Institut National
Polytechnique de Lorraine
Inria Lorraine

Centre de Recherche en
Informatique de Nancy

Interprétation et apprentissage géométrique en vision par ordinateur

THÈSE

présentée et soutenue publiquement le **27 Janvier 1989**

pour l'obtention du

Doctorat de l'Institut National Polytechnique de Lorraine

(Spécialité Informatique)

par

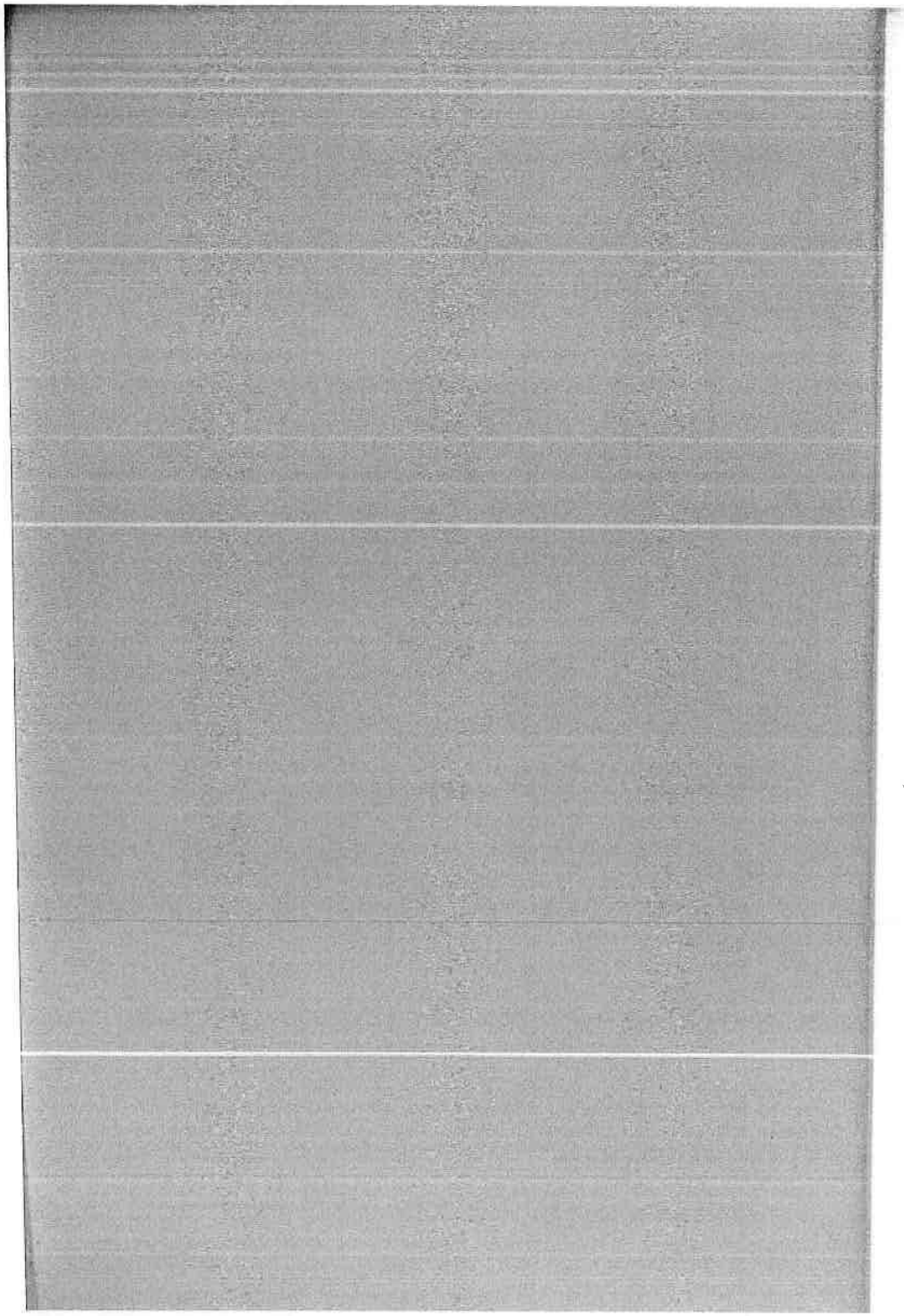
Eric Thirion

Composition du jury :

Président : Jean-Paul HATON

Olivier FAUGERAS
Claude PAIR

Examinateurs : Gerald MASINI
Roger MOHR



Cette thèse est pour moi l'occasion d'exprimer des remerciements à tous ceux qui m'ont conseillé et aidé à réaliser ce projet dans de bonnes conditions, et je suis heureux aujourd'hui de remercier plus particulièrement :

Monsieur Roger Mohr, Professeur à l'Institut National Polytechnique de Lorraine, qui m'a proposé un sujet de recherche passionnant, et m'a encadré et orienté tout au long de ces années de travail. Je lui suis particulièrement reconnaissant pour ses conseils, ses encouragements et également pour son enthousiasme.

Monsieur Gérald Masini, Chargé de Recherche au C.N.R.S, pour sa disponibilité, son aide dans l'utilisation et la conception d'outils de programmation, et pour ses très nombreux moments consacrés à la relecture de mes différents articles et de ce mémoire.

Monsieur Jean-Paul Haton, Professeur à l'Université de Nancy 1 et Directeur de Recherches INRIA au CRIN, qui m'a accueilli dans le laboratoire Reconnaissance des Formes et Intelligence Artificielle du CRIN, et qui me fait l'honneur de présider ce jury. Je le remercie de sa confiance et pour le grand intérêt qu'il a toujours porté à mes travaux.

Monsieur Olivier Faugeras, Directeur de Recherche à l'Institut National de Recherche en Informatique et en Automatique de Sophia-Antipolis et Monsieur Claude Pair, Professeur à l'Institut National Polytechnique de Lorraine, qui ont acceptés d'être rapporteurs de ce mémoire et de siéger à ce jury. Qu'ils trouvent ici l'expression de ma gratitude pour l'intérêt dont ils ont fait preuve à l'égard de ce travail.

Je veux aussi remercier les personnes des différents laboratoires impliqués dans le projet ORASIS *avec lesquelles j'ai collaboré à un moment ou un autre. Je tiens plus particulièrement à remercier Giorgio Toscani et Rachid Deriche pour leur contribution matérielle .*

Je dois des remerciement chaleureux à toute l'équipe Vision du CRIN et plus spécialement à mes étroits collaborateurs Gérald Masini et Long Quan pour leur aide.

Merci également à tous mes collègues et amis du CRIN, pour l'atmosphère de travail sympathique dont j'ai bénéficié.

Introduction générale

On peut constater que les robots actuels effectuent le plus souvent des tâches répétitives dans un environnement adapté et non évolutif. Cette limitation est essentiellement due à une absence de "perception". En particulier, si un robot ne "voit" pas les objets à sa portée il est incapable de les manipuler sans qu'ils soient à un endroit prévu à l'avance. C'est à ce niveau qu'intervient la vision artificielle ou *vision par ordinateur* qui est le domaine général dans lequel s'inscrit cette thèse.

On peut dire que le but de la vision par ordinateur est de réaliser des logiciels, appelés *systèmes de vision*, capables d'acquérir des informations utiles sur le monde physique par l'intermédiaire de capteurs : localisation d'objets, lecture d'un document, détection de défauts, surveillance, suivi de cible... Les applications de ce domaine de l'informatique sont donc nombreuses et dépassent le cadre de la robotique. Malgré cette diversité, l'objectif final est toujours le même : il s'agit d'*interpréter des images*. Commençons donc par définir ce qu'est une image en vision par ordinateur.

0.1 L'image

Une image est une codification, via un capteur, du rayonnement émis ou réfléchi par les objets placés dans le champ de ce capteur. Elle est généralement représentée par une matrice correspondant à un échantillonage discret bidimensionnel du signal délivré par le capteur. La valeur d'un élément de cette matrice, appelé *pixel* ou *pel*, dépend du type de capteur utilisé. Par exemple, elle représente une intensité lumineuse avec une caméra et une distance avec un télémètre laser.

Les images les plus couramment utilisées sont les images à niveaux de gris obtenues par camera video (fig. 0.1). L'image est alors représentée par une matrice à deux dimensions et la valeur de chaque pixel mesure l'intensité lumineuse perçue en ce point par le capteur.

0.2 Le système de représentation et le modèle

Pour interpréter une image, un système de vision doit avoir des connaissances sur son *univers*, c'est-à-dire sur l'ensemble des objets qu'il est susceptible de reconnaître et qui

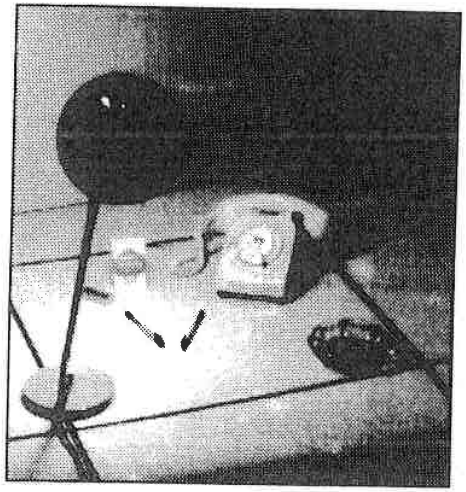


Figure 0.1. Exemple d'image à niveaux de gris (256 lignes × 256 colonnes).

dépend donc de son domaine d'application.

L'ensemble des connaissances que possède le système sur son univers et qui lui permet d'identifier les objets est appelé le *modèle*. Ces connaissances sont organisées selon un *système de représentation* qui définit :

1. le type de connaissance que l'on souhaite représenter. Exemple typique de telles connaissances :
   - des attributs caractérisant la forme des objets : dimensions, angles, ...
   - des attributs dépendant de la nature de la surface des objets : texture, couleur ...
   - des relations entre les objets :
     - la relation *partie-de* et la relation *est-un*
     - des relations exprimant la position relative des objets : *posé-sur*, *adjacent*, *au dessus* ...
2. la *représentation informatique* de ces connaissances : représentation par règles de production [New 73], représentation procédurale, utilisation de langages à objets comme KRL [Bob 77], SMALLTALK [Gol 83], SIMULA [Bri 83], CEYX [Hul 84], ...

Pour un univers donné, il existe donc un grand nombre de système de représentation possibles et un des problèmes fondamentaux de la vision est justement de trouver celui qui s'adapte le mieux à un univers donné.

0.3 Fonctions essentielles d'un système de vision

Nous décrivons dans ce paragraphe trois fonctions fondamentales d'un système de vision : la *segmentation*, l'*interprétation* et l'*apprentissage* (fig. 0.2).

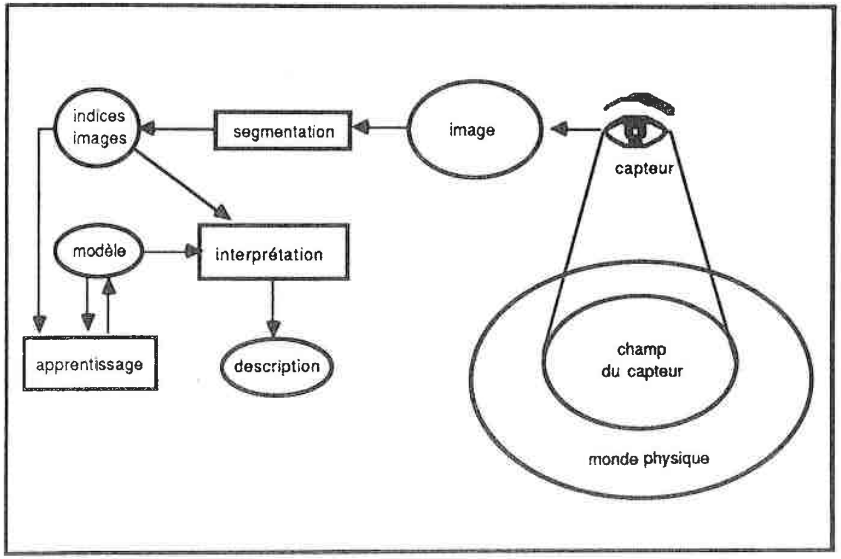


Figure 0.2. Schéma général d'un système de vision

0.3.1 La segmentation

La segmentation sert à découper l'image en des groupes de pixels appelées *indices image* (terminologie de A. Lux [Lux 85]) possédant *a priori* une signification.

Les *contours* sont des indices image fréquemment utilisés. Ils correspondent à des zones de l'image dans lesquelles la valeur des pixels varie brutalement (fig. 0.3).

Dans une image de niveau de gris, l'origine des contours dépend essentiellement de quatre facteurs [Mar 82, p41] : la géométrie des objets, la réflectance des surfaces visibles, l'illumination de la scène et le point de vue. Les entités physiques représentées par ces contours peuvent donc être (fig. 0.4) :

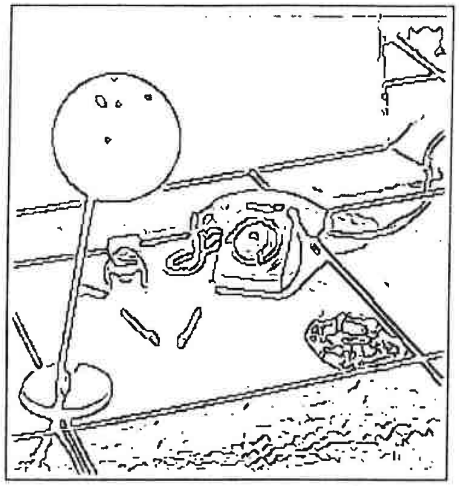


Figure 0.3. Image de la figure 0.1 segmentée en contours

- des lignes de discontinuité de surface
- des bords tangentiels de surface
- des limites d'ombres ou de reflets.
- des lignes de discontinuité de réflectance (dessins sur une surface, limite entre parties de couleur différente ...).

Dans la suite de cette thèse nous regrouperont toutes ces entités sous le nom de *contour physique*.

Les contours d'une image sont fréquemment approximés par des segments de droite que l'on peut également considérer comme des indices image (fig. 0.5).

Les *régions* sont des indices image également très utilisés. Ce sont des zones homogènes de l'image selon un certain critère (niveau de gris, couleur, texture, ...). Dans les images de niveau de gris (fig. 0.6) elles peuvent représenter des surfaces homogènes d'un point de vue géométrique (surfaces planes ou de courbure continue), ou de la la réflectance. On trouve également des régions correspondant simplement à des ombres ou à des reflets.

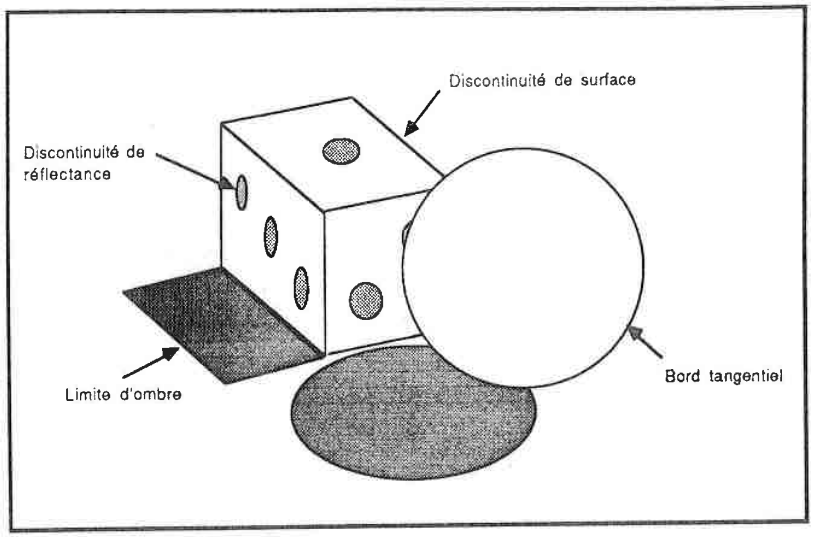


Figure 0.4. Contours physiques

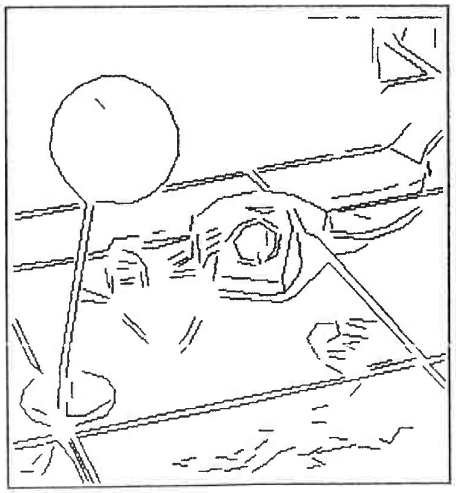


Figure 0.5. Approximation polygonale des contours de la figure 0.3

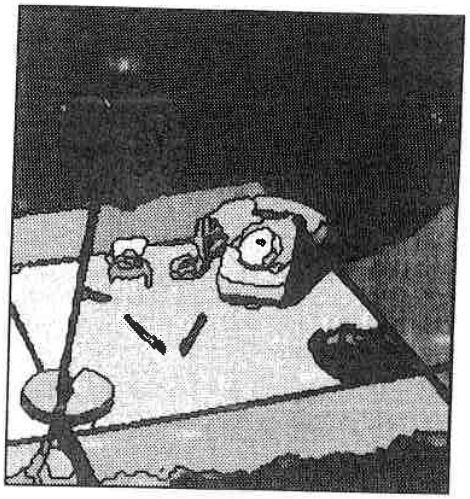


Figure 0.6. Image de la figure 0.1 segmentée en régions [Mon 87]

0.3.2 L'interprétation

L'interprétation est un processus de mise en correspondance entre une (ou des) image(s) et les objets de l'univers représentés dans le modèle.

Son objectif final est de donner une description du monde physique observé. Le contenu de cette description dépend du but du système. S'il s'agit uniquement d'identifier les objets présents, la liste des objets visibles et leurs positions dans l'image est suffisante. Mais quelques fois on souhaite avoir la position spatiale des objets par rapport aux capteurs (manipulation d'objets, navigation) ou bien des propriétés particulières caractérisant la forme ou la nature de la surface (détection de défaut).

0.3.3 L'apprentissage

Par apprentissage, nous entendons ici la faculté de construire et d'améliorer le modèle de manière automatique. Doter un système de vision d'apprentissage, permet de le rendre évolutif et de s'affranchir de la conception du modèle. Le système devient capable de s'améliorer en augmentant ses connaissances sur son univers.

Nous pensons qu'il faut distinguer deux type d'apprentissage selon la nature du modèle utilisé :

- *l'apprentissage symbolique* dont le but est de construire un modèle où les objets

sont représentés par des symboles et des relations entre ces symboles. Ce type d'apprentissage est réalisé par le programme ANALOGY de Winston [Win 84, 383–408] qui fonctionne à partir d'exemples et de contre–exemples décrits manuellement (figure 0.7), ou bien par le système plus récent de Connel et Brady [Con 87] produisant un modèle pour des objets bidimensionnels à partir d'images réelles (cf. figure 0.8).

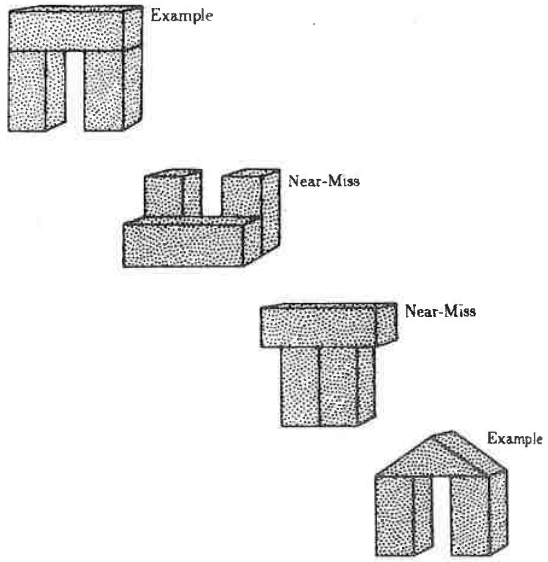


Figure 0.7. Exemples et contre–exemples utilisés pour apprendre le concept d'arche dans le programme de Winston [Win 84]

- *l'apprentissage géométrique* dont la fonction est d'engendrer une description purement géométrique de l'univers du système (par exemple une approximation par des segments de droites ou des portions de plans). Il s'agit donc simplement d'apprendre la forme des objets. L'apprentissage géométrique est plus simple que l'apprentissage symbolique dans la mesure où il ne fait intervenir que ce qui est directement perçu par le système. Il consiste généralement à prendre plusieurs vues des objets à modéliser afin de pouvoir les représenter en trois dimensions.

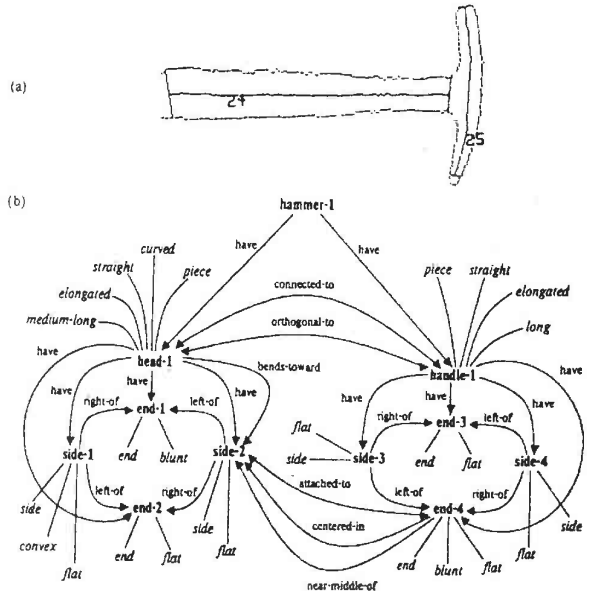


Figure 0.8. Modèle d'un marteau engendré par le programme de Brady et Connell. a) indices image représentant le marteau. b) modèle [Con 87]

0.4 Organisation de cette thèse

Cette thèse est divisée en trois parties. La première est consacrée aux problèmes de représentation et d'interprétation et la seconde au problème d'apprentissage. Ces deux premières parties sont associées à des projets de vision distincts : le projet TRIDENT pour la première et le projet ORASIS pour la seconde.

Dans la troisième partie nous essaierons de voir quelles sont les perspectives possibles des deux approches expérimentées. Nous verrons quelles sont les difficultés à franchir pour que les idées de TRIDENT puissent être réutilisés pour le projet ORASIS et inversement. Cette dernière partie établira donc un lien entre les deux premières et servira en même temps de conclusion.

PARTIE I

Interprétation

1
Introduction

Les sujets abordés dans cette première partie sont les problèmes d'interprétation et de représentation pour lesquels nous proposons une solution dans le contexte relativement général du projet TRIDENT. La difficulté essentielle de ce projet est l'absence d'hypothèse *a priori* sur le monde à percevoir. Les objets à reconnaître et à représenter sont donc quelconques. Par contre, le problème est simplifié par l'absence de bruit dans les données car les images traitées sont synthétiques.

1.1 Les idées

1.1.1 Origines

Les idées explorées dans TRIDENT proviennent essentiellement :

- des travaux de Roger Mohr [Moh 79] sur l'analyse dans un modèle structurel.
- du système d'interprétation de dessin MIRABELLE réalisé par Gérald Masini [Mas 78].
- de tous les travaux liés à l'approche syntaxique en reconnaissance de forme, que l'on peut considérer comme une extension de la théorie des langages formels.

  Les premiers travaux dans ce domaine ont porté sur la reconnaissance de formes bidimensionnelles relativement simples : reconnaissance de caractères [Gri 59] [Ede 62] [Nar 66], de chromosomes [Led 64], d'électro-cardiogrammes [Hor 75] et de dessins [Sha 69]

  Dans l'approche syntaxique un modèle peut être considéré comme une grammaire dans laquelle chaque objet est représenté par un ensemble de règles. Pour étendre les possibilités de représentation, différents auteurs [Vam 73] [Fu 74] [Mas 78] [Gon 78] [Hen 81] ont utilisés des grammaires attribuées. Ce type de grammaire, introduit par Knuth [Knu 71], permet de spécifier la sémantique d'un langage parallèlement à sa syntaxe. A chaque élément du vocabulaire est associé un certain nombre d'attributs.

Les valeurs de ces attributs sont définis par des *équations sémantiques* attachées à chaque règle.

Ce système de représentation est très proche de celui de TRIDENT, excepté le fait que les attributs ne sont pas nécessairement définis par des équations. La partie "sémantique" d'une règle contient, en plus des équations de définition des attributs, un ensemble d'inéquations linéaires appelées "contraintes".

- du système de vision ACRONYM [Bro 78]. Au départ nous nous sommes inspirés de ce système pour la manipulation des contraintes. Dans ACRONYM, ce traitement est réalisé par la méthode "SUP–INF" initialement introduite par Bledsoe [Ble 75] pour déterminer la validité de formules logiques. Elle a ensuite été améliorée par Shostak [Sho 77]. Les essais effectués dans TRIDENT avec cette méthode [Bel 86, 5.1–5.12] nous ont fait renoncer à son utilisation car les temps de calculs étaient trop importants. La méthode actuellement utilisée est l'algorithme de relaxation discrète de Mohr [Moh 85].

1.1.2 Principes d'interprétation

Au niveau de l'interprétation, les idées principales de TRIDENT sont :

- l'interprétation en parallèle de différentes parties de l'image produisant plusieurs descriptions locales de celle–ci.
- l'exploitation des liens contextuels entre ces différentes descriptions pour réduire le nombre d'hypothèses d'interprétation. Les liens considérés dans TRIDENT sont de deux types :
  - les *liens de compatibilité* exprimant le fait que les objets doivent appartenir à un même type de scène (par exemple une scène maritime, une scène urbaine, une scène d'intérieur, ...). Ce type de lien permet d'exploiter le contexte dans lequel on se trouve en excluant les objets qui ne peuvent pas y appartenir.
  - les *liens entre les attributs des objets* permettant d'exprimer des contraintes du type : la hauteur de la table est inférieure à la hauteur de la fenêtre, les quatre pieds de la table ont les mêmes dimensions, ... Ces liens permettent de prédire l'aspect d'un objet en fonction de l'aspect d'un objet déjà reconnu.
- l'utilisation de la propagation discrète pour exploiter ces liens. Le principe est d'éliminer des hypothèses dans des ensembles définis *a priori*. Pour exploiter les liens de compatibilité, ces ensembles sont des ensembles d'objets possibles pour

certaines parties de l'image. Dans le cas des attributs, ce sont des domaines de valeurs hypothétiques.

1.1.3 Principes de représentation

La représentation de TRIDENT est basée sur l'approche syntaxique avec une particularité importante : les attributs *prennent leurs valeurs dans un domaine discret et ordonné*. Par exemple un attribut de type couleur prendrait ses valeurs dans le domaine (rouge, orange, jaune, vert, bleu, violet). Cette particularité est une conséquence du choix de la propagation discrète pour exploiter les liens entre attributs.

1.2 Travaux effectués

Au CRIN, de nombreux travaux ont contribués à l'évolution du projet TRIDENT depuis 1981 (date de début du projet).

Une première maquette a été élaborée en 1981–1983 par F. Zaroli [Zar 83]. Cette première version était écrite en FORTRAN sur MITRA 15. La connexion entre la segmentation et l'interprétation était simulée et l'interprétation n'incluait pas la propagation des contraintes.

Nous avons repris ce travail en 1983 [Thi 84] [Mas 85]. Pour des raisons de changement de matériel, l'interpréteur a été complètement reécrit en Le–Lisp [Cha 86]. Depuis, il a évolué sur plusieurs machines (SM90, SPS9), jusqu'à la version actuelle sur SUN.

A l'origine le système était destiné à l'interprétation d'images de distance. Des travaux portant sur l'extraction de plans et de surfaces quadriques (cônes, sphères, cylindres) dans de telles images ont été réalisés par Y. Belaid [Bel 84].

La version de l'interpréteur réalisée en 1984 fonctionnait sur des images simples dont les surfaces étaient décrites par l'utilisateur. Ce premier travail a permis de dégager le principe de fonctionnement de l'interpréteur qui est encore valable aujourd'hui. Mais les images interprétées et le modèle étaient trop simples pour démontrer l'intérêt de ce principe. Nous avons donc essayé d'augmenter la complexité du modèle et de passer à des images plus proches de la réalité.

Pour cela nous avons utilisé des images synthétiques produites par un système de simulation que nous avions développé parallèlement, en collaboration avec Long Quan et Philippe Valery.

1.3 Organisation du système

L'organisation du système TRIDENT est résumé par la figure 1.1.

Les données proviennent d'une part de l'utilisateur qui conçoit le modèle décrivant les objets et d'autre part, du système de simulation produisant les données à interpréter. Ce système permet de construire des objets polyédriques, puis d'engendrer une vue tridimensionnelle de ces objets selon un point de vue quelconque donné par l'utilisateur.

La construction d'un modèle se produit en trois étapes : saisie, validation et compilation. Dans l'étape de saisie, la représentation externe donnée par l'utilisateur est traduite en une représentation interne sous forme de structures de données LISP. Le programme de saisie joue également un rôle de vérification limitée essentiellement à la syntaxe des règles.

Une vérification beaucoup plus forte est apportée par le programme de validation réalisé par M. Camonin [Cam 87]. Ce dernier permet de montrer que la représentation d'un objet est consistante. Pour cela, il recherche un arbre dérivant de la grammaire et contenant une description cohérente de l'objet. La représentation de l'objet est invalidée si cette recherche échoue.

Le compilateur transforme le modèle obtenu à la première étape en rendant explicites certaines informations implicites dans la première représentation. Le résultat est une forme interne étendue du modèle permettant une interprétation plus rapide. Les programmes de saisie et de compilation ont été réalisés par G. Masini.

Notre contribution se situe essentiellement au niveau de l'interpréteur dont la fonction est de donner une description structurée de la scène observée à partir d'une vue et de la forme interne du modèle. Le fonctionnement de ce programme peut être comparé de manière très schématique à celui d'un analyseur syntaxique produisant un arbre de dérivation (description de la scène) à partir d'un mot à analyser (la vue) et d'une grammaire (le modèle). Cet arbre est construit de manière "parallèle". Durant l'analyse, plusieurs arbres attachés à différentes parties du mot à analyser croissent en même temps. Ces arbres représentent des interprétations locales de la vue. La grammaire étant attribuée, les noeuds de ces différents arbres possèdent des attributs qui évoluent aux cours du temps par propagation discrète. Le résultat souhaité est un arbre unique dont les noeuds (les objets présents) sont caractérisés par des attributs à valeur discrète.

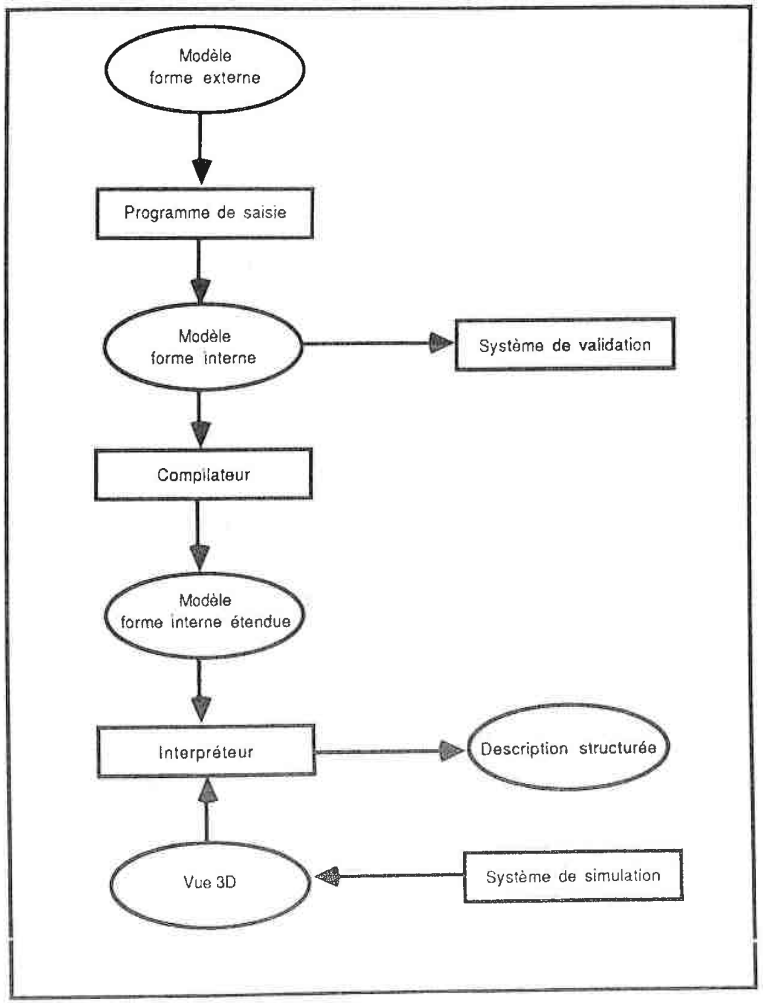


Figure 1.1 Organisation du système TRIDENT

2
Le système de représentation

2.1 Introduction

Le but de ce chapitre est de présenter et d'évaluer le système de représentation de TRIDENT. En premier lieu nous définissons de manière générale les systèmes de représentation syntaxiques dont le système de représentation de TRIDENT est un cas particulier. Cette définition est accompagnée de l'exemple du système d'interprétation de dessins MIRABELLE qui est en grande partie à l'origine de TRIDENT.

