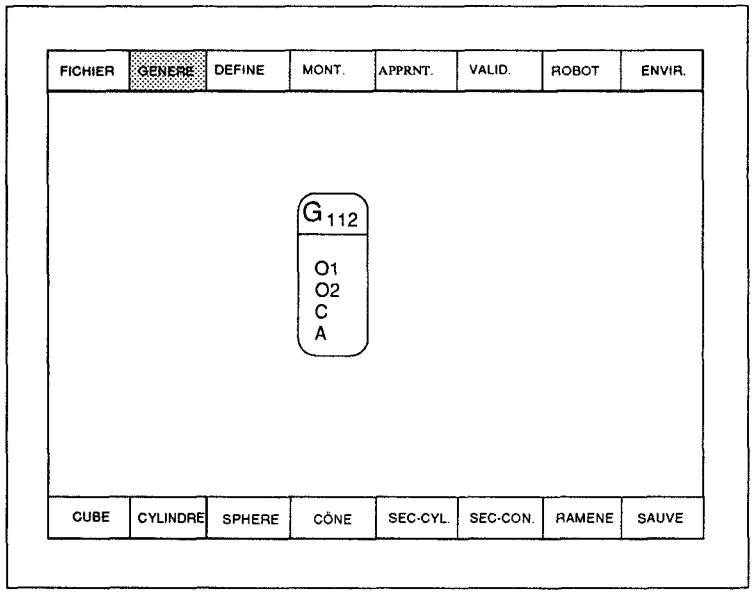


Fig. IV.7 - Le menu GENERE

- **CONE**: génération d'un cône

- **SEC-CYL.**: génération d'une section cylindrique

- **SEC-CON.**: génération d'un tronc conique

- **RAMENE**: chargement à partir de la mémoire de masse, d'un objet préalablement généré

- **SAUVE**: sauvegarde d'une instance générée

L'entrée des attributs pour une instance donnée, se fait par un menu-page correspondant. Ce menu-page comprend:

- une entête: le nom de l'instance objet générée
- des cases simples pour saisir les attributs simples de l'objet
- des tableaux de cases pour saisir les attributs listes de l'objet en question

On utilise ce format de menu-page pour la saisie des attributs des instances de toutes les classes de l'univers. La saisie d'un champ peut se faire explicitement ou implicitement par le

renseignement du nom d'un autre objet.

IV.7.3 - Le Menu DEFINE

L'écran de ce menu est présenté par la figure IV.8 . Ce menu utilise les méthodes Def- ( ) pour définir les pièces ainsi que la gamme d'assemblage. Il comprend les options suivantes:

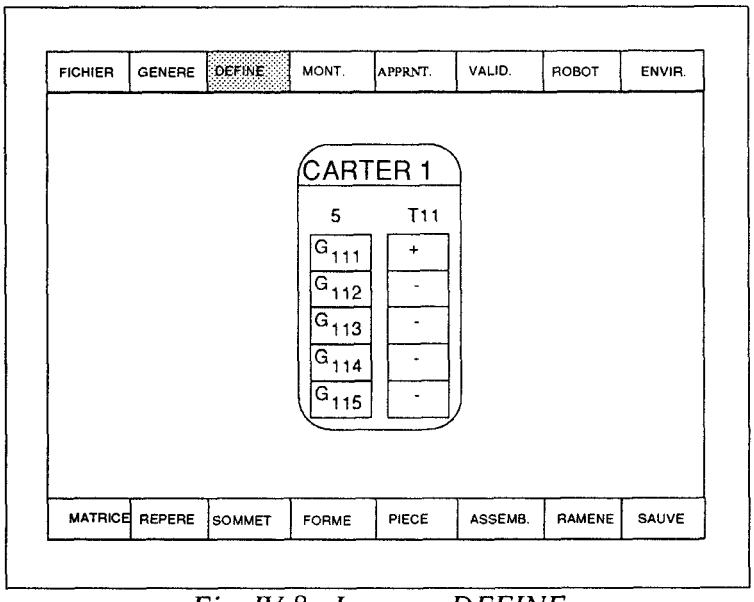


Fig. IV.8 - Le menu DEFINE

- **MATRICE**: pour instancier une matrice par saisie de ces attributs.

- **REPERE**: pour instancier des repères

- **SOMMET**: pour instancier des sommets

- **FORME**: pour instancier une forme (remarquons que la classe Forme est abstraite dans le présent travail)

- **PIECE**: pour définir une pièce donnée (instance de la classe Pièce). La saisie des attributs se fait par un menu-page (fig. IV.8). Ce menu-page comprend:

- une entête
- une case simple pour la saisie de l'attribut Nb-formes. Cet attribut est saisie explicitement.
- une case simple pour la saisie de l'attribut Mon-rep. La saisie de cet attribut se fait implicitement par l'entrée d'un nom d'une instance Repère.
- un tableau de cases pour la saisie implicite de la liste Mes-formes
- un tableau de cases pour la saisie explicite de la liste Signatures

- **ASSEMB.**: pour définir la gamme d'assemblage souhaitée: c'est à dire renseigner dans l'ordre, les pièces qui constituent l'assemblage et leurs positions au sein de cet assemblage (méthode Def-ass ( ) associée à la classe Assemblage). Cette définition se fait par instanciation d'un objet de la classe Assemblage. La saisie des attributs se fait par un menu-page.

- **RAMENE**: pour charger un objet déjà créer.
- **SAUVE** : pour sauvegarder un objet.

IV.7.4 - Le Menu APPRNT.

C'est le menu d'apprentissage graphique par démontage des pièces (méthode Apprent ( )). L'écran graphique de ce menu est présenté par la figure IV.9 . L'écran est divisé en quatre quadrants pour présenter les quatre vues de la scène:

- le quadrant supérieur gauche présente la vue de dessus.
- le quadrant supérieur droit présente la vue de droite.
- le quadrant inférieur gauche présente la vue de face.
- le quadrant inférieur droit présente la vue perspective 3D.

Les options de ce menu sont:

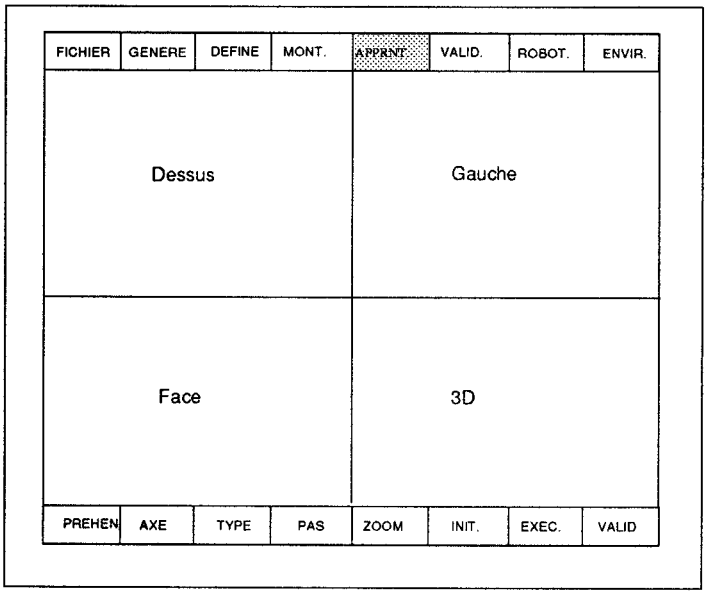


Fig. IV.9 - Le menu d'apprentissage graphique

- **PREHEN.**: pour régler le mode de préhension de l'effecteur (étape 3.2 du cycle d'apprentissage)

- **AXE**: pour choisir l'axe de déplacement (étape 3.3 du cycle d'apprentissage)

- **TYPE**: pour choisir le type de déplacement: translation, rotation, ... (étape 3.3 du cycle d'apprentissage).

- **PAS**: pour régler le pas de déplacement (étape 3.3 du cycle d'apprentissage).

- **ZOOM**: pour sélectionner une vue de travail parmi les quatres vues et effectuer un zoom.

- **INIT.**: pour initialiser un cycle d'apprentissage ou pour recommencer en cas de rejet de la commande de déplacement en étape 3.4 du cycle d'apprentissage.

- **EXEC.**: pour lancer la commande de déplacement (étape 3.4).

- **VALID**: pour valider le cycle d'apprentissage d'une pièce et sauvegarder les résultats après inversion des matrices (étape 3.6 et 4).

IV.7.5 - Le Menu MONT.

C'est le menu correspondant à la méthode Montage ( ) de la classe Assemblage. Il se charge du montage automatique des pièces de
l'assemblage. Cependant, ce menu supporte une option de montage manuel des pièces. L'écran graphique de ce menu est semblable à l'écran de l'apprentissage. Les options sont:

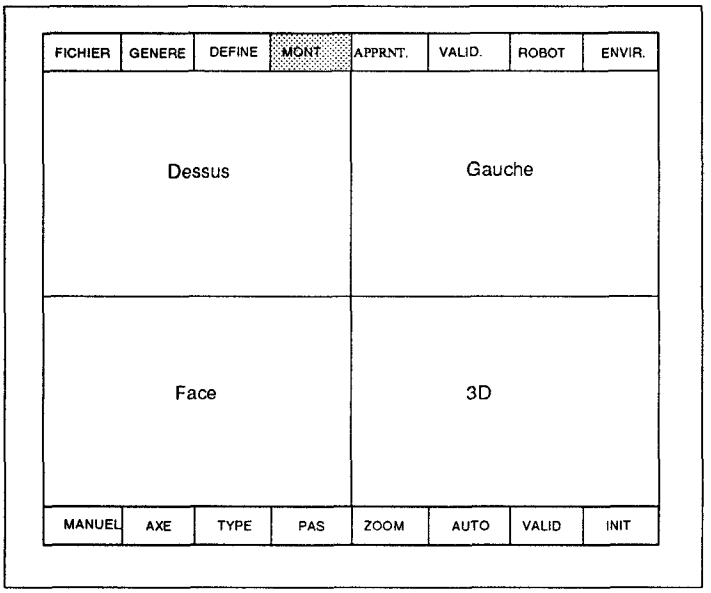


Fig. IV.10 - L'écran de montage des pièces

- **MANUEL**: montage manuel des pièces

- **AXE**: réglage de l'axe de déplacement

- **TYPE**: type de déplacement

- **PAS**: pas de déplacement

- **ZOOM**: effectuer un zoom sur une vue donnée

- **AUTO**: montage automatique des pièces (méthode Montage ( ))

- **VALID**: validation du montage

- **INIT**: initialisation du montage

IV.8 - Déroulement d'une Session de Programmation Graphique

Une session complète de programmation graphique d'un processus d'assemblage comporterait les étapes suivantes:

1) Acquisition, à partir du Bureau d'Etude de l'usine, de la description de tous les composants (pièces) de la base de travail: dessin technique, dimensions, éventuellement prototype, ...

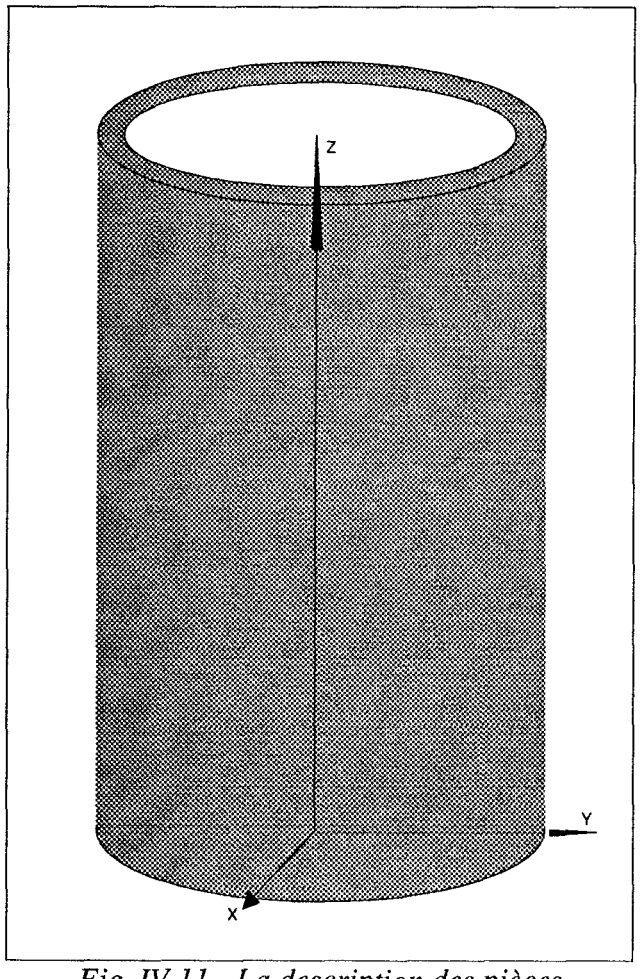


Fig. IV.11 - La description des pièces

2) Pour chaque pièce, repérage des "sommets" nécessaires pour la modélisation, et acquisition des données réelles de ces sommets: coordonnées, repères, matrices de positionnement, ...

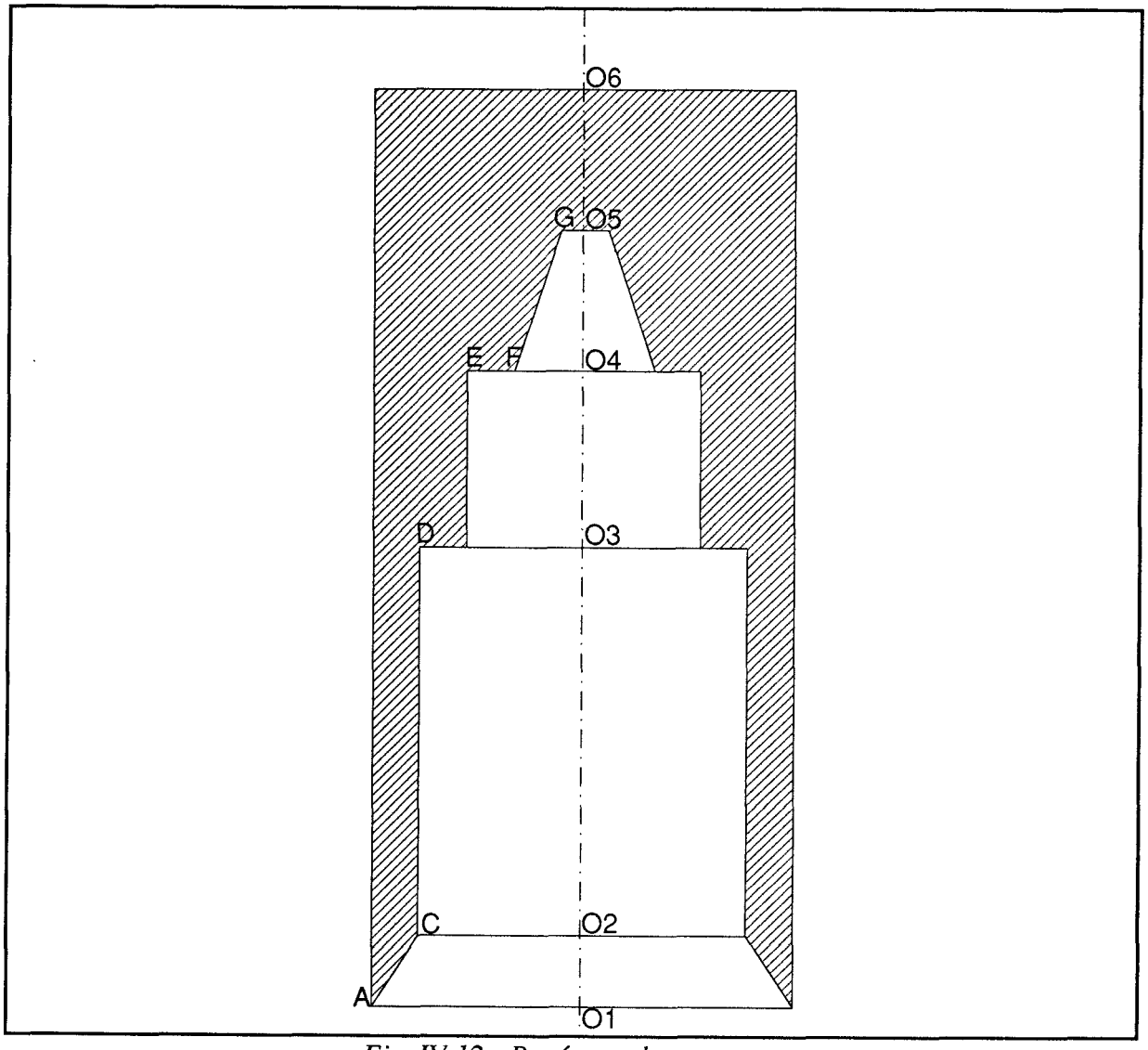


Fig. IV.12 - Repérage des sommets

3) Lancement du logiciel de programmation.

4) Réglage de l'environnement de l'application sous ENVIR. (échelle, synchronisation, ...).

5) Modélisation de la base de travail sous le menu DEFINE. Cette modélisation se déroule de la façon suivante:

a- on sélectionne l'option PIECE et on définit une instance pour chaque pièce de la base de travail. La définition d'une instance consiste à saisir les champs de cette instances. Pour les attributs qui sont eux même des objets (Mon-rep, Mes-formes), on saisie implicitement le champs. Ensuite, on définit à fur et à mesure ces attributs suivant leur classe d'appartenance.

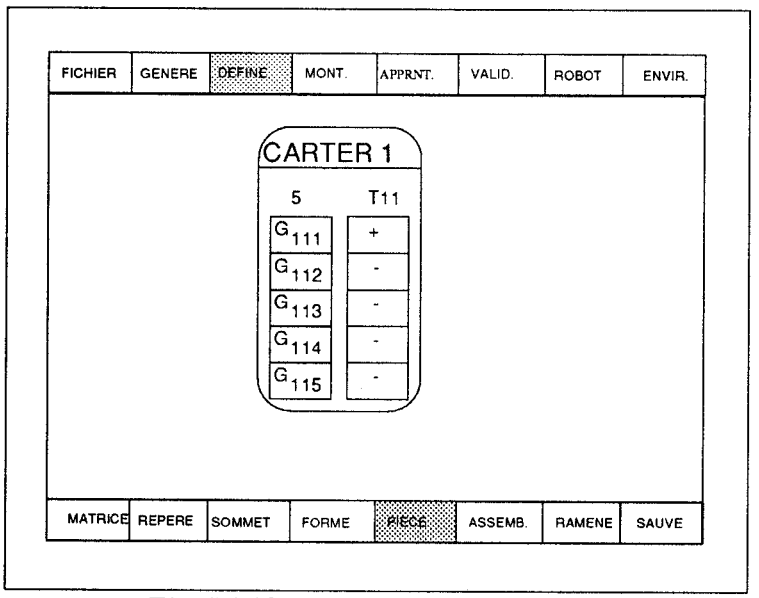


Fig. IV.13 - La définition des pièces

b- on sélectionne l'option REPERE et on définit explicitement les repères qui sont saisie en (a).

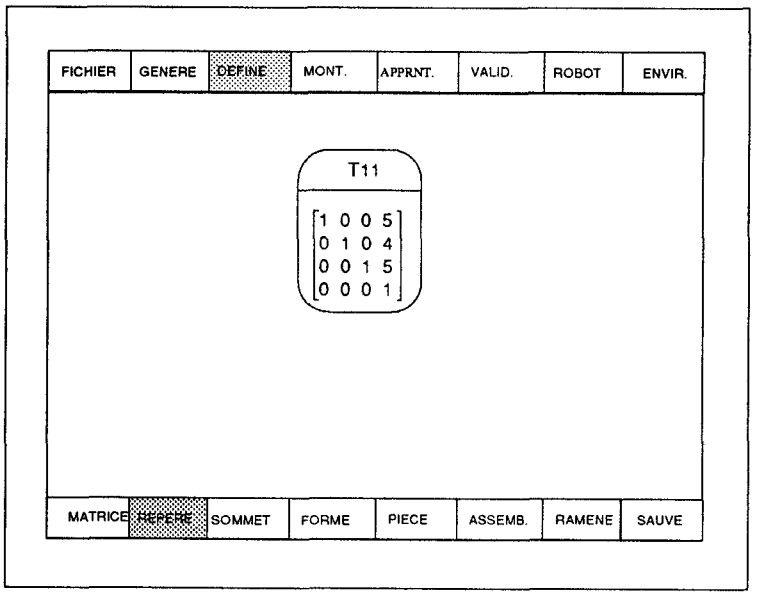


Fig. IV.14 - Définition des repères

c- sous GENERE, on génère les formes qui sont saisies en (a). La saisie de ces formes se fait implicitement par des Sommets.