Ensuite nous entrons dans la méthode de représentation que nous avons utilisée. Nous expliquons tout d'abord les particularités de cette méthode par rapport aux méthodes syntaxiques. Puis nous définissons le type de connaissances utilisées dans TRIDENT. Le chapitre se termine par une comparaison avec deux systèmes de visions généraux bien connus et par une évaluation de notre principe de représentation.

2.2 Les systèmes de représentation syntaxique

2.2.1 Définition

En représentation syntaxique chaque objet, appelé *forme* est défini par :

- son nom,
- ses propriétés spécifiques (dimensions, angles, coordonnées, ...) appelées *attributs* (l'attribut a d'une forme de nom N est noté $N.a$).
- les compositions possibles de cet objet par d'autres objets, chaque composition étant définie par :
  - une *partie syntaxique* donnant les noms des constituants de l'objet dans cette composition,
  - une *partie sémantique* constituée de contraintes sur les attributs de l'objet représenté et de ses sous objets.

D'un point de vue mathématique ces connaissances peuvent être formalisées par une grammaire dans laquelle :

- l'axiome est un objet contenant tous les autres objets représentés.
- les règles sont les compositions possibles des objets.
- les non-terminaux sont les objets décomposables en sous-objets et définis par une règle.
- les terminaux sont les objets qui ne sont définis par aucune règle et qui sont indécomposables en sous-objets.

En reconnaissance de forme les terminaux sont appelés *formes primitives* et les non-terminaux, les *formes non terminales.*

2.2.2 L'exemple de MIRABELLE

Pour illustrer la représentation syntaxique nous présenterons très brièvement le système de représentation utilisé dans MIRABELLE [Mas 78] [Mas 83].

Les formes primitives utilisées sont des segments de droites répartis en quatre catégories selon leur orientation (fig. 2.1).

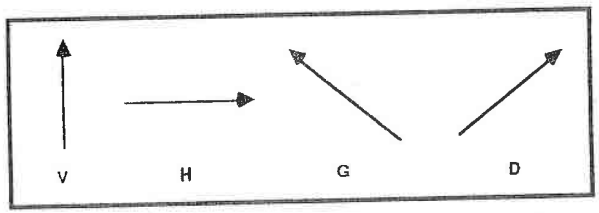


Figure 2.1. Quatres catégories de primitives utilisées dans Mirabelle.

Une forme F, dans le cas général, est une chaîne de segments possédant six attributs :

- une origine ($F.o$),
- une extrémité ($F.e$),
- quatre coordonnées extrêmes $F.xmin$, $F.ymin$, $F.xmax$, $F.ymax$ (fig. 2.2).

Ces attributs permettent de définir différentes relations entre les formes par l'intermédiaire de contraintes. Plus précisément, des contraintes d'égalité sur les extrémités des formes engendrent différentes relations de concaténation ($\circ, \times, +, F$), alors que des

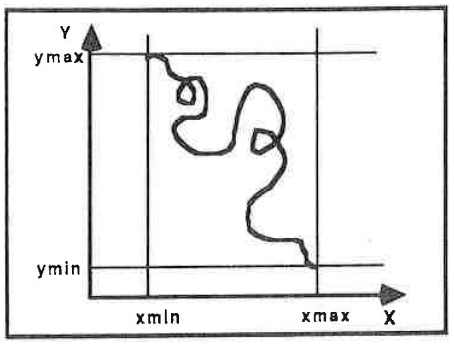


Figure 2.2. Coordonnées extrêmes d'une forme.

inégalités sur les coordonnées extrêmes définissent des relations de position comme *posé-sur, intérieur, à-gauche, ...*

La figure 2.3 montre certaines de ces relations et donne leur définition sous forme de contraintes.

Voici, pour terminer, un exemple de modèle syntaxique représentant une scène maritime (fig. 2.4) constituée d'un voilier sur la mer.

1: Scène-maritime → mer voilier

2: mer → vague mer

3: mer → vague

4: voilier → coque voile-gauche voile-droite
SUR (voile-gauche coque)
SUR (voile-droite coque)
A-GAUCHE (voile-gauche voile-droite)

5: coque → G bas-coque D haut-coque
+ (G bas-coque)
× (bas-coque trait-oblique-droit)
+ (haut-coque trait-oblique-droit)
∘ (haut-coque trait-oblique-gauche)

6: voile-gauche → D V H
× (D V)
+ (H V)
∘ (D H)

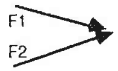
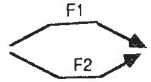
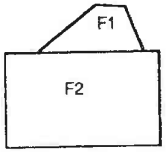
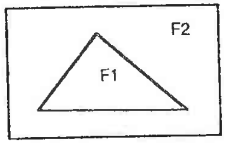
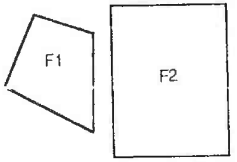
RELATIONS (F1 F2)	EXEMPLE	CONTRAINTES
+		F1.e = F2.o
x		F1.e = F2.e
O		F1.o = F2.o
F		F1.o = F2.o F1.e = F2.e
POSE-SUR		F1.ymin = F2.ymax F1.xmin >= F2.xmin F1.xmax <= F2.xmax
INTERIEUR		F1.xmin >= F2.xmin F1.xmax <= F2.xmax F1.ymin <= F2.xmin F1.ymax <= F2.ymax
A-GAUCHE		F1.xmax <= F2.xmin

Figure 2.3. Relation de connexité et relation de position dans Mirabelle.

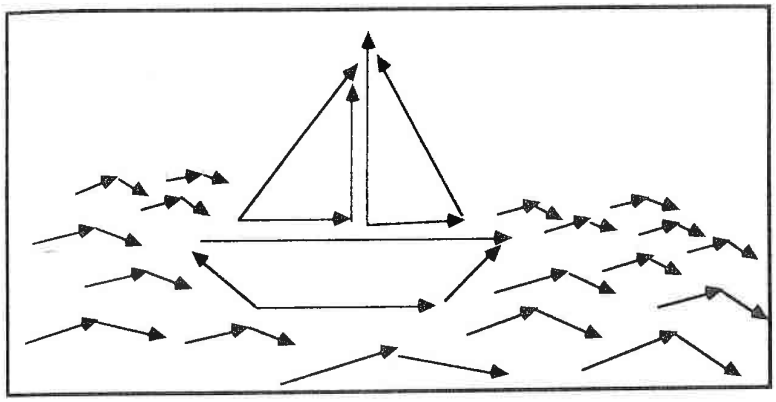


Figure 2.4. Exemple de scène pouvant être décrite par MIRABELLE.

7: voile droit → V G H
× (V G)
+ (H G)
∘ (V H)

8: vague → D G
× (D G)

9: bas coque → H

10 : haut coque → H

2.3 Le système de représentation de TRIDENT

2.3.1 Particularités

Une des particularités les plus importantes du système de représentation de TRIDENT est la discrétisation des attributs. Chaque attribut possède un type défini par une liste de valeurs ordonnées. De plus, la partie sémantique des règles est constituée de contraintes relativement simples. En effet, les seules contraintes autorisées dans notre représentation sont :

- Des contraintes exprimant une relation entre deux attributs a_1 et a_2. Celles-ci sont nécessairement de la forme :

$$a_1 = a_2$$

$a_1 <> a_2$

$a_1 \leq a_2$

- Des contraintes définissant directement les valeurs possibles d'un attribut a :

$$a = (v_1 \ldots v_n) \Longleftrightarrow (a = v_1) \vee \ldots \vee (a = v_n)$$

2.3.2 Connaissances prédéfinies

De manière générale il existe une dépendance entre le système de représentation et le type d'image, de capteurs et d'indices image. Par exemple, il serait inutile de modéliser la couleur dans un système où les capteurs ne permettraient pas de la percevoir. Les capteurs limitent donc les attributs possibles des objets. Il y a également une dépendance de la représentation vis à vis des indices images. Les indices images déterminent les formes primitives et il n'est pas facile de changer d'indices image car cela implique de changer les processus de segmentation permettant de les obtenir. Les systèmes de représentation utilisés en vision font donc intervenir des connaissances fixées *a priori*.

Dans TRIDENT, ces connaissances prédéfinies sont :

- les *types d'attributs*,
- les *attributs prédéfinis*,
- les *formes primitives*

Les types d'attributs

Les *types d'attributs* sont définis par les domaines de valeurs qui leurs sont associés. Deux types d'attributs existent actuellement : le type *angle*, noté A, avec le domaine de valeurs $(0, 45, 90)$ et le type dimension, noté D, avec les valeurs $(0, 10, 20, 40, 80 \ldots 640)$. Pour le type dimension, nous avons choisi un domaine dont les valeurs suivent une progression géométrique afin de pouvoir décrire une grande plage de valeurs avec un échantillonnage réduit. D'autre part cette répartition des valeurs permet de distinguer les petites dimensions avec plus de précision (absolue) que les grandes, ce qui est assez naturel.

Les attributs prédéfinis

Les *attributs prédéfinis* sont des attributs pouvant être associés implicitement à tous les objets. Actuellement, les seuls attributs prédéfinis sont les coordonnées extrêmes *xmin*, *ymin*, *zmin*, *xmax*, *ymax* et *zmax* d'un objet dans le référentiel du capteur

(l'axe des z de ce repère est supposé vertical et le plan d'équation $z = 0$ est supposé confondu avec le sol).

Les formes primitives

Les *primitives du modèle*. Chaque primitive est définie par les noms et les types respectifs de ses attributs. Les primitives de TRIDENT sont des *faces rectangulaires*, notées *RF* et des *faces non rectangulaires*, noté *NRF* (fig. 2.5). Les faces non rectangulaires possèdent uniquement un attribut (sans compter les coordonnées extrêmes) qui est l'angle *phi* de la normale avec la verticale. Les faces rectangulaires possèdent, en plus de cet attributs *phi*, les trois attributs suivants :

- La largeur l.
- La longueur L.
- L'angle *alpha* entre la droite portant le plus grand côté et la verticale. Cet angle sert par exemple à distinguer un rectangle dont le plan est vertical et dont le plus grand côté est horizontal (bord de table), d'un rectangle dont le plan est vertical et le plus grand côté vertical (tranche d'une porte).

2.4 Autres systèmes de représentation

Ce paragraphe compare la méthode de représentation de TRIDENT avec celle de deux systèmes de visions généraux bien connus : ACRONYM [Bro 81] et VISIONS [Han 78]. Nous insistons plus particulièrement sur ACRONYM car certaines idées de TRIDENT sont inspirées par ce système.

2.4.1 Représentation utilisée dans ACRONYM

Description

Dans ACRONYM [Bro 81], les objets sont définis en deux niveaux, le premier niveau servant de support à la définition de plusieurs classes d'objets dans le second.

Le premier niveau est appelé *graphe objet*. Un objet y est défini par :

- des propriétés relatives de deux types :
  - les positions relatives de l'objet par rapport à d'autres objets. Un repère tridimensionnel étant associé à chaque objet, la position relative de deux objets est le déplacement spatial (caractérisé par six paramètres) permettant de passer du repère du premier objet au repère du second.

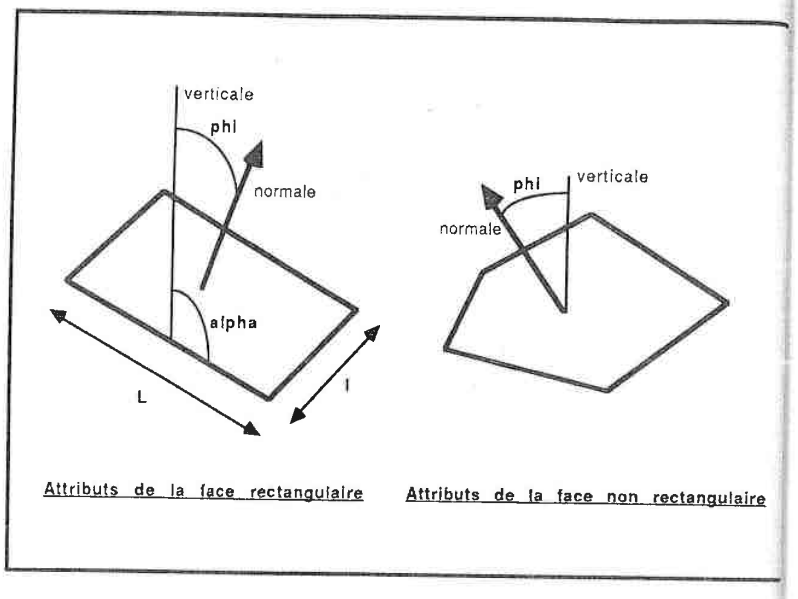


Figure 2.5. Les primitives de TRIDENT et leurs attributs.

- les sous-objets dont est constitué l'objet. La composition est accompagnée du nombre d'occurrences de chaque sous-objet.

- des propriétés spécifiques (dimensions, angles, ...).

Les valeurs attachées à ces propriétés sont des symboles ou des expressions algébriques. Il serait par exemple possible de définir un triangle rectangle par :

petit-cote c
grand-cote C
diagonale $\sqrt{c^2 + C^2}$

Ces expressions algébriques servent donc à exprimer les relations de dépendance entre les propriétés des objets.

Les objets primitifs sont les *cônes généralisés* (fig. 2.6). Ces éléments du graphe objet possèdent trois propriétés spécifiques, à valeur symbolique :

- le type de section : nom de la forme représentant la section.

- le type d'axe.

- le type de variation de la section le long de l'axe.

Par exemple, la représentation d'un cylindre sous forme d'un cône généralisé serait :

- type de section : cercle

- type d'axe : droit.

- variation de la section : aucune.

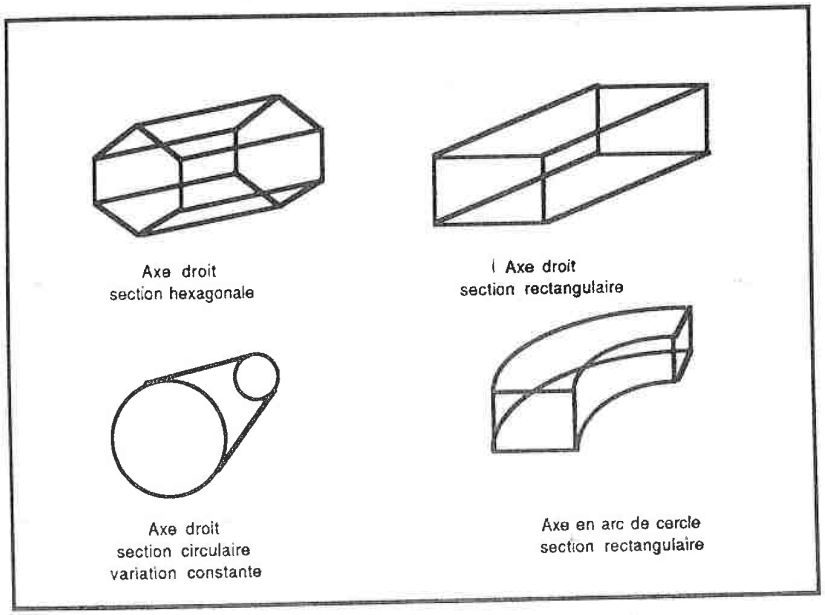


Figure 2.6. Exemples de cônes généralisés utilisés dans ACRONYM.

Le second niveau est le **graphe de restriction.** On y trouve la représentation de plusieurs classes d'objets pour chaque objet du premier niveau. Pour un objet donné, une classe est définie par un ensemble d'inéquations algébriques portant sur les paramètres intervenant dans sa définition.

Par exemple, une voile pourrait être représentée comme une classe particulière de triangle rectangle par les contraintes :

$c \geq 150 (cm)$
$C' \geq 300$

Toutes les classes d'un objet sont organisées de manière hiérarchique. Les sous-classes d'une classe donnée sont des classes plus réduites ou autrement dit, caractérisée par un ensemble d'inéquations plus important. Pour des raisons d'homogénéité, toute classe d'objet est sous-classe d'une classe particulière, appelée *base restriction node*, qui ne contient aucune contrainte.

Comparaison avec TRIDENT

La représentation d'ACRONYM se distingue principalement de celle de TRIDENT par la possibilité d'approcher la réalité de manière plus fine et ceci pour plusieurs raisons :

- les contraintes (inéquations algébriques) sont plus générales que celles de TRIDENT (inéquations simples). Elles permettent en particulier de représenter avec plus de précision le déplacement entre repères et par conséquent, la position relative des objets.
- les attributs numériques prennent leurs valeurs dans l'ensemble des réels alors que dans TRIDENT ils appartiennent nécessairement à un domaine de valeurs discret et très réduit. Ceci donne à ACRONYM la possibilité d'approcher la forme des objets avec plus de précision que dans notre représentation.
- les cônes généralisés permettent de définir un ensemble de formes primitives beaucoup plus important que celui de TRIDENT.

Néanmoins cette précision plus fine se paie par une exploitation plus difficile. Dans ACRONYM les liens entre attributs sont des inéquations algébriques. L'exploitation de ce type de lien demande obligatoirement des calculs assez importants alors que dans TRIDENT aucun calcul (au sens calcul numérique) n'est nécessaire : la propagation des valeurs des attributs par l'intermédiaire des contraintes est simplement un processus d'élimination de valeurs particulièrement rapide.

2.4.2 Représentation utilisé dans VISIONS

Description

Nous décrivons ici la représentation utilisée dans une extension récente du système VISIONS de Hanson et Riseman [Han 78] réalisée par Weimouth [Wey 86].

Dans ce système, un objet est décrit par (fig. 2.7) :

- des propriétés locales liées à sa forme (dimensions, angles) et à la nature de sa surface (texture, couleur).
- la méthode permettant de le reconnaître, codée sous forme procédurale.
- les objets dont il est composé.
- les objets dont il est une spécialisation.
- sa position spatiale par rapport à d'autres objets. Celle-ci n'est pas définie par un déplacement entre repère comme dans ACRONYM, mais par quelques relations types comme *au-dessus de, parallèle*, ...
- différent aspects typiques dépendants du point de vue. Chaque aspect est caractérisé par les relations bidimensionnelles entre les parties visibles de l'objet (à gauche, adjacent à, au-dessus, ...).

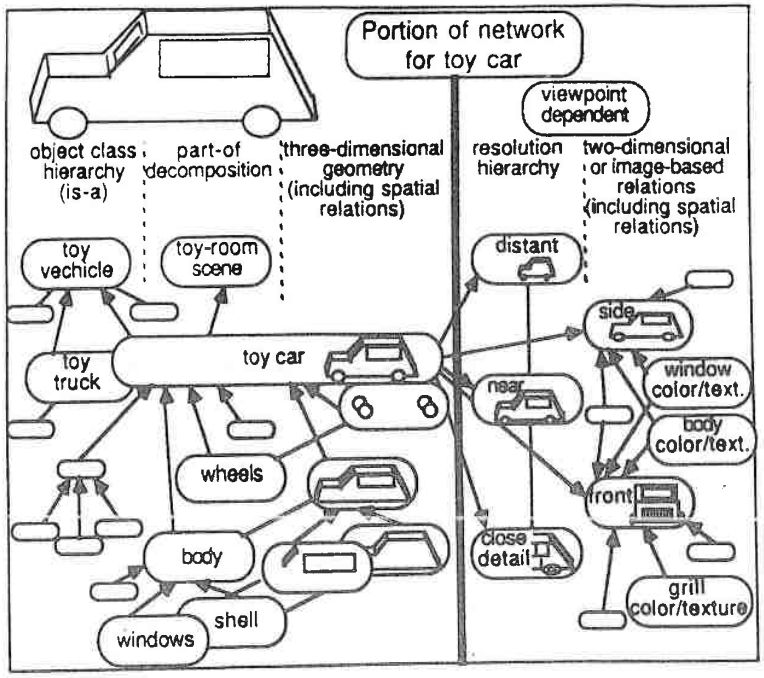


Figure 2.7. Représentation d'une voiture dans VISIONS

La représentation informatique de ces connaissances est un aspect important dans VISIONS. Elle n'est ni purement déclarative, ni purement procédurale mais mixte. Les connaissances associées à un objet sont implantées par une structure possédant plusieurs champs déclaratifs ou procéduraux. Les premiers servent à stocker les informations sur l'objet alors que les seconds permettent sa reconnaissance lorsqu'ils sont exécutés.

Comparaison avec TRIDENT

Les deux idées principales de VISIONS au niveau de la représentation sont : l'attachement de procédures de reconnaissance aux objets et la représentation 2D1/2, c'est-à-dire la description de différents aspects typiques des objets selon différents points de vue. Par rapport à la représentation 3D de TRIDENT, la représentation 2D1/2 de VISIONS a l'avantage de pouvoir être directement comparée avec l'image et nécessite donc moins de calculs. L'inconvénient est qu'il faut donner tous les aspects typiques de tous les objets du modèle.

L'attachement de procédure de reconnaissance à chaque objet permet une adaptation très ad hoc à chaque situation. Mais ceci implique de programmer une méthode de reconnaissance chaque fois qu'un nouvel objet est ajouté au modèle. Dans TRIDENT, au contraire la méthode de reconnaissance des objets est générale et indépendante de leur représentation. La conception et la mise à jour du modèle est donc plus aisée dans TRIDENT et l'indépendance vis à vis de l'univers d'application est plus grande.

2.5 Conclusion

Pour terminer ce chapitre, nous essaierons d'évaluer le système de représentation de TRIDENT selon sa *souplesse* et son *intérêt pour l'interprétation*.

2.5.1 La souplesse

La souplesse du système de représentation de TRIDENT ou autrement dit, la possibilité de pouvoir représenter des objets quelconques est limitée par :

- la définition des domaines de valeurs de chaque type d'attribut. Par exemple, dans la représentation actuelle, une dimension prend nécessairement sa valeur dans $\{0, 10, 20, \ldots, 640\}$. Ceci empêche de distinguer des objets dont les dimensions sont très proches. Bien entendu, il ne serait pas difficile de définir des domaines de valeurs plus importants avec une variation plus faible entre les valeurs consécutives. Mais ceci impliquerait d'une part, un encombrement mémoire plus important et d'autre part, un accroissement du temps de propagation.

- la forme très limitée des contraintes. Il n'y a que quatre contraintes possibles entre deux attributs X et Y : $X \leq Y$, $X \geq Y$, $X <> Y$ et $X = Y$. Ces contraintes ne représentent qu'une infime partie de toutes les contraintes possibles : contraintes linéaires, contraintes algébriques et toutes les contraintes que l'on ne peut pas représenter par des fonctions mathématiques connues.

- les formes primitives. La forme d'un objet quelconque n'est pas nécessairement une union des formes primitives de TRIDENT. Certains objets et surtout les objets naturels (plantes, animaux, corps humain, ...) ne peuvent être représentés que par approximation.

2.5.2 Intérêt pour l'interprétation

Le système de représentation de TRIDENT présente trois avantages importants pour l'interprétation :

- les liens contextuels (liens entre attributs et liens de compatibilité entre objets) sont contenus dans les règles. Ils permettent, comme nous le verrons au chapitre 5, de réduire les possibilités d'interprétation.

- le fait que les attributs soient à valeurs discrètes et la simplicité des contraintes permettent d'utiliser une méthode de propagation discrète efficace pour exploiter les liens entre attributs.

3
Conception d'un modèle

Dans ce chapitre nous décrivons la méthode de construction d'un modèle. Nous définissons les connaissances introduites par l'utilisateur ainsi que les connaissances calculées automatiquement par le compilateur.

3.1 Connaissances données par l'utilisateur

3.1.1 Les relations

Pour simplifier la tâche de l'utilisateur, certaines contraintes typiques entre attributs peuvent être remplacées par des relations. Par exemple, le parallélisme de deux faces f_1 et f_2 qui s'exprime par la contrainte :

$$f_1.phi = f_2.phi$$

peut être exprimé sous la forme ***parallele***$(F_1\ F_2)$.

Il existe ainsi tout un ensemble de relations n-aires prédéfinies permettant de simplifier l'écriture des contraintes. Les relations s'exprimant en fonction d'attributs prédéfinis peuvent être employées pour toutes les formes. $interieur_{xy}$ est une relation de ce type. Elle permet de représenter le fait qu'une forme F_1, vue du dessus, est à l'intérieur d'une forme F_2, vue du dessus également (fig. 3.1). Elle se définit de la manière suivante, en fonction des coordonnées extrêmes :

$$F_2.xmin \leq F_1.xmin$$
$$F_1.xmax \leq F_2.xmax$$
$$F_2.ymin \leq F_1.ymin$$
$$F_1.ymax \leq F_2.ymax$$

Remarquons au passage que ces contraintes sont rudimentaires. Ainsi F_1 est à l'intérieur de F_2 à la figure 3.2.

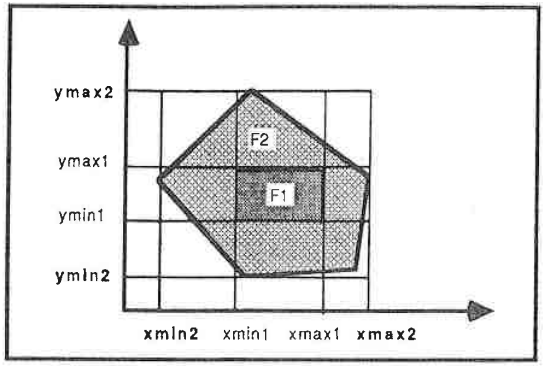


Figure 3.1. Relation $interieur_{xy}$ entre F_1 et F_2.

3.1.2 Syntaxe

Les règles définissant un modèle doivent être exprimées selon une syntaxe rigide. Cuelle-ci est décrite par les figures 3.3, 3.4 et 3.5 dans lesquelles les mots clefs sont en gras.

Bien que les attributs d'une forme soient définis au niveau d'une règle, ceux-ci ne peuvent être redéfinis dans une autre règle où apparait également cette forme. En fait, le compilateur suppose que les attributs d'une forme doivent être définis là où elle apparait pour la première fois.

Dans une règle les seules parties obligatoires sont le membre gauche et le membre droit.

Voici la syntaxe des contraintes pour :

- exprimer les relations entre attributs :

$$attribut = attribut$$

$$attribut <> attribut$$

$$attribut \leq attribut$$

$$attribut \geq attribut$$

- définir les valeurs possibles $v_1 \ldots v_n$ d'un attribut :

$$attribut = v_1 \ldots v_n$$

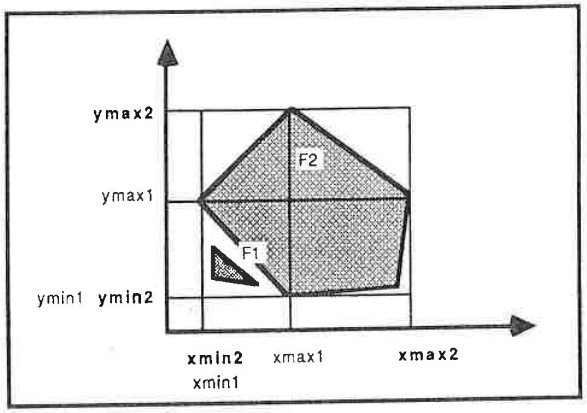


Figure 3.2. Limites de la sémantique de la relation $interieur_{xy}$.

(
MG *nom de la forme définie (membre gauche de la règle)*
MD *noms des formes composantes (membre droit de la règle)*
ATT *définition des attributs*
REL *relations entre les formes de la règle*
CTR *contraintes entre les attributs des formes de la règle*
)

Figure 3.3. Syntaxe de la définition d'une règle.

(
nom de forme (*attribut type* ... *attribut type*)
...
nom de forme (*attribut type* ... *attribut type*)
)

Figure 3.4. Syntaxe de la définition des attributs.

(
nom de relation (*forme* ... *forme*)
...
nom de relation (*forme* ... *forme*)
)

Figure 3.5. Syntaxe de la définition des relations entre objets.

- exprimer le fait qu'un attribut est inférieur ou supérieur à une valeur v :

$$attribut \leq v$$

$$attribut \geq v$$

- exprimer le fait qu'un attribut ne peut prendre aucune valeur dans un ensemble $\{v_1 \ldots v_n\}$:

$$attribut <> (v_1 \ldots v_n)$$

Toutes ces contraintes sont traduites par le compilateur sous une des quatres formes standard définies au paragraphe 2.3.1 :

- une contrainte de la forme $a_1 \geq a_2$ est remplacée par $a_2 \leq a_1$.
- une contrainte de la forme $a \leq v$ est remplacée par $a = v \ldots v_{max}$ où v_{max} est la plus grande valeur possible associée au type de l'attribut a.
- une contrainte de la forme $a \geq v$ est remplacée par $a = v_{min} \ldots v$ où v_{min} est la plus petite valeur possible associée au type de l'attribut a.
- une contrainte de la forme $a <> (v_1 \ldots v_n)$ est remplacée par $a = (v_{min} \ldots v_{max}) - (v_1 \ldots v_n)$.

3.2 Saisie des règles

Le programme de saisie des règles traduit les règles sous forme de structures de données LISP que nous ne décrirons pas ici. Parallèlement à cette traduction, un certain nombre de vérifications ont lieu qui permettent d'éviter des erreurs dans la description des objets. Voici les plus importantes :

- la syntaxe des règles définies précédemment doit être vérifiée.
- toute forme différente d'une forme primitive doit être définie (réduction inférieure)
- toute forme différente de l'axiome doit être utilisée (appartenir au membre droit) dans au moins une règle (réduction supérieure).
- tout attribut utilisé à un moment donné doit avoir été défini auparavant.
- les attributs intervenant dans une même contrainte doivent être de même type.

3.3 Exemples de règles

Voici à présent quelques exemples de règles extraites du modèle actuellement utilisé qui en contient une centaine. Ce modèle représente un univers d'intérieur. Les quatre premières règles du modèle (fig. 3.6) définissent les quatre types de scènes possibles (salon, cuisine, bureau et pièce vide).

```
(MG scene
 MD ( drawing_room ))
(MG scene
 MD ( kitchen ))
(MG scene
 MD ( office ))
(MG scene
 MD ( empty_room ))
```

Figure 3.6. Règles définissant les différents types de scènes.

Les pièces sont définies par les objets ou les ensembles d'objets qui peuvent leur appartenir. Par exemple, une cuisine est représentée par trois règles (fig. 3.7). Dans la première règle, elle est constituée d'une salle (room), d'un ensemble de lampes (lamp*), de tableaux (picture*), de tables (table*) et d'étagères (shelves*). Les deux règles suivantes précisent que les tableaux, les chaises et les étagères ne sont pas nécessairement présents.

```
(MG kitchen
 MD ( room lamp* picture* table* chair* shelves* ))
(MG kitchen
 MD ( room lamp* table* chair* ))
(MG kitchen
 MD ( room lamp* table* ))
```

Figure 3.7. Règles représentant le concept de cuisine.

La récursivité n'étant pas admise, les ensembles d'objets définissables sont nécessairement finis. Un exemple de représentation d'un ensemble d'objets est présenté en figure 3.8.

La figure 3.9 donne les règles définissant la notion de table. Les deux premières règles définissent les deux sortes de tables possibles : la table rectangulaire (rec-table) et la table hexagonale (hexa-table). La troisième règle dit qu'une table rectangulaire possède quatre

```
(MG lamp*
 MD ( lamp ))
(MG lamp*
 MD ( lamp lamp ))
(MG lamp*
 MD ( lamp lamp lamp ))
```

Figure 3.8. Règles utilisées pour définir un ensemble de lampes.

```
(MG table
 MD ( rec-table ))
(MG table
 MD ( hexa-table ))
(MG rec-table
 MD ( table-top table-leg table-leg table-leg table-leg )
 REL (XY-INSIDE ( table-leg table-top )))
(MG hexa-table
 MD ( hexa-table-top table-leg table-leg table-leg table-leg table-leg table-leg
 )
 REL (XY-INSIDE ( table-leg hexa-table-top )))
```

Figure 3.9. Règles définissant la notion de table.

pieds (table-leg) et un plateau (table-top). Les pieds, vus du dessus, sont à l'intérieur du plateau (xy-inside (table-leg table-top)). La table hexagonale est décrite par la quatrième règle comme composition d'un plateau hexagonale (hexa-table-top) et de six pieds avec la même relation entre les pieds et le plateau.