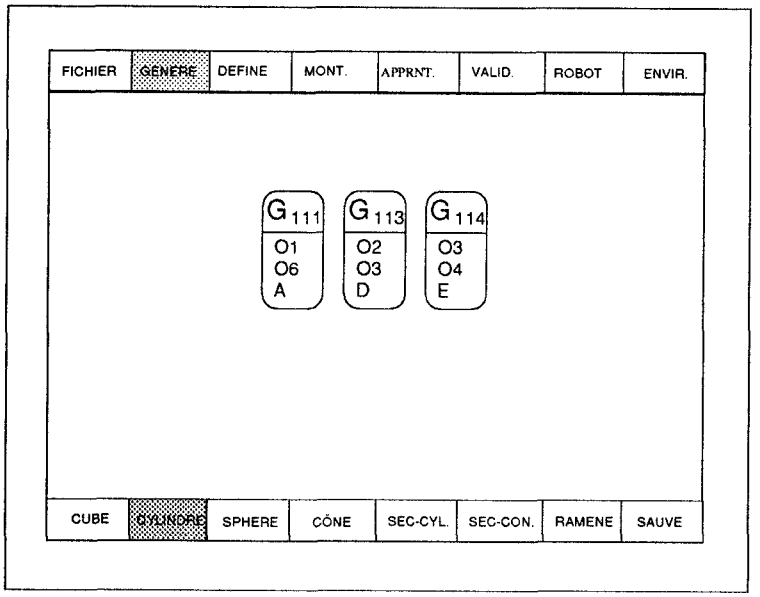


Fig. IV.15 - Génération des formes de type Cylindre

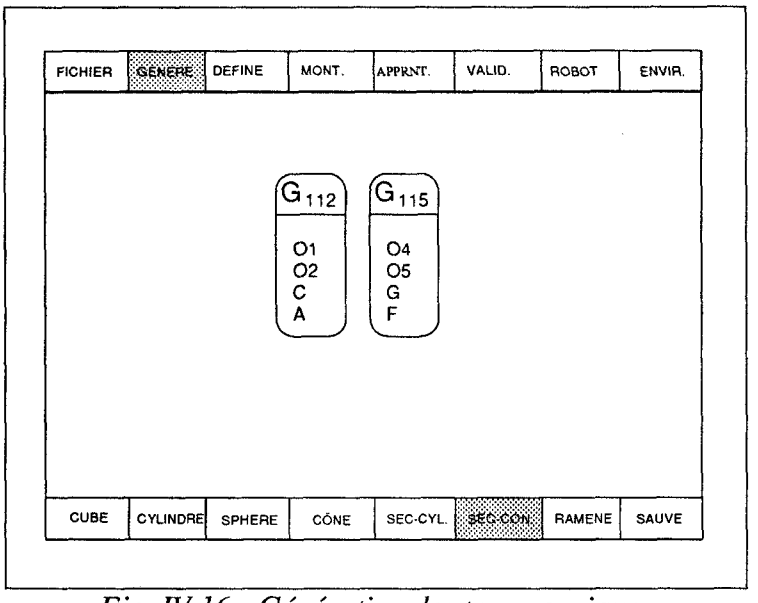


Fig. IV.16 - Génération des tronc coniques

d- sous DEFINE / SOMMET, on saisie explicitement les différents sommets de (c). Les coordonnées de ces sommets sont celles relevées en 2).

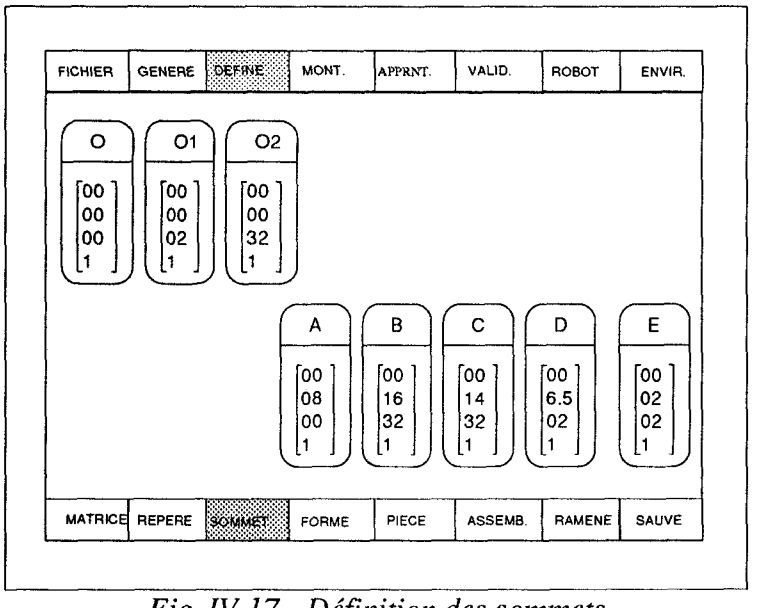


Fig. IV.17 - Définition des sommets

6) Définition de la gamme d'assemblage sous DEFINE / ASSEMB. On définit l'assemblage souhaité en saisissant les attributs d'une instance de la classe Assemblage. Les attributs implicites (Posi-pièces) de cet assemblage seront définis explicitement sous l'option DEFINE / MATRICE (en utilisant les données relevées en 2)).

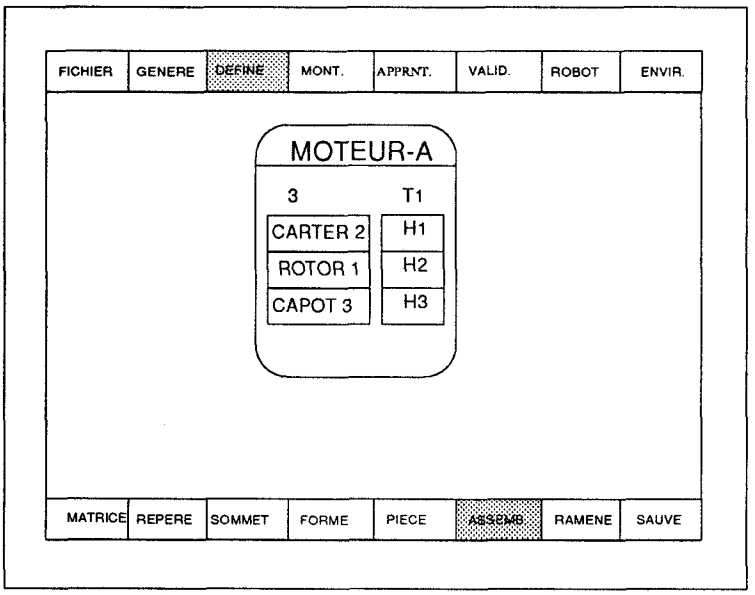


Fig. IV.18 - La selection de la gamme d'assemblage

7) Montage automatique des pièces sous MONT.

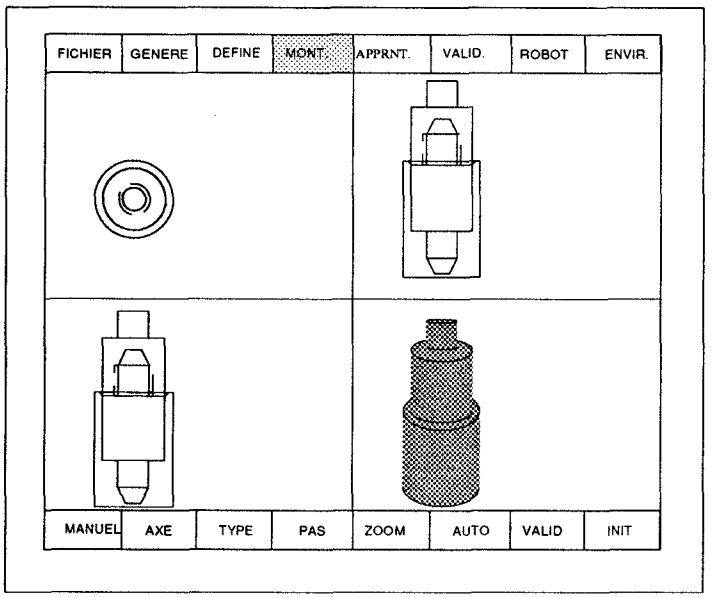
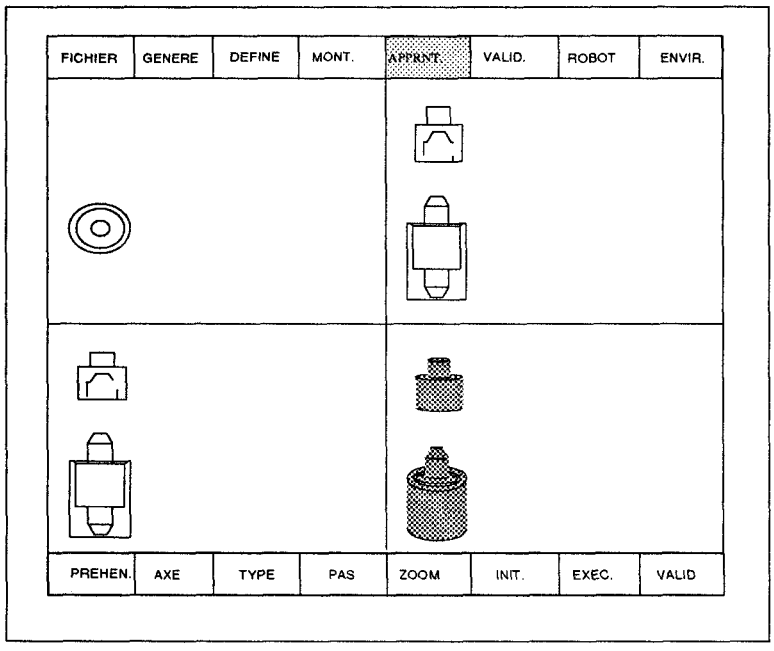


Fig. IV.19 - Réalisation du montage des pièces

8) Réglage du modèle robot et périrobotique sous ROBOT

9) Apprentissage graphique du processus d'assemblage sous APPRNT.



IV.9 - Conclusion

On a présenté dans ce chapitre le montage automatique des composants d'une gamme d'assemblage et la phase d'apprentissage graphique.

Le montage automatique consiste à amener la pièce, de sa position initiale à sa position finale, par un déplacement direct (sans tenir compte des collisions). L'algorithme est basé sur un calcul itératif des états progressifs de l'assemblage. En effet, à chaque sous assemblage correspond un état. L'état initial est l'assemblage vide. L'état final est l'assemblage complet. La transition d'un état à un autre se traduit par l'insertion d'une pièce supplémentaire.

La phase d'apprentissage graphique se déroule sur un écran, de manière interactive. Le cycle des opérations d'apprentissage par démontage des pièces a été présenté. Le démontage d'une pièce consiste à trouver un enchainement d'opérations de déplacements élémentaires directs de cette pièce, en vue de l'amener de sa position finale dans l'assemblage, vers sa position initiale de départ et cela en tenant compte des contraintes géométriques et mécaniques.

On a également décrit les menus et les écrans de l'interface de visualisation ainsi que le schéma global d'une session de programmation graphique des robots d'assemblage .

3ème Partie

CHAPITRE V

APPLICATION

V.1 - Introduction

On se propose dans ce dernier chapitre, de présenter une application industrielle de la méthode de programmation graphique des robots d'assemblage, développée précédemment.

Cette application porte sur l'assemblage d'une gamme de moteurs électriques. Pour cela, une cellule flexible d'assemblage conçue et installée au Centre d'Automatique de l'Université des Sciences et Techniques de Lille-Flandres-Artois, a été utilisée.

Chaque moteur est constitué de trois pièces élémentaires: un carter, un rotor et un capot. L'ensemble des pièces mécaniques disponibles comporte trois types de carters, deux types de rotors et trois types de capots. Ces pièces sont conçues de manière à permettre l'assemblage de n'importe quelle combinaison des trois types de composants. Ainsi, plusieurs gammes d'assemblage sont possibles.

La chaîne d'assemblage comporte quant à elle, deux robots PUMA 500 à configuration anthropomorphique, dont la présentation est faite en début de chapitre.

Dans un deuxième volet, on effectue la description et la modélisation de l'ensemble des pièces mécaniques formant la base de travail.

Ensuite, la gamme d'assemblage est sélectionnée (création de "MOTEUR-A" à partir des composants élémentaires). Puis le calcul des paramètres et la représentation graphique des pièces et des sous-assemblages sont réalisés.

Le déroulement des opérations de montage automatique est alors présenté. Finalement, l'apprentissage est réalisé par le démontage graphique du moteur.

V.2 - La Cellule d'Assemblage

La cellule expérimentale est constituée des matériels suivants (implantée selon le schéma de la figure V.1):

- deux robots PUMA 500 de STAUBLI UNIMATION. Ces robots ont une configuration anthropomorphique et sont équipés de deux types d'effecteurs.
- un convoyeur
- un système de vision EXPERT de ALLEN BRADLEY, pour l'acquisition et le traitement temps réel des images (extraction des contours, reconnaissance des formes, traitement morphologique, ...). Ce système est équipé de deux caméras matricielles de type PULNIX 760.
- un site d'alimentation sur lequel les pièces sont placées en vrac.
- un site d'évacuation.
- un micro-ordinateur IBM AT pour la supervision de la cellule. Ce calculateur va servir pour l'implantation des logiciels de simulation et pour l'apprentissage graphique des processus d'assemblages. Il communique avec les robots et le système de vision via des liaisons asynchrone V-24 (RS232-C).

La cellule était installée par la société SERPAC. Elle va servir pour valider expérimentalement la méthodologie de modélisation et d'apprentissage graphique. Nous allons décrire le déroulement du cycle d'apprentissage graphique comme présenté en section IV.8 .

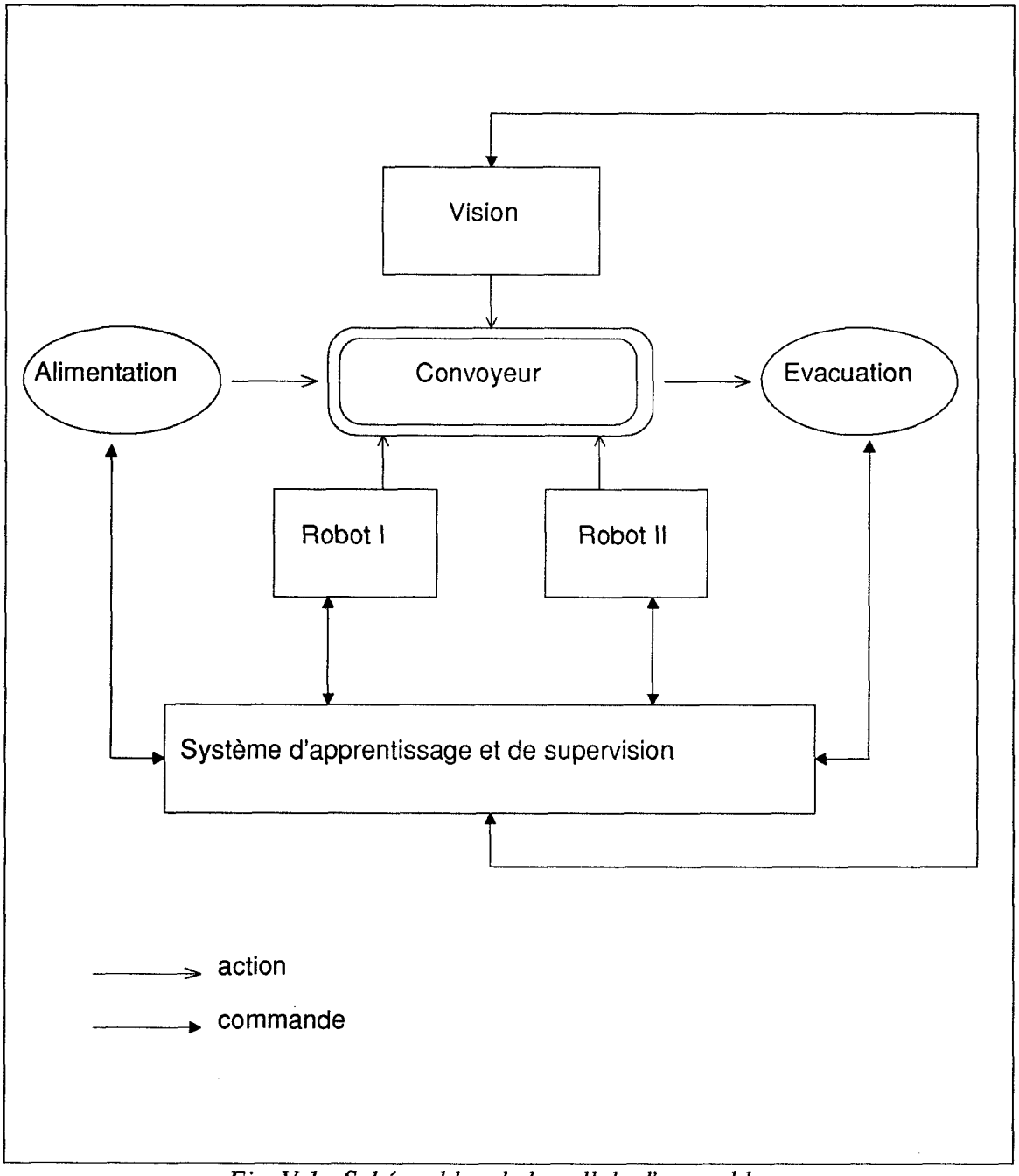


Fig. V.1 - Schéma bloc de la cellule d'assemblage

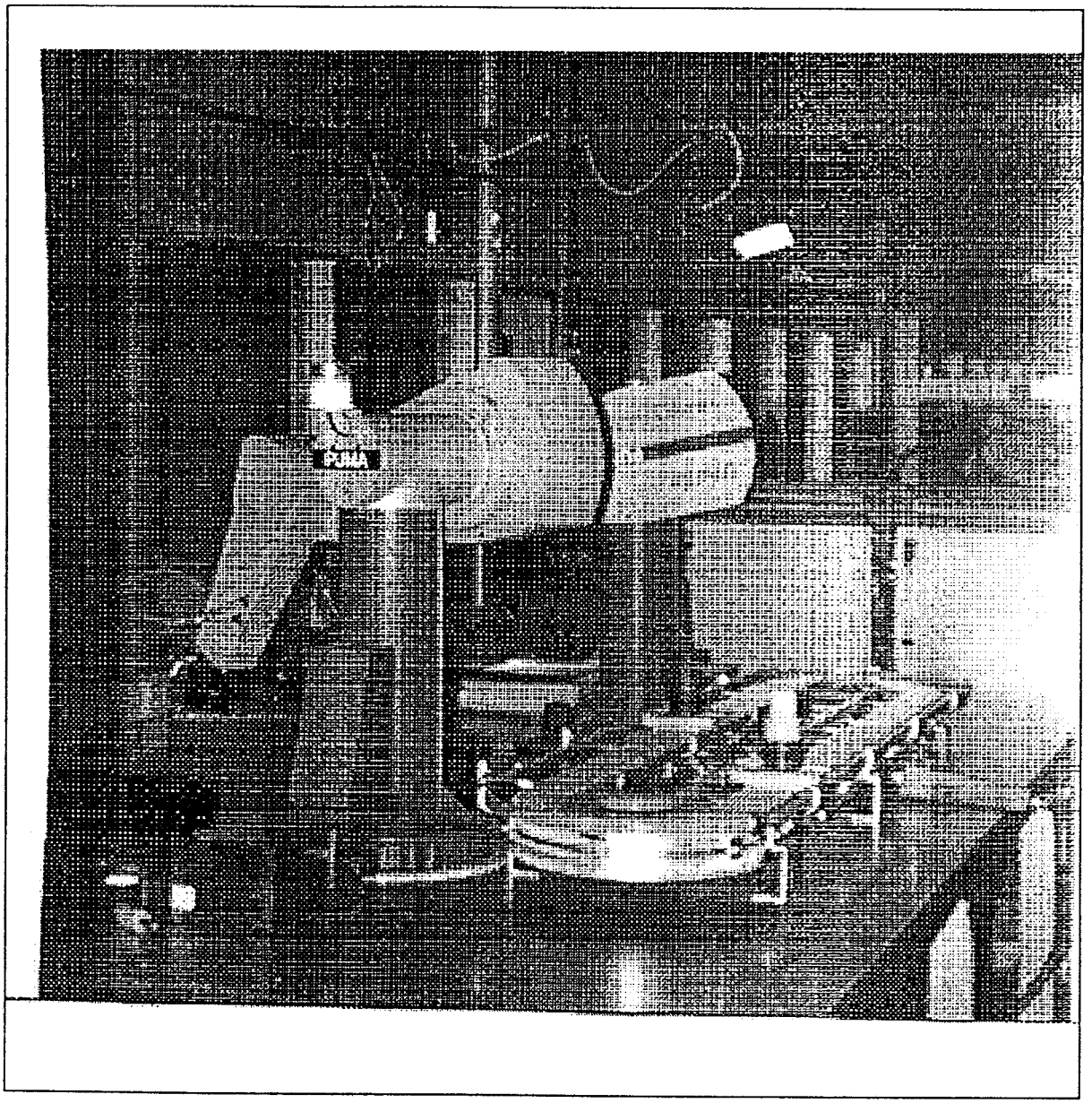


Fig. V.2 - La cellule d'assemblage

V.3 - Description de la Base de Travail

La base de travail est constituée de trois types de carters, deux types de rotors et trois types de capots, ainsi que l'illustre la figure V.3 .
Les dessins techniques des pièces sont donnés en vues de face, vues de profil et vues de dessus.

V.4 - Repérage des Données

En raison des différents types existant pour une pièce, on associe à chaque pièce de l'assemblage, un indice supplémentaire de type. Ainsi,

$P_i(j, k) = \{ P_{it}(j, k) \}$

- i: indice de l'ordre de la pièce dans l'assemblage

- t: indice du type de la pièce

- P_{it}: la ième pièce de l'assemblage, de type t

Les données des pièces (sommets, repères, ...) sont présentées sur les figures V.4 jusqu'à V.19.

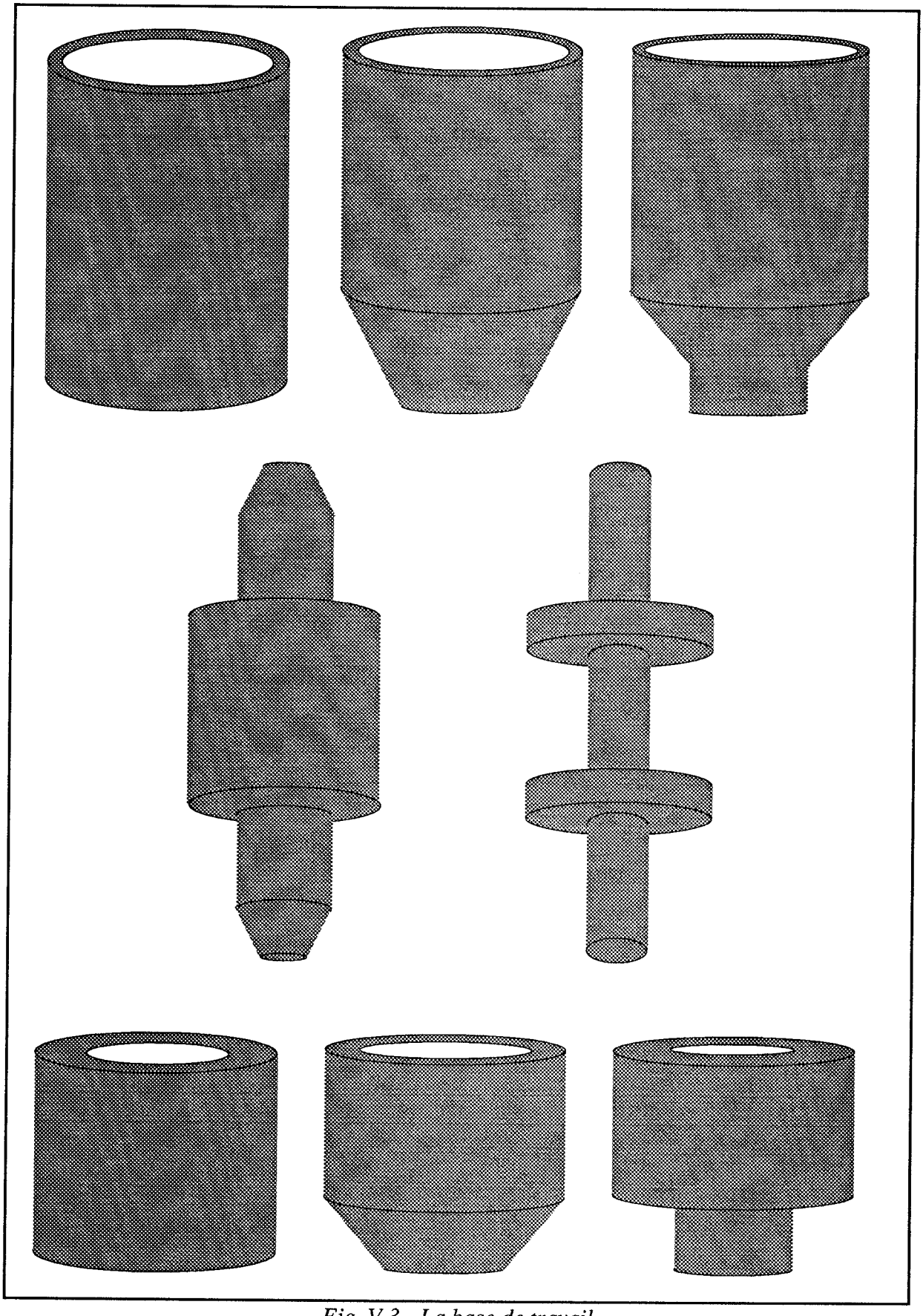


Fig. V.3 - La base de travail

CARTER TYPE I

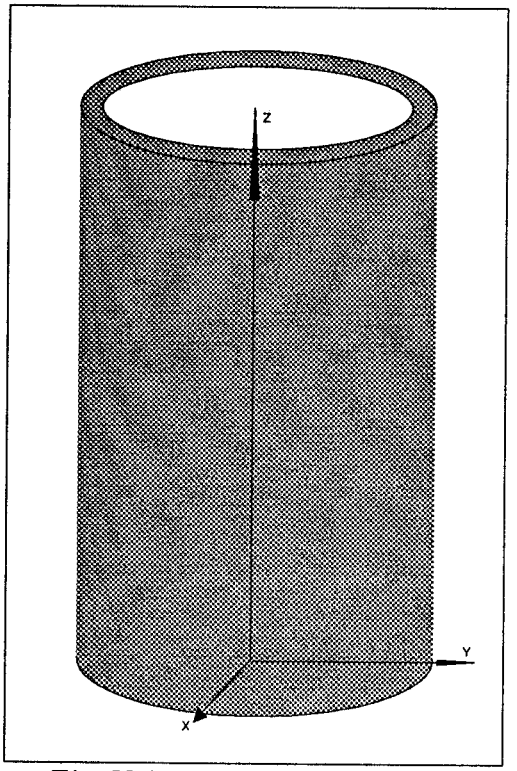


Fig. V.4 - vue 3D du CARTER I

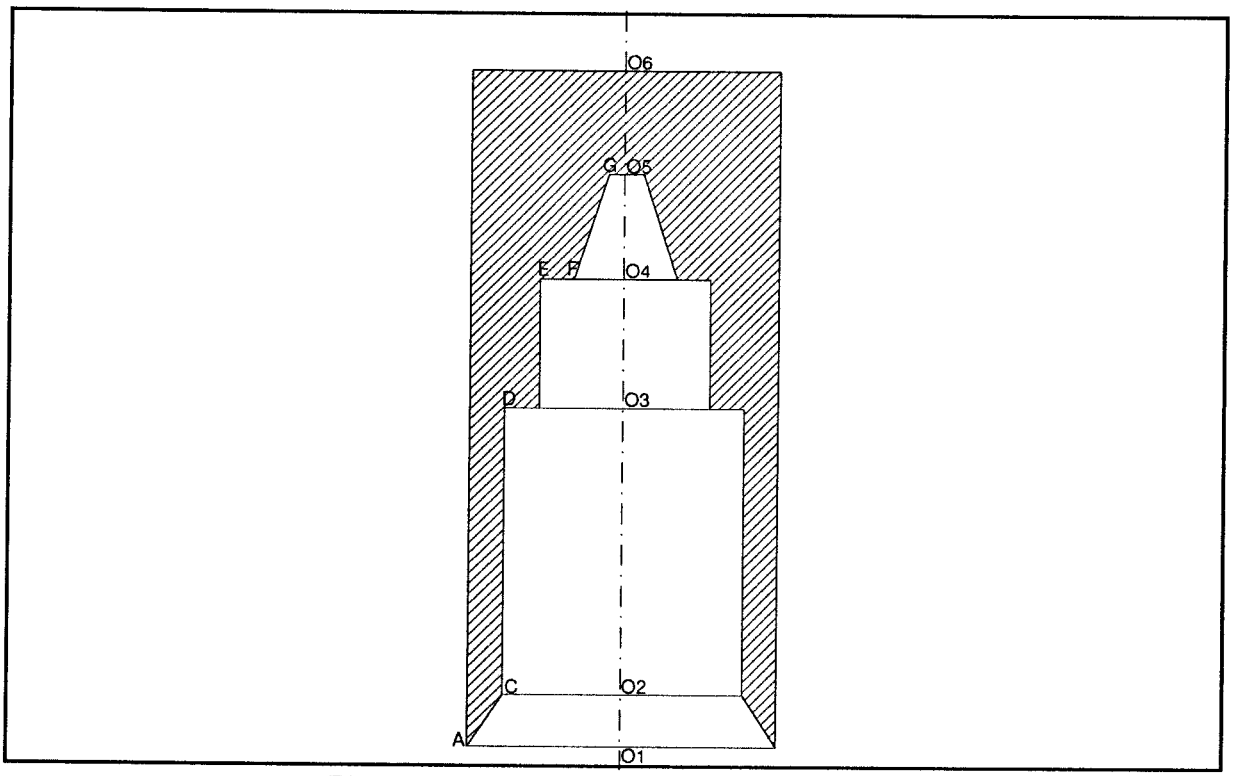


Fig. V.5 - CARTER I : repérage des sommets

CARTER TYPE II

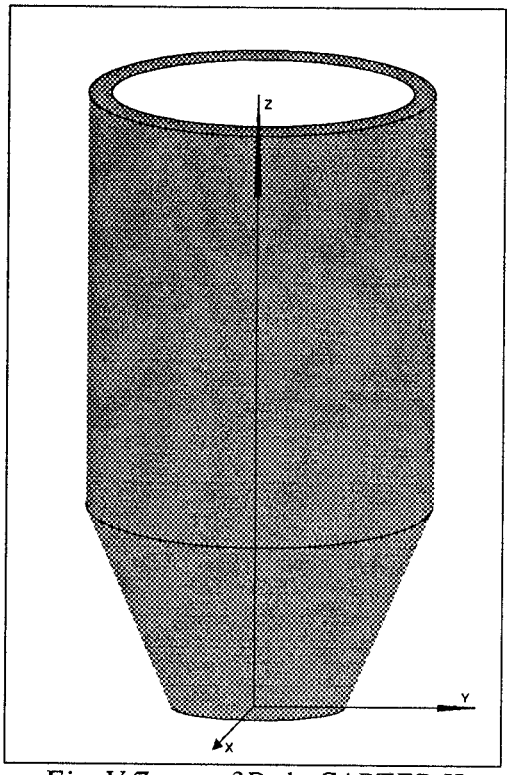


Fig. V.7 - vue 3D du CARTER II

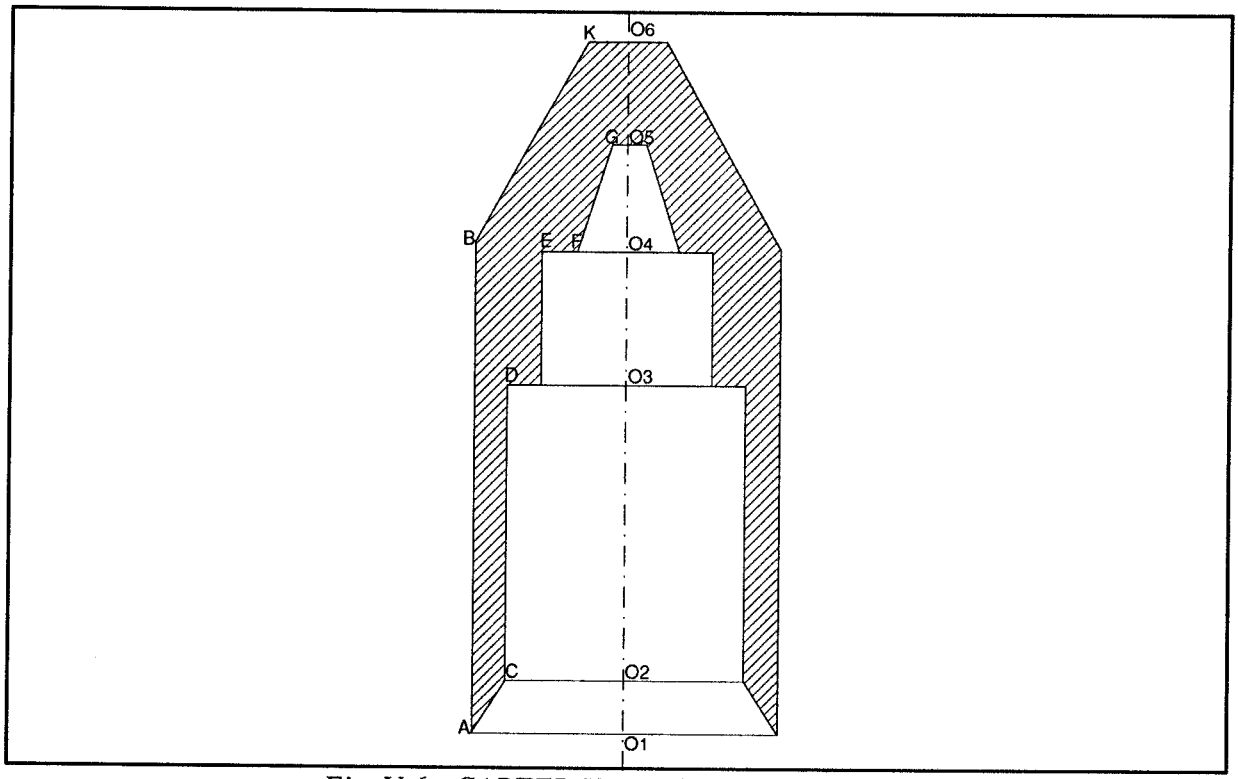


Fig. V.6 - CARTER II : repérage des sommets

CARTER TYPE III

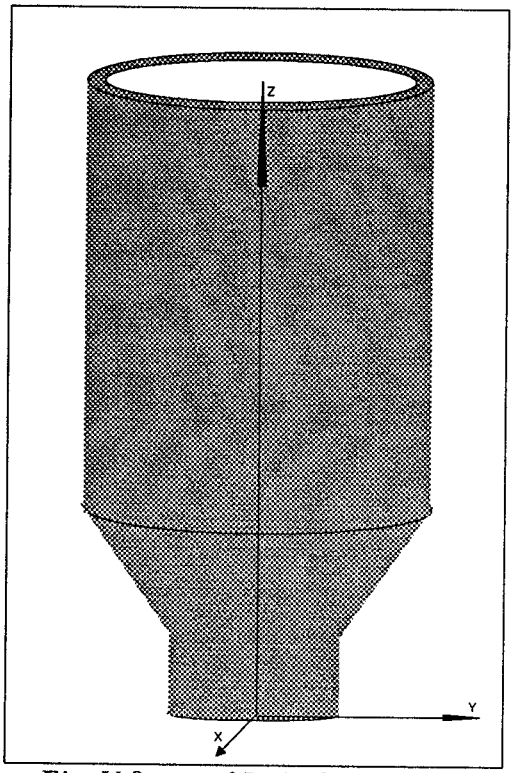