Pour finir, nous donnons la représentation de la forme *bloc* qui nous a servi à définir la plupart des objets. Un bloc est un parallélépipède non incliné, c'est-à-dire possédant quatre faces verticales et deux horizontales. Les faces opposées ont nécessairement la même forme. Il suffit donc de définir trois faces : la face horizontale (*h-face*) et deux faces verticales adjacentes (*v-face1* et *v-face2*).

La règle utilisée pour la face horizontale d'un bloc (fig. 3.10) décrit cette forme comme étant constituée d'un rectangle et possèdant les mêmes attributs que ce dernier (cf. paragraphe 2.3.2). Ces attributs sont : la largeur (l) et la longueur (L) de type dimension (D) et les deux angles phi et alpha de type angulaire (D) définissant l'orientation de la face. La relation unaire *horizontal* exprime le fait que la normale de la face est verticale


```
(MG h-face
 MD ( RF )
 ATT ( h-face ( l D L D phi A alpha A))
 REL (HORIZONTAL ( RF )))
```

Figure 3.10. Règle décrivant la face horizontale d'un bloc.

($RF.phi = 0$). La définition des deux faces verticales (fig. 3.11) suit le même principe.

```
(MG v-face1
 MD ( RF )
 ATT ( v-face1 ( L D l D phi A alpha A))
 REL (VERTICAL ( RF )))
(MG v-face2
 MD ( RF )
 ATT ( v-face2 ( L D l D phi A alpha A))
 REL (VERTICAL ( RF )))
```

Figure 3.11. Règles représentant les deux faces verticales d'un bloc.

Passons à la définition d'un bloc. Dans la règle décrivant cette forme (fig. 3.13), les attributs sont les suivants (fig. 3.12) :

- *hfw* : largeur de la face horizontale.
- *hfl* : longueur de la face horizontale.
- *vfw1* : largeur de la première face verticale.
- *vfl1* : longueur de la première face verticale.
- *vfw2* : largeur de la deuxième face verticale.
- *vfl2* : longueur de la deuxième face verticale.
- *alpha1* : angle avec la verticale du plus grand côté de la première face (0 ou 90 dans ce cas).
- *alpha2* : angle avec la verticale du plus grand côté de la deuxième face.

Le premier groupe de contraintes définit les attributs du bloc en fonction des attributs des faces. Les deux groupes suivant expriment les relations entre les coordonnées extrêmes du bloc et celles des faces.

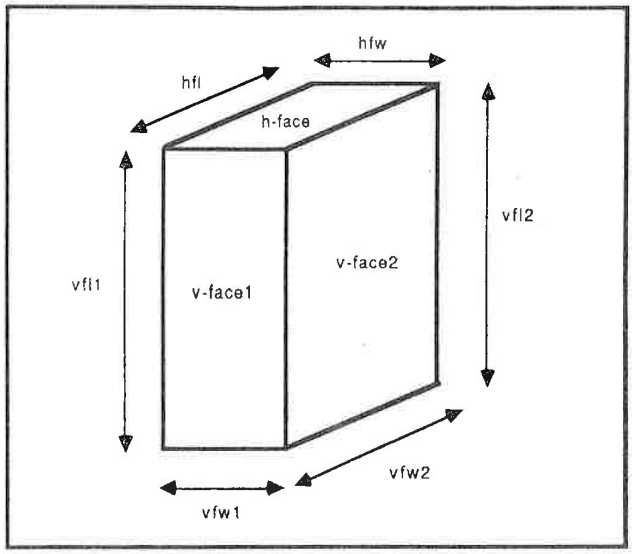


Figure 3.12. Attributs d'un bloc

Pour terminer, nous donnons un exemple de règle utilisant un bloc (figure 3.14). Dans cet exemple le bloc sert à définir un pied de table.

(**MG** bloc
MD (*v-face1 v-face2 v-face1 v-face2 h-face h-face*)
ATT (*block* (*hfw* **D** *hfl* **D** *vfw1* **D** *vfl1* **D** *vfw2* **D** *vfl2* **D**
alpha1 **A** *alpha2* **A**))
CTR ((*block.hfw* = *h-face.l*)
(*block.hfl* = *h-face.L*)
(*block.vfw1* = *v-face1.l*)
(*block.vfl1* = *v-face1.L*)
(*block.vfw2* = *v-face2.l*)
(*block.vfl2* = *v-face2.L*)
(*block.alpha1* = *v-face1.alpha*)
(*block.alpha2* = *v-face2.alpha*)
(*block.zmin* = *v-face1.zmin*)
(*block.zmax* = *v-face1.zmax*)
(*block.zmin* = *v-face2.zmin*)
(*block.zmax* = *v-face2.zmax*)
(*block.zmin* ≤ *h-face.zmin*)
(*block.zmax* ≥ *h-face.zmin*)
(*block.xmin* = *h-face.xmin*)
(*block.ymin* = *h-face.ymin*)
(*block.xmax* = *h-face.xmax*)
(*block.ymax* = *h-face.ymax*)
(*block.xmin* ≤ *v-face1.xmin*)
(*block.xmin* ≤ *v-face2.xmin*)
(*block.ymin* ≤ *v-face1.ymin*)
(*block.ymin* ≤ *v-face2.ymin*)
(*block.xmax* ≥ *v-face1.xmax*)
(*block.xmax* ≥ *v-face2.xmax*)
(*block.ymax* ≥ *v-face1.ymax*)
(*block.ymax* ≥ *v-face2.ymax*)))

Figure 3.13. Règle définissant un bloc.

(**MG** table_leg
MD (*block*)
CTR ((*block.zmax* = 80)
(*block.zmin* = 0)
(*block.rft1* = 80)
(*block.uft2* = 80)
(*block.ufw2* = 10)
(*block.vfw2* = 10)
(*block.hft* = 0)
(*block.hfw* = 0)
(*block.alpha1* = 0)
(*block.alpha2* = 0)))

Figure 3.14. Règle décrivant un pied de table.

3.4 Compilation d'un modèle

La compilation a pour but de déduire du modèle, une fois pour toutes, les connaissances qui sont implicitement contenues dans celui-ci et qui peuvent être exploitées de manière intéressante durant l'interprétation.

Ces connaissances sont des relations entre formes et entre règles qui ont été introduites par R. Mohr pour l'analyse de formes structurées [Moh 79] et utilisées dans MIRABELLE [Mas 78]. Nous n'en donnerons ici qu'une définition intuitive. Les définitions formelles et les algorithmes permettant d'obtenir ces informations à partir du modèle peuvent être trouvées dans [Moh 79].[Mas 83].

Nous appuyerons nos explications sur l'exemple de grammaire suivant :

1) $A \rightarrow B\ C$
2) $A \rightarrow D\ E$
3) $B \rightarrow a\ b$
4) $B \rightarrow b\ c$
5) $C \rightarrow c$
6) $D \rightarrow d$
7) $E \rightarrow e$

Les connaissances ajoutées lors de la compilation sont :

- les *sous-formes* d'une forme. Celles-ci permettent de retrouver directement les parties possibles d'un objet en tenant compte de toutes les alternatives au niveau des règles. Dans l'exemple, les sous-formes de B sont $\{B.a, b, c\}$ et les sous-formes de A sont $\{A.B.C.D.E.a.b.c.d.e\}$.

- les *formes caractéristiques* d'une forme. Une forme F_1 est caractéristique d'une forme F_2 si F_1 est une partie de F_2 et uniquement de F_2. L'observation de F_1 suffit donc à prédire la présence de F_2. Durant l'interprétation la connaissance des formes caractéristiques permet de reconnaître une forme très fortement occultée lorsqu'une (au moins) de ses sous-formes caractéristiques est visible. Dans l'exemple, les formes caractéristiques de B sont $\{B.b\}$.

- les *formes obligatoires* d'une forme. Ce sont les formes qui font nécessairement partie d'une forme donnée quelle que soit la règle la décrivant. Dans l'exemple, b est une forme obligatoire de B.

- les *formes exclues* d'une forme. Les formes exclues par une forme F sont celles qui ne peuvent pas apparaître dans une description (arbre dérivant de la grammaire) contenant F. La connaissance des formes s'excluant mutuellement permet de réduire

les ambiguïtés durant l'interprétation en éliminant les formes exclues par les formes déjà reconnues. La compilation ajoute donc au modèle les liens de compatibilité dont nous avions parlé en introduction. Dans l'exemple, les formes exclues par B sont $\{D, E, d, e\}$.

- les *règles exclues* par une règle. Selon le même principe que pour les formes, les règles exclues par une règle R sont les règles qui ne peuvent pas être utilisées dans une description utilisant R. Dans l'exemple, la règle 1 exclut $\{2, 6, 7\}$, 7 exclut $\{1, 3, 4, 5\}$.

3.5 Conclusion

La conception d'un modèle est facilitée par :

- les vérifications effectuées par le programme de saisie et par le système de validation.
- la modularité des règles. Chaque règle représente un module de représentation indépendant et par conséquent l'adjonction, la suppression ou la modification d'une règle ne remet pas en cause les autres règles.
- la présence de relations simplifiant l'écriture des règles.
- la simplicité des contraintes et la discrétisation des valeurs. Cette simplicité permet à l'utilisateur de décrire des objets sans avoir une connaissance précise de leur forme. Il lui suffit d'avoir une idée approximative des attributs géométriques des objets et de savoir les comparer.

Toutefois, il existe des difficultés :

- il n'est pas possible de s'assurer de la validité du modèle. Une bonne solution serait de pouvoir visualiser les objets représentés mais cela n'est pas possible étant donné que leur forme géométrique n'est pas entièrement définie. De plus, le système de validation n'apporte qu'une aide partielle car même une représentation sans contradictions internes peut être fausse.
- on ne sait jamais si les contraintes sont suffisantes pour pouvoir discerner les objets. Notre expérience nous a montré que les erreurs d'interprétation proviennent souvent d'une insuffisance de contraintes dans les règles. On est alors amener à ajouter des contraintes jusqu'à obtenir des résultats satisfaisants.

4
Construction d'une vue synthétique

L'objectif de ce chapitre est de définir précisément quelles sont les données contenues dans une vue à interpréter et comment elles sont obtenues. Donnons tout d'abord un aperçu général de la méthode.

4.1 Aperçu général

Dans TRIDENT, une vue est un ensemble de *d'instances de primitives* qui sont dans la version actuelle, des instances de faces rectangulaires et de faces non rectangulaires. Chacune d'entre elles représente une face visible d'un objet polyédrique.

Les vues de ce type sont obtenues à partir de vues constituées de segments tridimensionnels que nous appelons "vues fil de fer". Ces dernières sont obtenues à partir d'un système de simulation composé de deux programmes (fig. 4.2). Le premier (ARCHITECTE) permet de construire une représentation tridimensionnelle sommet-arête-face (modèle S.A.F.) d'un objet polyédrique. L'objet est décrit par l'utilisateur sous la forme d'un programme d'assemblage. Il s'agit d'un fichier dans lequel sont contenues des instructions de construction de formes. Les formes produites peuvent être des scènes relativement complexes, comme celle de la figure 4.1.

Le second programme (EXPLORATOR) produit une vue fil de fer d'un objet à partir d'un modèle S.A.F de celui-ci , de la position d'une caméra fictive.

La construction des instances de primitive à partir d'une vue fil de fer est réalisée par un programme d'interface qui utilise également le modèle S.A.F. correspondant.

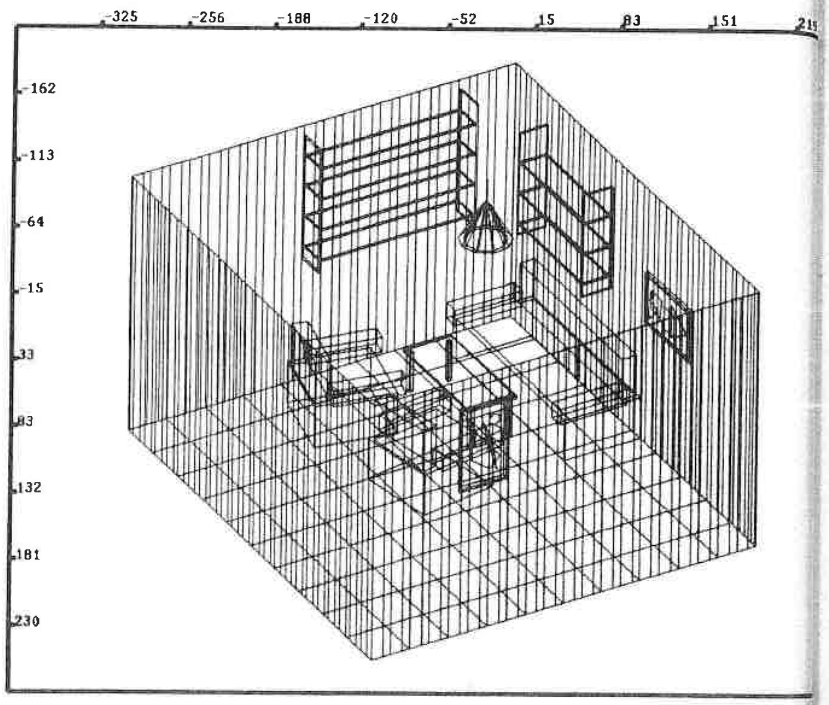


Figure 4.1. Forme 3D conçue à l'aide du système de simulation.

4.2 Construction d'un modèle S.A.F

4.2.1 Le programme d'assemblage

Les instructions de construction de formes contenues dans le programme d'assemblage ont la syntaxe suivante :

$$(C\ F\ p_1 \dots p_n)$$

où C est le nom d'un "constructeur de forme", F le nom de la forme construite et $p_1 \dots p_n$ sont les paramètres de ce constructeur. Les formes manipulées par ces instructions peuvent être bidimensionnelles ou tridimensionnelles.

Les constructeurs possibles sont :

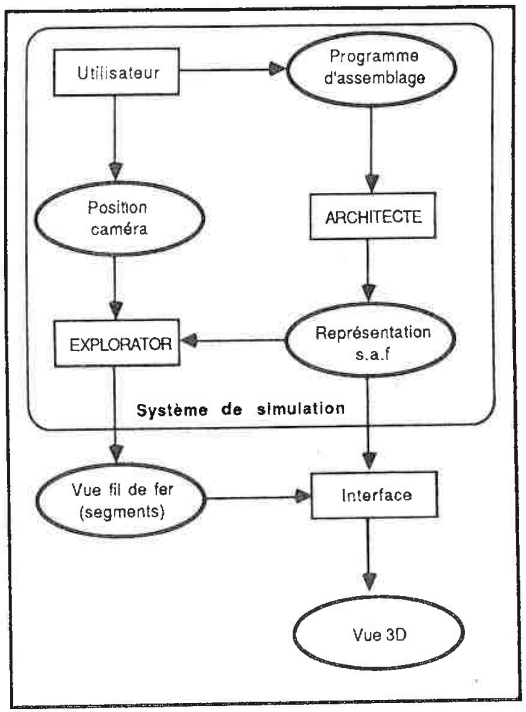
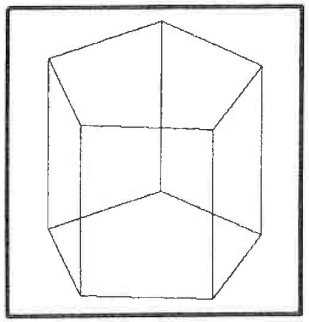


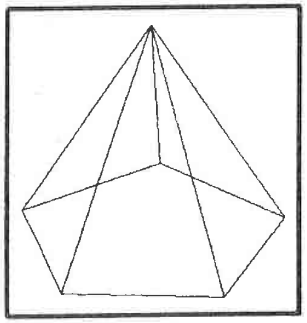
Figure 4.2. Programmes intervenant dans la construction d'une vue.

- Les constructeurs de formes primitives :
  - (**bloc** $F\ x\ y\ z$) : parallélépipède de dimensions x, y et z.
  - (**rectangle** $F\ x\ y$) : rectangle de dimensions x et y.
  - (**polygone** $F\ n\ l$) : polygone régulier de n côtés de longueur l.
- Les constructeurs permettant de construire des formes 3D à partir de formes 2D déjà définies :
  - (**prisme** $F\ b\ v$) (cf. figure 4.3) produit une forme tridimensionnelle F par translation d'une forme 2D b (la base du prisme), selon le vecteur v.
  - (**pyramide** $F\ b\ s\ h$) (cf. figure 4.3) construit une forme tridimensionnelle F en reliant chaque point d'une forme 2D b (base de la pyramide) à un point s

(sommet de la pyramide). Si h est donné, la pyramide est tronquée à cette hauteur.



Prisme obtenu à partir d'un pentagone.



Pyramide obtenue à partir d'un pentagone.

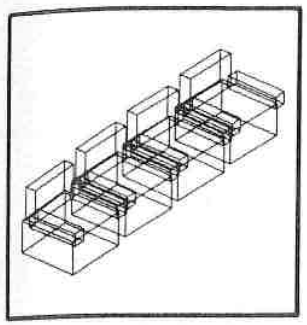
Figure 4.3. Exemple de formes 3D obtenues à partir de forme 2D

- Les constructeurs permettant d'assembler plusieurs formes :
  - (**union** F $(f_1\ p_1)\ldots(f_n\ p_n)$) (figure 4.4) permet d'assembler des formes différentes en fonction de leur position dans un référentiel commun. La position p_i d'une forme f_i est représentée par le déplacement 3D entre le repère local de f_i et le repère commun.
  - (**repetition** F X n d) (cf. figure 4.4) génère une forme F par n déplacements d d'une forme X déjà définie. Le déplacement est défini par un vecteur de translation, un angle et un axe de rotation.

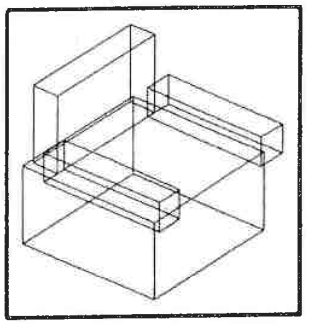
4.3 Fonctionnement du programme ARCHITECTE

Le programme ARCHITECTE exécute séquentiellement les instructions contenues dans le programme d'assemblage. L'exécution d'une instruction provoque la génération en mémoire d'une structure à trois champs représentant la forme résultat :

- le premier champ contient la liste des sommets avec leurs coordonnées,
- le second contient la liste des arêtes avec les deux sommets associés à chaque arête,
- le troisième donne la liste des faces de l'objet. Chaque face est définie par les arêtes qu'elle contient et par sa normale.



Répétition d'un fauteuil.



Fauteuil obtenu par union de blocs.

Figure 4.4. Exemple de formes 3D obtenues par assemblage.

Bien entendu, lorsque la forme construite est bidimensionnelle, le troisième champ est vide.

Après exécution du programme d'assemblage, l'utilisateur a la possibité de visualiser toutes les formes engendrées par le programme. De plus, la représentation S.A.F d'une forme quelconque peut être sauvegardée dans un fichier pour être utilisée par EXPLORATOR.

4.4 EXPLORATOR

4.4.1 Positionnement de la caméra

Pour créer une vue fil de fer d'un certain objet avec EXPLORATOR, l'utilisateur donne le nom du fichier contenant la représentation S.A.F de cet objet. Il doit ensuite préciser la position de la caméra fictive dans le repère de l'objet par cinq paramètres :

- Les trois coordonnées du centre focal de la caméra.
- L'orientation de l'axe de la caméra, défini par deux angles θ et ϕ (coordonnées sphériques).

Les paramètres intrinsèques de la c méra : dimensions de la rétine et distance focale, peuvent également être modifiés.

Pour vérifier la position de la caméra, l'utilisateur a la possibilité de visualiser sa position en projection sur le plan Oxy (vue du dessus), Oxz ou Oyz. La caméra est

matérialisée par trois segments de droite (fig. 4.5). Deux d'entre eux représentent le champ de visibilité et se rejoignent au centre focal de la caméra. Le troisième représente le plan rétinien.

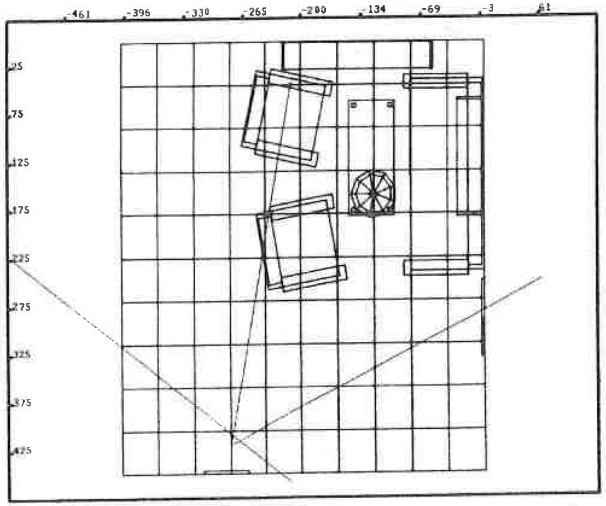


Figure 4.5. Vue du dessus de la position de la caméra dans la pièce de la figure 4.1

4.4.2 Construction d'une vue fil de fer

Lorsque la position de la caméra a été définie, le système calcule l'image observée par la caméra selon cette position. Celle-ci est obtenue par projection perspective des arêtes du modèle (fig. 4.6) et élimination des lignes cachées [Fol 82] (fig. 4.7). Le résultat est un ensemble de segments de droite représentant les parties visibles des arêtes du modèle. Chacun de ces segments est défini par les coordonnées 2D de ses extrémités dans l'image.

Une étape de calcul supplémentaire est nécessaire pour retrouver les coordonnées 3D des segments visibles dans le référentiel de la caméra. Ces dernières sont obtenues par projection perspective inverse des extrémités (fig. 4.8) : le point 3D correspondant à l'extrémité e d'une arête visible est l'intersection de la droite portant cette arête avec la droite des points pouvant se projeter en e.

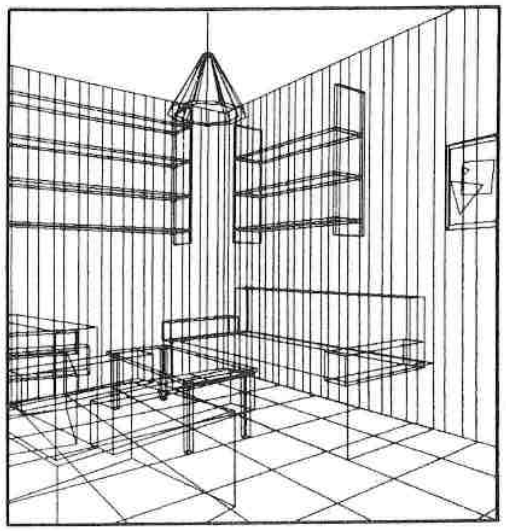


Figure 4.6. Projection perspective de la scène de la figure 4.1 avec la position de la caméra représenté en figure 4.5

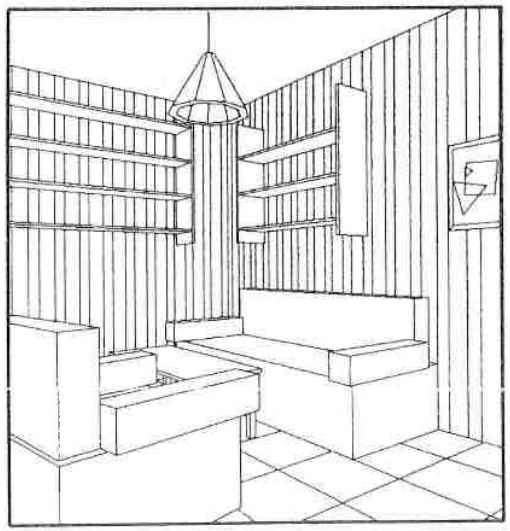


Figure 4.7. Segments visibles dans l'image de la figure 4.6

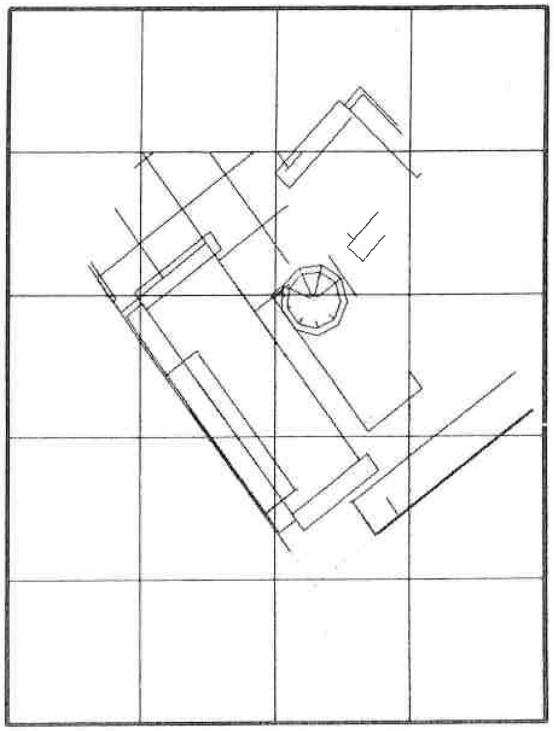


Figure 4.8. Projection verticale de la vue fil de fer correspondant à l'image de la figure 4.7

4.5 Extraction des instances de primitives

4.5.1 Contour visible d'une face

En partant uniquement des coordonnées des segments il serait assez difficile de retrouver ceux qui appartiennent à une même face. Mais, lors de la construction d'une vue fil de fer, EXPLORATOR mémorise pour chaque segment visible, l'arête du modèle qui lui correspond. En utilisant le modèle S.A.F, les instances de primitive peuvent donc être retrouvées très facilement. Il suffit de parcourir chaque face du modèle S.A.F et de créer une instance de primitive lorsque certaines des arêtes de la face sont visibles dans l'image. On mémorise ensuite l'association entre les parties visibles de ces arêtes, représentées par des segments dans la vue fil de fer, et l'instance de primitive engendrée. Les segments de la vue associés à une instance de primitive constituent son contour visible.

La figure 4.9 montre les instances de primitives obtenues à partir de la vue en fil de fer de la figure 4.7 et du modèle représenté par la figure 4.1.

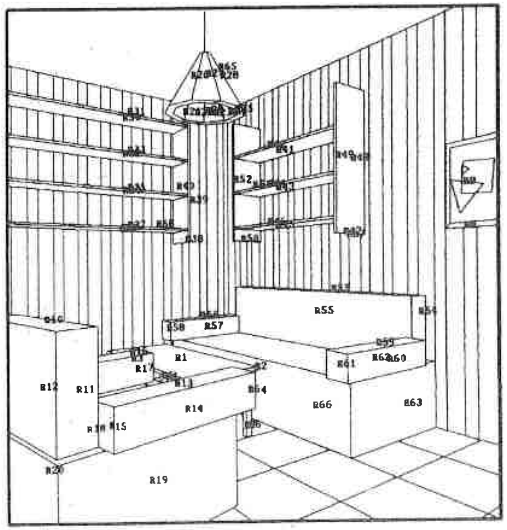


Figure 4.9. Instances de primitive extraites de la vue de la figure 4.7

4.5.2 Distinction entre face rectangulaire et face non rectangulaire

Le seul cas où l'on peut avoir la certitude qu'une face est rectangulaire est celui où la face est entièrement visible. En utilisant que le contour visible d'une face, la distinction entre face rectangulaire et face non rectangulaire n'est donc généralement pas possible. Mais

Attribut	Valeurs	Type	Valeurs possibles du type
α	(0 90)	angle	(0 45 90)
ϕ	(90)	angle	(0 45 90)
largeur	(20 40 80 160 320 640)	dimension	(0 10 20 ... 640)
longueur	(40 80 160 320 640)	dimension	(0 10 20 ... 640)
xmin	(0 10 20 40 80)	dimension	(0 10 20 ... 640)
ymin	(0)	dimension	(0 10 20 ... 640)
zmin	(0 10 20)	dimension	(0 10 20 ... 640)
xmax	(160)	dimension	(0 10 20 ... 640)
ymax	(20 40 80 160 320 640)	dimension	(0 10 20 ... 640)
zmax	(40)	dimension	(0 10 20 ... 640)

Tableau 4.1. Attributs d'une face rectangulaire

nous avons simplifié le problème en supposant qu'il n'existe pas de face non rectangulaire contenant des arêtes perpendiculaires. Avec cette hypothèse, une face est rectangulaire si et seulement si il existe au moins deux segments perpendiculaires dans son contour visible.

4.5.3 Caractérisation des faces

Les attributs d'une instance de primitive sont déduits des segments de son contour visible. Mais généralement, il n'est pas possible de leur donner une valeur exacte car cela dépend du degré de visibilité de la face considérée. Lorsque celle-ci n'est que partiellement visible, on ne peut donner que des valeurs approchées ou des domaines de valeurs.

Pour résoudre ce problème, tous les attributs sont définis par des listes de valeurs extraites de la liste des valeurs possibles du type correspondant. Pour fixer les idées, le tableau 4.1 donne les attributs de la face rectangulaire instancié par la face 20 de la figure 4.9 (face rectangulaire à l'arrière du fauteuil).

Le fait qu'un attribut puisse avoir plusieurs valeurs possibles n'est pas très gênant pour l'interprétation car, comme nous le verrons ultérieurement, ce type de donnée est tout à fait adapté au mécanisme de propagation utilisé par l'interpréteur.

Dimensions des instances de faces rectangulaires

Les dimensions des faces rectangulaires sont calculées par une analyse de cas qui est résumée par la figure 4.10. La dimension des côtés parallèles à une des deux directions du rectangle n'est connue que lorsque les deux côtés perpendiculaires à cette direction sont partiellement visibles : c'est la distance entre les droites portant les parties visibles de ces

deux côtés. Si un seul des côtés perpendiculaires est visible, alors la dimension est minorée par la distance entre la droite portant ce côté et le point le plus éloigné de cette droite appartenant à un des deux côtés que l'on veut mesurer.

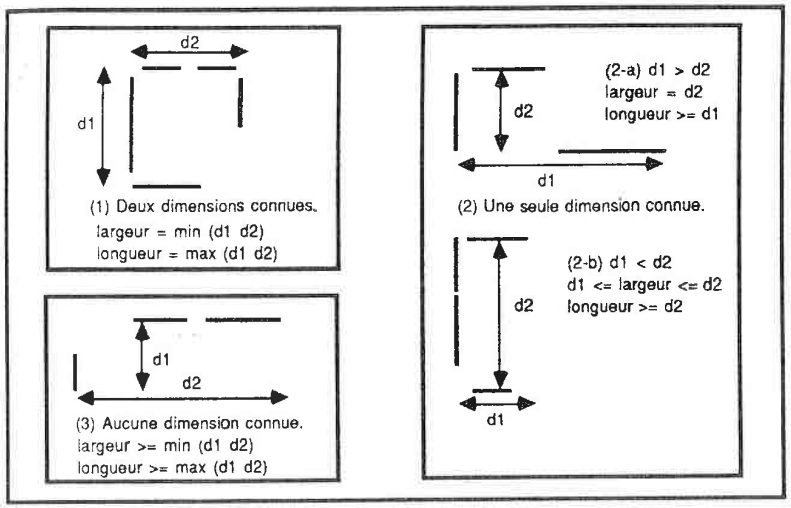


Figure 4.10. Les différentes configuration de visibilité d'une face rectangulaire.

Valeurs possibles des coordonnées extrèmes

Le calcul des coordonnées extrèmes suppose que toutes les faces d'objets sont convexes. Dans ce cas, les extrema locaux en x, y et z sont également des extrema globaux. Plus particulièrement, lorsqu'une face est polygonale, ses extrema sont forcément les coordonnées de certains sommets. Il suffit donc d'observer un sommet représentant un extremum local pour une certaine coordonnée, pour en déduire la valeur extrème de cette coordonnée sur toute la face. Ceci permet de trouver les coordonnées extrèmes d'une face, même lorsque celle-ci est fortement occultée.

Dans le cas où aucun coin ne permet de déduire la valeur extrème d'une certaine coordonnée, on lui associe une liste de valeurs, majorée ou minorée, selon le cas, par la valeur extrème de cette coordonnée sur tous les segments visibles.

Valeurs possibles des angles

L'angle ϕ d'une face peut être déterminé avec exactitude quelle que soit la configuration car chaque face doit posséder au moins deux segments non parallèles dans son contour pour être interprétée retenue. Cela permet de calculer la normale de la face par produit vectoriel des vecteur directeurs de ces deux segments. ϕ est donc donné par l'angle entre la verticale et la normale calculée de cette manière.

Par contre, l'angle α représentant l'inclinaison du plus grand côté d'un rectangle, ne peut pas toujours être défini. En effet, on ne peut pas toujours dire, d'après les parties visibles d'un rectangle, quels sont les segments qui appartiennent au petit côté et quels sont ceux qui appartiennent au grand côté.

4.6 Résumé

Dans TRIDENT, les données tridimensionnelles sur une scène sont des instances de faces caractérisées par leur forme (rectangulaire ou non) et des attributs spécifiques comme les coordonnées extrèmes, l'orientation de la normale ... A chaque attribut est associée une liste de valeurs possibles.