Fig. V.9 - vue 3D du CARTER III

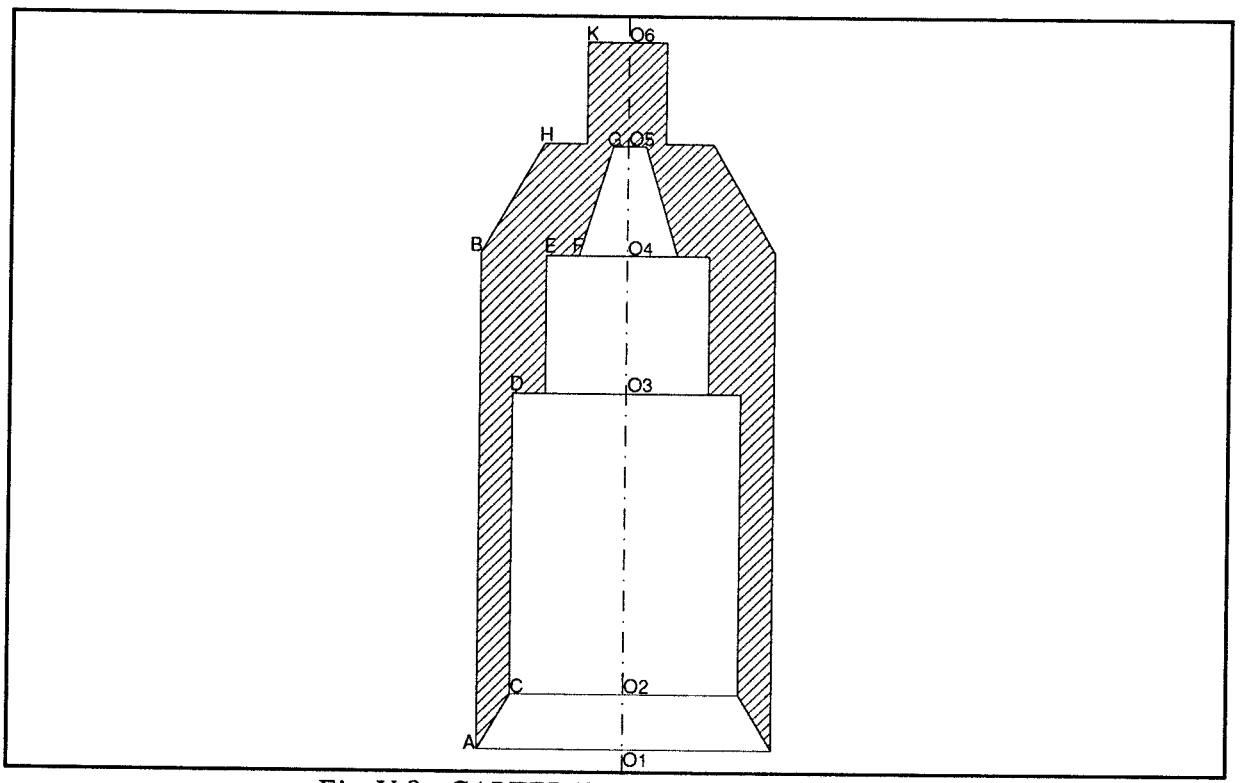


Fig. V.8 - CARTER III : repérage des sommets

ROTOR TYPE I

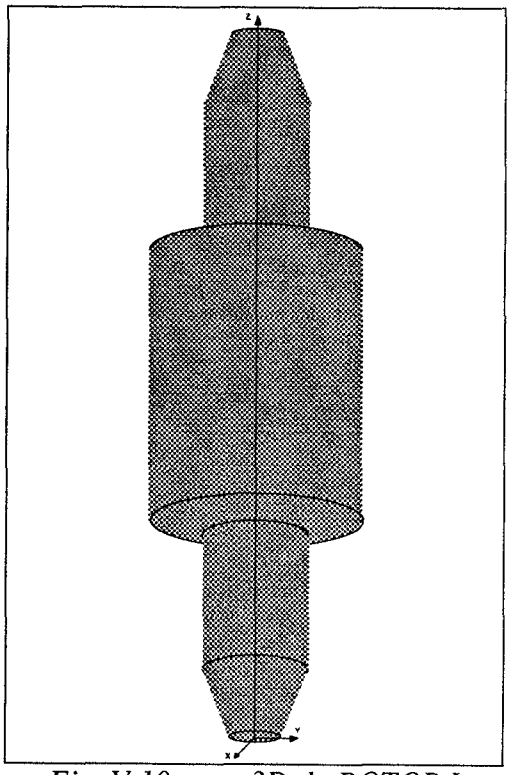


Fig. V.10 - vue 3D du ROTOR I

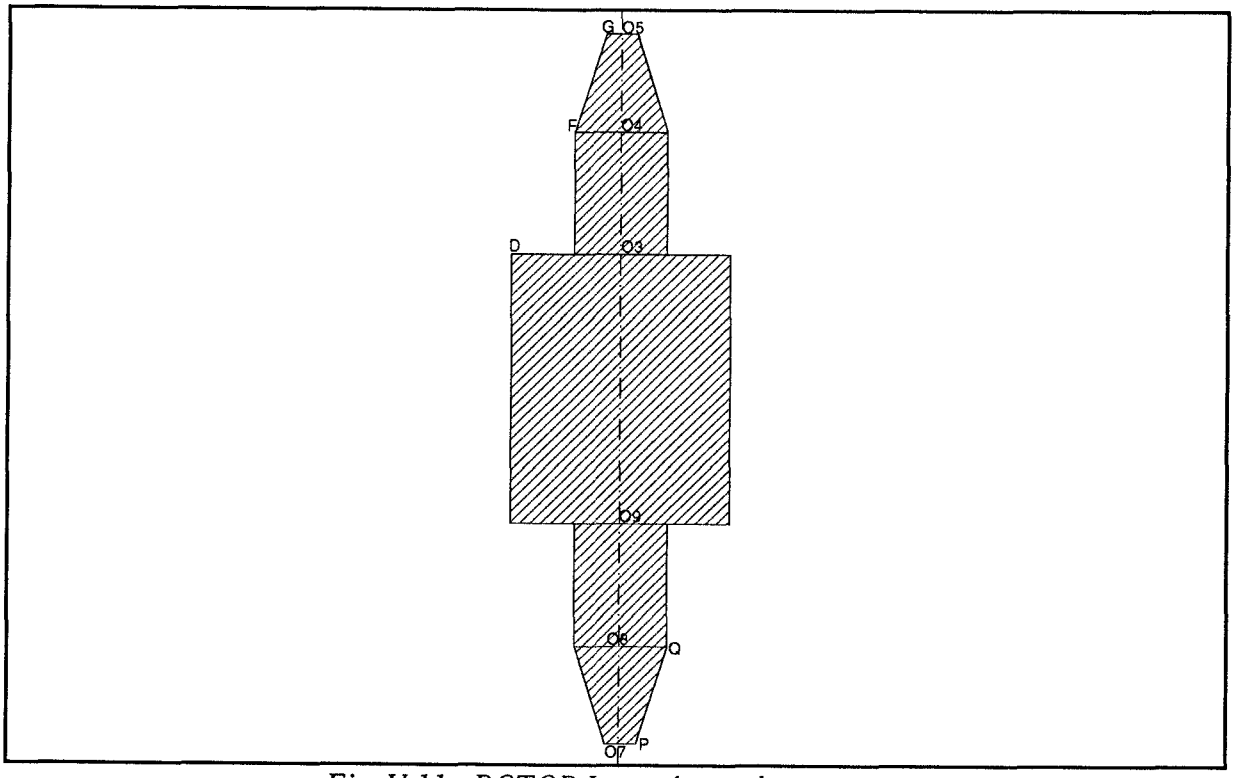


Fig. V.11 - ROTOR I : repérage des sommets

ROTOR TYPE II

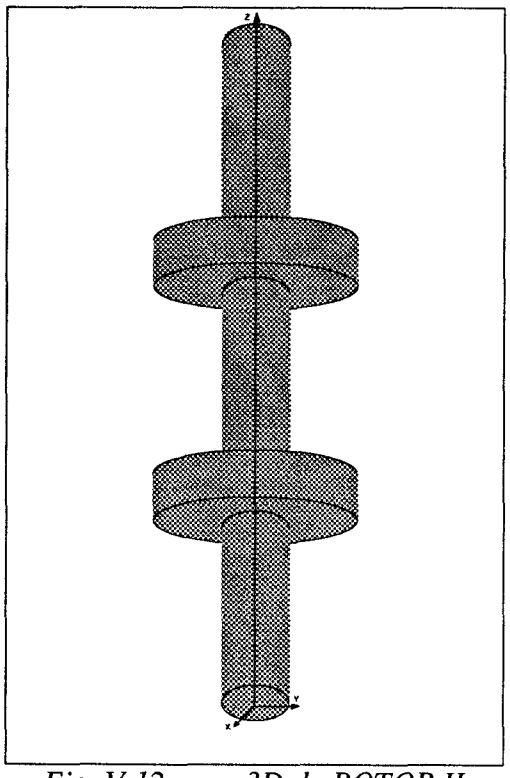


Fig. V.12 - vue 3D du ROTOR II

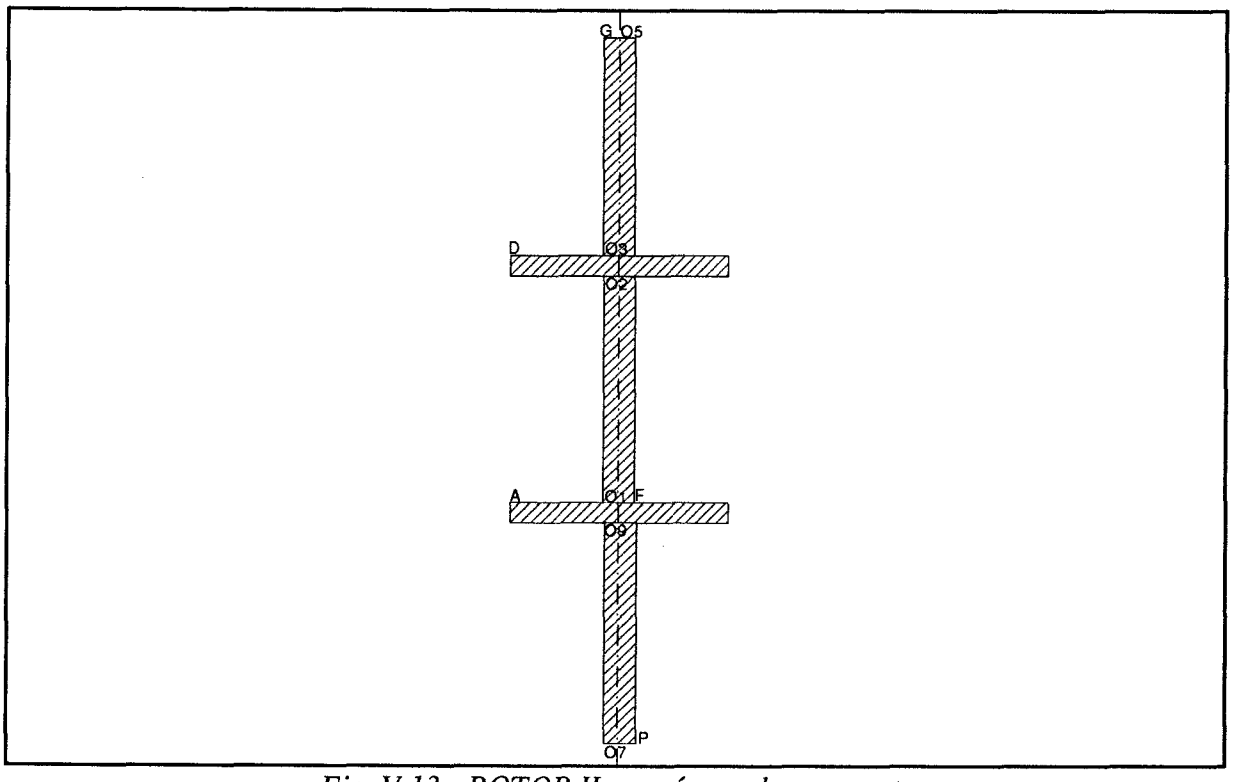


Fig. V.13 - ROTOR II : repérage des sommets

CAPOT TYPE I

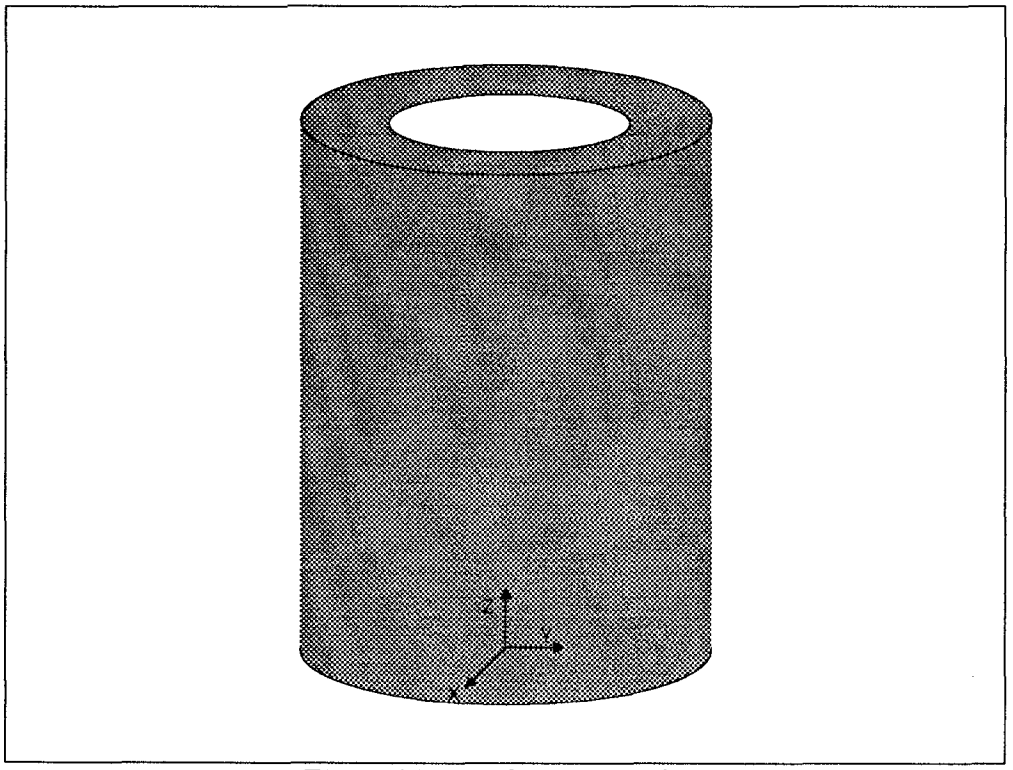


Fig. V.14 - vue 3D du CAPOT I

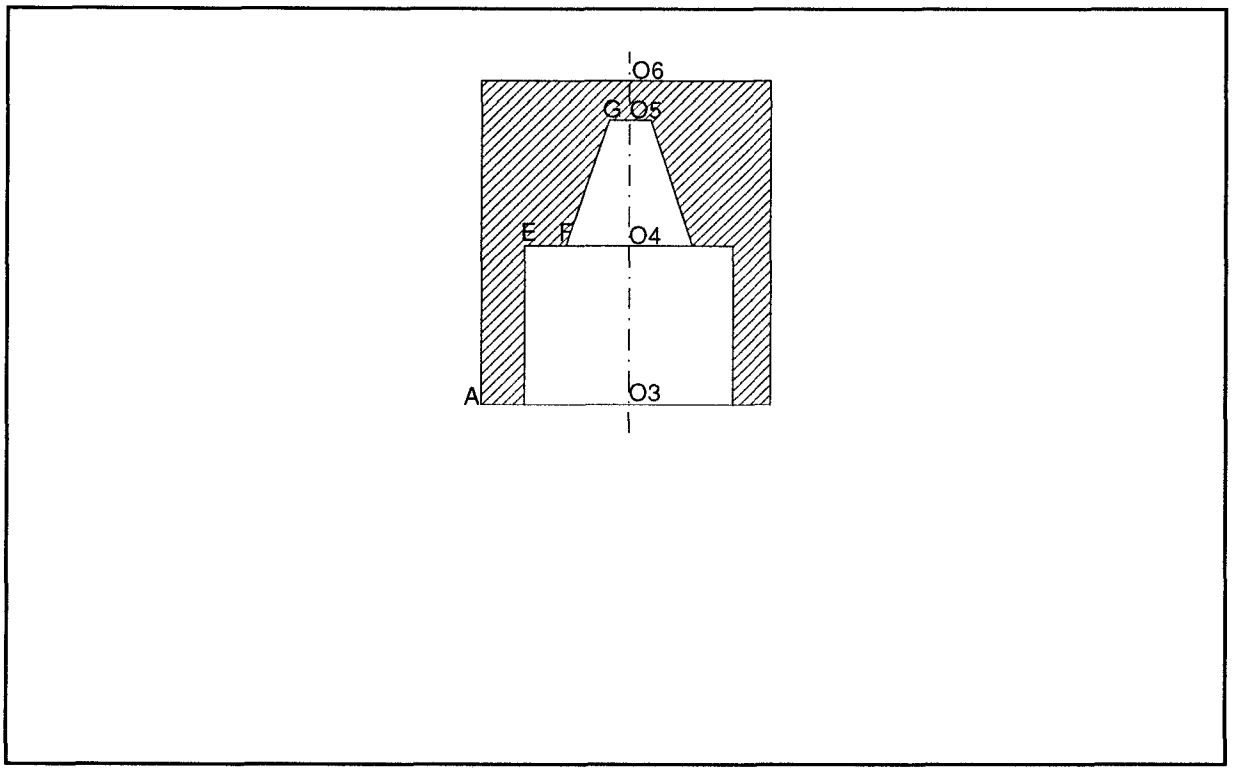


Fig. V.15 - CAPOT I : repérage des sommets

CAPOT TYPE II

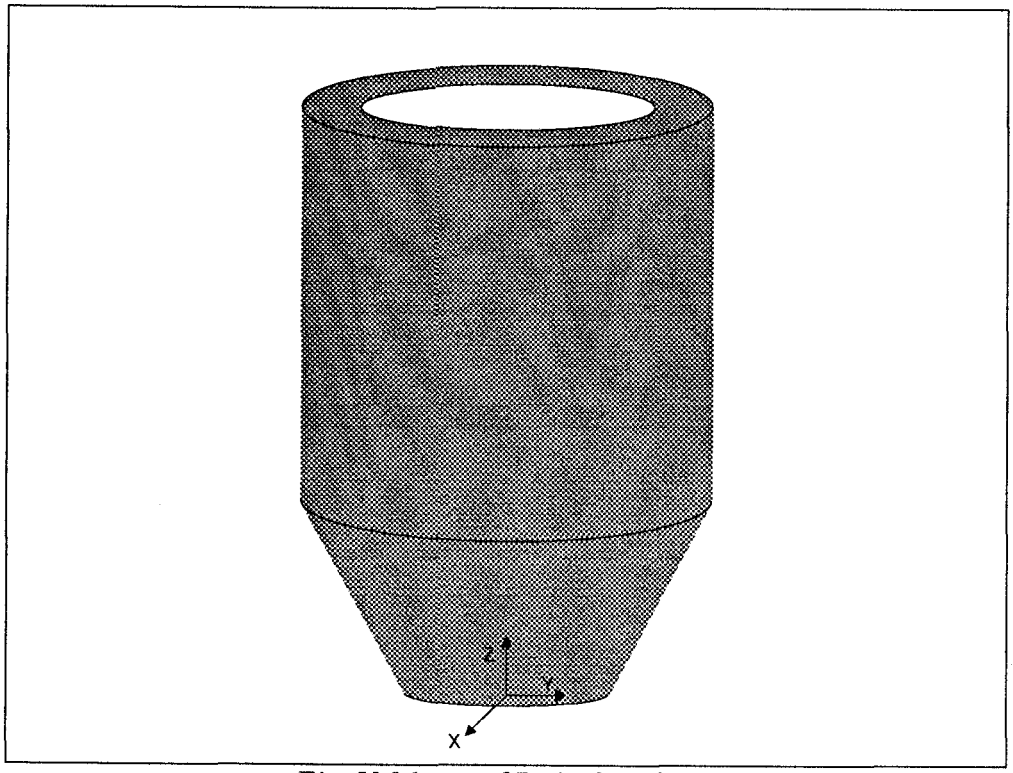


Fig. V.16 - vue 3D du CAPOT II

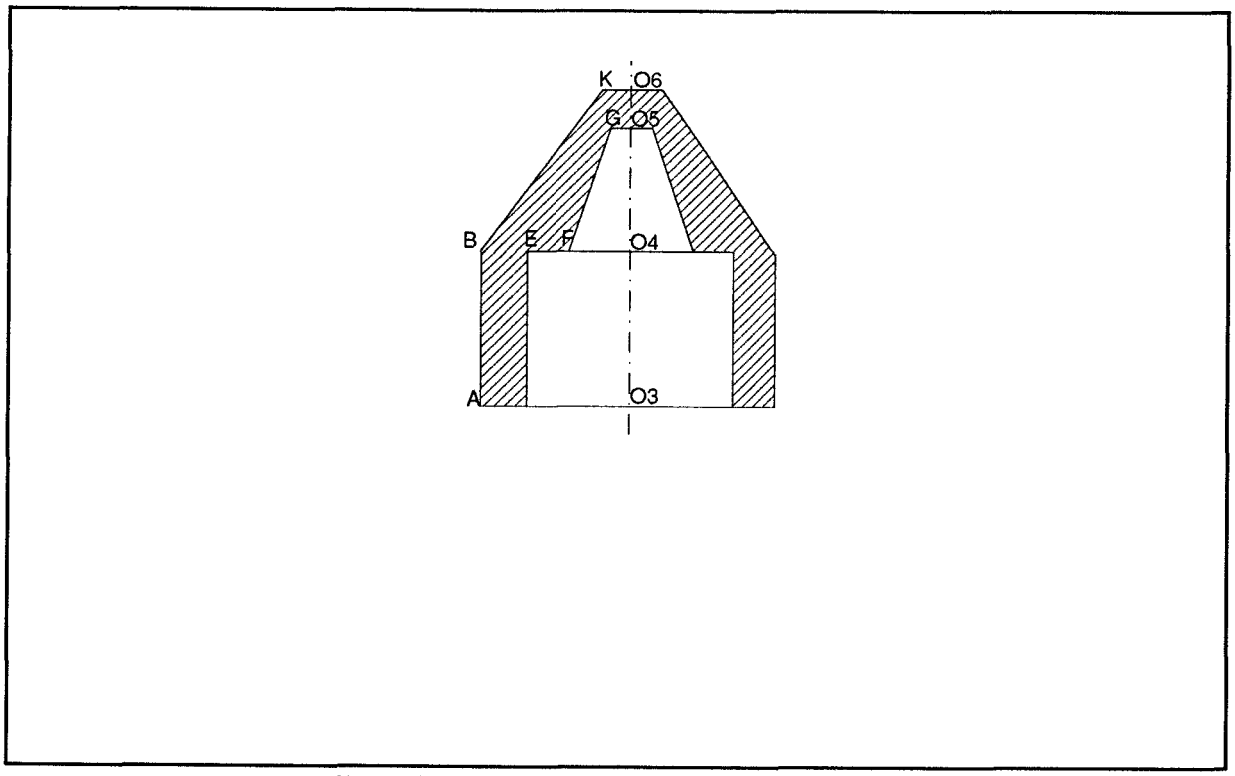


Fig. V.17 - CAPOT II : repérage des sommets

CAPOT TYPE III

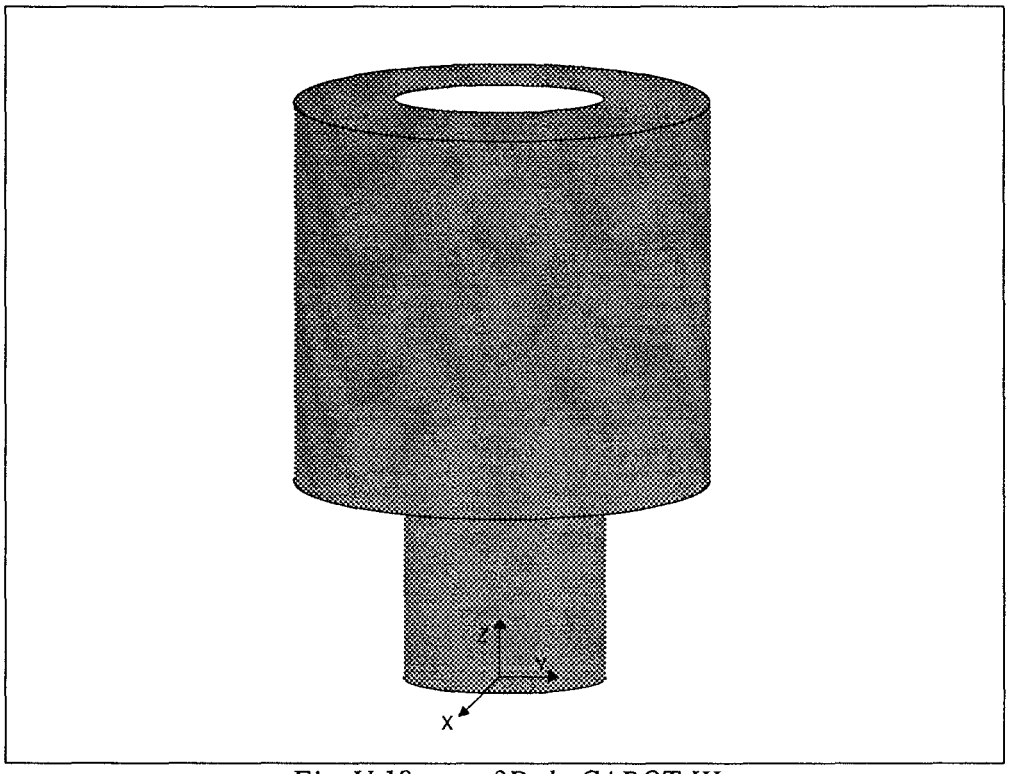


Fig. V.18 - vue 3D du CAPOT III

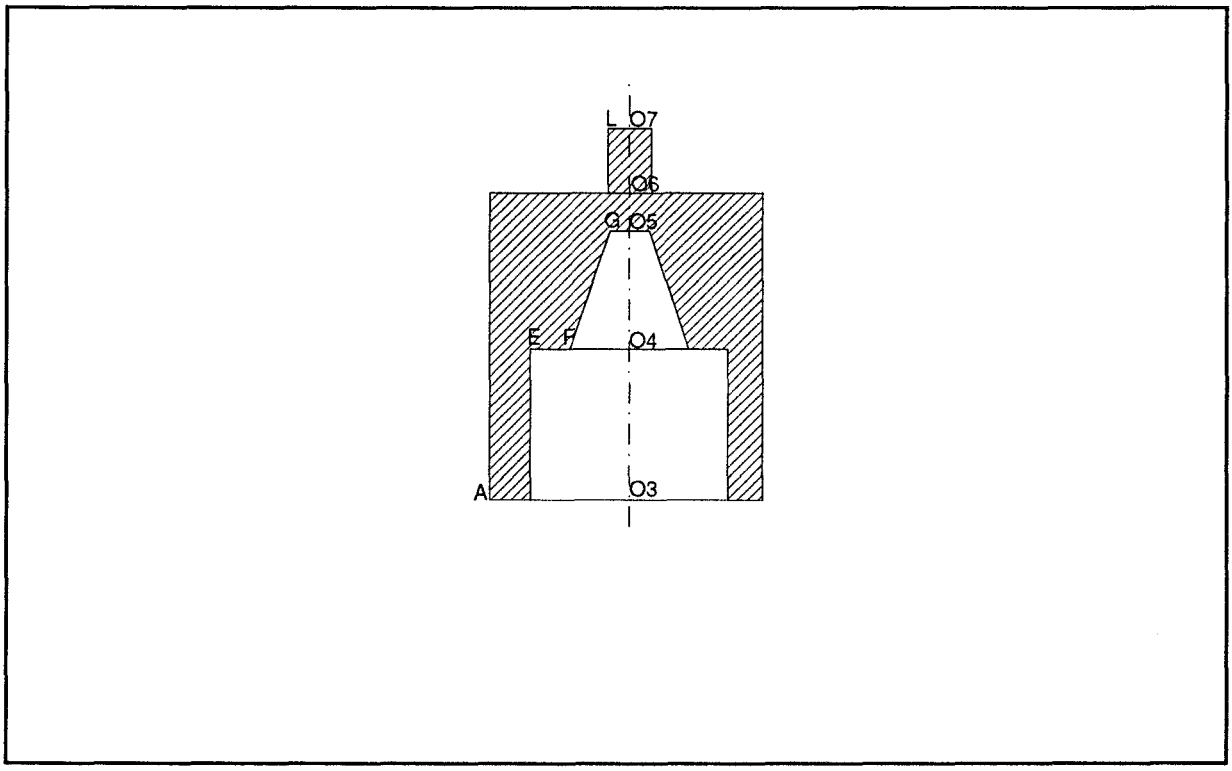


Fig. V.19 - CAPOT III : repérage des sommets

V.5 - Modélisation et Définition des Objets

Les instanciations des objets de la base de travail (pièces, formes, sommets, ...) sont présentées par les figures V.20, V.21, V.22, V.23 et V.24.

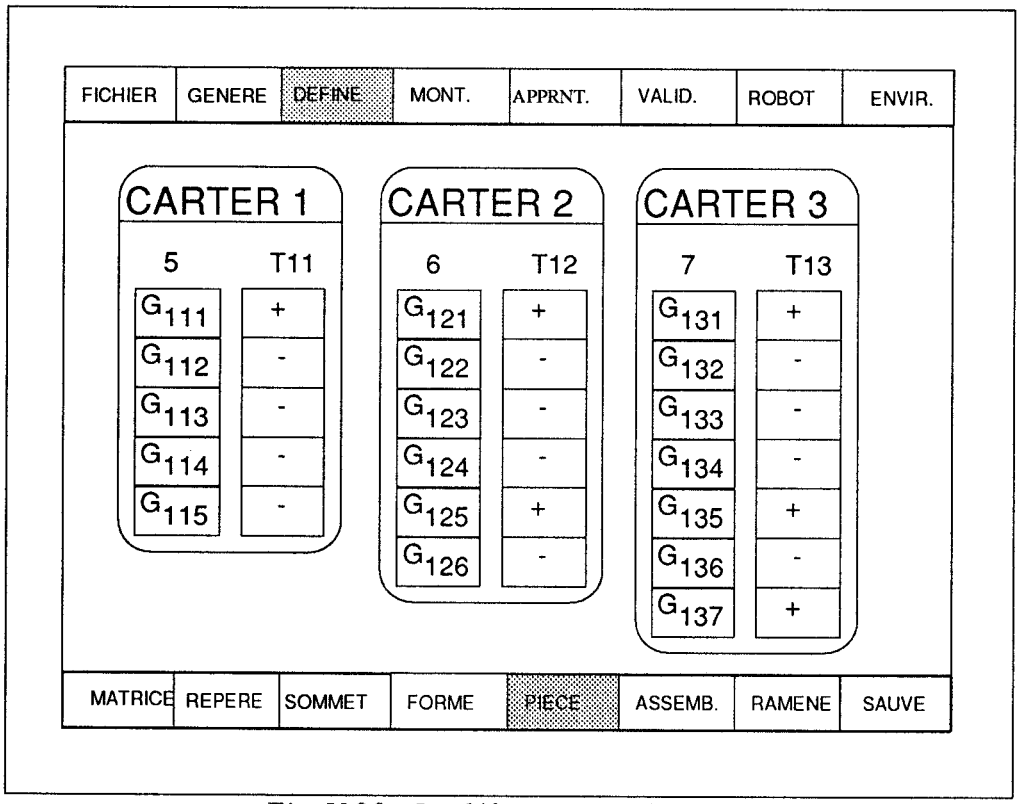


Fig. V.20 - La définition des CARTER

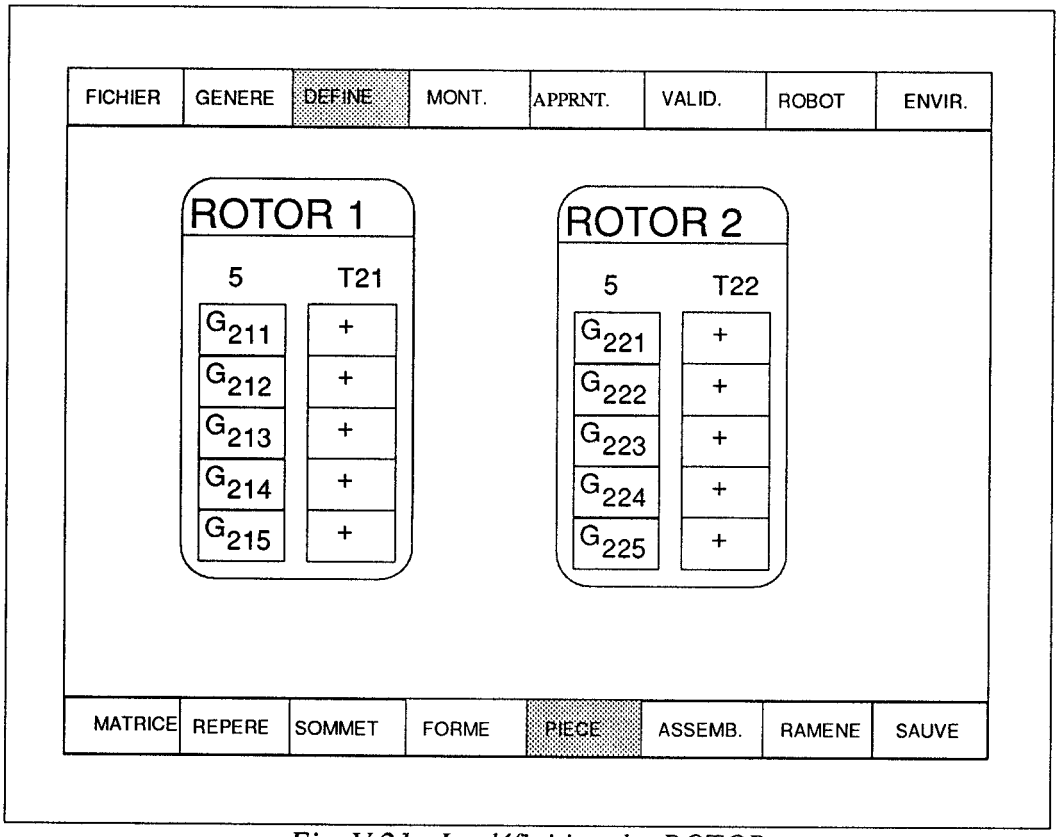


Fig. V.21 - La définition des ROTOR

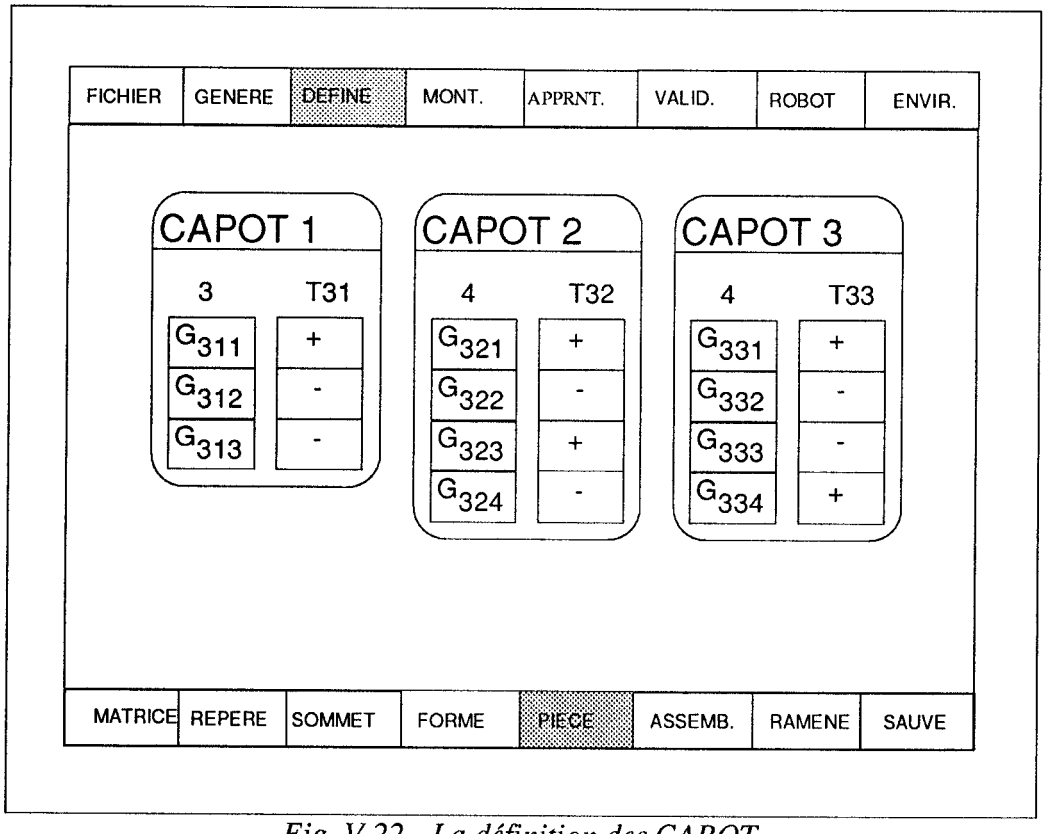


Fig. V.22 - La définition des CAPOT

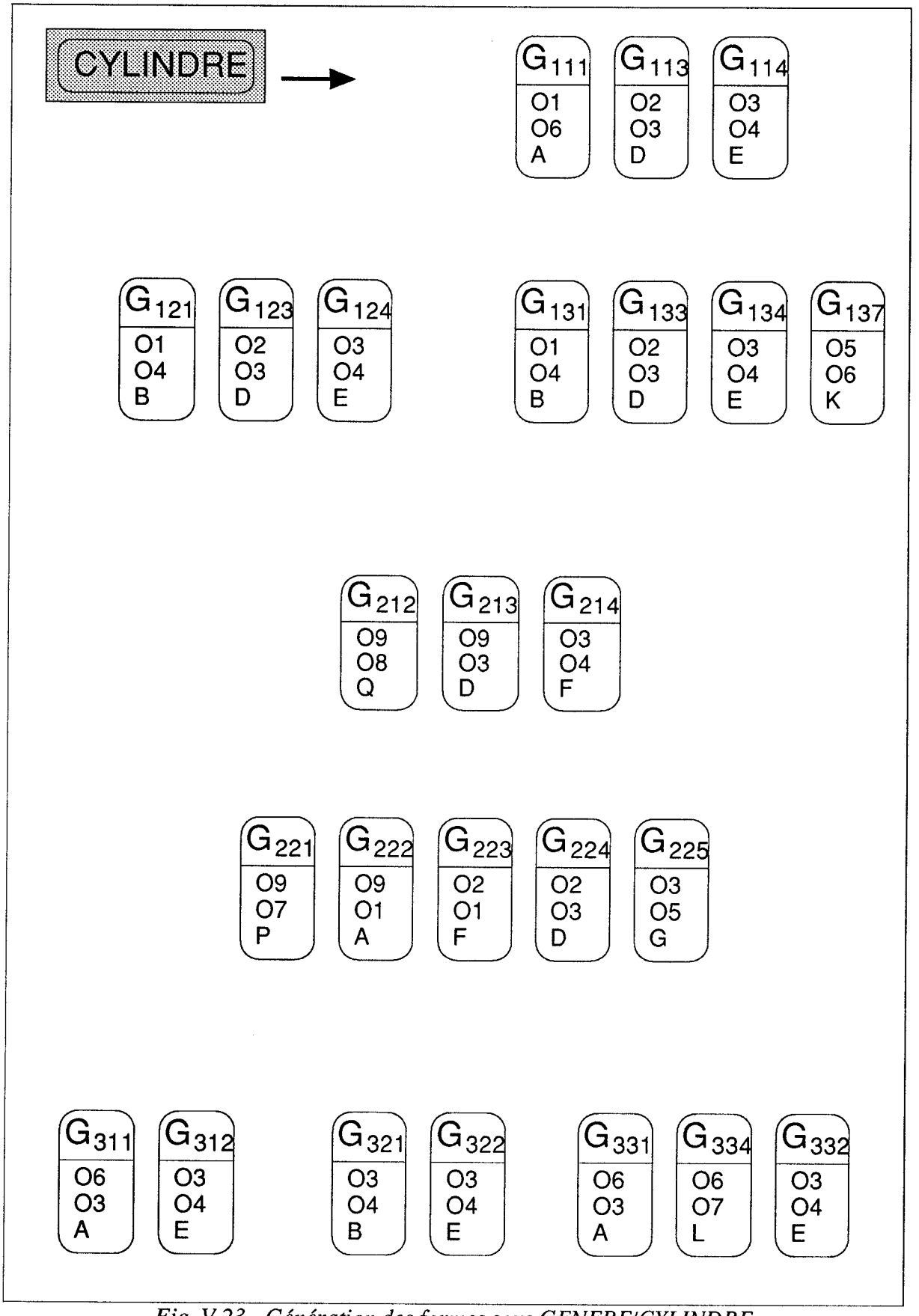


Fig. V.23 - Génération des formes sous GENERE/CYLINDRE

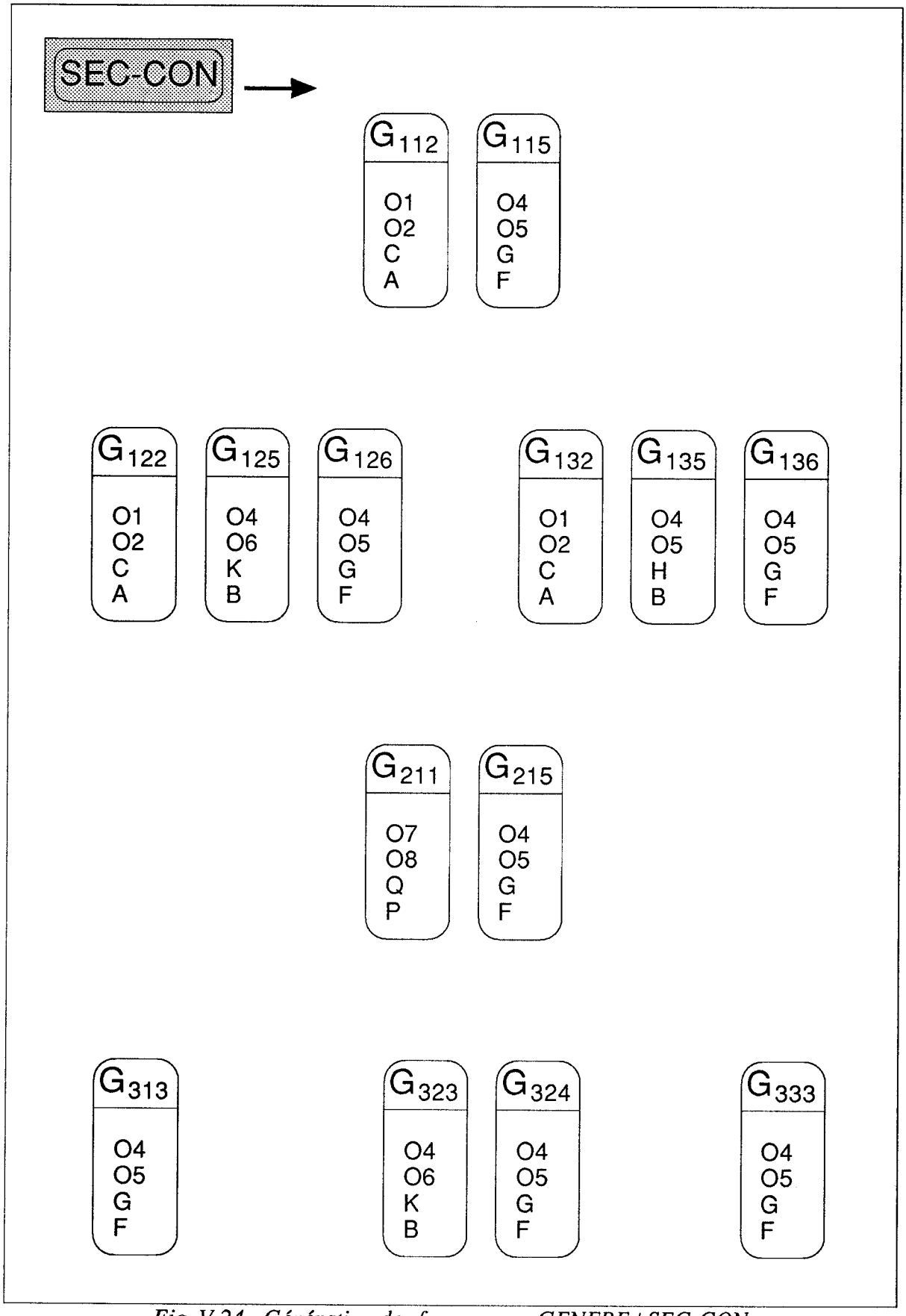


Fig. V.24 - Génération des formes sous GENERE / SEC-CON

V.6- Gamme D'Assemblage

A titre d'exemple, on se propose de réaliser l'assemblage d'un moteur, nommé MOTEUR-A, composé de CARTER 2, ROTOR 1 et CAPOT 3. L' écran de définition de cette gamme est présenté par la figure V.25 .