ARCHITECTE permet de créer une représentation tridimensionnelle de la scène par sommets, arêtes et faces à partir d'un programme d'assemblage donné par l'utilisateur. EXPLORATOR donne ensuite la possibilité d'engendrer une vue en fil fer de cette dernière selon un point de vue quelconque.

On extrait ensuite les instances de faces de cette vue fil de fer en s'aidant du modèle S.A.F de la scène. Puis chaque instance de face est identifiée comme une face rectangulaire ou non rectangulaire. En dernier lieu, les instances de faces sont décrites par domaines de valeurs calculés à partir de leur contour visible.

5
Interprétation

5.1 Introduction

5.1.1 But de l'interprétation

Rappelons que l'interprétation est un processus qui permet de déduire des informations sur une scène observée en fonction de données visuelles sur celle-ci et d'un modèle. Plus précisément, il s'agit de retrouver les objets présents et éventuellement des informations supplémentaires telles que la localisation de ces objets par rapport aux capteurs, leurs dimensions, etc.

Dans TRIDENT, ce rôle est réalisé par l'interpréteur dont le but est de donner une description structurée d'une scène à partir d'une vue de celle-ci et de la forme interne compilée du modèle.

La description que l'on souhaite obtenir est un arbre dérivant des règles du modèle et représentant la structure de la scène. Chaque nœud de cet arbre correspond à une instance d'une forme du modèle et est étiqueté par la forme qu'il instancie. En particulier, la racine de l'arbre est étiquetée par la forme axiome et les feuilles, par les formes primitives. La correspondance entre les formes de la description et la vue est représentée par des liens entre les *nœuds terminaux* (étiquetés par une forme primitive) et instances de faces.

En réalité, la description finale de la scène n'est pas toujours aussi complète. Durant l'interprétation, plusieurs arbres représentant différentes interprétations locales de la vue sont construits en parallèle. Ces différents arbres fusionnent au cours du temps et la forêt converge en principe vers une arborescence unique. Toutefois, l'interprétation peut s'arrêter avant la fusion de tous les arbres. La description finale peut donc être constituée de plusieurs arbres. De plus, ces arbres ne sont pas nécessairement complets, c'est-à-dire qu'ils peuvent avoir des *feuilles non terminales* (étiquetées par une forme non terminale).

Les propriétés des formes identifiées sont représentées par des *instances d'attribut* attachées aux nœuds. A chacune d'entre elle est associée un domaine de valeurs possibles. Les instances d'attribut d'un nœud N correspondent de manière biunivoque aux attributs

de la forme f qui l'étiquette. Si les attributs de f sont $a_1, \ldots, a_n$ alors les instances d'attribut attachées à N sont notés $N.a_1, \ldots, N.a_n$. Le domaine de valeur d'une instance d'attribut $N.a$ est noté $D(N.a)$.

5.1.2 Plan du chapitre

Ce chapitre décrit les mécanismes d'interprétation de TRIDENT. L'interprétation est considérée comme l'exécution d'une suite d'actions agissant sur des structures de données représentant la description actuelle de la vue. Nous commençons par décrire le contrôle central de ces actions qui définit l'architecture globale de l'interpréteur. Les paragraphes suivants sont consacrés au fonctionnement des différentes actions. Les principes stratégiques de TRIDENT, c'est-à-dire la manière d'ordonnancer les actions seront abordées dans les paragraphes 5.8 et 5.9. Enfin nous terminerons par une critique de notre réalisation, mettant en évidence les points non résolus ou ayant une solution non satisfaisante.

5.2 Contrôle de l'interprétation

Nous avons essayé de concevoir l'interpréteur comme un "outil de recherche", c'est-à-dire de manière à ce que différents points stratégiques puissent être implantés ou modifiés sans trop de répercussions sur l'ensemble du programme. Dans ce but, le principe des systèmes à règles de production [Lau 82] nous a semblé être une bonne solution. La représentation de la stratégie d'interprétation sous la forme d'un ensemble de règles *condition* → *action* est modulaire et par conséquent facile à modifier. Cependant la représentation des règles que nous avons adopté n'est pas déclarative mais procédurale. La partie condition d'une règle de production est une fonction, que nous appellons fonction de contrôle de l'action. Il existe également pour chaque action une *fonction de support* déterminant les objets sur lesquels elle doit agir.

L'interprétation comporte les trois phases cycliques habituelles des systèmes à règles de production :

- Une phase de **filtrage** déterminant les actions exécutables à un moment donné. Cette phase consiste simplement à évaluer les fonctions de contrôle de chaque action.
- Une phase de **sélection** déterminant l'action à exécuter parmi toutes les actions exécutables.
- Une phase d'**exécution** de l'action sélectionnée.

Pour des raisons d'efficacité, les actions considérées dans l'étape de filtrage sont uniquement celles qui sont contenues dans une file d'attente. Celle-ci est mise à jour après chaque

exécution d'une action. Lorsqu'une action est exécutée, elle "demande" l'exécution des actions qui peuvent logiquement la suivre en rajoutant ces nouvelles actions dans la file d'attente.

Le contrôle de l'interprétation est résumé par l'algorithme de la figure 5.1 avec les notations suivantes :

- *condition-arrêt* : fonction booléennne vraie si et seulement si la description est jugée satisfaisante.
- *file* : liste d'actions à exécuter.
- *filtrage* : fonction retournant la liste des actions exécutables parmi une liste donnée. Cette fonction exécute simplement les fonctions de contrôle de chaque action de la liste.
- *sélection* : fonction retournant une action parmi une liste donnée. Elle définit les critères de sélection de l'action à exécuter.
- *support* : fonction retournant la liste des paramètres d'une action donnée en exécutant la fonction de support de cette action.

Tant_que *non condition-arrêt* **faire**
  file ← filtrage (file)
  action ← sélection (file)
  file ← file − {action}
  paramètres ← support (action)
  action (paramètres)
Fin_tant_que

Figure 5.1. Boucle de contrôle des actions

Le contrôle est donc déterminé par un certain nombre de fonctions clefs (condition d'arrêt, fonction de contrôle, fonction de support, fonction de sélection) qui définissent la stratégie d'interprétation. Mais expliquons tout d'abord quelles sont les actions possibles de l'interpréteur. Il existe quatre types d'action :

- les *actions de construction*, qui font "croître" la forêt en ajoutant de nouveaux nœuds ;
- les *actions d'initialisation* qui évaluent les différentes possibilités de construction ;
- les *actions de propagation* qui éliminent directement ou indirectement des possibilités de construction et qui permettent de détecter des erreurs dans la description ;

- les *actions de gestion des erreurs*.

5.3 Les actions de construction

Les actions de construction sont celles qui font évoluer la description. Il y en a quatre :

- la phase ascendante :
- la phase descendante :
- démarrer une analyse :
- la fusion.

5.3.1 Phases ascendantes et descendantes

Les phases ascendantes et descendantes sont deux manières d'ajouter des nœuds à la forêt en utilisant une règle du modèle. Intuitivement, une phase descendante est la prédiction des constituants d'un objet déjà reconnu. Inversement, dans une phase ascendante, on prédit de quel objet fait partie un objet déjà reconnu et comment se décompose cet objet.

Illustrons ces deux actions par un exemple. Supposons que le modèle contient les règles suivantes :

1. *fauteuil* → *accoudoir accoudoir bas dossier*
2. *dossier* → *bloc*
3. *bas* → *bloc*
4. *accoudoir* → *bloc*

De plus, supposons que la forêt comprend un arbre unique constitué d'un seul nœud étiqueté par la forme *bloc* (fig. 5.2). Cet arbre peut être étendu par une phase ascendante en utilisant une règle dans laquelle la forme étiquetant la racine (ici la forme *bloc*) apparaît en membre droit. Dans ce cas, trois règles sont possibles : les règles 2, 3 et 4. Prenons la règle 4. Les nœuds ajoutés en phase ascendante sont :

- Un nœud étiqueté par la forme en membre gauche de la règle choisie. Ce nœud devient la nouvelle racine de l'arbre.
- Les nœuds instanciant les formes appartenant au membre droit de la règle (excepté l'ancienne racine de l'arbre). Ceux ci sont ajoutés comme nœuds fils de la nouvelle racine.

Après avoir utilisé la règle 4 en phase ascendante, la nouvelle racine de l'arbre est étiquetée par la forme *accoudoir*. Pour continuer la construction, la seule possibilité est d'effectuer une deuxième phase ascendante en utilisant la règle 1. Le nouvel arbre obtenu comprend des feuilles non terminales. Il devient donc possible d'exécuter une phase descendante :

1. choisir une feuille non terminale à étendre,
2. choisir une règle dont le membre gauche est la forme étiquetant la feuille choisie,
3. ajouter à la feuille des nœuds fils instanciant les formes appartenant au membre droit de la règle.

Dans notre exemple, la feuille étiquetée par la forme *dossier* peut être étendue de manière descendante en y "greffant" la règle 2.

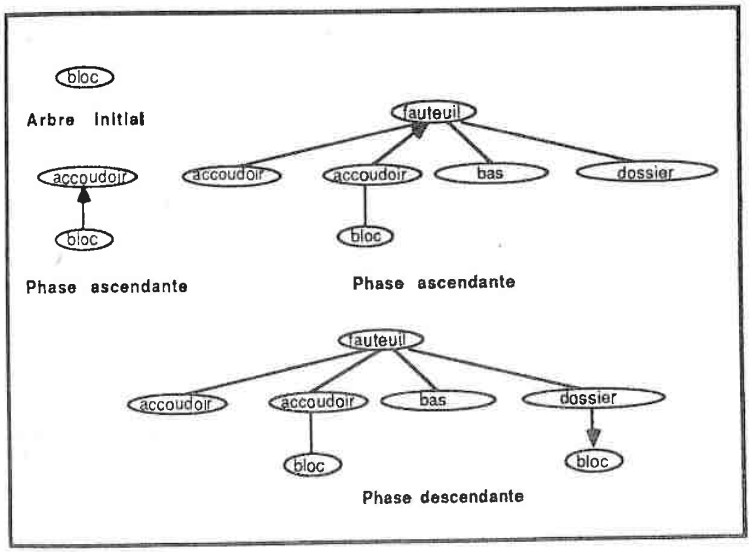


Figure 5.2 Exemples de phases ascendantes et descendantes

5.3.2 Démarrer une analyse

La suite de phases ascendantes - descendantes appliquées à un arbre donné est appelée une analyse. Pour démarrer une analyse il doit exister un nœud de départ. Celui-ci est engendré par l'action *démarrer une analyse* qui associe un nœud terminal à une instance

de face de la vue. Les instances d'attribut du nœud sont initialisées par les listes de valeurs associées aux attributs correspondants de la face et un lien est établi entre la face et le nœud (fig. 5.3)

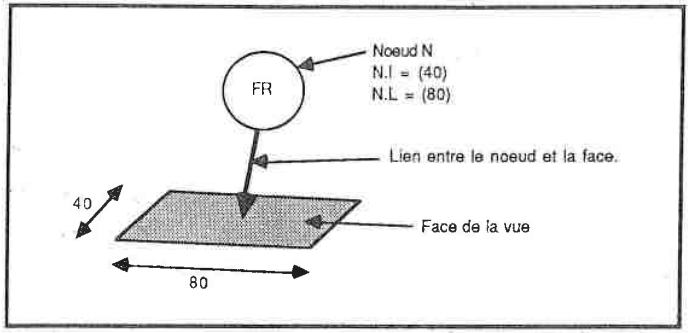


Figure 5.3. Démarrage d'une analyse. Exemple d'une face rectangulaire

5.3.3 La fusion

La fusion est une action de construction qui n'ajoute aucun nœud. Elle ne fait que "rassembler" des arbres existants et permet donc de faire évoluer la forêt vers une arborescence unique. Cette action consiste à substituer un arbre à une feuille, lorsque la feuille et la racine de l'arbre sont étiquetées par la même forme. Plusieurs arbres vérifiant cette condition peuvent exister et il peut donc y avoir plusieurs possibilités de fusion.

La figure (fig. 5.4) illustre ceci sur l'exemple précédent. L'arbre représentant le dossier du fauteuil est fusionné avec l'arbre représentant le fauteuil où le dossier est manquant.

5.4 Les actions d'initialisation

Du paragraphe précédent il ressort que l'interprétation fait intervenir trois types de choix :

- les *choix ascendants* dans lesquels on choisit une règle pour effectuer une phase ascendante.
- les *choix descendants* consistant à choisir une règle pour effectuer une phase descendante.
- les *choix de fusion* consistant à choisir un arbre parmi tous les arbres pouvant être fusionnés avec une certaine feuille de la description.

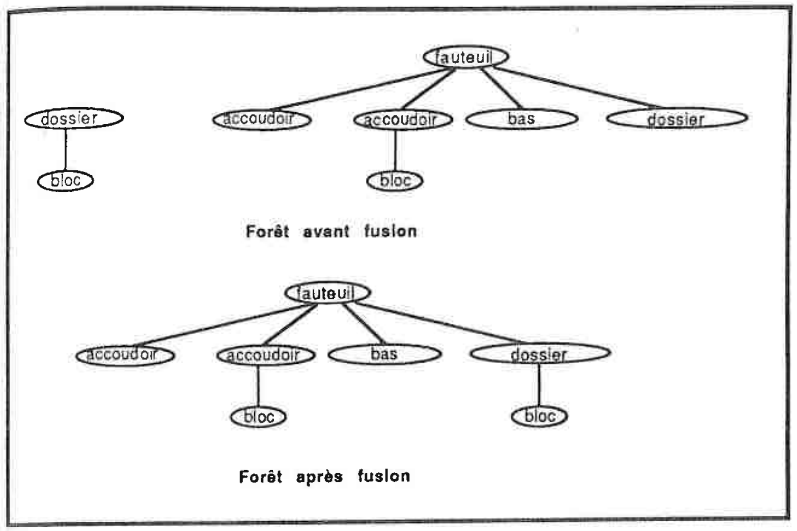


Figure 5.4. Fusion de deux arbres.

Pour chacun de ces types de choix il existe une action d'initialisation construisant l'ensemble des choix possibles pour effectuer l'action correspondante.

5.4.1 Initialisation d'une phase ascendante

Initialiser une phase ascendante pour un nœud N (nécessairement racine d'un arbre) consiste à définir un premier ensemble de règles qui pourront être utilisées en phase ascendante sur N. Cet ensemble, noté $ASC(N)$, contient au début toutes les règles du modèle compatibles avec les instances d'attribut de N et dont le membre droit contient la forme f représentée par N.

Pour vérifier la compatibilité des contraintes d'une règle avec les instances d'attribut de N, on ne tient compte que des contraintes de la forme $f.a = (v_1, \ldots, v_n)$. La règle est compatible, si pour chaque contrainte de ce type on a :

$$D(N.a) \cap \{v_1, \ldots, v_n\} \neq \emptyset$$

Supposons, par exemple, que la description contienne un arbre de racine N représentant un bloc dont la hauteur maximale est 80 cm ($zmax.bloc = (80)$). De plus, supposons que $ASC(N)$ contienne les règles suivantes :

1. *armoire* → *bloc*, avec la contrainte $zmax.bloc = (160\ 320)$;
2. *plateau table* → *bloc*, avec $zmax.bloc = (80)$;
3. *pied table* → *bloc*, avec $zmax.bloc = (80)$;
4. *plateau-chaise* → *bloc*, avec $zmax.bloc = (40)$.

Dans ce cas, les règles retenues dans $ASC(N)$ seront les règles 2 et 3.

5.4.2 Initialisation d'une phase descendante

L'initialisation d'une phase descendante suit le même principe. Il s'agit cette fois de définir les possibilités initiales de phases descendantes pour une feuille F. L'ensemble des règles possibles, noté $DES(F)$ contient les règles vérifiant les deux conditions suivantes :

- le membre gauche est la forme f étiquetant F,
- les contraintes de la forme $f.a = (v_1, \ldots v_n)$ sont compatibles avec les instances d'attribut de F.

5.4.3 Initialisation des fusions

Une feuille non terminale de la description peut être fusionnée avec tous les arbres dont la racine est étiquetée par la même forme qu'elle. Lorsqu'une nouvelle feuille F apparaît dans la description on lui associe donc un ensemble noté $FUS(F)$ contenant toutes les racines pouvant être fusionnées avec elle. Ce sont toutes les racines étiquetées par la même forme f que F et compatibles avec F par leurs propriétés. Une racine R est compatible avec F si pour chaque attribut a de f on a :

$$D(F.a) \cap D(R.a) \neq \emptyset$$

Autrement dit, il doit exister au moins une valeur commune pour chaque instance d'attribut correspondant au même attribut de f.

De même, lorsqu'une nouvelle racine R apparaît, on lui associe, selon le même principe, l'ensemble $FUS(R)$ de toutes les feuilles pouvant être fusionnées avec elle.

5.5 Les actions de propagation

Les actions de propagation éliminent des hypothèses et influencent par là les possibilités d'évolution de la description. Il y en a deux :

- la propagation syntaxique qui élimine des possibilités de choix de règles (ascendants ou descendants) dans les ensembles $ASC(N)$ et $DES(N)$ associés aux nœuds de la description,
- la propagation des contraintes qui élimine des valeurs dans les listes de valeurs associées aux instances d'attribut. Etant donné que l'initialisation des phases ascendantes, des phases descendantes et de la fusion dépend directement des attributs, cette action influence toutes les possibilités de choix.

Ces deux actions vérifient en même temps la cohérence de la description. La *cohérence syntaxique* vérifiée par la propagation syntaxique est la possibilité de construire un arbre unique dérivant du modèle à partir de tous les arbres présents par une suite de phases ascendantes, descendantes et de fusions. La *cohérence des attributs*, vérifiée par la propagation des contraintes, est la possibilité de donner une valeur à chaque attribut de manière à ce que toutes les contraintes soient vérifiées.

La vérification de la cohérence syntaxique et la vérification de la cohérence des attributs sont des cas particuliers du problème d'étiquetage consistant d'un graphe [Mon 74] [Ros 76] [Har 78] [Mac 85].

5.5.1 Etiquetage consistant d'un graphe

Le problème d'étiquetage consistant d'un graphe G peut être posé comme suit :

- soit N et E deux ensembles représentant respectivement l'ensemble des noeuds de G et l'ensemble des étiquettes,
- soit A un sous-ensemble de $N \times N$ représentant l'ensemble des arcs de G.

Chaque nœud i est associé à un ensemble d'étiquettes possibles $E(i) \subset E$ et chaque arc (i, j) est associé à un prédicat binaire P_{ij} allant de $E \times E$ dans $\{vrai, faux\}$.

Le problème est de trouver pour chaque nœud i une étiquette $e_i \in E(i)$ de telle sorte que tous les prédicats soient vérifiés, c.a.d :

$$\forall (i, j) \in A.\ P_{ij}(e_i, e_j)$$

Dans le cas de la cohérence syntaxique :

- N est l'ensemble des nœuds feuilles ou racines de la description,
- $A = N \times N$,
- l'ensemble $E(i)$ des étiquettes possibles d'un nœud i est $ASC(i)$ si i est une racine et $DES(i)$ si i est une feuille.

- pour $e_i \in E(i)$ et $e_j \in E(j)$, $P_{ij}(e_i, e_j)$ est vrai si et seulement si e_i et e_j sont des règles compatibles.

Dans le cas de la cohérence des attributs :

- N est l'ensemble des instances d'attribut,
- A est l'ensemble des couples (i, j) d'instances d'attribut liés par une contrainte,
- l'ensemble $E(i)$ des étiquettes possibles d'un nœud i est $D(i)$, c'est-à-dire le domaine de valeurs associé à l'instance d'attribut i,
- pour $e_i \in E(i)$ et $e_j \in E(j)$, $P_{ij}(e_i, e_j)$ est vrai si et seulement si e_i et e_j vérifient toutes les contraintes liant i et j.

5.5.2 L'algorithme de consistance d'arcs

Pour prouver la validité de la description, il faudrait théoriquement exhiber un étiquetage consistant dans les deux cas, mais la recherche d'un étiquetage consistant demande l'exploration d'un très grand nombre d'étiquetages (la seule solution est d'utiliser les algorithmes de recherche classiques de coût exponentiel). Par conséquent, au lieu de montrer la consistance globale de la description en exhibant une solution, nous nous sommes contentés d'un algorithme permettant de vérifier une consistance locale.

Il s'agit de l'algorithme AC4 de Mohr [Moh 85] qui vérifie l'existence d'un étiquetage consistant par arc: pour chaque nœud i du graphe, pour chaque étiquette $e_i \in E(i)$ et pour chaque nœud j, voisin de i, il doit exister au moins une étiquette e_j de $E(j)$ compatible avec e_i. S'il n'existe pas une telle étiquette, e_i peut être supprimée.

On vérifie donc simplement une *consistance locale* contrairement au problème d'étiquetage consistant d'un graphe dans lequel on recherche une *solution globale*. L'optimalité de AC4 pour ce problème a été démontrée. Le coût est en $O(ae^2)$, a étant le nombre d'arcs et e le nombre maximal d'étiquettes par nœud.

L'idée de base est d'associer un compteur noté C_{ijx} à chaque couple arc-étiquette. Ce compteur représente le nombre d'étiquettes de $E(j)$ compatibles avec l'étiquette x pour le nœud i. De plus, on mémorise l'ensemble $S(j, y)$ de tous les couples nœud-étiquette (i, x) compatibles avec (j, y).

Ces ensembles servent à retrouver les compteurs C_{ijx} qui doivent être décrémentés de 1 lorsque y est éliminé au nœud j.

L'algorithme se déroule en deux étapes. La première étape sert simplement à initialiser les compteurs et construire les ensembles $S(j, y)$. Une liste L, initialement vide, enregistre tous les couples nœud-étiquette devant être supprimés. L'algorithme est donné par la figure 5.5.

Pour $(i, j) \in A$ **faire**
  Pour $x \in E(i)$ **faire**
    Pour $y \in E(j)$ **faire**
      Si $P_{ij}(x, y)$ **Alors**
        $C_{ijx} \leftarrow C_{ijx} + 1$
        $S(j, y) \leftarrow S(j, y) \cup \{(i, x)\}$
      Fin_si
    Fin_pour
    Si $C_{ijx} = 0$ **Alors**
      $L = L \cup \{(i, x)\}$
    Fin_si
  Fin_pour
Fin_pour

Figure 5.5. Étape d'initialisation dans AC4

La mise à jour des ensembles d'étiquettes associés aux nœuds, a lieu dans la deuxième étape. Lorsqu'un compteur C_{ikx} est nul, il n'existe plus d'étiquette de $E(k)$ compatible avec l'étiquette x pour le nœud i. Il faut donc supprimer x de $E(i)$ puis signaler à tous les nœuds ayant une étiquette compatible avec x que celle-ci n'existe plus. Autrement dit, il faut décrémenter de 1 le compteur $[(j, i), y]$ pour tout couple $(j, y) \in S(i, x)$. Cette opération est répétée chaque fois qu'un nouveau compteur passe à zéro. Si un ensemble $E(i)$ devient vide, on a prouvé l'inconsistance du graphe.

L'algorithme est présenté en figure 5.6 sous la forme d'une fonction booléenne prenant en argument une liste L de couple nœud-étiquette à supprimer et retournant vrai si et seulement si le graphe est arc consistant.

Nous ne donnons pas dans cette thèse l'adaptation de AC4 à la vérification de la cohérence syntaxique et de la cohérence des attributs. L'algorithme de propagation de contraintes conçu par G. Masini peut être trouvée dans [Bel 86, 5.15-5.28].

5.5.3 Utilisation des actions de propagation

Les actions de propagation sont utilisées chaque fois qu'il y a modification du graphe G. La propagation syntaxique est donc appelée lorsque de nouveaux ensembles de règles apparaissent (après initialisation d'une phase ascendante ou descendante) ou après une phase ascendante ou descendante. Dans ce dernier cas on considère que l'ensemble de règles correspondant est réduit à la règle choisie.

De même, la propagation des contraintes est exécutée dès que de nouvelles contraintes

```
Fonction AC4 (L)
 consistance = vrai
 Tant_que L ≠ ∅ et consistance faire
  L = L − {(i, x)}
  E(i) = E(i) − {x}
  Si E(i) = ∅ Alors
   consistance ← faux
  Sinon
   Pour (j, y) ∈ S(i, x) faire
    Si y ∈ E(j) Alors
     C_jiy ← C_jiy − 1
     Si C_jiy = 0 Alors
      E(j) ← E(j) − {y}
      L = L ∪ {(j, y)}
     Fin_si
    Fin_si
   Fin_pour
  Fin_si
 Fin_tant_que
 Retourner consistance
Fin_fonction
```

Figure 5.6. Algorithme de propagation discrète AC4.

ou de nouveaux domaines de valeurs apparaissent. Plus exactement, trois actions demandent une propagation de contraintes :

- lorsqu'il y a fusion d'une feuille F avec un arbre de racine R, les instances d'attribut associées à F et R doivent être égaux. Les valeurs possibles appartiennent donc nécessairement à l'intersection des domaines de valeurs correspondant. Par conséquent, pour chaque attribut a de la forme f étiquetant F et R, on propage la contrainte $R.a = D(R.a) \cap D(F.a)$:
- en cas de phase ascendante ou descendante, les domaines de valeurs des instances d'attribut $N.a_1, \ldots, N.a_n$ de chaque nouveau nœud N sont initialisés par les valeurs possibles du type correspondant, c'est à dire :
  - $(0, 45, 90)$ pour les instances d'attribut du type angle,
  - $(0, 10, 20, 40, \ldots, 640)$ pour les instances d'attribut du type dimension.

  Ensuite, il y a propagation de chaque contrainte associée à la règle utilisée. Par exemple, supposons que la règle suivante soit utilisée :

$$scène \text{ — } armoire\ bureau\ chaise$$
$$zmax.armoire \geq zmax.bureau$$
$$zmax.bureau = (80)$$

Si l'armoire est instanciée par le nœud N_1 et le bureau par le nœud N_2, on propage les contraintes :

$$N_1.zmax \geq N_2.zmax$$
$$N_2.zmax = (80)$$

5.5.4 Utilité des actions de propagation

Les actions de propagation ont deux intérêts principaux. Le premier est la réduction de l'indéterminisme. La propagation syntaxique élimine des choix de règle et la propagation des contraintes réduit indirectement les possibilités de choix de règles et de fusion en supprimant des valeurs dans les domaines de valeurs des instances d'attribut.

Le second intérêt est la possibilité de détecter des incohérences dans la description. Celles-ci sont signalées à l'interpréteur par des *échecs*. Ce sont des variables générées dynamiquement et véhiculant un message d'erreur. Deux types d'échecs sont possibles :

- les *échecs syntaxiques*, engendrés par la propagation syntaxique, signalent la présence d'un ensemble de règles vide en précisant le nœud associé à cet ensemble.
- les *échecs liés aux attributs*, créés par la propagation des contraintes, informent l'interpréteur de l'existence d'un domaine de valeurs vide et précisent l'instance d'attribut correspondant.

5.6 Les actions de gestion des erreurs

Actuellement, il n'existe qu'une seule action de gestion des erreurs : la suspension d'une analyse. Celle ci consiste simplement à stopper la croissance d'un arbre en marquant tous ses nœuds. En cas d'échec, l'interpréteur renonce à interpréter une partie de la vue.

5.7 Enchaînement possible des actions

Nous avons vu au paragraphe 5.2 que chaque action "demande" l'exécution de certaines actions qui peuvent la suivre ou les plaçant dans une file d'attente.

Les actions demandées par une action A sont toutes les actions pouvant agir sur les nouveaux objets produits par A (nœuds, arbres, attributs, contraintes, échecs ...). Le tableau 5.1 résume les objets nécessaires et les objets produits par chaque action. La figure 5.7 montre les demandes qui en résultent sous la forme d'un graphe.

Action	Objets nécessaires	Objets produits
Démarrer-analyse	face de la vue	racine
Initialiser phase ascendante	racine	racine avec choix de règles
Initialiser phase descendante	feuille	feuille avec choix de règles
Initialiser choix fusion	racine ou feuille	nœud avec choix de fusions
Phase ascendante	racine avec choix de règles	racine, feuilles, contraintes
Phase descendante	feuille avec choix de règles	feuilles, contraintes
Fusion	nœud avec choix de fusions	contraintes
Propagation de contraintes	contraintes	echec éventuel
Propagation syntaxique	ensemble de règles	échec éventuel
Stopper-analyse	échec	

Tableau 5.1. Objets nécessaires à chaque action et produits par chaque action

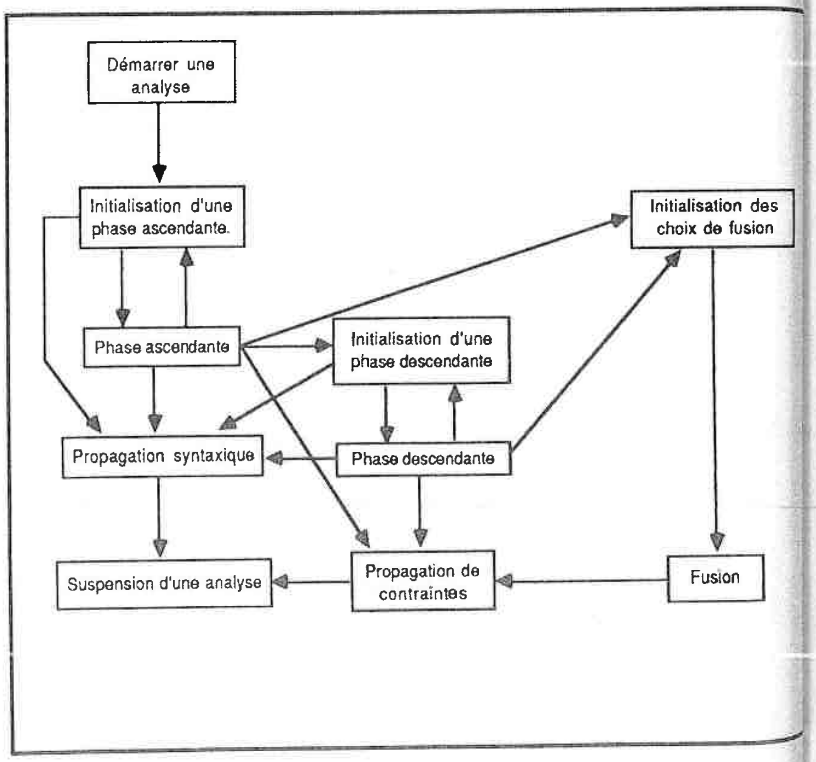


Figure 5.7. Demandes des actions. Une flèche relie deux actions A_1 et A_2 si A_2 est demandée par A_1.

Action	Objets prédéfinis	Objets choisis
Démarrer-analyse	face de la vue	
Initialiser phase ascendante	racine	
Initialiser phase descendante	feuille	
Initialiser choix fusion	racine ou feuille	
Phase ascendante	racine	règle
Phase descendante	feuille	règle
Fusion	nœud	nœud
Propagation de contraintes	ensemble des contraintes existantes	
Propagation syntaxique	ensembles de règles existants	
Stopper-analyse		arbre

Tableau 5.2. Tableau résumant les objets prédéfinis et les objets choisis pour chaque action.

Après avoir sélectionné une action, l'interpréteur choisit les objets sur lesquels celle-ci devra agir. Une partie de ces objets sont définis à l'avance par les demandes de chaque action. Par exemple, l'initialisation d'une phase ascendante pour une racine R demandera l'exécution d'une phase ascendante sur R. Seule la règle à utiliser pour cette action fera l'objet d'un choix, défini par la fonction de support de l'action phase ascendante. Cette manière de procéder évite de rechercher à chaque itération tous les paramètres d'une action.

La table 5.2 résume les différents objets associés aux actions, en distinguant ceux qui sont déterminés à l'avance (objets prédéfinis) de ceux qui sont déterminés par les fonctions de support (objets choisis).

5.8 Principes de sélection

Les principes de sélection d'une action sont contenus dans la fonction de sélection qui établit, à chaque itération, une priorité sur les actions appartenant à la file d'attente.

5.8.1 Priorités des différents types d'action

Rappelons qu'il existe quatre types d'action. Ceux-ci sont ordonnés par priorités décroissantes de la manière suivante :

1. les actions de gestion des échecs,
2. les actions de propagation,
3. les actions d'initialisation.

4. les actions de construction.

Ces priorités s'expliquent par des principes stratégiques qui relèvent simplement du "bon sens" :

- *Retardement des choix* : les actions sans risque d'erreur sont prioritaires. Or les seules actions impliquant des choix importants sont les actions de construction. D'où la priorité minimale qui leur est associée. L'intérêt de ce principe de sélection est que les actions déterministes, comme les actions de propagation, peuvent éventuellement réduire les possibilités de choix associées aux actions de construction en attente et diminuer ainsi au maximum les risques d'erreurs. Il y a toutefois une exception pour l'action *démarrer une analyse* qui est une action de construction sans possibilité d'erreur. En effet, pour chaque instance de face la forme primitive associée est unique. Cette action est donc placée au niveau de priorité 3 avec les actions d'initialisation.