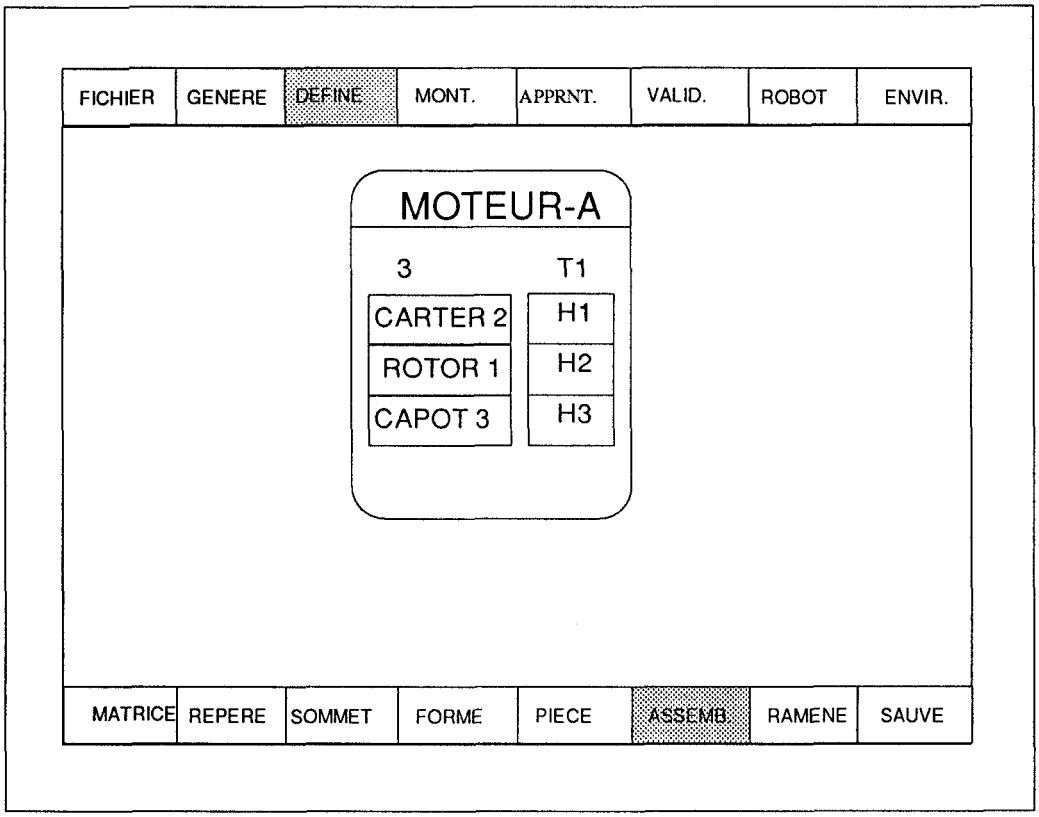


Fig. V.25 - Définition de la gamme d'assemblage

V.7- Montage des Pièces

V.7.1- Les Ecrans de Montage

Montage de la première pièce (CARTER 2):

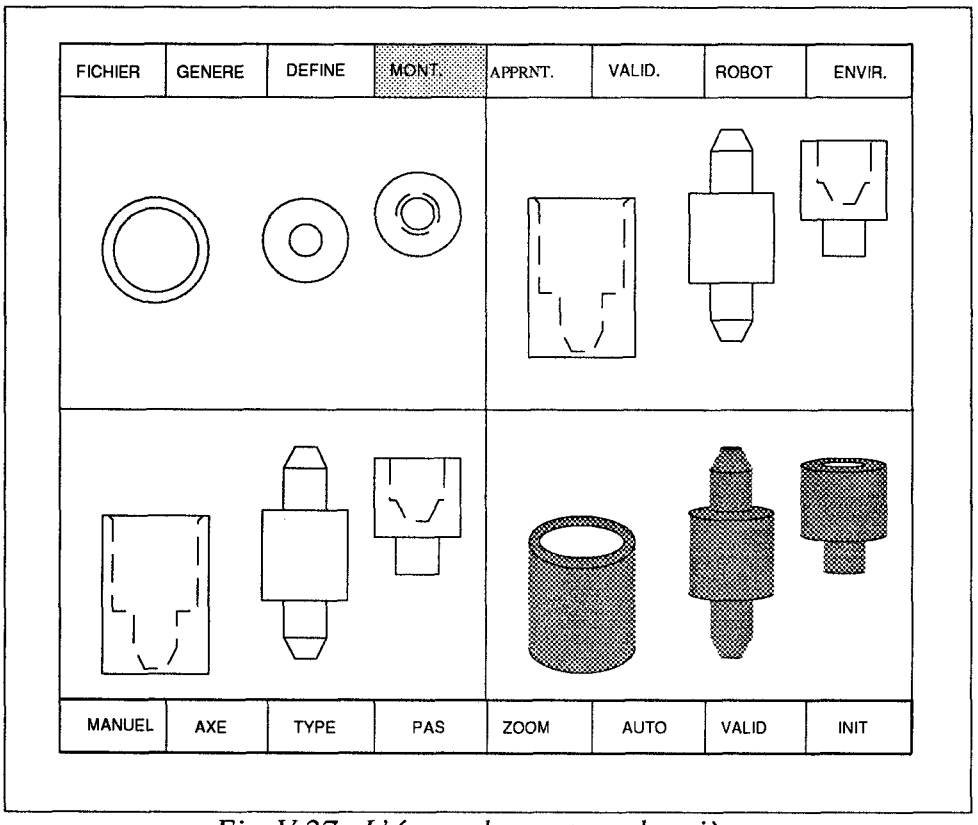


Fig. V.27 - L'écran de montage des pièces

Montage de la deuxième pièce (ROTOR 1):

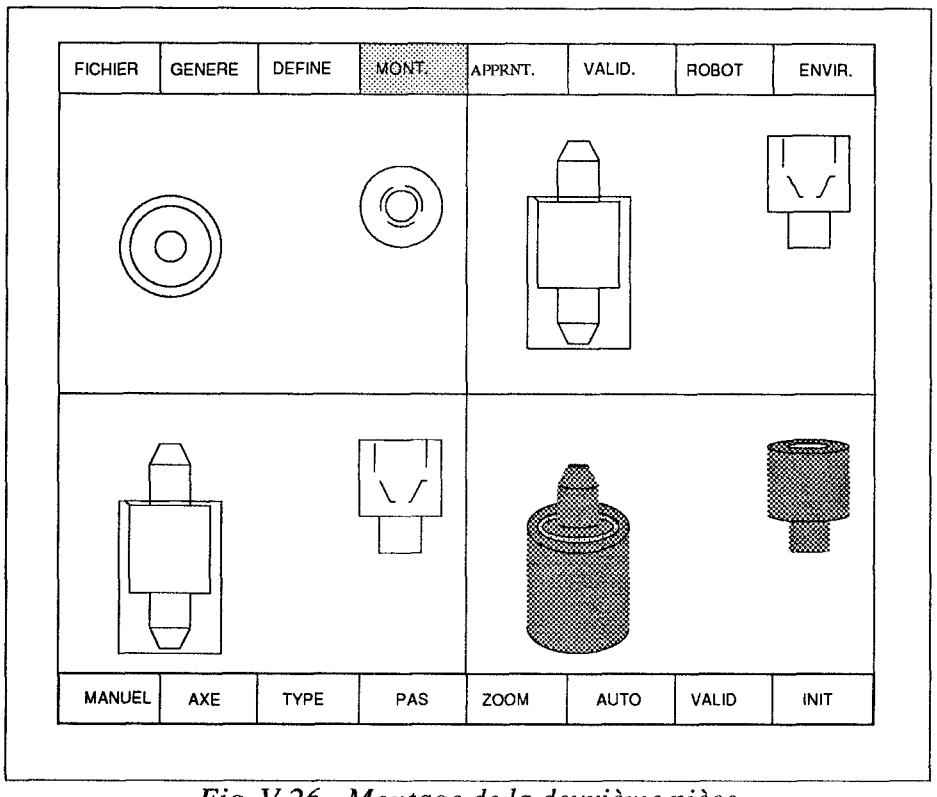


Fig. V.26 - Montage de la deuxième pièce

Montage de la troisième pièce (CAPOT 3):

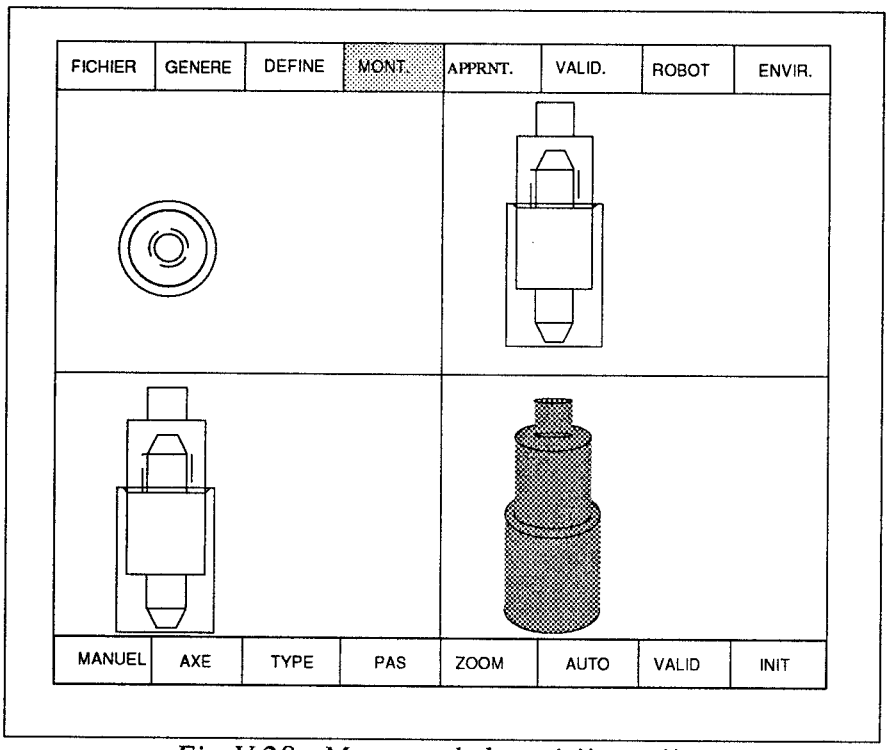


Fig. V.28 - Montage de la troisième pièce

V.7.2- Exemple de Réduction

Les figures V.29 et V.30 présentent un exemple de réduction de structures pour le sous assemblage A2.

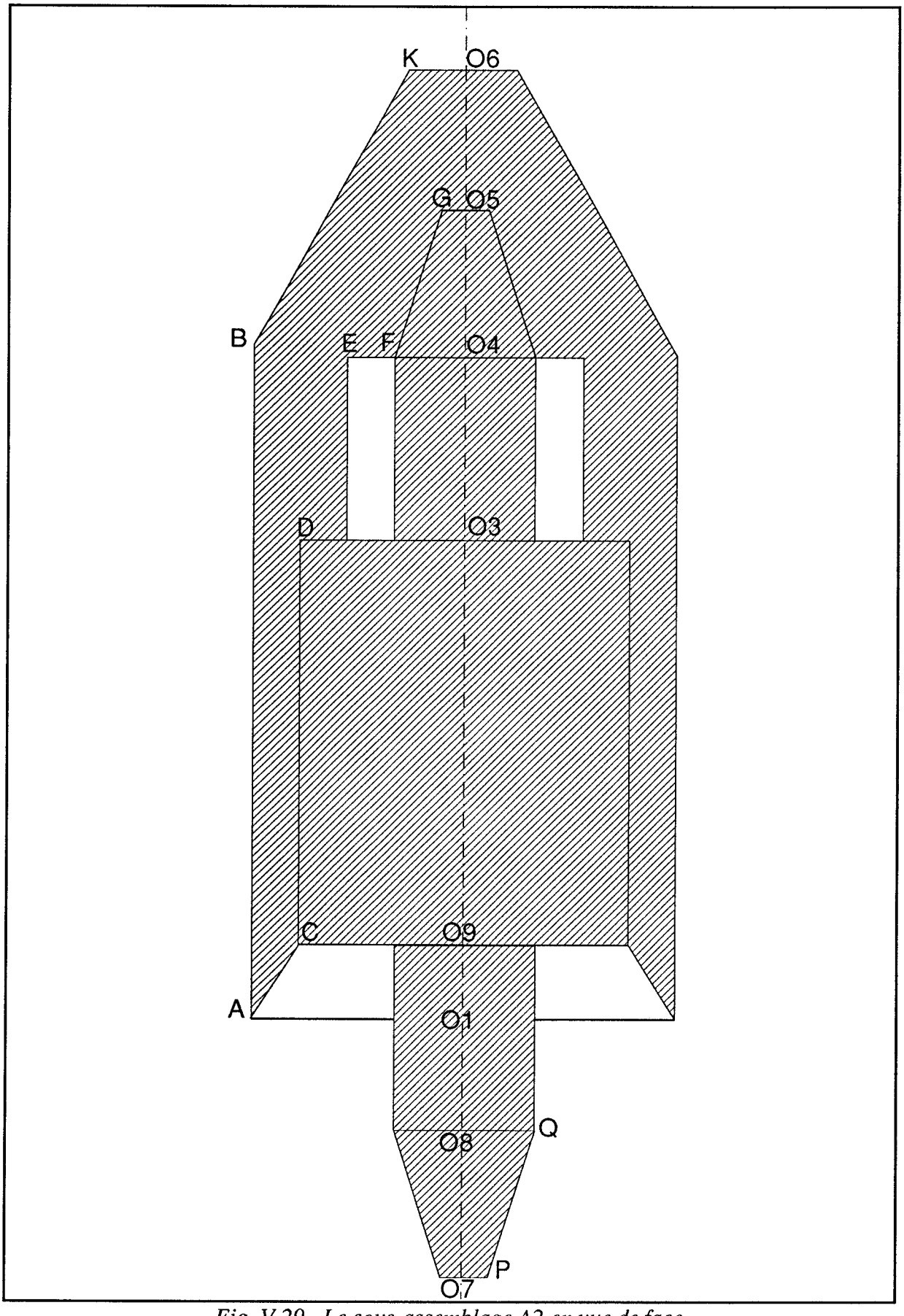


Fig. V.29 - Le sous-assemblage A2 en vue de face

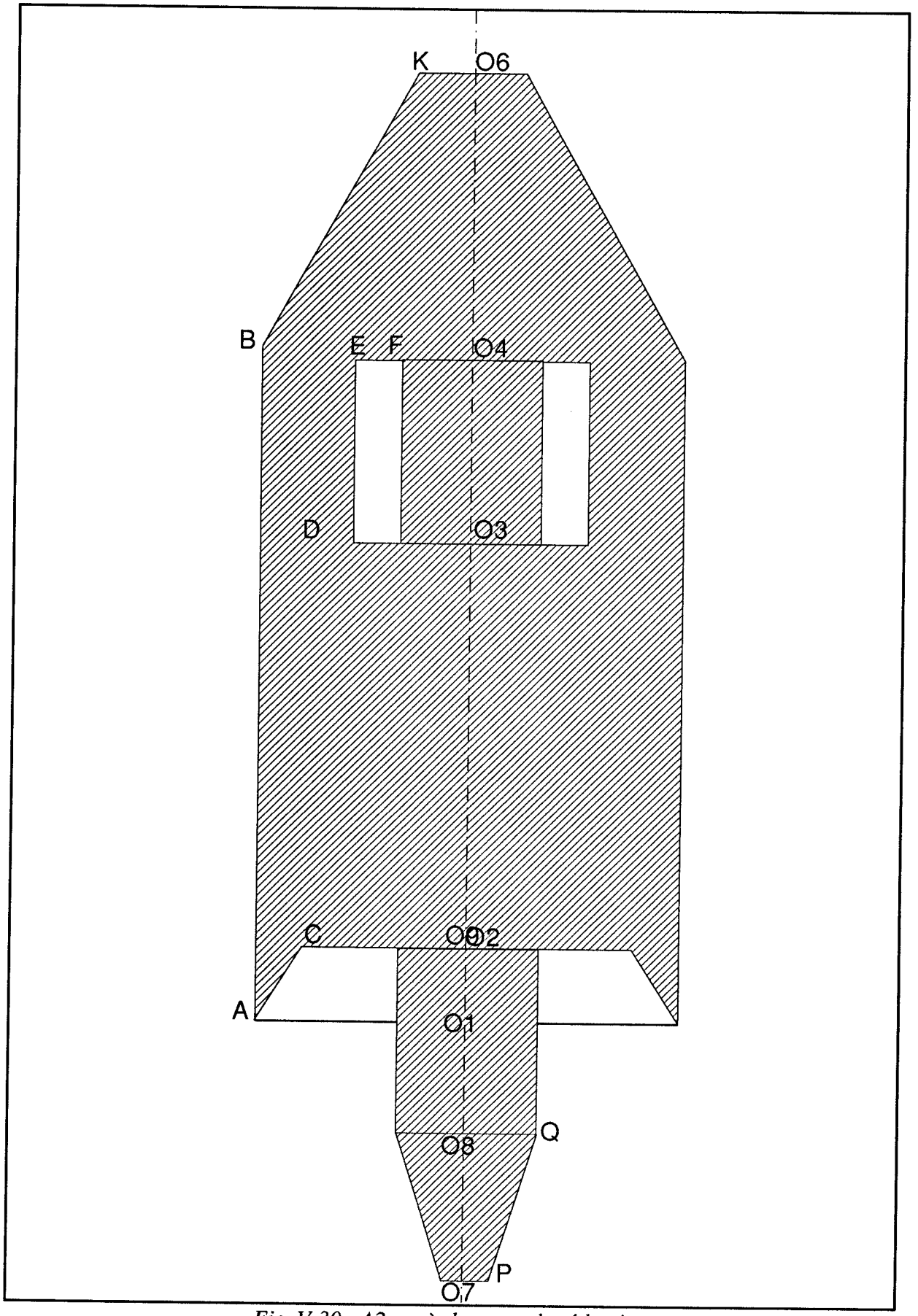


Fig. V.30 - A2 après les tests de réduction

V.8 - Apprentissage Graphique

Les écrans de la phase d'apprentissage graphique sont les suivants:

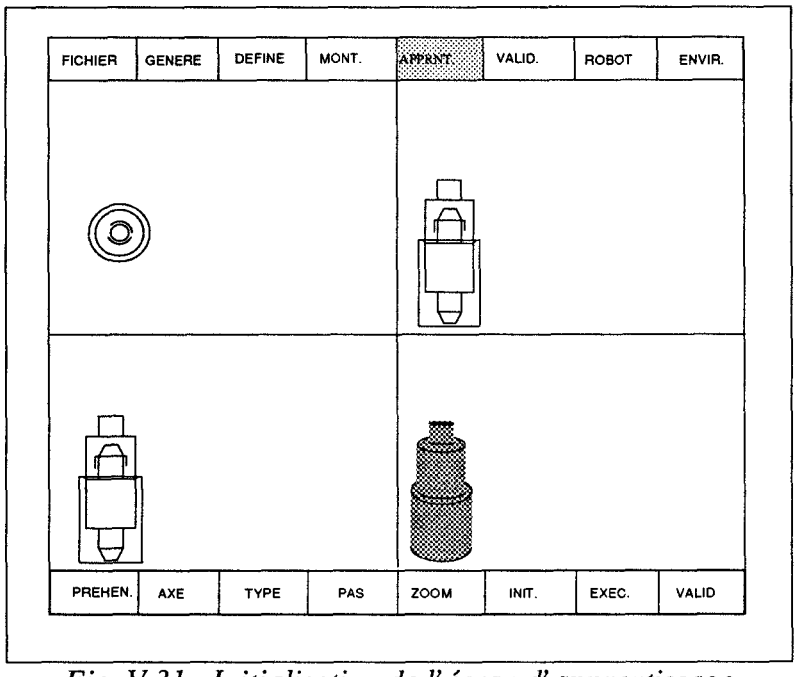


Fig. V.31 - Initialisation de l'écran d'apprentissage

* Démontage du CAPOT:

- 1er déplacement:

AXE = Z
TYPE = Translation
PAS = +5
EXEC.