- *Prise en compte immédiate des échecs* : lorsqu'un échec se produit, il faut absolument éviter que les effets de l'erreur se propagent. Un mauvais choix de règle peut par exemple supprimer des choix de règles corrects par propagation syntaxique ou induire de fausses valeurs d'attributs par propagation des contraintes. Les actions les plus prioritaires sont donc les actions de prise en compte des échecs (stopper une analyse).

- *Détection immédiate des erreurs* : les dommages impliquées par une erreur sont minimisés si celle-ci est détectée le plus rapidement possible. Par conséquent, les actions de propagation sont placées en deuxième position dans l'ordre des priorités. Cela permet également, comme nous l'avons déjà souligné, de réduire au minimum les possibilités de choix avant d'entreprendre toute autre action.

5.8.2 Sélection des actions de construction

S'il ne reste plus que des actions de construction dans la file d'attente, la règle de sélection est la suivante :

1. *Priorité maximale aux fusions* pour résoudre le conflit entre phase ascendante, phase descendante et fusion. En effet, lorsqu'une fusion est possible entre une feuille F et une racine R (cf. exemple de la figure 5.8), il est également possible d'effectuer une phase ascendante sur R ou une phase descendante sur F.

   Or, si l'on fusionne F et R, les deux autres actions deviennent impossibles et réciproquement. Il s'agit en fait de décider si les nœuds F et R instancient le même objet (la forme A dans la figure 5.8), ou s'ils représentent deux occurrences distinctes de cet objet. Ce problème est délicat. En choisissant la fusion, on décide que l'on a une seule occurrence de l'objet, alors que dans le cas contraire, on décide qu'il y en a deux. La priorité maximale accordée à la fusion règle le conflit mais elle ne résoud pas réellement le problème. Il est évident que la fusion n'est pas toujours la bonne solution.

2. *Priorité aux actions de choix minimal* :

   - S'il y a plusieurs actions de fusion en attente, on choisit celle qui offre le moins de possibilités.
   - S'il ne reste que des actions impliquant un choix de règle (phases ascendantes et descendantes) on choisit l'action qui a une probabilité de succès maximale ou autrement dit, celle pour laquelle le nombre de règles possibles est le plus réduit.

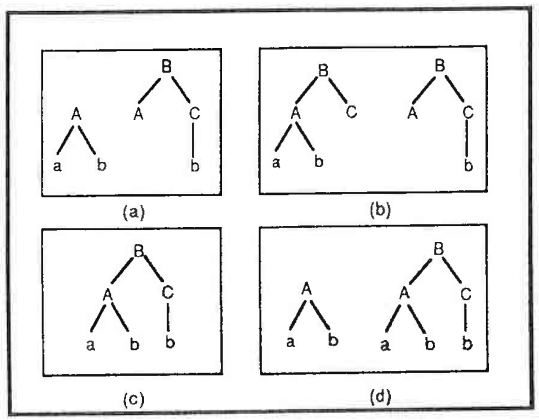


Figure 5.8. Conflit des phases ascendantes et descendantes avec la fusion. (a) Situation initiale. (b) Résultat après phase ascendante. (c) Résultat après fusion. (d) Résultat après phase descendante.

5.9 Ordre de construction de la description

5.9.1 Importance de l'ordre de construction

La description produite par l'interpréteur peut être considérée comme le résultat de l'exécution d'une suite d'actions de construction (démarrer une analyse, effectuer une

phase ascendante, une phase descendante ou une fusion). Une telle suite d'actions définit l'ordre de construction de la description. Il est important de remarquer que pour une même description, il existe un grand nombre d'ordres de construction possibles qui, de plus, n'ont pas la même probabilité de succès.

Illustrons ceci par un exemple. Le modèle (fig. 5.9) représente une forme X par trois règles. Celle-ci est constituée de deux faces rectangulaires (*face1 et face2*), la largeur de la première face étant inférieure à celle de la seconde face.

1. $bloc \rightarrow face1\ face2$
   $face1.largeur \leq face2.largeur$
2. $face1 \rightarrow RF$
   $face1.largeur = RF.l$
3. $face2 \rightarrow rectangle$
   $face2.largeur = RF.l$

Figure 5.9. Modèle pour illustrer l'importance de l'ordre de construction

La figure 5.10 illustre la description obtenue à partir de ce modèle. Montrons qu'il existe deux ordres de construction de cette description n'ayant pas la même probabilité de réussite. Nous donnons uniquement les probabilités de succès (en supposant l'équiprobabilité de chaque règle) pour les actions à risque d'erreur :

- Première solution :
  1. Démarrer une analyse sur la face $F1$ en produisant le nœud $N1$.
  2. Démarrer une analyse sur la face $F2$ en produisant le nœud $N2$.
  3. Effectuer une phase ascendante sur $N1$ avec la règle 2. Probabilité de succès : 1/2.
  4. Effectuer une phase ascendante sur $N3$ avec la règle 1.
  5. Effectuer une phase descendante sur $N4$ avec la règle 3.
  6. Effectuer une fusion entre le nœud (absent de la description finale) fils de $N4$, étiqueté par la forme RF et le nœud $N2$.

  Si l'on effectue le produit des probabilités, la probabilité totale de succès est 1/2.

- Deuxième solution :
  1. Démarrer une analyse sur la face $F1$ en produisant le nœud $N1$.
  2. Démarrer une analyse sur la face $F2$ en produisant le nœud $N2$.

  3. Effectuer une phase ascendante sur $N1$ avec la règle 2 en produisant le nœud $N3$. Probabilité de succès : 1/2.
  4. Effectuer une phase ascendante sur $N2$ avec la règle 3 en produisant le nœud $N4$. Probabilité de succès : 1/2 (en supposant que la règle 2 n'est pas exclue par propagation syntaxique).
  5. Effectuer une phase ascendante sur $N3$ avec la règle 1.
  6. Effectuer une fusion entre le nœud (absent dans la description finale) fils de $N5$, étiqueté par la forme *face2* et le nœud $N4$.

  Dans ce cas le produit des probabilités est 1/4.

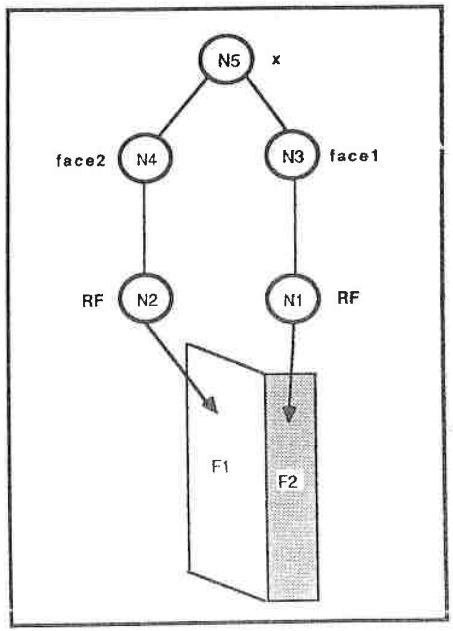


Figure 5.10. Description obtenue avec le modèle de la figure 5.9

La différence de probabilité ent du fait qu'il n'est pas possible de décider de manière locale quelle est la face la moins large. Or dans la deuxième solution, cette décision locale est prise deux fois, au lieu d'une dans la première solution.

5.9.2 Différentes approches possibles

Concernant l'ordre de construction, différentes approches sont possibles. On oppose classiquement :

- *l'approche descendante* qui consisterait dans notre cas à générer un premier nœud instanciant la forme axiome, puis à construire la description par une suite de phases descendantes Une approche totalement descendante n'est pas possible dans notre cas car les premiers nœuds générés sont nécessairement des nœuds terminaux.
- *l'approche ascendante* dans laquelle on autorise uniquement des phases ascendantes et des fusions. Nous pouvons interpréter une vue de cette manière en définissant la fonction de contrôle de la phase descendante de telle sorte qu'elle rende toujours la valeur *faux*. On peut remarquer que ce type de construction implique l'existence de plusieurs arbres déconnectés, contrairement à la méthode descendante.
- *l'approche ascendante/descendante* dans laquelle plusieurs arbres sont développés parallèlement et peuvent être complétés de manière ascendante ou descendante selon le choix de l'interpréteur. Cette méthode est la plus générale puisque les deux méthodes précédentes en sont des cas particuliers.

5.9.3 Notre approche

Le fait de définir *a priori* le type d'interprétation en interdisant certaines actions limite les chances de succès. L'exemple du paragraphe 5.9.1 le montre. Dans ce cas, en imposant une démarche ascendante (deuxième solution) les chances de succès seraient limitées à 1/4. L'approche ascendante/descendante est donc préférable. En laissant le libre choix des actions de constructions à l'interpréteur, celui-ci a toujours la possibilité de trouver la suite d'actions la plus sûre. Par conséquent, il n'y a pas d'avantages à imposer des contraintes sur l'ordre de construction de la description. Nous verrons au chapitre 6. que ce principe de "liberté de construction" est confirmé pas les résultats expérimentaux.

5.10 Conclusion

5.10.1 Le problème du contrôle

Le contrôle de la description par la compatibilité des règles n'est pas toujours suffisant. La figure 5.12 montre par exemple une description incohérente (relativement au modèle de la figure 5.11) qui ne peut être détectée par notre méthode de propagation syntaxique. En effet, les trois arbres A_1, A_2 et A_3 sont compatibles deux à deux mais ils ne peuvent exister simultanément dans une même description. Or, la propagation syntaxique ne vérifie que

la compatibilité d'ordre deux et ne produira donc pas d'échec dans ce cas. Bien que la vérification effective de cohérence syntaxique (existence d'un étiquetage consistant) ne soit pas souhaitable car elle demanderait trop de calculs, il serait intéressant d'introduire quelques possibilités de vérification supplémentaires. Par exemple, lors de la compilation du modèle on pourrait calculer le nombre maximal d'occurrences de chaque forme et de chaque règle dans une même description. Cette information supplémentaire permettrait de détecter des incohérences syntaxiques analogues à celle de l'exemple précédent.

1. $F \rightarrow F1\ F1$
2. $F \rightarrow F1\ F2$
3. $F \rightarrow F2\ F3$
4. $F \rightarrow F1\ F3$
5. $F1 \rightarrow A\ B$
6. $F2 \rightarrow A\ B$
7. $F3 \rightarrow c$
8. $A \rightarrow a$
9. $B \rightarrow b$

Figure 5.11. Modèle pour illustrer l'insuffisance de la propagation syntaxique.

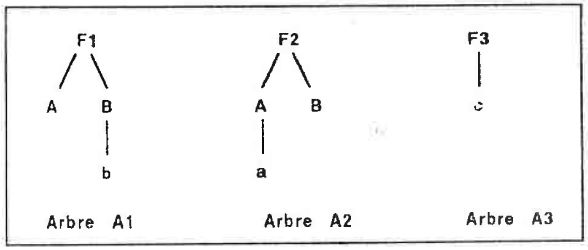


Figure 5.12. Incompatibilité non détectée par l'exclusion des règles

5.10.2 Questions non résolues

Un certain nombre de questions demeurent pour l'instant non résolues :

- Il manque des critères fiables ou à défaut des heuristiques pour guider le choix des paramètres des actions (ces critères sont en principe codés dans les fonctions de support des actions) :

- choix de l'arbre à fusionner avec une feuille dans le cas où plusieurs arbres sont possibles.
- choix de la règle à utiliser en phase ascendante ou descendante.
- choix de l'analyse à stopper en cas d'échec. Ce problème est très délicat. Pour l'instant, l'analyse suspendue est celle où s'est produit l'échec. Dans le cas où l'échec est un ensemble vide de choix de règles pour un certain nœud, l'analyse suspendue est celle de l'arbre contenant ce nœud. De même, lorsqu'il s'agit d'un domaine de valeurs vide pour un certain attribut d'un certain nœud (échec consécutif à une propagation de contraintes), on stoppe la croissance de l'arbre correspondant.

  En réalité, il faudrait stopper la croissance de l'arbre dans lequel s'est produit le mauvais choix de règle ou de fusion à l'origine de l'échec. Mais il est difficile de retrouver ce choix. La difficulté vient du fait qu'un mauvais choix peut en entraîner d'autres.

- lorsqu'une une racine d'un certain arbre et une feuille d'un autre arbre sont étiquetées par la même forme f, la fusion n'est possible que si ces deux nœuds représentent effectivement la même occurrence physique de la forme f. Or, nous ne savons pas comment vérifier ceci (cf. paragraphe 5.8.2).

6
Résultats expérimentaux

Les résultats expérimentaux décrits ici ont été obtenus sur douze vues de différentes scènes d'intérieur synthétiques. Le modèle utilisé est donné dans l'annexe A.

6.1 Résultats obtenus avec différentes stratégies

Rappelons le principe de "liberté de construction" énoncé au paragraphe 5.9.3 : ne pas imposer de contraintes sur l'ordre de construction de la description pour ne pas interdire les suites de choix de probabilité de succès maximale. Pour vérifier ce principe, nous avons effectué des essais avec les stratégies suivantes :

- Stratégie 1 : ascendante/descendante en parallèle.
- Stratégie 2 : analyse totalement ascendante et parallèle.
- Stratégies 3 et 4 : ces deux stratégies sont analogues à la stratégie 1, excepté le fait que le degré de parallélisme est plus faible, c'est-à-dire que les choix ascendants ne sont autorisés que sur certains arbres privilégiés. Pour cela, avant de démarrer l'interprétation, on retient une proportion p de faces, qui seront considérées comme les meilleures faces de départ. Le critère retenu pour désigner ces faces est la dimension des domaines de valeurs. Afin de réduire l'indéterminisme des premiers choix ascendants, les meilleures faces sont donc celles qui minimisent la somme du nombre de valeurs possibles pour chaque attribut. Pendant l'interprétation les arbres privilégiés sont ceux qui sont instanciés par au moins une des meilleures faces.

  La stratégie 3, à un degré de parallélisme de 25% ($p = 0.25$), alors que le degré de parallélisme de la stratégie 4 est de 50% (remarquez qu'en prenant $p = 1$, on retrouve la stratégie 1).

Les résultats obtenus avec ces différentes stratégies sont évalués par la qualité Q de la description et le degré d'indéterminisme I. Pour mesurer la qualité de la description nous

ne faisons intervenir que les formes représentant les objets les plus importants de la scène (armoires, tables, chaise ...). Pour chaque occurence O_i ($i = 1 \ldots K$) d'un tel objet, nous comparons l'ensemble des faces ER_i qui l'instancient en réalité avec l'ensemble ED_i des faces qui l'instancient dans la description. La qualité de la description est calculée par la formule suivante :

$$Q = \frac{1}{K} \sum_{i=1}^{K} \frac{n_i}{N_i}$$

n_i étant le nombre de faces communes à ED_i et ER_i et N_i, le nombre de faces de E_i.

L'indéterminisme I est le nombre moyen de choix possibles chaque fois qu'une action de construction est exécutée

Q varie entre 0 (reconnaissance nulle) et 1 (reconnaissance parfaite) et I est compris entre 1 (analyse totalement déterministe) et $+\infty$.

L'indéterminisme et le taux de reconnaissance sont optimaux pour la stratégie la moins contrainte, c'est-à-dire la stratégie ascendante/descendante sans réduction du parallélisme (stratégie 1, $Q = 0.95$ et $I = 1.5$). Ils sont nettement moins satisfaisants pour l'analyse totalement ascendante (stratégie 2, $Q = 0.78$ et $I = 2.2$).

D'autre part, on constate que l'indéterminisme augmente et que le taux de reconnaissance diminue, lorsque le degré de parallélisme diminue également. Ceci est illustré par les résultats associés aux stratégie 1, 4 et 3 :

- pour la stratégie 1 le parallélisme est de 100 %, $Q = 0.95$ et $I = 1.5$.
- pour la stratégie 4 le parallélisme est de 50 %, $Q = 0.84$ et $I = 1.5$.
- pour la stratégie 3 le parallélisme est de 25 %, $Q = 0.5$ et $I = 1.7$.

Ceci confirme encore une fois le principe de liberté de construction, puisque la stratégie 3 est un cas particulier de la stratégie 4, qui est elle même un cas particulier de la stratégie 1.

6.2 Résultats détaillés pour la stratégie 1

6.2.1 Résultats numériques

Le tableau 6.1 donne des informations détaillées sur l'interprétation des douze vues (illustrées par les figures 6.1 ... 6.12) avec la stratégie 1. Les valeurs données dans ce tableau (excepté le taux de reconnaissance et le degré d'indéterminisme) sont du même ordre pour les autres stratégies.

Dans ce tableau, t_1 désigne le temps total d'interprétation en secondes (sur SUN3/260 et en Le_Lisp interpreté), t_2 est le temps utilisé par la propagation des contraintes. F est

vue	t_1	t_2	F	A	N	Q	I
1	68	19	37	2183	369	0.89	1.16
2	63	13	38	2039	353	0.9	1.26
3	163	36	60	2852	486	0.97	1.32
4	249	33	88	3362	565	0.97	1.63
5	211	33	82	3242	545	0.97	1.50
6	224	50	71	3451	599	0.93	1.37
7	233	33	94	3467	613	1	1.61
8	25	6	25	742	148	1	1.99
9	59	13	33	1753	315	0.85	1.51
10	106	20	54	2118	376	1	1.52
11	186	39	68	3930	681	0.95	1.34
12	140	23	67	3708	642	1	1.33

Tableau 6.1. Résultats pour la stratégie 1

le nombre de faces de la vue, A est le nombre d'attributs de la description et N le nombre de noeuds.

6.2.2 Étiquetage des différentes vues

Les figures 6.1 ... 6.12 montrent les étiquetages de chaque vue interprétée avec la stratégie 1. Les étiquettes sont localisées sur le centre de gravité des faces instanciant chaque objet.

6.2.3 Exemple de description structurée

En étiquetant les faces par les formes, il est difficile de visualiser la totalité de la description obtenue. Pour avoir une visualisation plus complète, il est possible d'afficher l'arborescence finale dans sa totalité. La figure 6.14 montre par exemple la description complète de la vue 8. Les arcs obtenus par phase ascendante sont en gras. Ceux qui proviennent de phases descendantes sont en trait simple. Enfin, les arcs doublés d'un trait parallèle correspondent à des fusions. En fait, lors d'une fusion entre un noeud R et une feuille F, les deux noeuds sont conservés avec un pointeur de R sur F. Les arcs correspondant à une fusion visualisent donc ces pointeurs. Les noeuds sont étiquetés par les formes correspondantes, sauf pour les formes primitives et les faces de polyèdres afin de ne pas surcharger la figure. Les feuilles étiquetées Rxx sont les faces de la vue 8 (fig. 6.13).

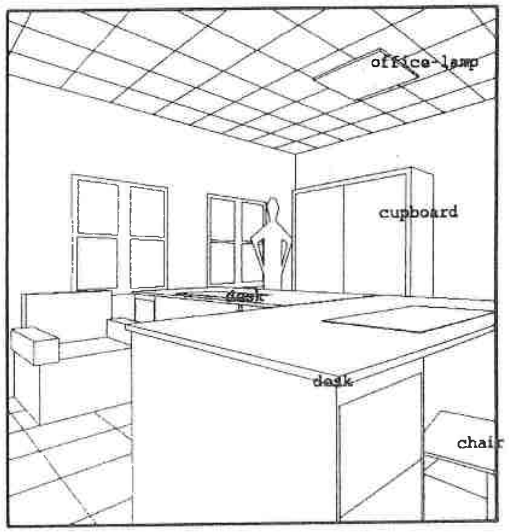


Figure 6.1. Interprétation de la vue 1

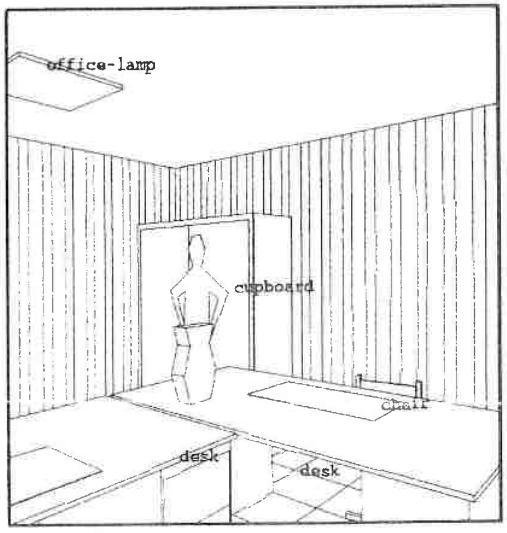


Figure 6.2. Interprétation de la vue 2

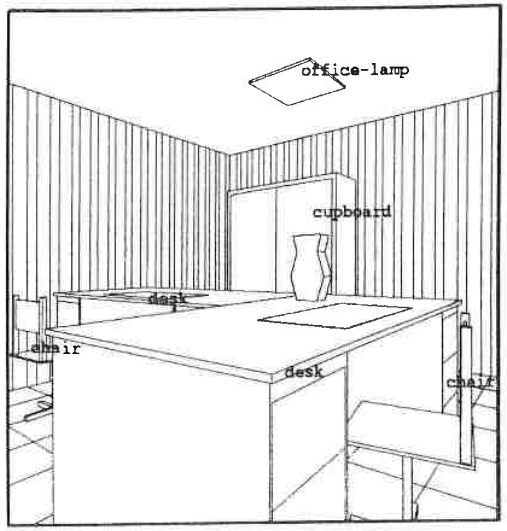


Figure 6.3. Interprétation de la vue 3

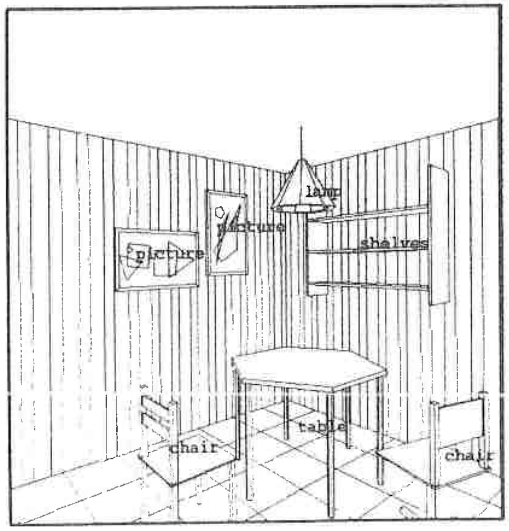


Figure 6.4. Interprétation de la vue 4

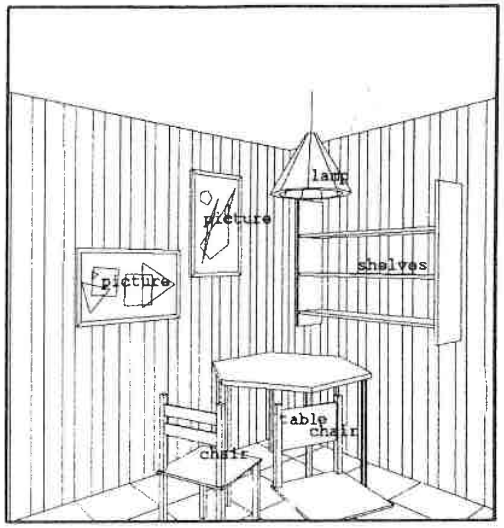


Figure 6.5. Interprétation de la vue 5

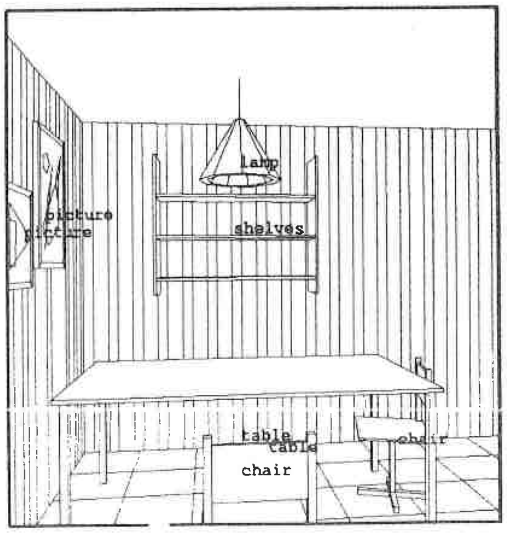


Figure 6.6. Interprétation de la vue 6

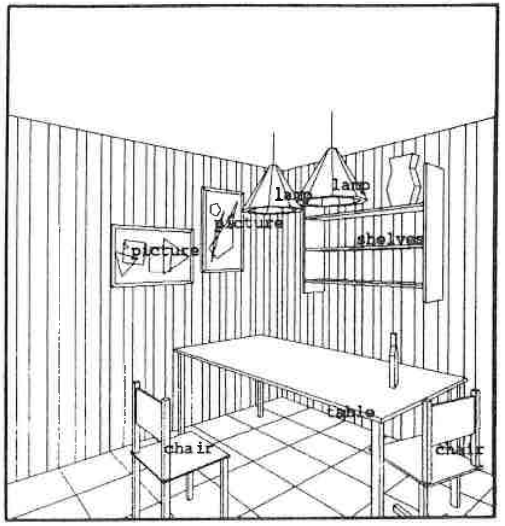


Figure 6.7. Interprétation de la vue 7

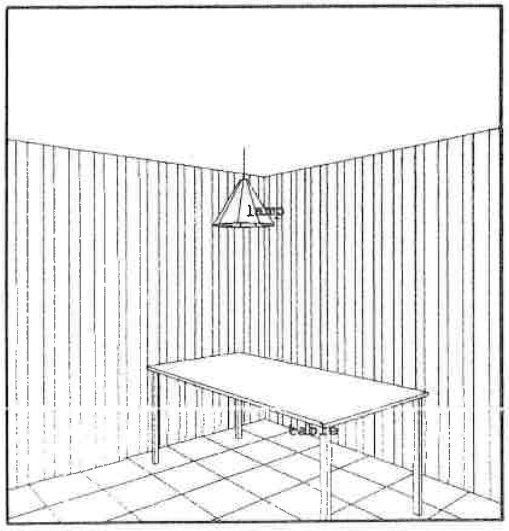


Figure 6.8. Interprétation de la vue 8

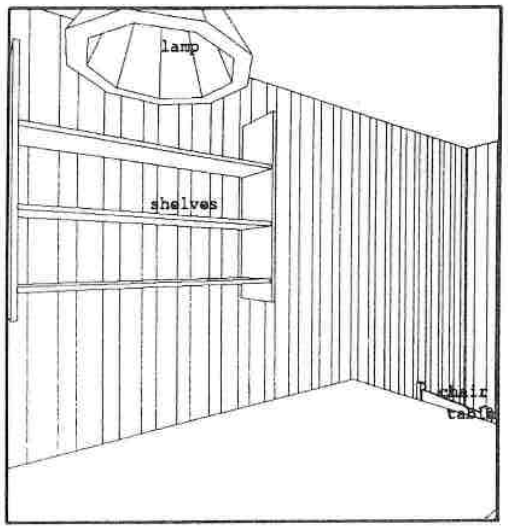


Figure 6.9. Interprétation de la vue 9

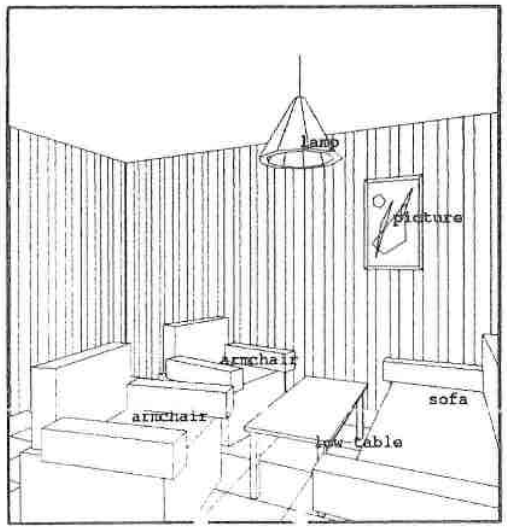


Figure 6.10. Interprétation de la vue 10

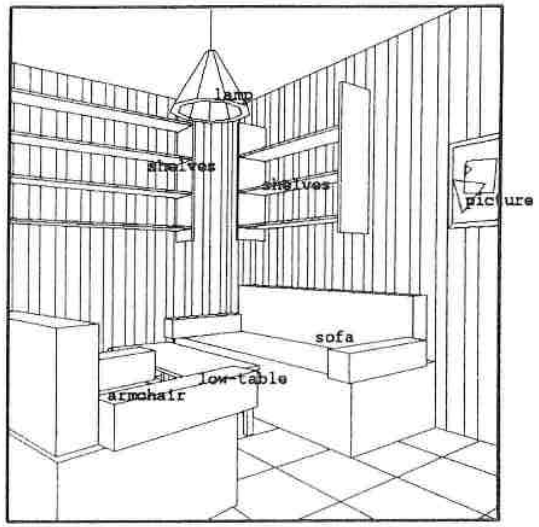


Figure 6.11. Interprétation de la vue 11

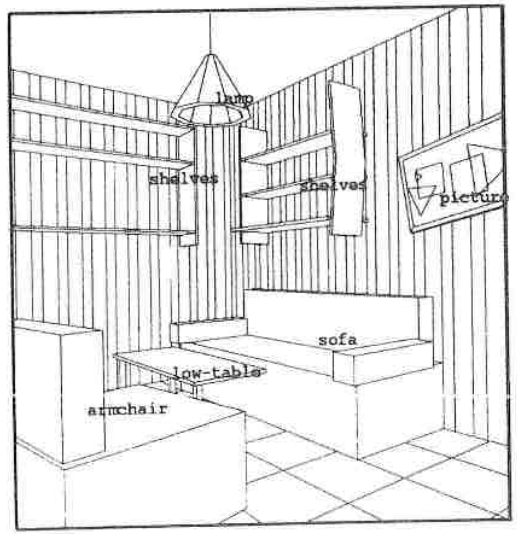


Figure 6.12. Interprétation de la vue 12

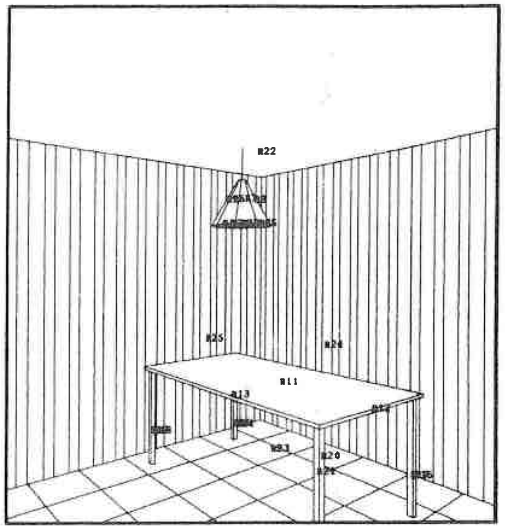


Figure 6.13. Faces de la vue 8

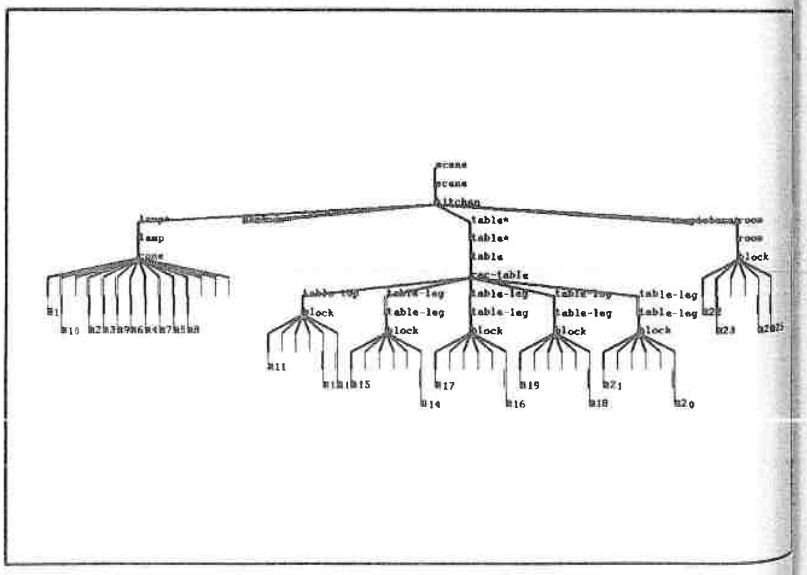


Figure 6.14. Description structurée de la vue 8

6.2.4 Problèmes rencontrés

Les problèmes rencontrés sont de différents types :

Mauvaise instanciation

Les objets ne sont pas toujours bien instanciés. Par exemple, la table identifiée dans la vue de la figure 6.9 est mal instanciée. Les deux barres de dossiers de la chaise sont interprétées comme des pieds de la table (fig. 6.15).

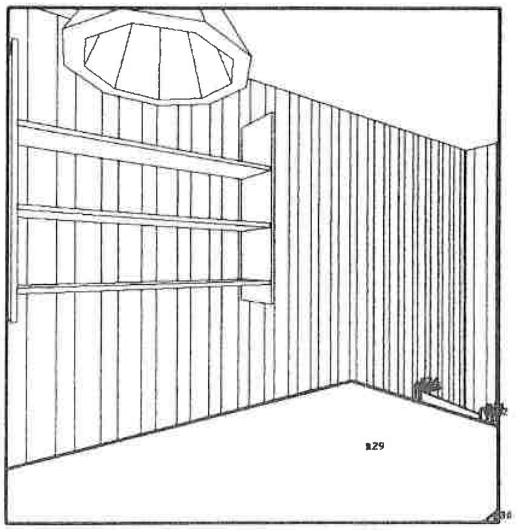


Figure 6.15. Faces instanciant la table dans la vue 9.

Cette erreur est essentiellement due au fait que la représentation d'une table dans le modèle est trop incomplète ou trop peu restrictive (règles des figures 3.9 et 3.11 du chapitre 2). Avec cette représentation, les faces en gras dans la figure 6.15 représentent effectivement une table.

L'erreur serait évitée avec une définition plus fine de la relation $interieur_{xy}$ (cf. paragraphe 2.3.2). Avec la définition actuelle de cette relation, les barres de dossier de la chaise sont bien à l'intérieur du plateau. Malheureusement, une définition plus fine de la relation *à l'intérieur de* est difficile à obtenir avec notre représentation.

Détection d'objets absents

On trouve également des descriptions contenant des objets absents de la scène. Cette erreur se produit dans l'interprétation de la vue 6 (fig. 6.16). L'interpréteur reconnaît deux tables au lieu d'une car les deux barres de dossier de la chaise sont considérées comme les deux pieds d'une table dont le plateau et les deux autres pieds sont invisibles.

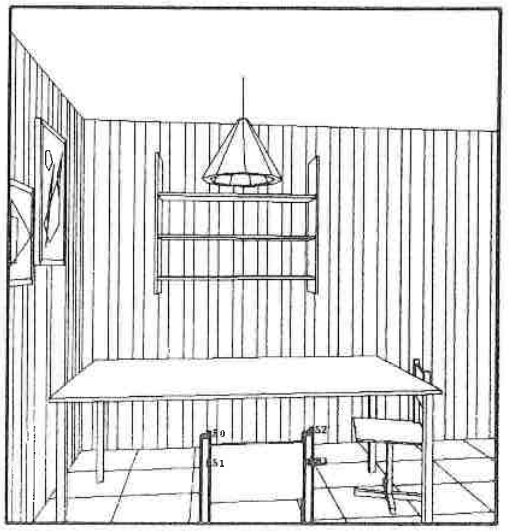


Figure 6.16. Table "parasite" détectée dans la vue 6.

Cette erreur fait apparaître une autre faiblesse du système. Pour l'interpréteur, la non visibilité de certaines parties d'un objet n'est jamais considérée comme une indication d'erreur. Ce principe permet de reconnaître des objets fortement occultés mais d'un autre côté, il réduit le pouvoir de contrôle de la description.

Objets non reconnus

Le contrôle de la cohérence syntaxique peut empêcher la reconnaissance d'objets qui ne se trouvent pas dans leur contexte habituel. Dans la figure 6.1 on voit que le fauteuil n'est pas reconnu. Ceci est du au fait que le modèle exclut la présence d'un fauteuil dans un bureau. La propagation syntaxique a donc éliminé l'hypothèse fauteuil et empêché la reconnaissance de cet objet.

7
Conclusion

7.1 Points faibles

TRIDENT comprend essentiellement deux points faibles :

- un manque de souplesse.
- un contrôle insuffisant.

7.1.1 Manque de souplesse

Le manque de souplesse se retrouve à plusieurs niveaux. Au niveau de la représentation on peut critiquer la forme trop restrictive des contraintes qui nous empêche d'exprimer certaines relations (comme la relation "à l'intérieur de") avec suffisamment de finesse. De plus, la définition discrète des domaines de valeur des attributs implique nécessairement une approximation assez grossière.

Au niveau de l'interprétation, on peut critiquer la "brutalité" de la propagation syntaxique qui empêche de reconnaître des objets lorsqu'ils ne sont pas dans leurs contexte habituel (exemple du fauteuil de la figure 6.1). Dans la réalité, on trouve rarement des objets s'excluant de manière absolu. Il y a surtout des scènes types dans lesquelles on trouve fréquemment les mêmes objets. Le modèle devrait donc plutôt exprimer la probabilité d'observer tel objet lorsqu'on en a déjà reconnu un autre. Pour ce point précis, il serait donc préférable d'utiliser la relaxation continue : associer des probabilités de présence aux objets et faire varier ces probabilités selon le contexte durant l'interprétation.

7.1.2 Contrôle insuffisant

Les erreurs d'interprétation données au chapitre 6 illustrent bien l'insuffisance du contrôle. Ce problème a deux origines :

- notre système de représentation ne permet pas de d'approximer les formes possibles des objets avec suffisament de précision. Autrement dit, les contraintes contenues

dans le modèle sont trop "lâches" pour permettre un bon contrôle.

- les mécanismes de contrôle utilisant les contraintes du modèle (propagation syntaxique et propagation des contraintes) sont limités. En effet, les méthodes de propagation n'exploitent pas totalement les liens contextuels et les liens entre attributs du modèle. La vérification de la compatibilité est toujours locale (la compatibilité des couples d'hypothèses), alors qu'une compatibilité plus globale serait quelquefois souhaitable.

7.2 Points forts

Les points forts du système sont :

- une faible sensibilité à l'occultation (par exemple : chaise de la figure 6.1, chaise de la figure 6.2, chaise de la figure 6.9, table de la figure 6.11).
- une certaine tolérance au bruit confirmée par l'interprétation de la vue 12 (fig. 6.12). Dans cette vue, certaines parties d'objets ont été supprimées (accoudoirs du fauteuil, planche d'étagère, pied de table) et d'autres déformées (coté et planche inférieure de l'étagère de droite), sans empêcher leur reconnaissance. Ceci laisse envisager une extension du système à des données réelles.
- des temps de calculs raisonnables grâce à l'utilisation de AC4 pour la propagation discrète.
- la possibilité d'interpréter des scènes contenant des objets inconnus du système :
  - la forme humaine dans les figures 6.1, 6.2.
  - le vase dans les figures 6.2, 6.3, 6.7.
  - la bouteille sur la table de la figure 6.7.

  Ceci est possible grâce à la suspension locale de l'interprétation en cas d'échec. En outre, les objets non identifiés peuvent être localisés comme le montre la figure 7.1. Il suffit pour cela de rechercher dans la description les arbres stoppés.
- la capacité de reconnaître des objets avec un modèle générique, c'est à dire un modèle qui ne donne pas la forme exacte des objets mais qui leur associe simplement une famille de formes admissibles. Or, il est important de pouvoir exploiter une telle représentation car elle permet de représenter les objets que nous utilisons tous les jours. On constatera aisément que la plupart des objets n'ont pas une forme unique, mais prennent au contraire des formes très variés. La généricité s'impose donc dans

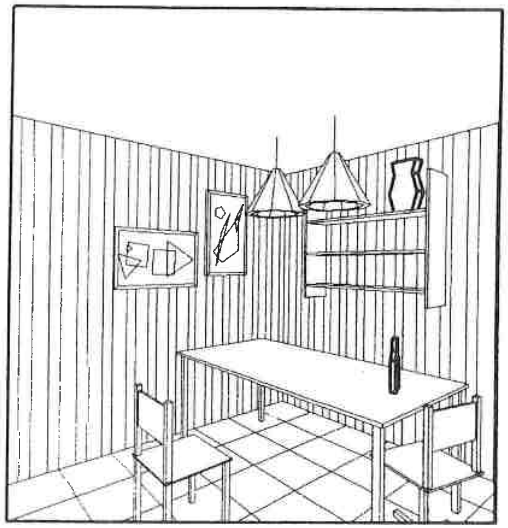


Figure 7.1. Objets non identifiés dans la vue 7.

ce cas, car il serait difficile de représenter toutes les formes possibles de ces objets de manière exacte.

Dans TRIDENT la généricité apparaît à plusieurs niveaux :

- au niveau de la structure (composition en sous-objets) des objets. Le modèle ne donne que les structures possibles des objets et la structure précise d'un objet observée n'est déterminée que lors de l'interprétation.
- au niveau des attributs spécifiques des objets qui sont définis avec une certaine marge de tolérance par les contraintes. Quelques fois le modèle ne donne pas une valeur unique pour un attribut, mais plusieurs valeurs possibles (contraintes de la forme $a = \{v_1, \ldots, v_n\}$). Les attributs peuvent également être défini par simple comparaison avec un autre attribut (contraintes de la forme $a_1 \leq a_2$, $a_1 = a_2$ ou $a_1 \neq a_2$), ce qui leur laisse également une grande marge de variation.

La difficulté d'interprétation est plus grande avec un modèle de ce type qu'avec un modèle définissant exactement la forme de chaque objet. En effet, si la forme d'un objet est connue, il est possible prédire son aspect dans l'image avec une bonne précision dès que certaines parties de cet objet sont identifiées. Ceci permet un bon contrôle de l'interprétation en vérifiant la "superposition" de la forme prédite avec la

forme observée. Or, ce type de contrôle n'est pas possible avec un modèle générique. Dans ce cas il faut donc trouver d'autres moyens de vérification. Une bonne solution consiste à s'aider du contexte. Ceci est réalisé dans TRIDENT par l'intermédiaire de la propagation syntaxique (exploitation du type de scène pour exclure des objets) et par l'intermédiaire de la propagation de contraintes (liens entre attributs).

7.3 Principes à retenir

Que faut-il retenir de ce travail ? Il y a tout d'abord la représentation discrète des attributs qui, bien qu'elle limite le pouvoir de représentation, permet un contrôle rapide et efficace et une réduction importante de l'indéterminisme. Par ailleurs, cette approche est assez proche du raisonnement humain qui est plutôt qualitatif que quantitatif. Nous sommes par exemple incapables d'évaluer les distances et les dimensions des objets au centimètre près mais nous pouvons les approximer par des valeurs discrètes (entre 10 et 15 centimètres, à plus de 2 mètres, ...). Intuitivement, il nous semble donc que les systèmes de visions "intelligents" se situent plutôt dans cette voie.

Un autre point à retenir est le principe de liberté de construction. La manière de produire la description n'est pas fixée *a priori* mais déterminée durant l'interprétation. Avec ce principe, l'interpréteur a toujours la possibilité d'effectuer la suite de choix la plus fiable. De ce principe il ressort également que l'interprétation en parallèle de plusieurs parties de l'image est préférable à une analyse unique.

Mais le parallélisme est également intéressant pour une autre raison. Différentes parties de l'image constituent des sources d'information indépendantes les unes des autres, mais leurs sémantiques sont liées. Pour avoir une interprétation plus fiable, il est donc intéressant de produire plusieurs interprétation locales de l'image pouvant se confirmer ou s'infirmer par le biais des liens contextuels.

PARTIE II

Apprentissage géométrique

1
Introduction

1.1 L'apprentissage géométrique

1.1.1 Objectif

Le problème général abordé dans cette seconde partie est celui de l'apprentissage géométrique, que nous avons défini dans l'introduction générale comme la faculté d'apprendre la forme des objets par la vision. Nous considérons ici un cas particulier d'apprentissage géométrique dont le but est de construire un modèle tridimensionnel complet d'un objet par un moyen de perception tridimensionnelle (télémétrie laser [Bah 81], stéréovision [Aya 88] [Her 86], ultrasons [Cro 85] ...). Il existe également des approches utilisant une séquence d'images bidimensionnelles de l'objet [Mar 83]. Nous n'en parlerons pas ici. De plus, nous supposerons que l'objet à modéliser est *rigide*, c'est-à-dire qu'il ne se déforme pas au cours du temps.

En déplaçant le ou les capteur(s) autour de l'objet à modéliser, on obtient plusieurs *vues tridimensionnelles* de celui-ci. Chacune d'entre elles est constituée d'un ensemble de *primitives tridimensionnelles* dont les caractéristiques géométriques s'expriment dans un repère local aux capteurs. Ces primitives sont généralement des entités géométriques simples (segments de droite, facettes planes, ...) approchant une partie élémentaire de l'objet. Le problème est donc de combiner ces différentes vues tridimensionnelles afin d'obtenir un modèle tridimensionnel de l'objet.

1.1.2 Approche non structurée

Principe

Nous dirons qu'un système d'apprentissage géométrique est basé sur une approche non structurée si le modèle produit, ainsi que toute vue, est simplement un ensemble de primitives tridimensionnelles définies dans un certain repère.

Étant données deux vues V_1 et V_2, un système basé sur cette approche doit pouvoir remplir les fonctions suivantes (fig. 1.1) :

- estimer le déplacement D entre le repère de V_1 et celui de V_2.
- *mettre entre correspondance* les primitives de V_1 avec celles de V_2. Pour chaque primitive π de V_1 représentant une partie p de l'objet, il faut déterminer l'ensemble des correspondants de π, c'est-à-dire les primitives $\pi_1, \ldots \pi_n$ de V_2 représentant des parties $p_1, \ldots p_n$ de l'objet ayant une intersection non vide avec π.
- *fusionner* V_1 et V_2 afin d'obtenir une vue V_3 plus complète. Cette opération nécessite la connaissance du déplacement D entre V_1 et V_2 et des correspondants dans V_2 de chaque primitive de V_1. La manière la plus simple de procéder est de confondre le repère de V_3 avec celui de V_2. Etant donné l'hypothèse de rigidité sur l'objet, le déplacement D donne alors la position théorique de chaque primitive de V_1 dans V_3. Les primitives de V_3 sont donc :
  - les primitives de V_1 sans correspondants dans V_2, définies dans le repère de V_2 à l'aide du dépacement D.
  - les primitives de V_2 sans correspondants dans V_1.
  - les primitives obtenues par fusion. Chaque primitive possédant un ensemble de correspondants non vide est déplacée dans le repère de V_2 puis fusionnée avec ses correspondants.

A partir de là, un modèle tridimensionnel M peut être construit de manière incrémentale à partir d'une séquence de vues $V_1, \ldots V_n$. Au départ, on pose $M = V_1$, puis on répète les opérations suivantes pour les vues $V_2, \ldots V_n$:

- Estimer le déplacement D entre V_i et M.
- Effectuer une mise en correspondance entre les primitives de V_i et celles de M.
- Fusionner V_i et M en un nouveau modèle M' et poser $M = M'$.

Exemple

J. Crowley défend un système de représentation non structurée basée sur une primitive unique, appelée pavé de surface, servant à homogénéiser les données de différentes sources d'information 3D.

Un pavé de surface (fig. 1.2) représente une partie de surface plane perçue par le système et caractérisée par :

- Une ellipse d'incertitude W délimitant la surface.
- Sa normale N.

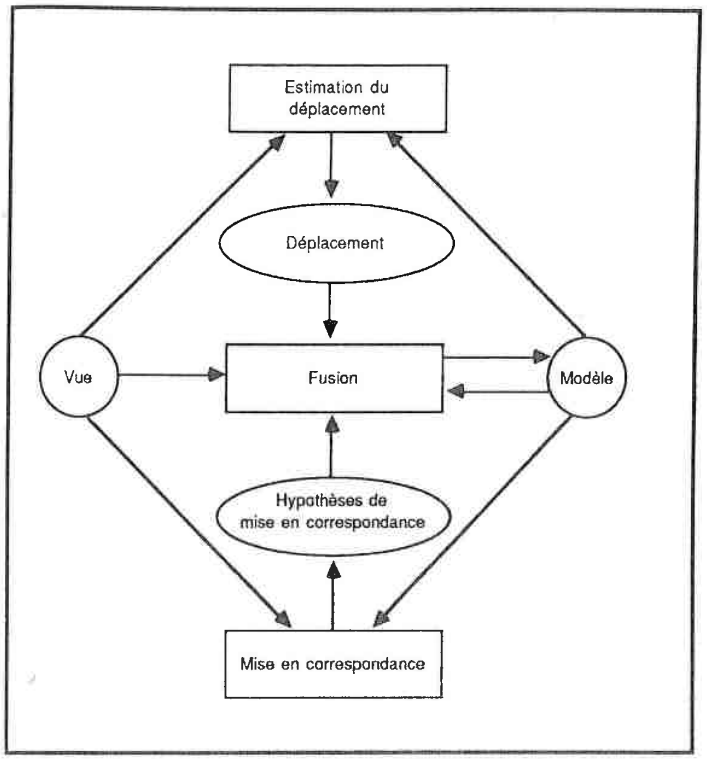


Figure 1.1. Construction incrémentale d'un modèle 3D : approche non structurée.

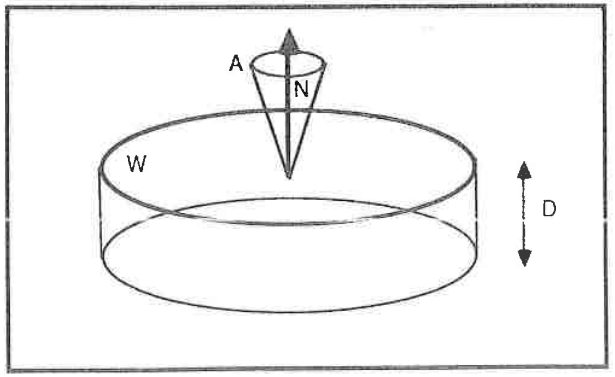


Figure 1.2. Le pavé de surface

- Un cône d'angle A, représentant l'incertitude sur l'orientation de la surface.
- Un cylindre de hauteur D dont l'axe est perpendiculaire à la surface et dont la base est W. Ce cylindre est un domaine d'incertitude contenant toutes les positions possibles de l'ellipse.
- Un coefficient de confiance CF, exprimant la certitude d'existence du pavé de surface.

Expliquons brièvement comment sont effectuées la mise en correspondance et la fusion de deux vues V_1 et V_2. Soient :

- D, le déplacement entre le repère de V_1 et celui de V_2 ;
- P_1, un pavé de surface de V_1 ;
- P_2, un pavé de surface de V_2 ;
- P_1', le pavé obtenu en déplacant P_1 par D ;
- W_1', l'ellipse d'incertitude de P_1' ;
- W_2, l'ellipse d'incertitude de P_2.

P_2 est un correspondant possible de P_1 si :

- Les cylindres d'incertitude de P_1' et P_2 ont une intersection non vide.
- Les ellipses d'incertitude de P_1' et P_2 se recouvrent. Géométriquement, cela signifie que la projection orthogonale de W_1' sur le plan portant W_2 a une intersection non vide avec W_2 (fig. 1.3).

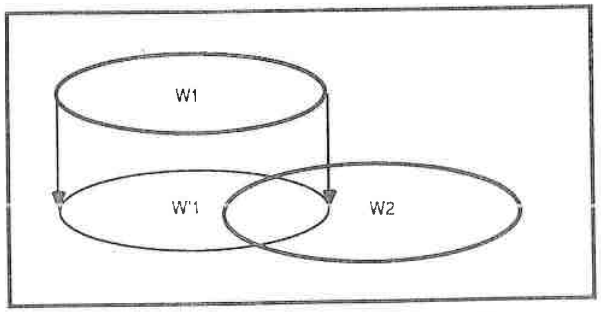


Figure 1.3. Condition de recouvrement des ellipses d'incertitude

Les pavés de la nouvelle vue obtenue par fusion de V_1 et V_2 sont :

- les pavés de V_2 sans correspondants.
- les pavés de V_1 sans correspondants placés dans le repère de V_2 par D.
- les pavés de V_1 avec correspondants fusionnés avec leur meilleur correspondant dans V_2 (recouvrement maximal des ellipses).

1.1.3 Approche structurée

Dans l'approche structurée, le modèle de l'objet et une vue de celui-ci sont deux représentations tridimensionnelles de natures différentes. Un modèle comprend des entités regroupant plusieurs primitives, qui peuvent à leur tour appartenir à des entités de niveau plus élevé. Toutes ces entités sont définies dans le même repère 3D.

Afin que les opérations de mise en correspondance et de fusion soient plus aisées, on essaie de construire un modèle à partir des primitives de la vue. Ce modèle est une représentation structurée de la partie visible de l'objet que nous appelons *modèle du champ d'observation*.

A partir de là, les problèmes intervenant dans la construction d'un modèle sont les mêmes que ceux rencontrés dans l'approche non structurée :

- Estimation du déplacement entre les repères respectifs de deux modèles M_1 et M_2.
- Mise en correspondance entre les entités de M_1 et celles de M_2.
- Fusion de M_1 avec M_2 pour obtenir un nouveau modèle M_3 contenant :
  - les entités de M_1 sans correspondants dans M_2 définies dans le repère de M_2 à l'aide du déplacement D.
  - les entités de M_2 sans correspondants dans M_1.
  - les entités de M_1 fusionnées avec leurs correspondants dans M_2.

Un modèle tridimensionnel M peut alors être construit de manière incrémentale à partir d'une séquence de vue $V_1, \ldots V_n$. On commence par construire le modèle M_1 correspondant à V_1. On pose $M = M_1$. Puis on répète les opérations suivantes pour les vues $V_2, \ldots V_n$:

- construire le modèle M_i correspondant à V_i (phase de structuration).
- Estimer le déplacement D entre M_i et M.
- Effectuer une mise en correspondance entre les entités de M_i et celles de M.
- Fusionner M_i et M en un nouveau modèle M' et poser $M = M'$.

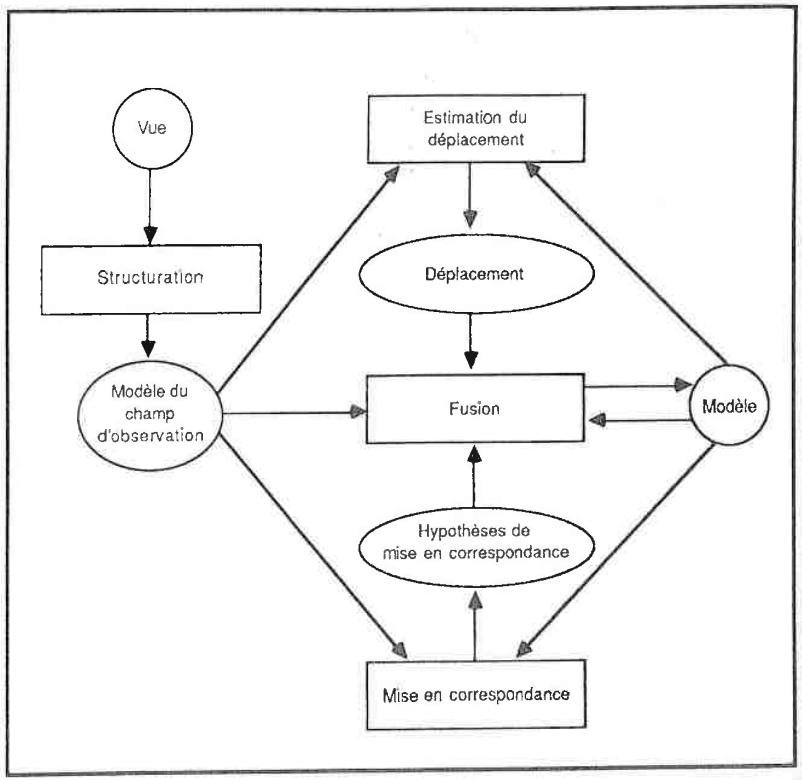


Figure 1.4 Construction incrémentale d'un modèle 3D : approche structurée.

Exemple

Le système MOSAIC [Her 86] est un système d'apprentissage visuel utilisant une représentation structurée. Il a été appliqué à des vues aériennes de Washington pour modéliser des bâtiments. La fig. 1.5 montre l'organisation du système. Une vue tridimensionnelle est obtenue à partir de segments observés par deux caméras en combinant stéréovision et analyse monoculaire. Le résultat est un ensemble de segments de droite dont on connaît la position spatiale dans un repère 3D lié aux caméras.

Ce système est également prévu pour des tâches d'interprétation et de navigation pour un robot mobile. Mais pour l'instant, seules les parties de mise à jour du modèle, de stéréovision et d'analyse monoculaire ont été réalisées.

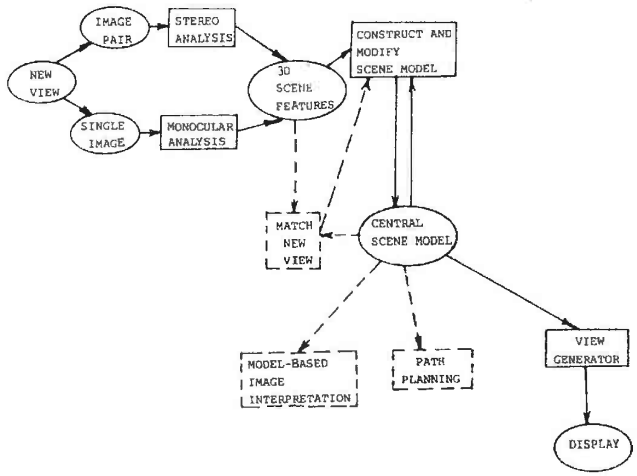


Figure 1.5. Schéma du système MOSAIC

Le système de représentation de MOSAIC utilise trois primitives géométriques : le point, la droite et le plan ainsi que des entités topologiques : coin, segment, groupe de segments, face et objet nécessairement polyédrique.

Les groupes de segments et les faces sont des ensembles de segments connexes et coplanaires formant une ligne ouverte dans le premier cas et fermée dans le second cas.

Les cinq entités topologiques sont reliées par des relations du type partie-de, induisant cinq niveaux de hiérarchie dans le modèle (fig. 1.6)

Les primitives servent à définir ou à contraindre la position des entités topologiques du type coin, segment ou face. Par exemple, lorsqu'un plan P contraint une face F cela signifie que tous les segments de F doivent appartenir à P.

1.2 Définition du problème abordé

Le travail que nous avons effectué en apprentissage géométrique est lié au projet de vision ORASIS. Ce projet a pour but d'accroître l'autonomie des robots par l'intermédiaire de la vision et en particulier leur permettre d'acquérir un modèle tridimensionnel de l'environnement dans lequel ils se déplacent. C'est à ce niveau que se situe notre con-

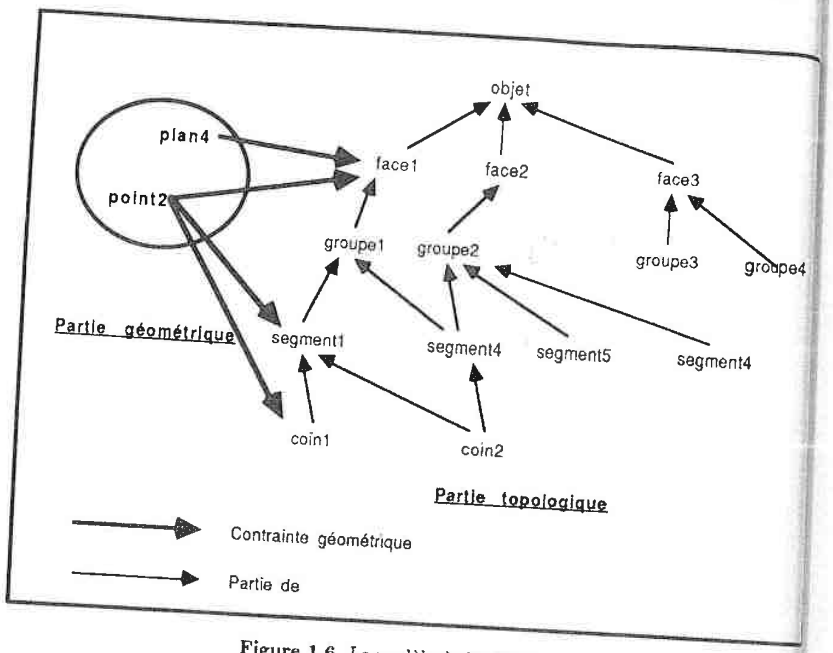


Figure 1.6. Le modèle de MOSAIC

tribution.

1.2.1 Hypothèses sur le type d'objet

Le système que nous avons réalisé est prévu pour fonctionner dans un environnement *polyédrique* (hypothèse 1), *orthogonal* (hypothèse 2) et *statique* (hypothèse 3).

Hypothèse 1 *L'environnement contient une très forte proportion d'objets polyédriques.*

Hypothèse 2 *Il existe trois directions majoritaires dont l'une verticale et les deux autres horizontales et perpendiculaires telles que la majorité des arêtes des objets soient parallèles à l'une d'entre elles.*

Hypothèse 3 *Il existe un repère dans lequel une grande partie des objets ont une position constante.*

Ces hypothèses sont conformes au projet ORASIS. Premièrement, ce projet est limité à un univers d'intérieur. Or, on peut constater :

1. que les objets présents dans ce type d'environnement sont fréquemment polyédriques.
2. que les bâtiments sont généralement construits selon trois directions : la verticale et deux directions horizontales perpendiculaires parallèles aux murs. Ceci justifie l'hypothèse 2.

Deuxièmement, l'hypothèse de staticité fait partie des hypothèses de départ du projet. L'environnement du robot peut donc être considéré comme un seul objet rigide.

1.2.2 Hypothèses sur la perception 3D

Voyons à présent quelles sont les hypothèses sur les capteurs et sur le moyen de perception tridimensionnelle utilisé.

Hypothèses sur les capteurs

Hypothèse 4 *Le robot possède au moins deux caméras.*

Hypothèse 5 *Chaque caméra vérifie le* modèle sténopé *: l'image délivrée par chacune d'entre elles est le résultat d'une projection perspective parfaite de centre C sur un plan P (fig. 1.7).*

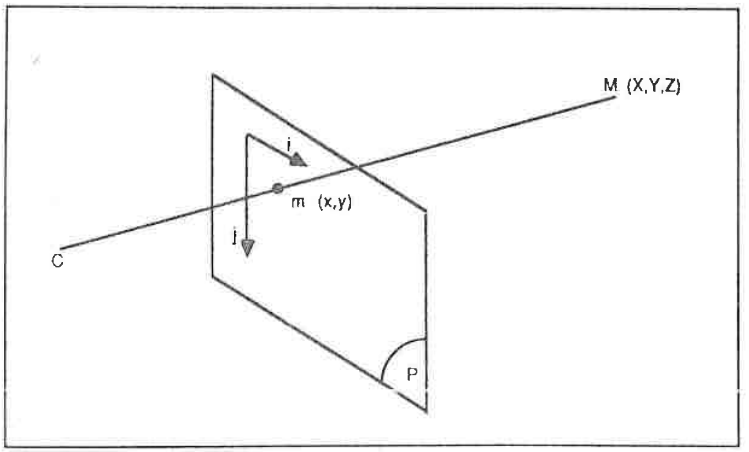


Figure 1.7. Le modèle sténopé

L'hypothèse suivante fait intervenir la notion de *matrice de projection d'une caméra*. Définissons donc tout d'abord cette matrice. Soient (fig. 1.7) :

- c, une caméra ;
- M, un point de l'espace de coordonnées (X, Y, Z) dans un repère R ;
- r, le *repère image*, c'est-à-dire le repère dans lequel s'expriment les coordonnées en pixels des points perçus par c ;
- x et y, les coordonnées dans le repère r de la projection m de M sur le plan image de c.

La matrice projection MP de c est une matrice 4×3 de la forme :

$$MP = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{pmatrix}$$

Cette matrice définit entièrement la projection perspective associée à c dans le repère R par les relations :

$$x = \frac{a_{11} X + a_{12} Y + a_{13} Z + a_{14}}{a_{31} X + a_{32} Y + a_{33} Z + a_{34}}$$

$$y = \frac{a_{21} X + a_{22} Y + a_{23} Z + a_{24}}{a_{31} X + a_{32} Y + a_{33} Z + a_{34}}$$

Si l'on représente le point image en coordonnées homogènes (U, V, S), la projection perspective devient une transformation linéaire et la relation entre (U, V, S) et (X, Y, Z) peut s'écrire :

$$\begin{pmatrix} U \\ V \\ S \end{pmatrix} = MP \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

avec :

$$x = \frac{U}{S}$$

$$y = \frac{V}{S}$$

Hypothèse 6 *On connaît la matrice de projection d'une des caméras dans un repère 3D appelé repère stéréoscopique.*

Nous appelons cette caméra *caméra de référence*.

Hypothèse 7 *La hauteur de l'origine du repère stéréoscopique varie très peu.*

Hypothèse 8 *L'axe des y du repère stéréoscopique est approximativement vertical et dirigé vers le bas.*

Méthode de perception

Notre étude est limitée à une perception 3D par *stéréovision* avec des *segments de droite* extraits d'images à niveaux de gris. Ce moyen de perception permet d'obtenir des segments de droites tridimensionnels, appelés segments stéréoscopiques, en observant la scène avec au moins deux caméras. Ils représentent une approximation de tous les contours physiques (cf. paragraphe 0.3.1 de l'introduction) visibles par chaque caméra.