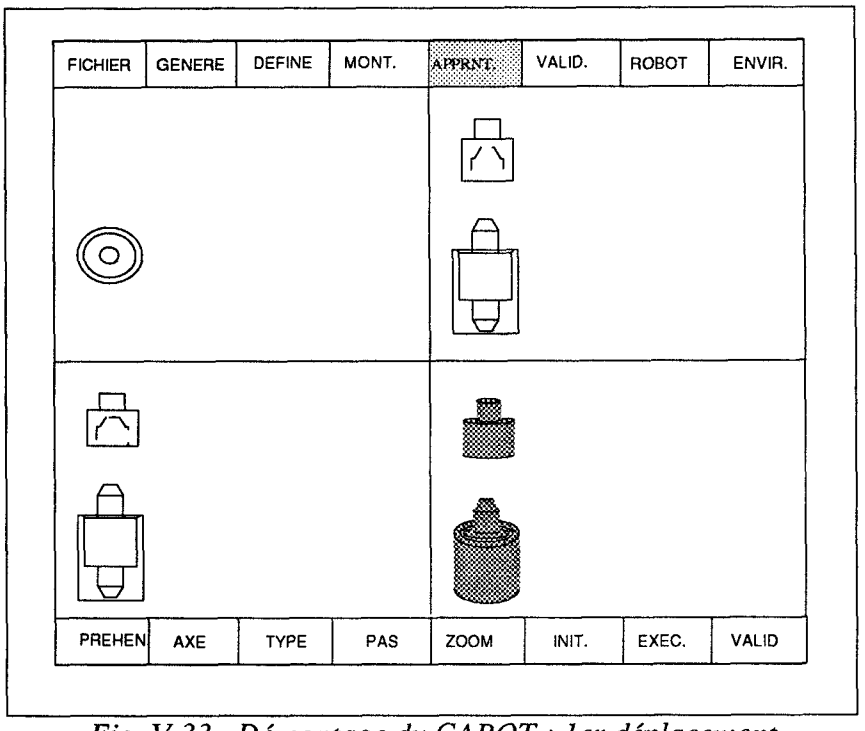


Fig. V.33 - Démontage du CAPOT : 1er déplacement

- 2ème déplacement:

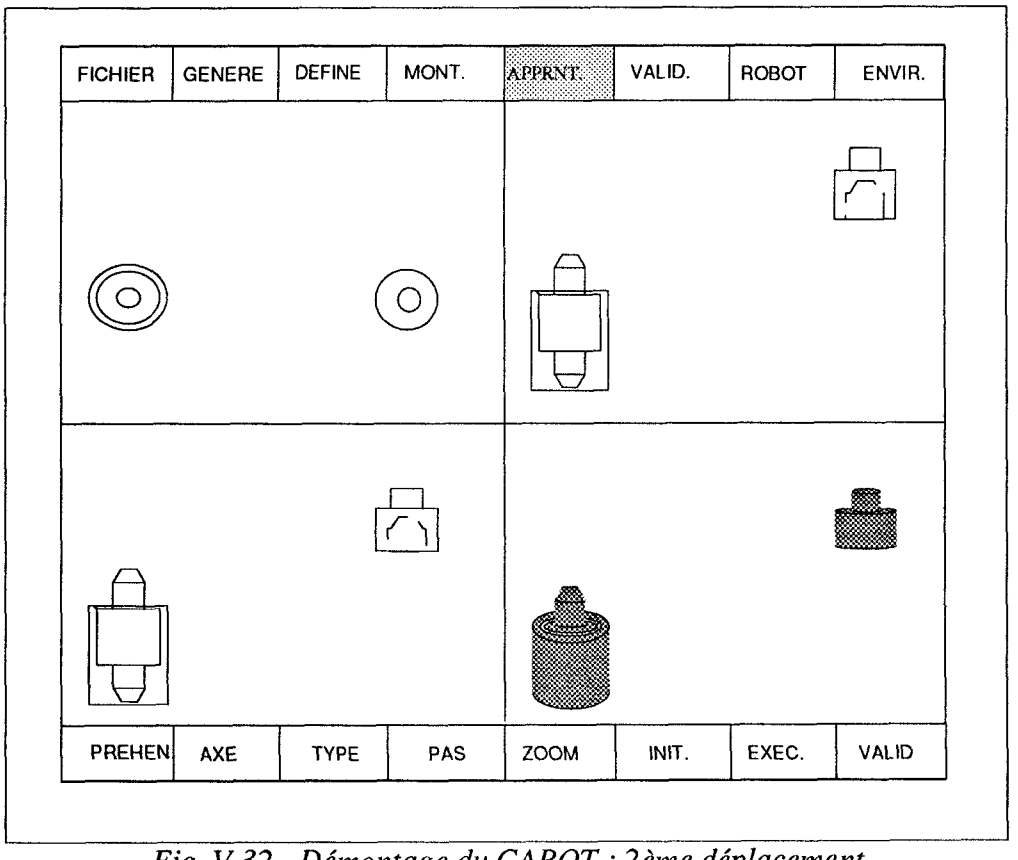


Fig. V.32 - Démontage du CAPOT : 2ème déplacement

AXE = Y

TYPE = Translation

PAS = +8

EXEC.

- 3ème déplacement:

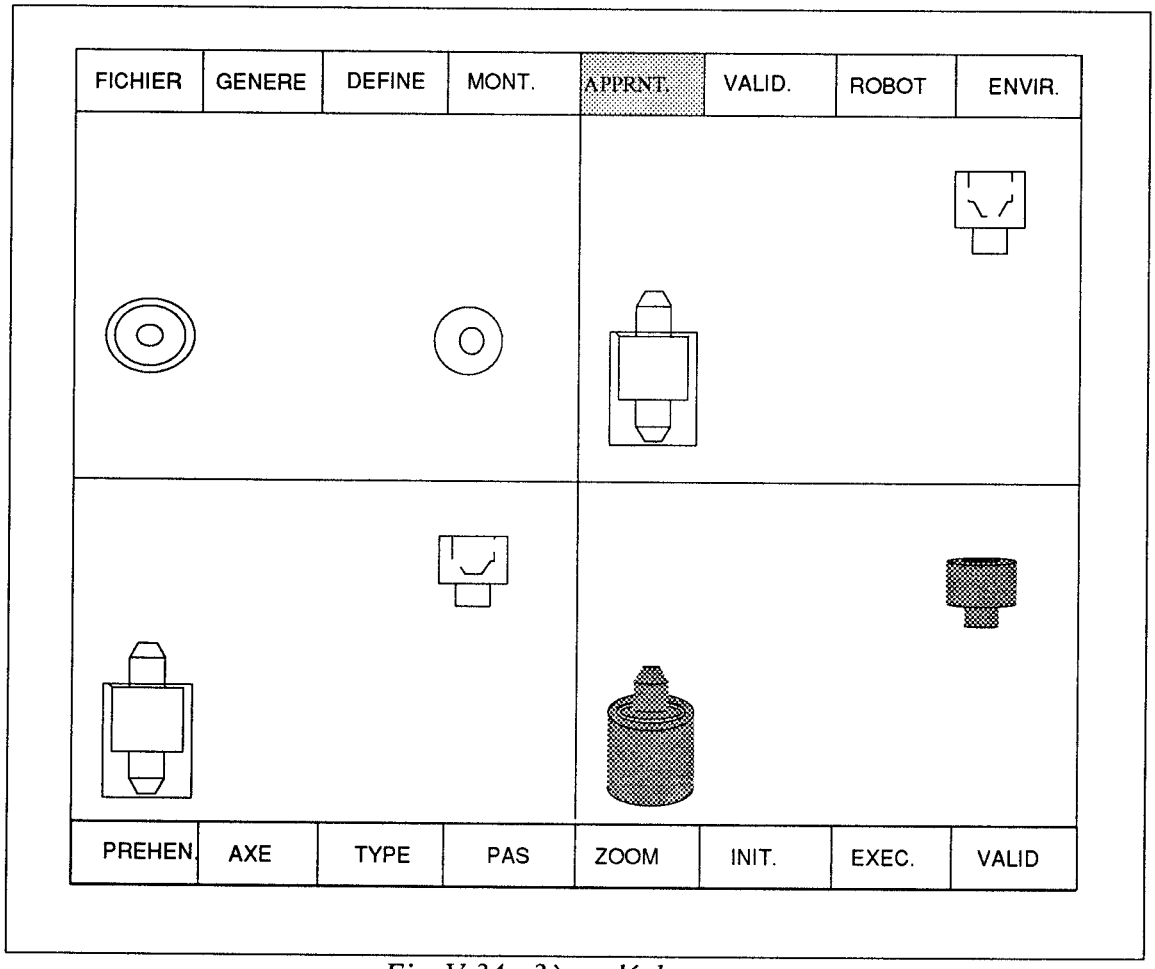


Fig. V.34 - 3ème déplacement

AXE = X

TYPE = Rotation

PAS = +180

EXEC.

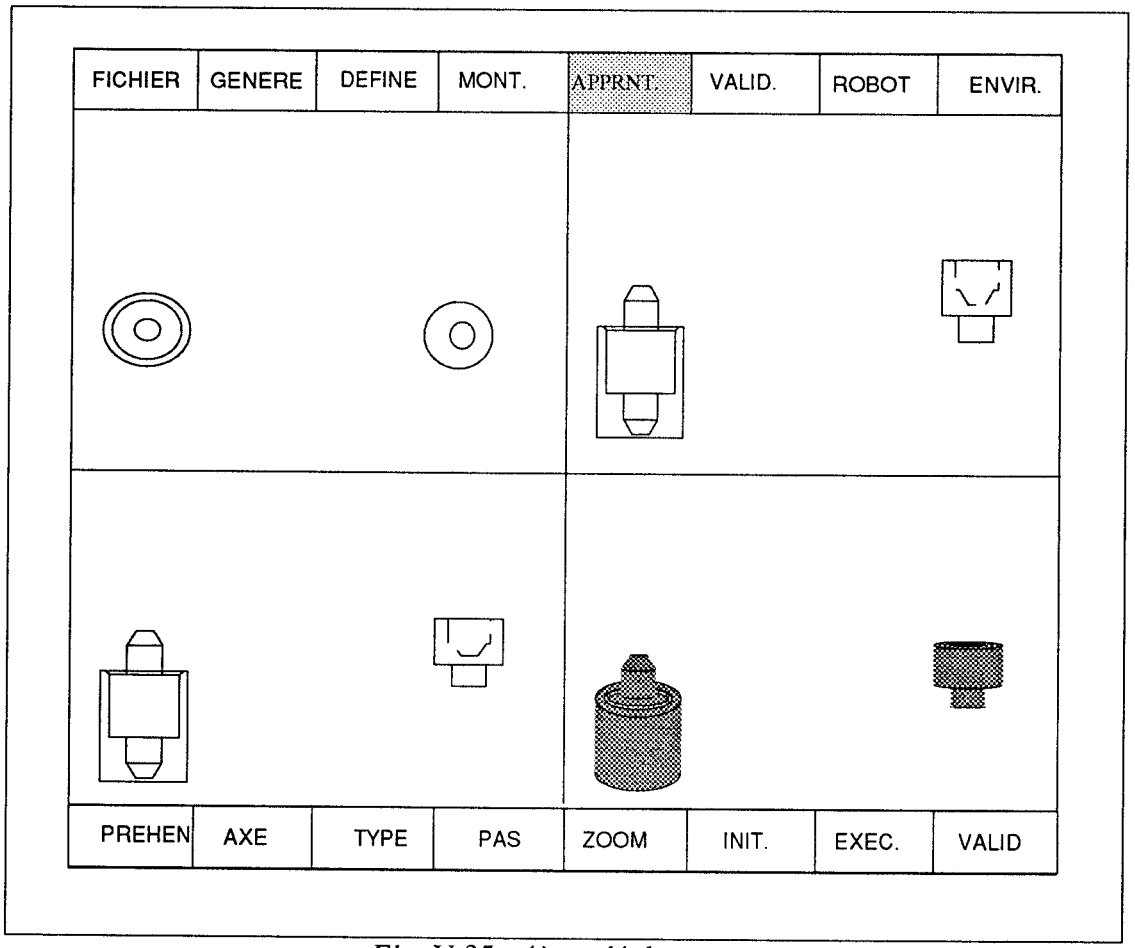


Fig. V.35 - 4ème déplacement

- 4ème déplacement:

AXE = Z

TYPE = Translation

PAS = -7

EXEC.

Les démontages du ROTOR et du CARTER se déroulent de la même facon. L'écran du démontage total est présenté par la figure V.36.

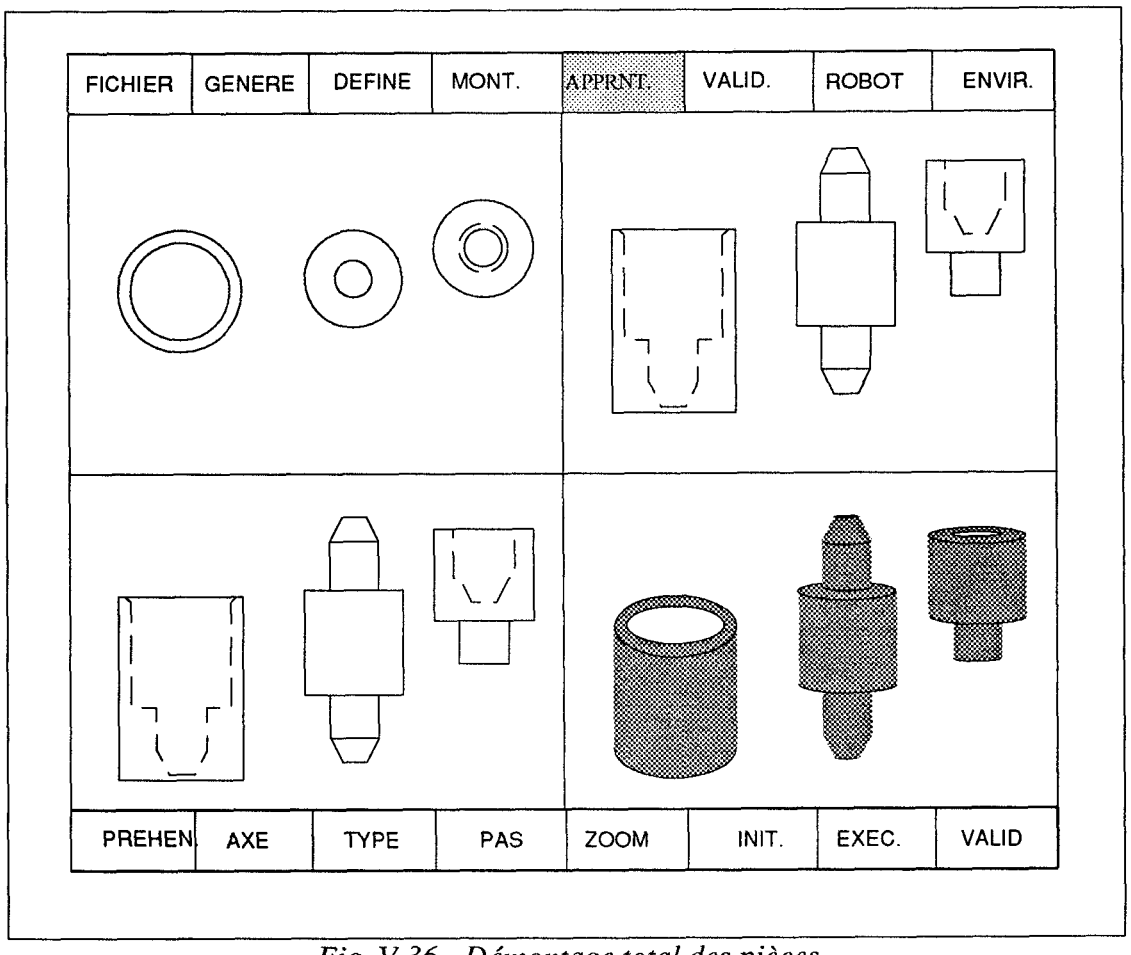


Fig. V.36 - Démontage total des pièces

V.9 - Conclusion

On a présenté dans ce chapitre une application de la méthodologie d'apprentissage graphique. Cette méthodologie est appliquée au processus d'assemblage d'une gamme de moteurs électriques.

On a décrit la cellule d'assemblage installée au Centre d'Automatique de l'Université de Lille. Ensuite, on a modélisé la base de travail et on a procédé au montage automatique des composants d'une gamme de moteurs, appelée MOTEUR-A. Finalement, l'apprentissage des opérations s'effectue par démontage interactif des pièces.

Cette méthodologie d'apprentissage est retenue pour le projet de développement d'une cellule flexible d'assemblage robotisé. Ce projet se déroule actuellement au Centre d'Automatique de Lille. L'apprentissage graphique des tâches d'assemblage sera utilisé lors de l'installation définitive de cette cellule.

CONCLUSION GENERALE

L'introduction des robots sur les processus de production industriels augmente la flexibilité et mène à une nouvelle génération d'usines. On a proposé dans ce travail, une méthodologie de programmation graphique interactive des robots d'assemblage.

Après une présentation de l'état de l'art de la robotique d'assemblage, on a abordé la programmation des robots d'assemblage. Une méthodologie de programmation des robots d'assemblage par l'utilisation de la simulation graphique a été présentée. Cette méthode nous a amené à formuler une modélisation informatique tridimensionnelle du processus d'assemblage et une représentation graphique de la base de travail. Le bon fonctionnement de la méthode a nécessité également l'élaboration d'un interface homme - machine approprié.

Les caractéristiques de la méthode sont:

- séparation entre la modélisation mathématique du processus et sa représentation graphique
- modélisation tridimensionnelle des produits et de la base de travail
- modélisation des tâches d'assemblage par simples équations et matrices mathématiques
- possibilité d'intégrer des paramètres non géométriques dans la modélisation des objets
- introduction de la notion de matière négative pour pouvoir modéliser l'absence de matières, les cavités, les trous, ...
- manipulation structurée de l'univers par l'utilisation d'une approche "objets"
- représentation graphique simplifiée et possibilité d'intégrer des systèmes de CAO
- souplesse et ergonomie au niveau de l'interface homme-machine
- simplicité de réalisation: implantation sur micro-ordinateur
- intégration de la méthode au niveau du système de CFAO de l'usine
- liberté de choix pour l'acquisition des données

La modélisation tridimensionnelle est la contribution majeure du présent travail à la programmation graphique des robots. Dans ce sens, on vient compléter les travaux sur

l'assemblage déjà menés au Laboratoire d'Automatique de Lille (BIENFAIT dans [BIE 87], DARRAS dans [DAR 88], ...).

Le choix actuel de représentation graphique bidimensionnelle en trois vues a été retenu pour causes de simplicité et d'économie. Cependant, le développement rapide des systèmes de CAO et la disponibilité sur le marché, à des prix compétitifs, des logiciels tridimensionnels de simulation graphique, pourraient altérer ce choix. Une éventuelle amélioration consisterait à intégrer des systèmes entièrement tridimensionnels pour la représentation graphique de la base de travail.

D'autre part, le choix du mode d'acquisition des données reste libre. On pourra envisager d'intégrer un système d'acquisition automatique par vision artificielle et traitement d'images. Une autre alternative consiste à acquérir les données directement à partir du système de CFAO de l'usine. L'assemblage d'une pièce sera pris en compte et programmé dès la phase de conception matérielle de cette pièce.