Expliquons brièvement comment ces segments sont obtenus. Initialement, une extraction de contours est appliquée aux différentes images de la scène. Ces contours sont ensuite approchés par des segments de droites appelés *segments image*. Puis, une mise en correspondance entre les segments image associés aux différentes caméras permet de retrouver les *segments homologues*, c'est-à-dire ceux qui proviennent d'un même contour physique.

L'étape finale est le calcul des segments stéréoscopiques. Le but de cette étape est de déterminer les coordonnées des extrémités de ces segments dans le repère stéréoscopique. Puisqu'on connaît la matrice de projection de chaque caméra dans le repère stéréoscopique (hypothèse 6), il est possible de calculer un segment stéréoscopique S à partir de deux segments homologues s_1 et s_2. Une des solutions permettant de déterminer S est de calculer l'intersection des deux portions de plan P_1 et P_2 qui sont les projections inverses respectives de s_1 et s_2 (fig. 1.8).

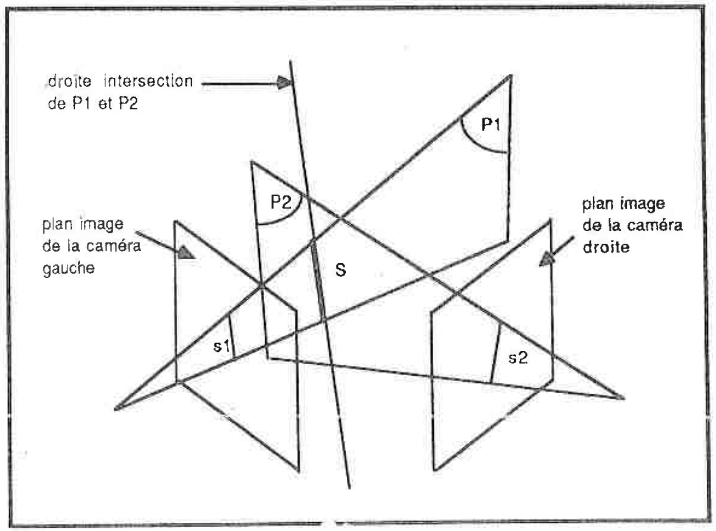


Figure 1.8. Calcul des coordonnées 3D d'un segment en stéréovision

1.3 Utilisation des points de fuite

Dans chaque vue de l'environnement nous utilisons, en plus des coordonnées des segments stéréoscopiques, les points de fuite associés à la caméra de référence.

1.3.1 Définition et propriétés

La notion de point de fuite est attachée à une propriété de la projection perspective : les droites parallèles sont transformées en droites concourantes (fig. 1.9). Un *point de fuite* est donc le point d'intersection de droites parallèles transformées selon une certaine projection perspective.

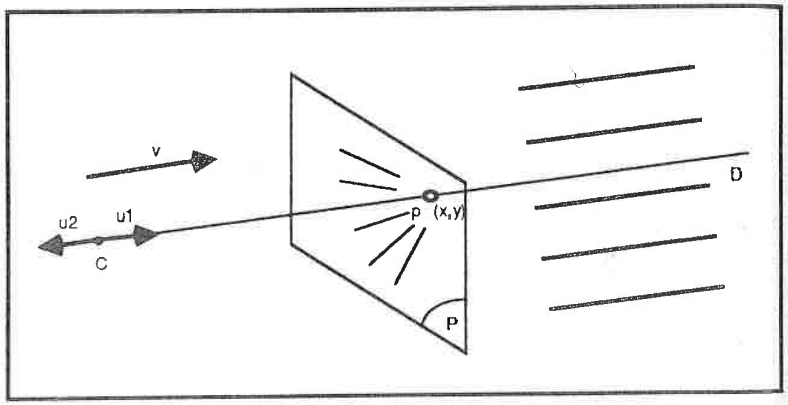


Figure 1.9. Point de fuite

On peut également définir un point de fuite p par la donnée d'un vecteur $\vec{v}$ de $\mathbb{R}^3$ (représentant la direction des droites) et d'une projection perspective de centre C sur un plan P. Les coordonnées (x, y) de p dans le plan de projection sont données par :

$$x = \frac{a_{11}X + a_{12}Y + a_{13}Z}{a_{31}X + a_{32}Y + a_{33}Z}$$

$$y = \frac{a_{21}X + a_{22}Y + a_{23}Z}{a_{31}X + a_{32}Y + a_{33}Z}$$

où les a_{ij} sont les coefficients de la matrice de projection et (X, Y, Z) les coordonnées de $\vec{v}$.

D'un point de vue géométrique, p est l'intersection avec le plan P, de la droite D passant par C et de vecteur directeur $\vec{v}$ (fig. 1.9).

Réciproquement, les équations précédentes permettent de retrouver la direction spatiale des droites, lorsque leur point de fuite est donné. En ajoutant l'équation :

$$X_i^2 + Y_i^2 + Z_i^2 = 1$$

on obtient un système de trois équations à trois inconnues, dont les solutions sont les deux vecteurs unitaires opposés $\vec{u_1}(p)$ et $\vec{u_2}(p)$ portés par la droite D (fig. 1.9).

1.3.2 Méthode utilisée

Le calcul des points de fuite a été réalisé grâce à un programme de L. Quan. Celui-ci prend en entrée un ensemble de segments image et fournit un ensemble de points de fuite caractérisés par leurs coordonnées dans le plan image. Le programme donne également les segments associés à chaque point de fuite.

La méthode de L. Quan [Qua 87] [Qua 88] est inspirée de celle de Barnard [Bar 83]. On définit un repère tridimensionnel $R = (O, \vec{I}, \vec{J}, \vec{K})$ dont l'origine O est confondue avec le centre optique de la caméra et tel que $\vec{I}$ soit parallèle au vecteur $\vec{i}$ du repère image, $\vec{J}$, parallèle au vecteur $\vec{j}$ du repère image et $\vec{K}$, perpendiculaire au plan image et dirigé vers celui-ci (fig. 1.10). Chaque segment s est représenté dans R par un cercle C_s appartenant à la sphère gaussienne. Ce cercle est l'intersection avec la sphère gaussienne, du plan projection inverse de la droite d portant s. Il définit l'ensemble des vecteurs unitaires possibles correspondant au point de fuite associé à s.

Avec cette représentation, les points de fuite correspondent à deux points antipodaux de la sphère gaussienne par lesquels passent un nombre important de cercles. Dans la sphère gaussienne, rechercher un point de fuite revient donc à rechercher les points de passage maximaux des cercles associés aux segments image. L'idée essentielle de L. Quan est de rechercher ces points par transformée de Hough rapide. A la $i^{ème}$ itération on considère une portion de sphère P_i limitée par un intervalle angulaire $d\alpha_i = [\alpha^i_{min}, \alpha^i_{max}]$ (angle avec $\vec{I}$) et par un intervalle $d\phi_i = [\phi^i_{min}, \phi^i_{max}]$ (élévation par rapport au plan $(O, \vec{I}, \vec{J})$). Cette portion de sphère est associée à l'ensemble S_i des segments s tels que C_s coupe P_i.

P_i est ensuite divisée en quatre parties P_i^1, P_i^2, P_i^3 et P_i^4 définies respectivement par les couples d'intervalles $(d\phi_i^1, d\alpha_i^1)$, $(d\phi_i^2, d\alpha_i^2)$, $(d\phi_i^3, d\alpha_i^3)$ et $(d\phi_i^4, d\alpha_i^4)$ avec :

$$d\alpha_i^1 = [\alpha^i_{min}, \frac{\alpha^i_{min} + \alpha^i_{max}}{2}] \quad d\phi_i^1 = [\frac{\phi^i_{min} + \phi^i_{max}}{2}, \phi^i_{max}]$$

$$d\alpha_i^2 = [\frac{\alpha^i_{min} + \alpha^i_{max}}{2}, \alpha^i_{max}] \quad d\phi_i^2 = [\frac{\phi^i_{min} + \phi^i_{max}}{2}, \phi^i_{max}]$$

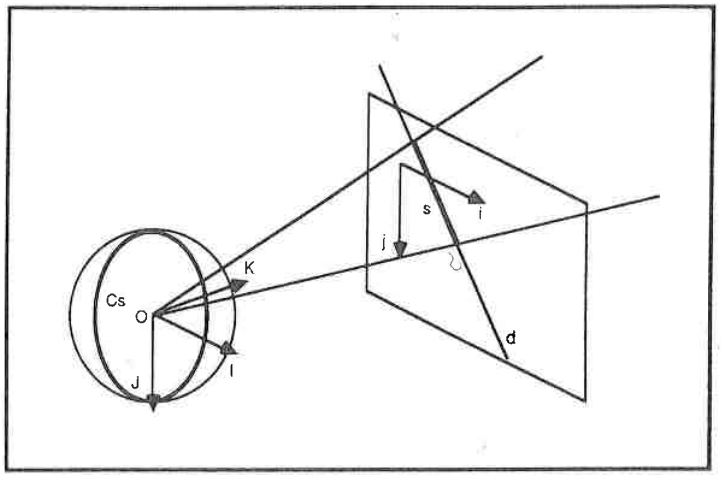


Figure 1.10. Représentation d'un segment sur la sphère gaussienne.

$$d\alpha_i^3 = [\alpha_{min}^i, \frac{\alpha_{min}^i + \alpha_{max}^i}{2}] \quad d\phi_i^3 = [\phi_{min}^i, \frac{\phi_{min}^i + \phi_{max}^i}{2}]$$

$$d\alpha_i^4 = [\frac{\alpha_{min}^i + \alpha_{max}^i}{2}, \alpha_{max}^i] \quad d\phi_i^4 = [\phi_{min}^i, \frac{\phi_{min}^i + \phi_{max}^i}{2}]$$

Chaque partie P_i^j est associée à un compteur c_i^j et un ensemble de segments S_i^j. Pour déterminer c_i^j et S_i^j,on considère chaque segment s de S^i. Si s coupe P_i^j, on incrémente c_i^j et s est ajouté à S_i^j.

Ceci permet de définir la portion de sphère P_{i+1} à utiliser pour la prochaine itération : c'est la partie P_i^j qui maximise c_i^j.

La figure 1.11 montre des exemples de divisions successives de la sphère. Initialement, il suffit de considérer la demi-sphère dirigée vers le plan image ($z > 0$) et de l'associer à l'ensemble des segments image. Autrement dit P_0 est définie par les deux intervalles $d\alpha_0 = [0, \pi]$ et $d\phi_0 = [-\pi/2, \pi/2]$.

L'itération s'arrête lorsque S_i contient un nombre insuffisant de segments (moins d'un quart des segments donnés) ou bien lorsque la dimension de P_i est suffisament petite (la dimension des intervalles $d\phi_i$ et $d\alpha_i$ est inférieure à 5 degrés). Dans le premier cas, on conclut qu'il n'existe pas de point de fuite. Dans le second, P_i est considéré comme un

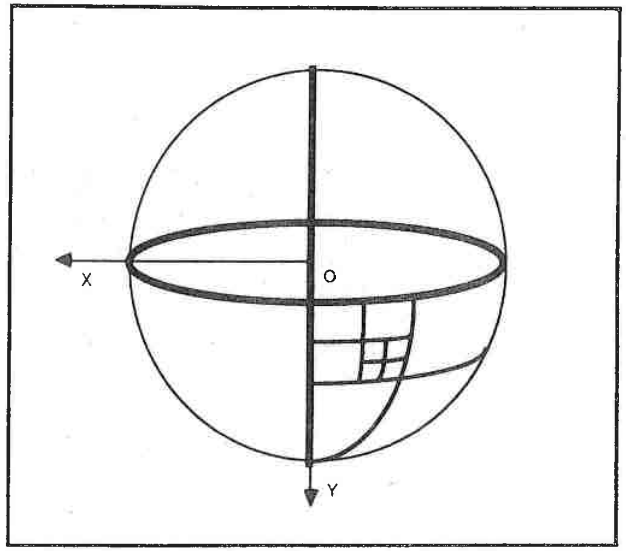


Figure 1.11. Divisions de la sphère gaussienne.

point de passage maximal. La position du point de fuite correspondant est calculée par moindre carrés en fonction des segments image contenus dans S_i. C'est le point le plus proche de toutes les droites portant les segments de S_i. Le résultat retourné est donc ce point associé à S_i.

On peut remarquer que cette méthode ne donne qu'un seul point de fuite. C'est le point de fuite correspondant à un nombre maximal de segments. Pour en trouver un autre, il suffit d'enlever les segments de S_i dans l'ensemble initial de segments et de recommencer avec ce nouvel ensemble de segments.

1.4 Les données utilisées

Les résultats de stéréovision que nous avons utilisés proviennent de l'INRIA Rocquencourt et sont obtenus par une méthode de stéréovision trinoculaire (à trois caméras) par prédiction/vérification développée par N. Ayache [Aya 87].

Les différentes vues stéréoscopiques ont été prises à partir d'un robot portant trois caméras (fig. 1.12) disposées triangulairement. La caméra de référence est la plus haute des trois caméras.

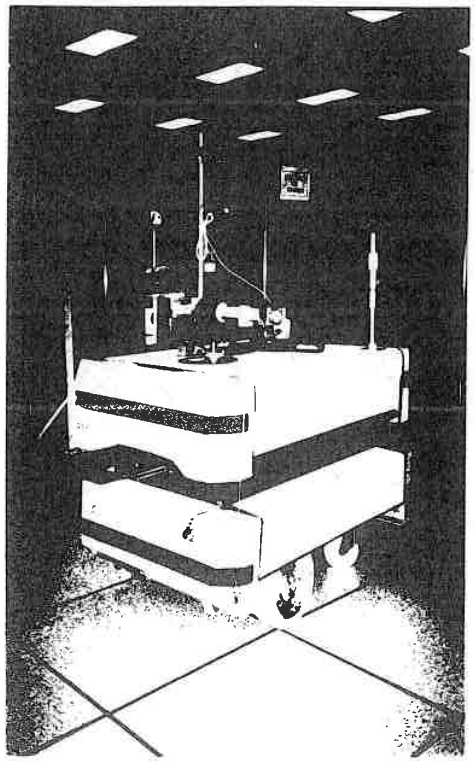


Figure 1.12. Le robot mobile d'ORASIS

Nous avons reproduit en simulation la pièce de l'INRIA où les prises de vues ont été effectuées. La figure 1.13 montre une projection orthogonale de cette pièce.

Parmi les différentes observations de cette pièce, 16 vues, que nous notons $V_0, \ldots, V_{15}$ correspondent à des positions $p_0 \ldots p_{15}$ obtenues par rotation successives de 5 degrés du robot. La figure 1.14 visualise approximativement le champ d'observation de la caméra de référence dans la position 0 du robot.

La figure 1.15 montre l'image délivrée par la caméra de référence dans cette même position.

Nous n'avons utilisé que les huit premières vues stéréoscopiques $V_0, \ldots, V_7$, correspon-

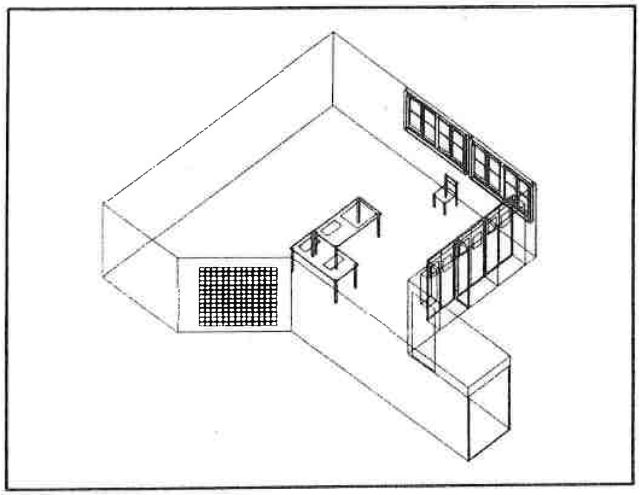


Figure 1.13. Projection orthogonale de la pièce de l'INRIA reproduite en simulation

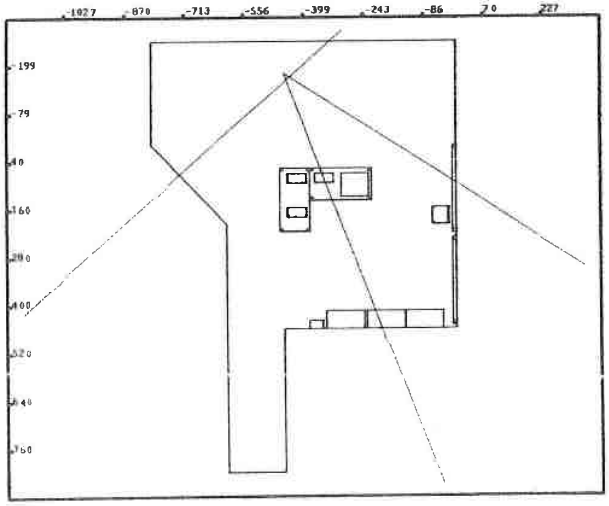


Figure 1.14. Position p_0 du robot.

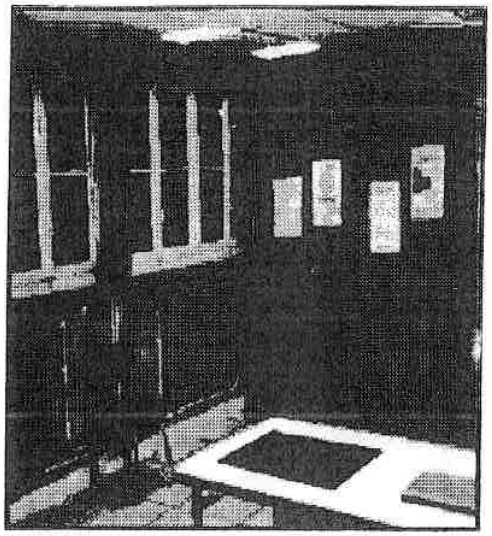


Figure 1.15. Image délivrée par la caméra de référence dans la position 0

dant aux positions $p_0 \ldots p_7$ du robot, ainsi que la vue 15, qui correspond à une grille de calibration.

Les figures de haut de page de l'annexe B présentent les projections orthogonales de ces différentes vues.

1.5 Description globale du système réalisé

1.5.1 Modules du système

Le système réalisé est basé sur l'approche structurée. Il comprend donc trois modules principaux (fig. 1.16) : le module de structuration, le module de mise en correspondance et le module de fusion.

Le module de structuration construit un modèle représentant la partie de l'environnement qui se trouve dans le champ des caméras (modèle du champ d'observation). Il utilise pour cela la matrice de projection de la caméra de référence, les coordonnées des points de fuite associés à cette même caméra ainsi que les coordonnées des segments stéréoscopiques.

Etant donnés deux modèles M_1 et M_2, le module de mise en correspondance produit un ensemble d'hypothèses de correspondance entre les segments de M_1 et ceux de M_2. La

méthode de mise en correspondance utilisée donne en même temps le déplacement entre les repères de M_1 et M_2.

Enfin, connaissant le déplacement entre deux modèles M_1 et M_2 ainsi que les hypothèses de correspondance entre les segments de M_1 et ceux de M_2, le module de fusion produit un nouveau modèle M_3, union des modèles M_1 et M_2.

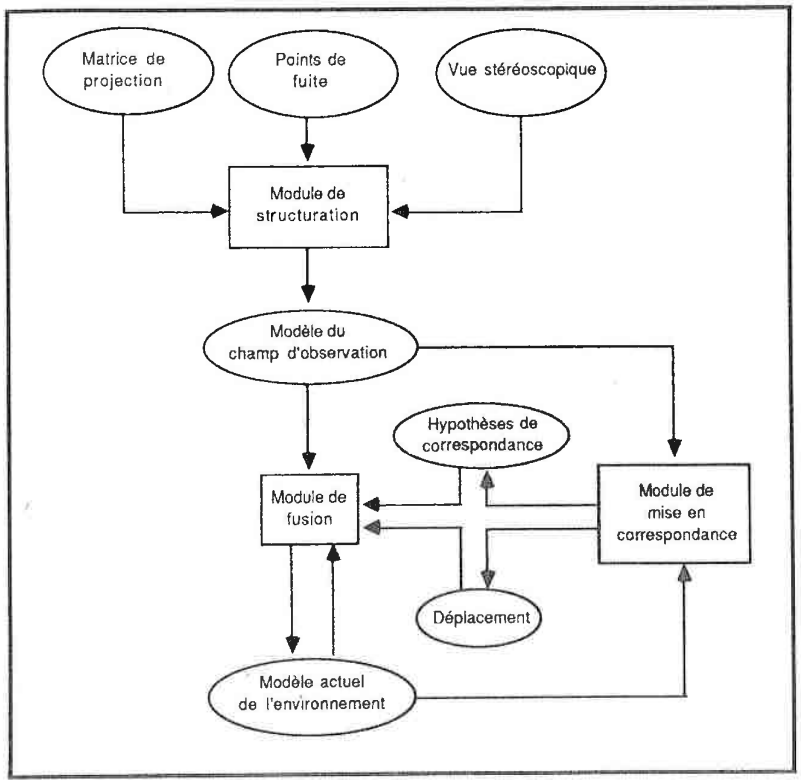


Figure 1.16. Schéma fonctionnel du système

1.5.2 Enchaînement des modules

Ces différents modules sont utilisés de la manière suivante afin de produire un modèle de l'environnement ME_N à partir de plusieurs vues $V_0, V_1, \ldots, V_N$:

1. on construit un premier modèle ME_0 en appliquant le module de structuration à V_0.

2. on crée les modèles ME_i pour i allant de 1 à N :

   (a) en utilisant le module de structuration pour construire un modèle MV_i à partir de V_i.

   (b) en appliquant le module de mise en correspondance à MV_i et ME_{i-1}.

   (c) en utilisant les résultats de la mise en correspondance (déplacement entre les repères de MV_i et ME_{i-1}, correspondances entre les segments de MVi et ceux de ME_{i-1}) pour fusionner MV_i et ME_{i-1} et produire ME_i.

2
Définitions et notations

Ce chapitre définit les fonctions géométriques que nous utilisons dans les chapitres suivants. Ces fonctions sont regroupées selon les entités géométriques sur lesquelles elle portent : les points, les droites, segments de droite et plans de $\mathbb{R}^n$.

Chaque fonction est définie par :

- sa notation,
- son appellation,
- le type de ses arguments,
- le type de son résultat.

Dans les cas non triviaux, nous indiquons la formule permettant de calculer la fonction ou bien sa signification intuitive.

2.1 Points

2.1.1 Centre de gravité

- Notation : $C_g(E)$
- Appellation : centre de gravité de E.
- Arguments : ensemble de points de $\mathbb{R}^n$.
- Résultat : point de $\mathbb{R}^n$.

2.1.2 Projection orthogonale sur une droite

- Notation : $P^{\perp}(p, d)$
- Appellation : projection orthogonale de p sur d.

- Arguments :
  - p : point de $\mathbb{R}^n$.
  - d : droite de $\mathbb{R}^n$.
- Résultat : point de $\mathbb{R}^n$.
- Formule :

$$p'_i = m_i + \frac{v_i\left[(p_1 - m_1)\,v_1 + \ldots + (p_n - m_n)\,v_n\right]}{v_1^2 + \ldots + v_n^2}$$

m_i, v_i, p_i et p'_i étant respectivement la ième coordonnée d'un point de d, d'un vecteur directeur de d, du point p et la projection orthogonale de p sur d.

2.1.3 Projection orthogonale sur un plan

- Notation : $P^{\perp}(p, P)$
- Appellation : projection orthogonale de p sur P.
- Arguments :
  - p : point de $\mathbb{R}^3$.
  - P : plan de $\mathbb{R}^3$.
- Résultat : point de $\mathbb{R}^3$
- Formule :
  si p a pour coordonnées (x_p, y_p, z_p) et si P est le plan d'équation $a\,x+b\,y+c\,z+d=0$, les coordonnées (x'_p, y'_p, z'_p) de la projection orhtogonale de p sur P sont données par :

$$x'_p = x_p - a\mu$$

$$y'_p = y_p - b\mu$$

$$z'_p = z_p - c\mu$$

avec :

$$\mu = \frac{a\,x_p + b\,y_p + c\,z_p + d}{a^2 + b^2 + c^2}$$

2.1.4 Projection perspective

- Notation : $P^{<}(p, m)$
- Appellation : projection perspective de p selon la projection définie par la matrice m.

- Arguments :
  - p : point de $\mathbb{R}^3$.
  - m : matrice 4×3.
- Résultat : point de $\mathbb{R}^2$.
- Formule :
  les coordonnées x et y du point projeté sont données par :

$$x = \frac{a_{11}\,X + a_{12}\,Y + a_{13}\,Z + a_{14}}{a_{31}\,X + a_{32}\,Y + a_{33}\,Z + a_{34}}$$

$$y = \frac{a_{21}\,X + a_{22}\,Y + a_{23}\,Z + a_{24}}{a_{31}\,X + a_{32}\,Y + a_{33}\,Z + a_{34}}$$

X, Y, Z étant les coordonnées de p et m la matrice :

$$m = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{21} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{pmatrix}$$

2.1.5 Perspective inverse

- Notation : $P^{>}(p, m)$
- Appellation : projection perspective inverse de p par la projection définie par la matrice m.
- Arguments :
  - p : point de $\mathbb{R}^2$.
  - m : matrice 4×3.
- Résultat : droite $\mathbb{R}^3$.
- Formule :
  la projection perspective inverse de p est la droite d'équation :

$$x = \frac{a_{11}\,X + a_{12}\,Y + a_{13}\,Z + a_{14}}{a_{31}\,X + a_{32}\,Y + a_{33}\,Z + a_{34}}$$

$$y = \frac{a_{21}\,X + a_{22}\,Y + a_{23}\,Z + a_{24}}{a_{31}\,X + a_{32}\,Y + a_{33}\,Z + a_{34}}$$

x, y étant les coordonnées de p et m la matrice :

$$m = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{21} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{pmatrix}$$

2.1.6 Distance à un point

- Notation : $\|p_1, p_2\|$
- Appellation : distance euclidienne entre p_1 et p_2.
- Arguments : points de $\mathbb{R}^n$.
- Résultat : nombre réel positif.

2.1.7 Distance à une droite

- Notation : $\|p, d\|$
- Appellation : distance entre p et d.
- Arguments :
  - p : point de $\mathbb{R}^n$.
  - d : droite de $\mathbb{R}^n$.
- Résultat : nombre réel positif.
- Formule :

$$\|p, d\| = \|p, P^{\perp}(p, d)\|$$

2.2 Droites

2.2.1 Pseudo–intersection

- Notation : $\sqcap(d_1, d_2)$
- Appellation : pseudo-intersection de d_1 et d_2.
- Arguments : droites de $\mathbb{R}^3$.
- Résultat : point de $\mathbb{R}^3$.
- Signification : milieu du couple de points constitué du point de d_1 le plus proche de d_2 et du point de d_2 le plus proche de d_1. Dans le cas ou d_1 et d_2 sont coplanaires, $\sqcap(d_1, d_2)$ représente l'intersection des deux droites aux sens habituel du terme.

2.3 Segments

2.3.1 Milieu

- Notation : $M(s)$
- Appellation : milieu de s.
- Arguments : segment de $\mathbb{R}^n$.
- Résultat : point de $\mathbb{R}^n$.

2.3.2 Longueur

- Notation : $L(s)$
- Appellation : longueur de s.
- Arguments : segment de $\mathbb{R}^n$.
- Résultat : nombre réel positif.

2.3.3 Droite support d'un segment

- Notation : $D(s)$
- Appellation : droite support de s.
- Arguments : segment de $\mathbb{R}^n$.
- Résultat : droite de $\mathbb{R}^n$.
- Signification : droite passant par les deux extrémités de s.

2.3.4 Distance

- Notation : $\delta(s_1, s_2)$
- Appellation : distance entre s_1 et s_2.
- Arguments : segments de $\mathbb{R}^n$.
- Résultat : nombre réel positif.
- Signification : distance maximale entre les milieux de chaque segment et la droite supportant l'autre segment (fig. 2.1).

- Formule :

$$\delta(s_1, s_2) = max(d_1, d_2)$$

avec :

$$d_1 = \|M(s_1), D(s_2)\|$$

$$d_2 = \|M(s_2), D(s_1)\|$$

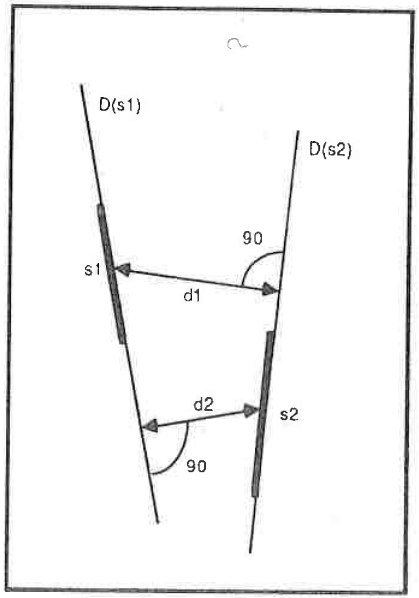


Figure 2.1. Distance entre deux segments

2.3.5 Projection orthogonale sur une droite

- Notation : $P^{\perp}(s, D)$
- Appellation : projection orthogonale de s sur D.
- Arguments :
  - s : segment de $\mathbb{R}^n$.
  - D : droite affine de $\mathbb{R}^n$.
- Résultat : segment de $\mathbb{R}^n$.

- Signification : segment dont les extrémités sont les extrémités de s projeté orthogonalement sur D.

2.3.6 Projection orthogonale sur un plan

- Notation : $P^{\perp}(s, P)$
- Appellation : projection orthogonale de s sur P.
- Arguments :
  - s : segment de $\mathbb{R}^3$.
  - P : plan affine de $\mathbb{R}^3$.
- Résultat : segment de $\mathbb{R}^3$.
- Signification : segment dont les extrémités sont les extrémités de s projeté orthogonalement sur P.

2.3.7 Recouvrement

- Notation : s_1/s_2
- Appellation : recouvrement de s_1 par s_2.
- Arguments : segments de $\mathbb{R}^n$.
- Résultat : nombre réel.
- Signification :
  - soit s'_1 la projection orthogonale de s_1 sur la droite portant s_2 (fig. 2.2).
  - si $s_1/s_2 > 0$, l'intersection de s_2 avec s'_1 est un segment. s_1/s_2 mesure dans ce cas la longueur de ce segment.
  - si $s_1/s_2 = 0$, l'intersection de s_2 avec s'_1 est un point.
  - si $s_1/s_2 < 0$, l'intersection de s_2 avec s'_1 est vide. s_1/s_2 mesure dans ce cas la distance entre les extrémités de s'_1 et s_2 les plus rapprochées.
- Formule :
  Soit :
  - a_2 et b_2, les extrémités de s_2 ;
  - D, la droite support de s_2 ;
  - a_1 et b_1, les projections orthogonales des extrémités de s_1 sur D :

– λ_a^1, λ_b^1, λ_a^2, λ_b^2 les coordonnées paramétriques respectives des points a_1, b_1, a_2, b_2 sur la droite D.

$$s_1/s_2 = min\,(\lambda_{max}^1, \lambda_{max}^2) - max\,(\lambda_{min}^1, \lambda_{min}^2)$$

avec :

$$\lambda_{max}^1 = max\,(\lambda_a^1, \lambda_b^1)$$
$$\lambda_{max}^2 = max\,(\lambda_a^2, \lambda_b^2)$$
$$\lambda_{min}^1 = min\,(\lambda_a^1, \lambda_b^1)$$
$$\lambda_{min}^2 = min\,(\lambda_a^2, \lambda_b^2)$$

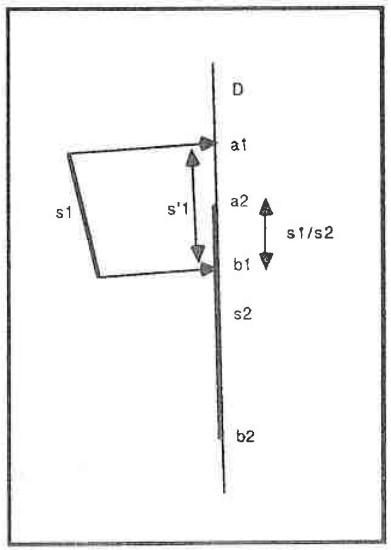


Figure 2.2. Recouvrement de deux segments

3
Système de représentation

3.1 Vue globale

Ce chapitre donne une définition d'un modèle de l'environnement et des entités qu'il contient, dans notre système de représentation.

Nous définissons un modèle comme un ensemble de segments de droite et de *motifs*.