Il est à noter finalement que ce travail se situe dans le cadre d'un projet global de conception et de réalisation d'une cellule flexible d'assemblage robotisé. Un tel projet doit prévoir de gérer la conformité entre les différents modules qui sont mis en jeu: gamme d'assemblage, programmation des tâches, contrôle du process (par vision et autres), CFAO, supervision, ... Les composantes matérielles de la cellule étant déjà installées au Centre d'Automatique de l'Université des Sciences et Techniques de Lille, les travaux de recherche s'orientent actuellement suivant trois axes:

- commande des robots, dans ce sens on note les résultats de V. KONCAR sur la commande par suréchantillage dans [KON 90] et [KON 91].
- intégration des systèmes et contrôle réparti, on retrouve les travaux de M. SOUAM, D. FOURMAUX et M.COUVREUR [COU 89], [FOU 91].
- systèmes de programmation, dans lequel le présent travail se situe.

Une synthèse de tous les travaux menés dans ce domaine parait donc nécessaire...

BIBLIOGRAPHIE

[AND 83] J.R. ANDERSON
The Architecture of Cognition
Harvard University Press, Massachussetts, U.S.A., 1983

[ANS 85] S. ANSALDI, B. DE FLORIANI, B. FALCIDIENO
Geometric modeling of solid objects by using a face adjacent graph representation
Computer Graphics, july 1985, pp 131 - 140

[BAE 79] A. BAER, C. EASTMAN, M. HENRION
Geometric modeling : a survey
CAD, vol. 11, no 5, pp. 253 - 273, sep. 1979

[BAY 88] M. BAYART
Conception d'un système de pilotage temps réel pour une cellule d'assemblage automatisé
Thèse de Doctorat, Lille, 1988

[BER 88] G. BERRY, G. GONTHIER
The ESTEREL synchronous programming language: design, semantics and implementation
Technical Report, n 842, INRIA, 1988

[BHP 87] C. BRAESCH, A. HAURAT, J.L. PERRARD, C. RENAUD
Outils et concepts du système LMAC multi-tâche
Congrès GAMI, ENS Cachan, jan. 1987

[BIE 87] E. BIENFAIT
Programmation de Robot Assisté par Traitement d'Image et Camera
Thèse de Doctorat, Lille, 1987

[BON 82] S. BONNER, K. G. SHIN
A comparative study of robot languages
IEEE Computer pp. 82 - 96, dec. 1982

[BOU 84] A. BOURJAULT
Contribution à une approche méthodologique de l'assemblage automatisé : élaboration automatique des séquences opératoires
Thèse de Doctorat, Besançon, 1984

[BOU 84-b] J.P. BOURRIERES
Intrinsic compliance of position-controlled robot. Application in assembly
Proc. of the 5th Int. Conf. on Assembly Automation, Paris, 1984

[BOU 86] A. BOURJAULT, LHATTE
Modélisation d'un processus d'assemblage
A.P.I.I., vol. 20, numero 2, 1986

[BOU 87] A. BOURJAULT
Détermination des sous assemblage d'un produit, à partir des séquences temporelles d'assemblage
A.P.I.I., vol. 21, numero 2, 1987

[BRA 78] I. BRAID
New directions in geometric modeling
CAM workshop on geometric modeling, Arlington - Texas, 1979

[BRA 82] M. BRADY, J.M. HOLLERBACH, ..
Robot Motion: Planning and Control
MIT Press, ENGLAND, 1982

[BRA 85] M. BRADY
Artificial Intelligence and Robotics
Artificial Intelligence, n 26, 1985, pp. 79 - 121

[BRO 83] R.A. BROOK
Planning collision free motions for pick and place operations
1st Int. Symposium on Robotic Research, Bretton Woods, Aug. 1983

[BRU 81] H. VAN BRUSSEL, H. THILEMANS, J. SIMONS
Further developments of the active adaptable compliant wrist for robot assembly
Proc. 11th Int. Symposium on Industrial Robot, Tokyo, oct. 1981

[BRU 87] H. VAN BRUSSEL, L. VAN AKEN, J. DE SHUTTER
LOLA, an advanced end-point off-line robot programming system
Annals of the CIRP, vol. 36/1, 1987

[BUR 86] BURNS, WORTHINGTON
Practical Robotics, 1984

[CAL 79] H. MAC CALLION, G.R. JOHNSON, D.T. PHAM
A compliant device for inserting a peg in a hole
The Industrial Robot, vol. 6, n 2, june 1979, pp. 81 - 87

[CAS 86] J. P. CASSAR
Simulateur graphique pour la programmation de cellule de soudage robotisé : l'apprentissage hors ligne
5th European Conference on CAD/CAM and Computer Graphics, 1986

[CAS 87] J. P. CASSAR
Poste de programmation graphique de cellule de soudage robotisée
Thèse de Doctorat, Lille, 1987

[CIM 89] CIMSTATION: overview
Documentation SILMA, juin 1989

[COU 87] J. COUTAZ
PAC, an implementation model for dialog design
Interact'87, Sttutgart - RFA, sep. 1987, pp. 431 - 436

[COU 88] J. COUTAZ
Interface homme - ordinateur: conception et réalisation
Thèse de Doctorat d'Etat, Grenoble, 1988

[COU 89] M. COUVREUR, D. FOURMAUX, ..
Partition d'un graphe de commande global et implantation répartie
39th International Mini and Micro Computers, Zurich, 1989

[COU 90] J. COUTAZ
Architectures et outils pour la programmation des systèmes interactifs
Bulletin de l'INRIA, n 127, mai 1990

[CRI 85] CRITCHLOW
Introduction to Robotics
California State University, Macmillan, 1985

[DAN 88] W. DANG
Formal specification of interactive languages using Definite Clause Grammar
Réunion internationale sur la programmation logique, Orléans, mars 1988

[DAR 88] J.P. DARRAS
Programmation graphique des Robots
Rapport de DEA, USTLFA, Lille, 1988

[DCC 88] G. DEJONGHE, D. CHAIGNE, A. COSSIC
ARES/ projet de développement
Rapport CEA - DEMT/88/342, 7 oct. 1988

[DIL 86] R. DILLMAN, B. HORNUNG, M. HUCK
Interactive programming of robots using textual programming and simulation techniques
Brussel, 1986

[DIL 86] R. DILLMAN, B. HORNUNG, M. HUCK
Interactive programming of robots using textual programming and simulation techniques
Brussel, 1986

[DIL 88] R. DILLMAN
Mobile robots in industrial environment
Acts of 19th ISIR, Sidney, avril 1988, pp. 79 - 89

[DIL 89] R. DILLMAN
A programming system for flexible robot assembly cells
Acts of 20th ISIR, Tokyo, oct. 1989, pp. 793 - 802

[DOM 81] E. DOMBRE
Analyse des performances des robots manipulateurs flexibles et redondants - contribution à leur modélisation et à leur commande
Thèse de Doctorat d'Etat, Montpellier, 1981

[DON 86] M. R. DONY
Graphisme Scientifique sur Micro-ordinateur
Masson, 1986

[DWI 85] DWIVEDI, SATYANARAYANA
Practical problems in the application of industrial robots
American Society for Testing and Materials, 1985

[ELM 87] H. ELMARAGHY
Artificial Intelligence and Robotic Assembly
Enginnering with computers 2, 1987, pp. 147 - 155

[EVA 77] J. M. EVANS, J. S. ALBUS, A. J. BARBERA
Robotics Research Workshop, Washington DC, 1977

[FAS 86] D. FASSOTTE
Conception des synoptiques sur CRT destinés au contrôle de processus: aspect de l'ergonomie en informatique
Edition de l'université de Bruxelles, BELGIQUE, 1986

[FOU 91] D. FOURMAUX
Contribution à la conception des systèmes de contrôle répartie
Thèse de Doctorat, Lille, 1991

[GAR 83] GARDAN, LUCAS
Techniques Interactives et CAO
Hermes, 1983

[GAR 85] GARDAN
Mathématiques et CAO, Méthodes de Base
Hermes, 1985

[GOL 84] A. GOLDBERG
Smalltalk-80: the interactive programming environment
Addison - Wesley, 1984

[GRO 84] GROOVER, ZIMMERS
CAD/CAM Computer Aided Design and Manufacturing
Prentice Hall, 1984

[GUE 86] R. A. GUEDJ
Methodology in Computer Graphics
Proceeding IFIP, workshop hell, Seillac - France, 1986

[GUI 83] J.C. GUINOT
Réalisation d'un préhenseur tridigital doté de capteurs d'efforts tactiles et mise en oeuvre de sa commande en boucle fermée
Rapport annuel ARA, Besançon, nov. 1983

[GUY 85] H. GUYENNET
LMAD-G : un logiciel d'apprentissage graphique pour la programmation des robots
Thèse Docteur Ingénieur, Université de Franche - Comté, 1985

[HAL 85] J.F. HALLEY
La fonction robotique dans CATIA
Congrès AFRI MICADO, jan. 1985, pp. 43 - 49

[HAU 84] A. HAURAT, M. C. THOMAS
Les logiciels en Robotique
Cours école, La Martinique, 1984

[HEN 85] L. HENNINGER
SIMIR-L : un interpréteur pour l'intégration en robotique
Thèse Docteur Ingénieur, Université de Paris Sud, 1985

[HOP 83] HOPGOOD, DUCE, GALLOP SUTCLIFFE
Introduction to the GKS
Academic Press, New York, 1983

[HUL 86] J.H. HULLOT
SOS Interface, un générateur d'interfaces homme - machine
AFCET Langages orientés objet, Bigre + Globule, jan. 1986, pp. 69 - 78

[HUR 89] W.D. HURLEY, J.L. SIBERT
Modeling user interface - application interaction
IEEE Software, jan. 1989,pp. 71 - 77

[IGR 89] IGRIG: an introduction
Document Technique DENEB - 1989

[IRI 86] J. IRIGOYEN
Commande en position et force d'un robot manipulateur d'assemblage
Thèse de Doctorat, Toulouse, 1986

[JAM 86] JAMES, GUPTON
Computer Controlled Industrial Machines, Processes and Robots
Prentice Hall, 1986

[JEA 85] P. JEANNIER
Caractéristiques Opératoires des Robots D'Assemblage
Thèse de Doctorat, Université de Franche - Comté, 1985

[KAR 87] S. KARSENTLY
Graffiti, un outil interactif et graphique pour la construction d'interfaces homme - machine adaptables
Technical Report, Thèse 3ème cycle, Orsay, 1987

[KAR 90] S. KARSENTLY
Les Spécifications d'Interfaces
Bulletin de l'INRIA, n 127, mai 1990

[KAS 81] M. KASAI, K. TAKEYASU, M. UNO, K. MURAOKA
Trainable assembly system with an active sensoru table processing 6 axes
Proc. 11th Symp. on Industrial Robots, Tokyo, 1981

[KEM 85] K. G. KEMPF
Manufacturing and Artificial Intelligence
Robotics 1, pp. 13 - 25, 1985

[KOL 89] C. KOLSKI
Contribution à l'ergonomie de conception des interfaces graphiques homme - machine dans les procédés industriels
Thèse de Doctorat, Valenciennes, 1989

[KON 90] V. KONCAR, C. VASSEUR
Commande numérique par suréchantillage, application à la réalisation des cartes d'axes
ISMM International Symposium Mini and Micro Computers, Lugano - SUISSE, juin 1990

[KON 91] V. KONCAR
Commande numérique par suréchantillage et coordination en série de sous systèmes
Thèse de Doctorat, Lille, 1991

[KOR 87] KORSAKOV, ZAMYATIN
Assembly Practice in Machine Building
MIR - MOSCOW
Traduit par KOLYKHMATOV et KOCHIN, 1987

[LAM 84] LAMMINEM, CORNILLI
Industrial Robots, 1984

[LAT 82] J. C. LATOMBE
Survey of advance general purpose software for robot manipulators
IMAG Grenoble, RR no. 330, 1982

[LAT 85] J. C. LATOMBE, C. LAUGIER
Système de programmation pour la robotique
4ème conférence européenne MICAD, Paris, pp 400 - 420, 1985

[LAU 82] C. LAUGIER
LISP 3D : logiciel graphique pour la manipulation et la visualisation des scènes tridimensionnelles
Grenoble, 1982

[LAU 85] C. LAUGIER, J. PERTIN-TROCCAZ
S.H.A.R.P.: a system for automatic programming of manipulation robots
3d International Symposium of Robotics Research, FRANCE, oct. 1985

[LAU 87] C. LAUGIER
Raisonnement géométrique et méthodes de décision en robotique. Application à la programmation automatique des robots
Thèse de Doctorat d'Etat, INPG Grenoble, 1987

[LEM 86] L. LEMARCHAND
Auto-adaptativité élastique passive en robotique de manutention et de montage
Thèse de Doctorat, Besançon, 1986

[LIE 84] C. A. LIEGEOIS
Analyse de Performances et CAO
Les Robots, tome 7, Hermes, 1984

[LJM 87] T. LOZANO-PEREZ, J.L. JONES,..
HANDEY:a robot system that recognizes, plans and manipulates
IEEE, 1987, pp. 843 - 849

[LOZ 84] T. LOZANO-PEREZ
Robot Motion : Planning and Control
MIT Press, 1984

[MAG 82]
Les ateliers flexibles de montage
Thèse Docteur Ingénieur, ENSAM Paris, 1982

[MAN 88] M. MANTYLA
Introduction to Solid Modeling
Computer Science Press, 1988

[MAS 81] M. T. MASON
Compliance and force control for computer control manipulators
IEEE vol. 11, no. 6, pp. 418 - 433, june 1981

[MAS 89] MASAI.I.O
L'éditeur interactif d'interfaces graphiques
ILOG, 1989

[MCD 89] Mc DONNELL DOUGLAS
Présentation des modules de ROBOTICS
Doc. Mc Donnell Douglas, 1989

[MIC 88] A. MICHARD
An office representation framework for planning and monitoring office activities
Rapport de recherche INRIA, n 891, Rocquencourt, août 1988

[MIC 90] A. MICHARD
Modélisation du dialogue homme - machine
Bulletin de l'INRIA, n 127, mai 1990

[MIL 87] P. MILLOT, D. WILLAEYS
Les approches intelligence artificielle dans les systèmes de supervision
ENAC I. I. et Contrôle de la navigation aérienne, Toulouse, 1987

[MOL 84] A. MOLLON
Tendances et évolution de la CFAO
Revue Eléctronique Industrielle, pp 56 - 60, 1984

[MOR 81] T. MORAN
The Command Language Grammar
Int. J. Man - Machine Studies, 1981, pp. 3 - 50

[MOU 90] J.C. MOUREAU
Génération de plans d'actions en robotique manufacturière
Thèse de Doctorat, Besançon, 1990

[NEV 79] J.L. NEVINS, D.E. WHITNEY
What's Remote Center Compliance and what it can do
Proc. of 9th Symposium on Industrial Robot, Washington, 1979

[NEW 86] A. NEWELL, S. CARD
Straightening Out Softening Up
Human Computer Interaction, 2 (3), 1986, pp. 251 - 267

[NOR 86] D.A. NORMAN, S.W. DRAPPER
User Centered System Design
Laurence Erlbaum Associates Publishers, 1986

[OLS 86] D.R. OLSEN
MIKE: the Menu Interaction Kontrol Environment
ACM Trans. on Graphics, oct. 1986, pp. 318 - 344

[OSO 85] A. OSORIO, L. HENINGER, A. MELLER
Programmation d'une cellule flexible d'assemblage
Actes des journées SM 90, Versailles, 1985

[OWE 85] T. OWIN
Assembly Robots, 1985

[PAR 83] PARENT, LARGEAU
Les Robots, Langages et Méthodes de Programmation
Tomes 5 - 7, 1983

[PFA 85] G. E. PFAFF
User Interface Management System
Eurographics Seminars, Springer - Verlag, 1985

[RAD 87] J. RADHAKRISHNAN, P.M. DEISENROTH
An interactive programming system for the IBM-7545 robot
Comp. Ind. Eng., vol. 12, n 4, 1987, pp. 275 - 282

[RAM 90] F. RAMOS
Une contribution à la plannification des chemins en robotique mobile
Thèse de Doctorat, Besançon, 1990

[REQ 77] REQUICHA, VOELCKER
Geometric modeling of physical parts and processes
IEEE Computer Graphics, vol. 10 - 2, pp 48 - 57, 1977

[REQ 80] A. A. G. REQUICHA
Representation of solid objects : theory, methods and systems
ACM Computer Surveys, pp. 437 - 464, dec 1980

[REQ 82] REQUICHA, VOELCKER
Solid modeling : a historical summary and contemporary assessement
IEEE Computer Graphics, mars 1982

[ROB 89] ROBCAD
Présentation des modules de ROBCAD, TECHNOMATIX, sept. 1989

[ROB 86] A. ROBERT DE ST VINCENT
Perception et modélisation de l'environnement d'un robot mobile: une approche par stéréovision
Thèse de Doctorat, Toulouse, 1986

[ROB 86-b] A. ROBERT
Commande hybride position - force
Thèse de Doctorat, ENSAE Toulouse, 1986

[ROU 83] J.P. ROUGET
Contribution à l'étude structurelle et fonctionnelle d'organes compliants passifs pour robots d'assemblage
Thèse de Doctorat, Besançon, 1983

[ROU 87] C.C. ROUKANGAS, W.A. GUTH MILLER,..
Off line programming motion and process commands for robotic welding of space shuttle man engines
Journal of Robotic Systems, 4 (3), 1987, pp. 355 - 375

[SAL 80] J.R. SALISBURY
Active stiffness control of a manipulator in cartesian coordinates
Proc. of 19th IEEE Conf. on Decision and Control, vol. 1, New Mexico, dec. 1980, pp. 95 - 100

[SEP 85] M. SEPASER
Temps de Cycle d'un Poste d'Assemblage Robotisé
Diplôme universitaire de formation à la recherche, université de Franche - Comté, 1985

[SIB 86] J.L. SIBERT, W.D. HURLEY, T.W. BLESER
An object oriented user interface management system
SIGGRAPH'86, n 20(4), 1986, pp. 259 - 268

[SOE 88] R. SOENEN, C. TAHON, M. RHORAB, J.P. LECAT
User guide of the graphic interactive of data structure design
Document VITAMIN, 11/7/1988

[SOL 84] E.J. SOL, J. VAN DEN BROEK
Progress in CAD tools for robot based flexible automation systems
Proceeding of 14th ISIR, Sweden, 1984

[SOR 83] B.I. SOROKA
What can't robot language do ?
Proceeding of 13th international symposium of industrial robot, Chicago, 1983

[TAY 82] P.H. TAYLOR, P.D. SUMMERS, J.M. MEYER
AML: a manufacturing language
Int. Jour. of Rob. Research, vol. 1, n 3, 1982, pp. 19 - 41

[THE 88] P. THEVENEAU
Utilisation du raisonnement géométrique pour la plannification en robotique d'assemblage: PAMELA
Thèse de Doctorat, INPG Grenoble, 1988

[TIL 81] R.B. TILOVE
Exploiting Spatial and Structural Locality in Geometric Modeling
Ph. D., University of Rochester, 1981

[TEM 85] M. TEMANI
Contribution à la modélisation des processus d'assemblage: élaboration automatique des gammes
Thèse 3ème cycle, INSA Lyon, 1985

[TOD 86] TODD
Fundamentals of Robot Technology
New York, 1986

[TRO 85] J. TROCCAZ
S.M.G.R.: a geometric and relational modeller for robotics
International Conference on Advanced Robotics, Tokyo, sep. 1985

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

[WAN 87] H.P. WANG, R.A. WYSK
Intelligent reasonning for process planning
Comp. in Industrie, 8, 1987, pp. 293 - 309

[WAN 91] J. WANG
Conception d'un système de gestion d'interface utilisateur dans l'environnement CIM
Thèse de Doctorat, Valenciennes, 1991

[WIL 85] J. P. WILHELMS
Graphical Simulation of the Motion of Articulated Bodies
Ph. D. Computer Sciences, U. C. Berkeley, 1985

[WHI 82] The mathematics of coordinated control of arms and manipulators
Jour. of dynamic systems, measurement and control, pp 303-309, dec. 1982

[WON 83] WONG HUDSON
The Australian Robotic Sheep Shearing
Robotics, 1983

[WON 86] WONG, PUGH
Machine Intelligence and Knowledge Engineering for Robotic Applications
NATO ASI Series, série F : computer and system sciences, vol. 33, 1986

[USI 90] Usine Nouvelle, 1990 et 1991

[AXE 90] Axes Robotiques, 1990 et 1991