Les segments sont une représentation bruitée des contours physiques de l'environnement (cf. paragraphe 0.3.1). Etant donné que les objets sont polyédriques (hypothèse 1), les contours physiques sont également des segments de droite que nous appelons *arêtes*. Les arêtes représentent la réalité physique parfaite, alors que les segments, obtenus par stéréovision, sont une perception bruitée de cette réalité.

Les motifs représentent des ensembles de segments définis par des propriétés géométriques fréquemment observables dans les scènes d'intérieur telles que le parallélisme, la coplanarité, la colinéarité et la connexité. Nous définissons ces propriétés en fonction des arêtes associées aux segments et non pas en fonction des segments, qui ne les vérifient que de manière approximative.

Les définitions données dans ce chapitre ne donnent donc aucune indication sur la manière de construire les motifs à partir des segments. L'objectif est simplement de donner leur signification.

3.2 Définition des motifs

Les six types de motifs que nous utilisons sont les *groupes directionnels*, les *groupes coplanaires*, les *groupes colinéaires*, les *couples connexes*, les *couples colinéaires* et les *couples parallèles*. Voici leur définition :

- les *groupes directionnels* sont des ensembles de segments associés à des arêtes parallèles.

- les *groupes coplanaires* sont des ensembles de segments associés à des arêtes coplanaires,
- les *groupes colinéaires* sont des ensembles de segments associés à des arêtes colinéaires.
- les *couples connexes* (fig. 3.1–a) sont des motifs formés de deux segments représentant des arêtes partageant un même sommet,
- les *couples parallèles* (fig. 3.1–b) sont des motifs constitués de deux segments représentant des arêtes parallèles, proches (la distance qui les sépare doit être inférieure à un seuil S_{pr}) et qui se recouvrent (cf. définition du recouvrement au paragraphe 2.3.7).
- les *couples colinéaires* (fig. 3.1–c) sont des motifs composés de deux segments représentant des arêtes colinéaires.

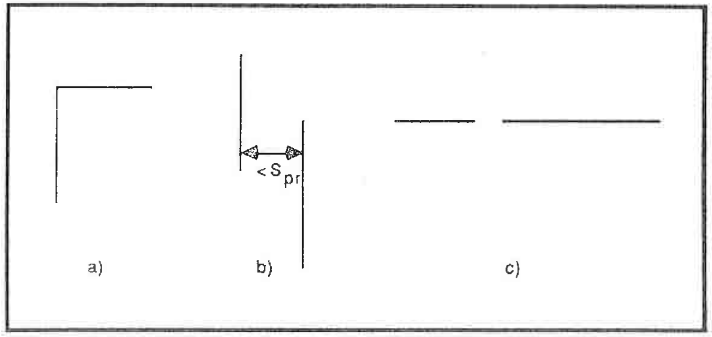


Figure 3.1. Les trois types de couples : (a) connexe (b) parallèle (c) colinéaire.

3.3 Attributs géométriques des entités

A tout modèle est associé un repère unique $R = (O, \vec{I}, \vec{J}, \vec{K})$ dans lequel s'expriment les positions de toutes les entités.

Toute entité e (segment, couple ou groupe) est définie géométriquement dans R par trois attributs : son *centre* (point de $\mathbb{R}^3$), sa *direction* (vecteur de $\mathbb{R}^3$) et sa norme (nombre réel positif). Ces trois attributs sont notés respectivement $C(e)$, $\uparrow(e)$ et $\|e\|$. Voici leur définition (fig. 3.2) :

Le centre

- Le centre d'un segment est son milieu.
- Le centre d'un motif dépend de son type :
  - si c'est un un groupe directionnel, c'est l'origine O du repère ;
  - si c'est un couple connexe, c'est le sommet commun des arêtes associées aux deux segments de ce couple.
  - sinon, le centre est simplement le centre de gravité des centres des segments du motif.

La direction

La direction d'un segment est un vecteur unitaire qui lui est parallèle. La direction d'un motif dépend de son type :

- pour un couple parallèle, un couple colinéaire, un groupe colinéaire ou un groupe directionnel, la direction est un vecteur unitaire parallèle aux arêtes représentées par ses segments :
- pour un couple connexe et un groupe coplanaire, la direction est un vecteur unitaire orthogonal au plan formé par ses segments.

La norme

La norme d'un segment est la moitié de sa longueur. Celle d'un motif est simplement la somme des normes de ses segments.

3.4 Résumé

Dans ce chapitre nous avons défini un système de représentation pour un environnement polyédrique et orthogonal. Dans ce système de représentation, un modèle d'environnement est un ensemble de segments et de motifs. Les motifs sont des segments regroupés selon des propriétés géométriques vérifiées par les arêtes représentées par leurs éléments. On y trouve :

- les groupes directionnels : ensemble de segments correspondant à des arêtes parallèles.
- les groupes coplanaires.

- les groupes colinéaires.
- les couples de segments colinéaires, connexes et parallèles.

Les positions de toutes ces entités sont définies dans un repère unique par trois attributs géométriques : leur centre, leur direction et leur norme.

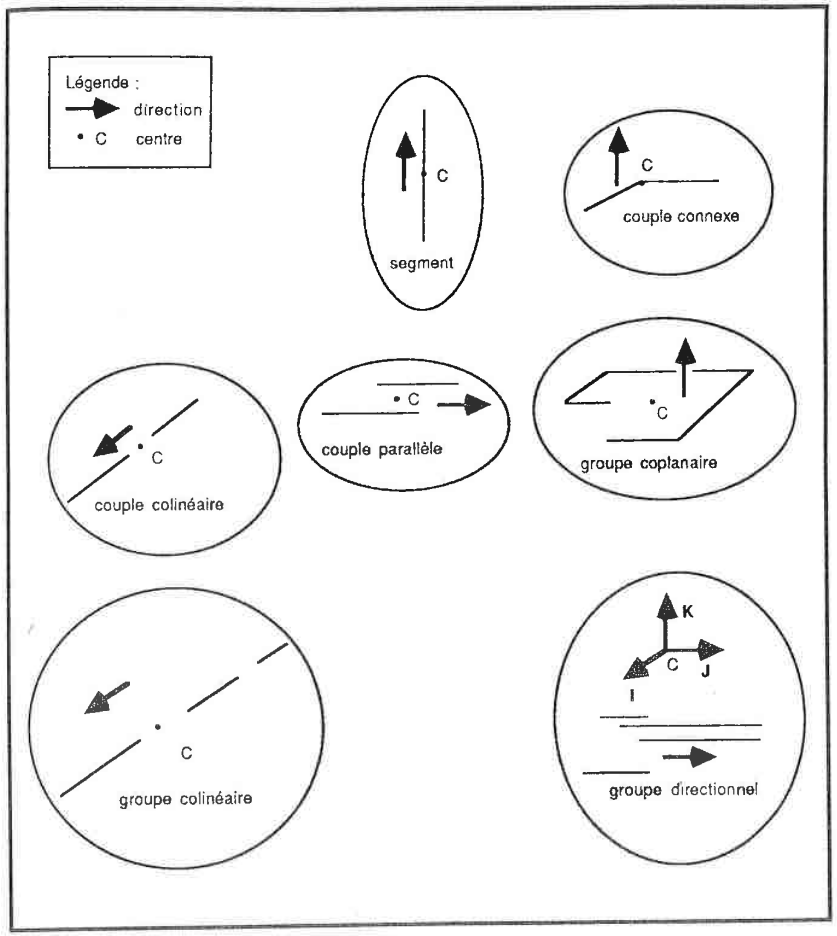


Figure 3.2. Attributs géométriques des entités.

4
Le module de structuration

Dans ce chapitre nous montrons comment les différentes entités définies au chapitre précédent peuvent être extraites d'une vue stéréoscopique. Cette opération est réalisée par le module de structuration. Ce module :

1. définit tout d'abord le repère du modèle à partir des points de fuite.
2. exprime ensuite les coordonnées des segments de la vue dans ce nouveau repère.
3. construit parallèlement les groupes directionnels et les segments du modèle.
4. créé les différents couples de segments.
5. en déduit les groupes colinéaires et coplanaires.
6. corrige les segments et calcule les attributs géométriques des motifs.

Décrivons à présent chacune de ces étapes.

4.1 Construction du repère

4.1.1 Objectif

Le repère R du modèle est défini de manière à simplifier, lors de l'étape de mise en correspondance, l'estimation du déplacement entre les repères de deux modèles :

- l'origine O est confondue avec celle du repère stéréoscopique $(O_R, \vec{I}_R, \vec{J}_R, \vec{K}_R)$. D'après l'hypothèse 7, la hauteur de O_R varie très faiblement d'une vue à l'autre. La convention $O = O_R$, nous permet donc supposer (en première approximation) que la composante z du déplacement entre deux repères de modèle est nulle.
- les trois vecteurs $(\vec{I}, \vec{J}, \vec{K})$ sont définis de la manière suivante :
  - $\vec{K}$ est parallèle à la direction de la verticale et dirigé vers le haut.

- $\vec{I}$ est parallèle à une des deux directions majoritaires horizontales de l'environnement.
- $\vec{J}$ est perpendiculaire à $\vec{I}$ et $\vec{K}$.

Avec cette définition, le repère d'un modèle n'a que quatre orientations possibles par rapport à l'environnement (fig. 4.1). Connaissant une de ces orientations, les trois autres s'en déduisent par des rotations de $\pi/2$, π et $3\pi/2$ autour d'un axe vertical.

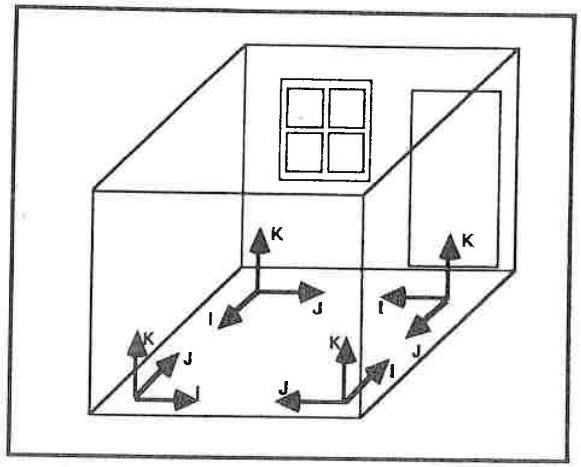


Figure 4.1. Orientation possible du repère d'un modèle

4.1.2 Méthode

Nous donnons ici une méthode permettant de calculer les coordonnées de $(\vec{I}, \vec{J}, \vec{K})$ dans le repère stéréoscopique à partir des points de fuite associés à la caméra de référence et de sa matrice de projection.

D'après l'hypothèse d'orthogonalité (hypothèse 2), il y a dans toute image de l'environnement au plus trois points de fuite correspondant aux directions majoritaires.

D'autre part, connaissant la matrice de projection de la caméra de référence dans le repère stéréoscopique on peut calculer, pour tout point de fuite p, les deux vecteurs unitaires $\vec{u_1}(p)$ et $\vec{u_2}(p)$ définissant la direction associée à p (la formule est donnée au paragraphe 1.3).

L'hypothèse 8 sur le repère stéréoscopique nous permet de différencier le point de fuite p_v associé à la direction verticale par rapport aux autres points de fuite. En effet, si le vecteur $\vec{J}_R$ du repère stéréoscopique est approximativement vertical, p_v est le point de

fuite dont les deux vecteurs unitaires maximisent $|Y|$. Les autres sont considérés comme des points de fuite associés aux directions horizontales.

Partant de p_v et d'un point de fuite associé à une direction horizontale p_h, les trois vecteurs du repère peuvent être définis de la manière suivante (fig. 4.2) :

- $\vec{I}$ est un des deux vecteurs unitaires $\vec{u_1}(p_h)$, $\vec{u_2}(p_h)$ associé à p_h (cf paragraphe 1.3).
- le vecteur $\vec{K}$ est déduit des vecteurs unitaires $\vec{u_1}(p_v)$, $\vec{u_2}(p_v)$ associés au point de fuite correspondant à la verticale. Il faut tout d'abord déterminer lequel de ces deux vecteurs est dirigé vers le haut. On utilise ici le fait que le vecteur $\vec{J}_R$ du repère stéréoscopique est dirigé vers le bas (hypothèse 8).

  Soit K' le vecteur unitaire associé à p_v et dirigé vers le haut (Y négatif). Etant donné l'erreur sur le calcul de p_v et p_h, $\vec{K'}$ n'est pas nécessairement exactement perpendiculaire à $\vec{I}$. On définit donc $\vec{K}$, comme le vecteur appartenant au plan $(\vec{I}, \vec{K'})$ et perpendiculaire à $\vec{I}$:

$$\vec{K} = \frac{\vec{I} \wedge \vec{K'}}{\|\vec{I} \wedge \vec{K'}\|} \wedge \vec{I}$$

- $\vec{J} = \vec{I} \wedge \vec{K}$.

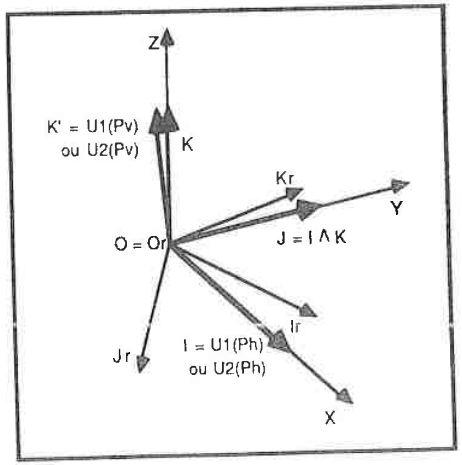


Figure 4.2. Définition du repère à partir des points de fuite.

4.2 Changement de repère

Le changement de repère consiste à passer du repère stéréoscopique $(O_R, \vec{I}_R, \vec{J}_R, \vec{K}_R)$ au repère $(O, \vec{I}, \vec{J}, \vec{K})$. Par définition O est confondu avec O_R. La transformation permettant d'effectuer le changement de repère est donc une rotation R dont l'axe passe par O_R. Un vecteur directeur $\vec{A}$ de cet axe est donné par :

$$\vec{A} = (\vec{I} - \vec{I}_R) \wedge (\vec{J} - \vec{J}_R)$$

L'angle de rotation θ est l'angle entre les deux vecteurs $\vec{A} \wedge \vec{I}$ et $\vec{A} \wedge \vec{J}$.

Cette rotation est appliquée à toutes les extrémités des segments stéréoscopiques.

4.3 Formation des groupes directionnels et des segments

D'après l'hypothèse 2, la majorité des segments stéréoscopiques sont parallèles à une des trois directions majoritaires de l'environnement qui sont également les directions des trois vecteurs $\vec{I}$, $\vec{J}$ et $\vec{K}$ du repère. Le module de structuration ne forme donc que les trois groupes directionnels parallèles correspondant à ces trois directions.

Les segments du modèle et les trois groupes directionnels sont formés simultanément. Pour chaque segment stéréoscopique s de milieu m et de longueur l :

- si s fait un angle inférieur à S_a avec $\vec{K}$, on lui associe un segment du modèle de centre m, de norme $l/2$ et de direction $\vec{K}$. Ce segment est ajouté au groupe directionnel V.
- si s fait un angle inférieur à S_a avec $\vec{I}$, on lui associe un segment du modèle de centre m, de norme $l/2$ et de direction $\vec{I}$. Ce segment est ajouté au groupe directionnel H_1.
- si s fait un angle inférieur à S_a avec $\vec{J}$, on lui associe un segment du modèle de centre m, de norme $l/2$ et de direction $\vec{J}$. Ce segment est ajouté au groupe directionnel H_2.
- sinon, s est ignoré.

Les segments n'appartenant à aucun groupe directionnel ne sont donc pas représentés dans le modèle.

4.4 Formation des couples

4.4.1 Principe général

L'erreur sur la position des segments image provoque une erreur sur les coordonnées des segments stéréoscopiques qui s'amplifie avec la distance. Ceci pose un problème important pour la construction des couples. Mais, deux faits intéressants permettent de contourner ce problème :

- en projetant les segments stéréoscopiques (fig. 4.3) sur le plan image d'une des caméras, on obtient des segments moins bruités (fig. 4.4).

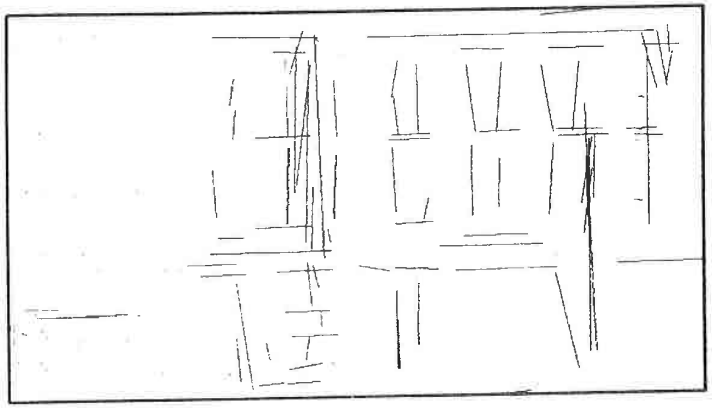


Figure 4.3. Projection verticale de segments stéréoscopiques

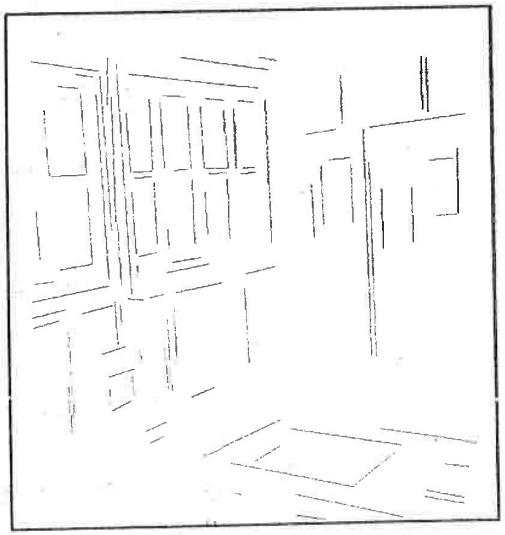


Figure 4.4. Segments stéréoscopiques projetés

- ces segments contiennent encore une grande partie de l'information nécessaire à la formation des couples.

Le principe de construction des couples est donc le suivant :

1. les segments sont projetés sur le plan image de la caméra de référence en utilisant sa matrice m exprimée dans R.
2. les couples sont formés en utilisant des conditions assez restrictives sur les projections des segments et des conditions beaucoup moins restrictives sur leurs attributs 3D.

4.4.2 Conditions associées à chaque type de couple

Couples colinéaires

Deux segments s_1 et s_2 forment un couple colinéaire si (fig. 4.5) :

- ils appartiennent au même groupe directionnel.
- leur distance est inférieure à S_{cl} : $\delta(s_1, s_2) < S_{cl}$.
- leurs projections sont approximativement colinéaires :
  - la distance entre les segments projetés est inférieure à S'_{cl} :
    $$\delta(P^<(s_1, m), P^<(s_2, m)) < S'_{cl}$$
  - ils ne se recouvrent pas : $P^<(s_1, m)/P^<(s_2, m) < 0$

Couples connexes

Deux segments s_1 et s_2 forment un couple connexe si (fig. 4.6) :

1. ils appartiennent à des groupes directionnels distincts.
2. leurs extrémités les plus proches sont à une distance inférieure à S_{cn}.
3. leurs projections sont approximativement connexes. Cette condition fait intervenir :
   (a) les droites portant les segments projetés :
   $$d_1 = D(P^<(s_1, m))$$
   et
   $$d_2 = D(P^<(s_2, m))$$

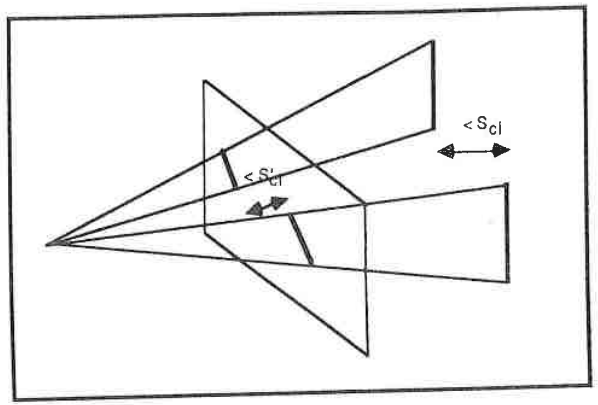


Figure 4.5. Conditions associées à un couple colinéaire

   (b) le point d'intersection I de d_1 et d_2.
   (c) l'extrémité e_1 de $P^<(s_1, m)$ la plus proche de I.
   (d) l'extrémité e_2 de $P^<(s_2, m)$ la plus proche de I.

   Elle est vérifiée lorsqu'une des deux conditions suivantes est réalisée :

   (a) la distance entre p_1 et p_2 est inférieure à S'_{cn} (fig. 4.6-a).
   (b) la distance entre p_1 et p_2 est inférieure à S''_{cn} (avec $S''_{cn} > S_{cn'}$) et la distance de d_1 à p_2 (ou la distance de d_2 à p_1), inférieure à S'''_{cn} (fig. 4.6-b). Autrement dit, la droite portant un des deux segments doit passer très près d'une extrémité de l'autre segment. Cette condition permet de retrouver la connexité même lorsque l'un des deux segments s'arrête assez loin du point d'intersection.

Couples parallèles

Deux segments s_1 et s_2 forment un couple parallèle si (fig. 4.7) :

- leur distance est inférieure à S_{pr} : $\delta(s_1, s_2) < S_{pr}$.
- ils appartiennent au même groupe directionnel.
- leurs projections respectives sont proches :
  $$\delta(P^<(s_1, m), P^<(s_2, m)) < S'_{pr}$$
- et se recouvrent : $P^<(s_1, m)/P^<(s_2, m) > 0$.

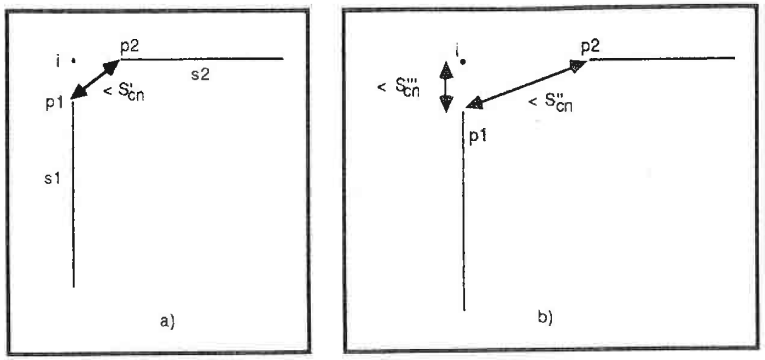


Figure 4.6. Conditions associées aux deux segments projetés d'un couple connexe

4.5 Formation des groupes colinéaires et coplanaires

Les groupes colinéaires et coplanaires sont formés à partir d'un groupe initial que l'on étend itérativement. A chaque itération, tous les segments n'appartenant pas déjà à un groupe sont passés en revue et ajoutés au groupe courant s'ils vérifient une certaine condition d'adjonction. Lorsqu'il n'y a plus d'adjonction possible, on créé un nouveau groupe initial que l'on étend de la même manière, ...

Le résultat dépend donc essentiellement du critère de choix des segments servant à former le groupe initial et du critère d'adjonction d'un segment à un groupe.

Pour les groupes colinéaires, le groupe initial est un couple colinéaire. Un segment peut être ajouté au groupe courant s'il appartient à un couple colinéaire dont l'un des deux éléments appartient déjà au groupe.

Pour un groupe coplanaire, le groupe initial est un couple connexe et un segment peut être ajouté à un groupe en cours de formation si :

1. il appartient à un couple colinéaire ou connexe dont l'un des deux éléments appartient déjà au groupe.

2. il appartient à un des deux groupes directionnels associés aux segments du groupe initial.

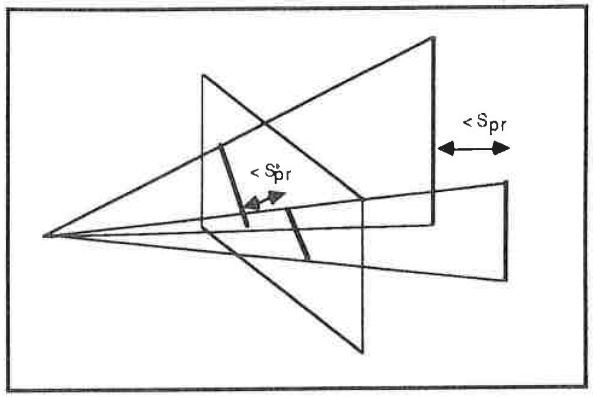


Figure 4.7. Conditions associées à un couple parallèle

4.6 Correction des segments et attributs géométriques des motifs

Le but de cette étape est de calculer les attributs géométriques (centre, direction, norme) des motifs tout en corrigeant les attributs des segments de manière à ce que :

1. les segments appartenant à un même groupe coplanaire soient effectivement coplanaires.

2. les segments appartenant à un même groupe colinéaire soient effectivement colinéaires.

3. les segments appartenant à un couple connexe aient une extrémité commune.

4.6.1 Suite d'opérations

Les opérations de correction et de calcul d'attributs se déroulent dans l'ordre suivant :

1. calcul des attributs des groupes coplanaires.

2. correction de la coplanarité.

3. calcul des attributs des groupes colinéaires.

4. correction de la colinéarité.

5. calcul des attributs des couples connexes.

6. correction de la connexité.

7. calcul des attributs des couples parallèles et colinéaires.

4.6.2 Calcul des attributs

Les attributs géométriques d'un motif m sont calculés de la manière suivante :

- sa norme est la somme des normes de ses segments.
- si c'est un groupe colinéaire, un couple colinéaire ou un couple parallèle, sa direction est celle d'un de ses segments. Si s'agit d'un groupe coplanaire ou d'un couple connexe, sa direction est le vecteur du repère qui n'est pas une direction d'un de ses segments.
- si c'est un couple connexe, son centre est la pseudo-intersection (paragraphe 2.2) des droites portant ses deux segments.
- sinon, son centre est simplement le centre de gravité des centres de ses segments.

4.6.3 Corrections

La correction d'un segment s consiste à modifier sa norme et son centre en fonction des attributs géométriques du motif auquel il appartient. Cette modification est effectuée en considérant successivement chaque motif. Définissons à présent la nouvelle norme n' et le nouveau centre c' d'un segment s pour chaque type de motif.

Groupe coplanaire

Soit :

- g un groupe coplanaire,
- P le plan de normale $\uparrow(g)$ passant par $C(g)$,
- p_1 et p_2 les extrémités de s,
- m la matrice de projection de la caméra de référence,
- $p'_i = P^{<}(p_i, m)$ la projection perspective de p_i selon m,
- $d_i = P^{>}(p'_i, m)$ la droite projection inverse de p'_i selon m,
- p''_i l'intersection de d_i avec P,

On définit n' et c' par :

$$n' = \frac{\|p''_1.p''_2\|}{2}$$

$$c' = C_g(\{p''_1.p''_2\})$$

La correction d'un segment consiste donc à le projeter sur le plan image de la caméra de référence, puis à reprojeter le segment image obtenu sur le plan P, par projection perspective inverse (fig. 4.8).

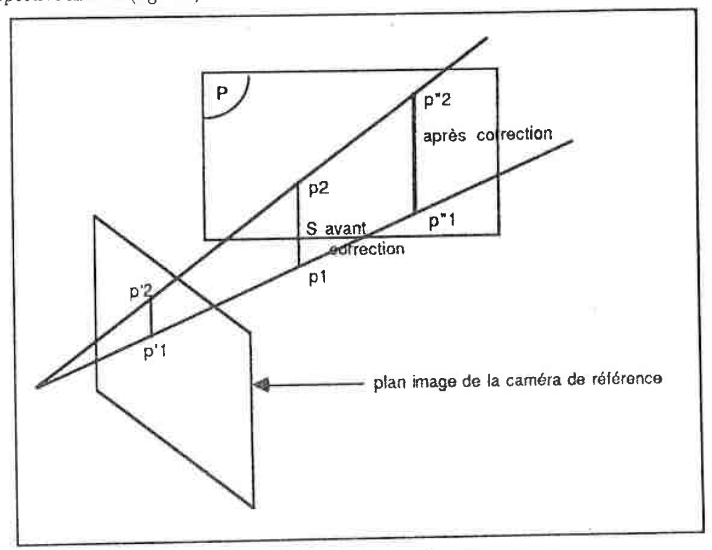


Figure 4.8. Correction de la coplanarité d'un segment.

Groupe colinéaire

Chaque segment d'un groupe colinéaire g est projeté orthogonalement sur la droite D de vecteur directeur $\uparrow(g)$ passant par $C(g)$ (fig. 4.9).

n' et c' sont donc donnés par les relations :

$$n' = \frac{L(P^{\perp}(s, D))}{2}$$

$$c' = M(P^{\perp}(s, D))$$

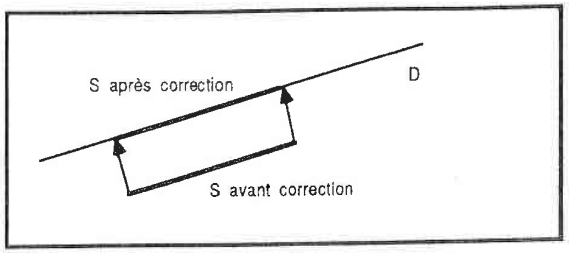


Figure 4.9. Correction de la colinéarité d'un segment.

Couple connexe

Par construction, tout couple connexe est nécessairement inclu dans un groupe coplanaire. Après correction de la coplanarité, les deux segments s_1 et s_2 appartenant à un couple connexe c sont donc nécessairement coplanaires. Il s'ensuit que le centre du couple est exactement l'intersection des deux droites portant s_1 et s_2. Soit alors I, l'intersection de ces deux droites et p_i le point de s_i le plus éloigné de I. La correction de la connexité d'un segment s_i consiste à remplacer son extrémité la plus proche de I par I (fig. 4.10). On a donc :

$$n' = \frac{\|I, p_i'\|}{2}$$

$$c' = C_g(\{I, p_i'\})$$

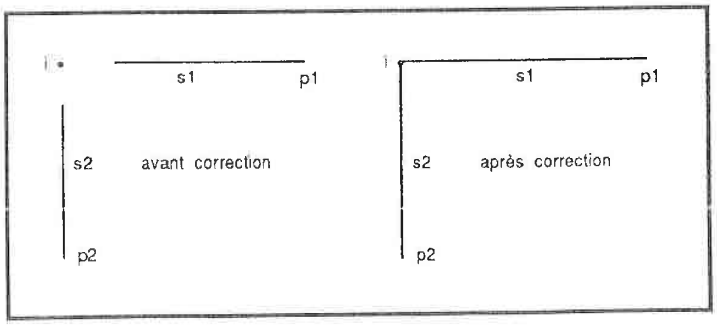


Figure 4.10. Correction de la connexité d'un segment

4.7 Importance et détermination des seuils

Les seuils utilisés par le module de structuration ont une influence sur la qualité et la quantité des motifs construits. En augmentant la valeur de ces seuils, le risque de ne pas retrouver certains motifs diminue mais le risque d'erreur croît. Nous avons donc déterminé les seuils de structuration de manière à obtenir un nombre suffisant de motifs avec un nombre minimal de motifs erronés. Ceci a été effectué de manière expérimentale en visualisant les résultats pour différentes valeurs de seuil. Voici un récapitulatif des différents seuils utilisés et de leur valeur :

- S_a
  - Signification : seuil relatif à la formation des groupes directionnels (cf. paragraphe 4.3).
  - Valeur fixée à : 20 degrés.
- S_{cl}, S'_{cl}
  - Signification : seuils relatifs à la formation des couples colinéaires (cf. paragraphe 4.4.2).
  - Valeur fixée à : 50 cm pour S_{cl} et 3 pixels pour S'_{cl}.
- S_{cn}, S'_{cn}, S''_{cn}, S'''_{cn}.
  - Signification : seuils relatifs à la formation des couples connexes (cf. paragraphe 4.4.2).
  - Valeur fixée à : 50 cm pour S_{cn}, 7 pixels pour S'_{cn}, 15 pixels pour S''_{cn} et 2 pixels pour S'''_{cn}.
- S_{pr}, S'_{pr}
  - Signification : seuils relatifs à la formation des couples parallèles (cf. paragraphe 4.4.2)
  - Valeur fixée à : 20 cm pour S_{pr} et 15 pixels pour S'_{pr}.

4.8 Résultats

Nous commentons ici les résultats de la structuration des vues 0 (annexes C), 1 (annexe D) et 15 (grille de calibration, annexe E) produisant les modèles MV_0, MV_1 et MV_{15}.

4.8.1 Formation des motifs

Pour visualiser les motifs, les conventions suivantes ont été adoptées :

- les segments sont présentés par projection perspective sur le plan image de la caméra de référence.
- les segments appartenant à un motif sont affichés en gras.
- les segments appartenant à un même groupe sont étiquetés par le numéro du groupe auquel ils appartiennent.
- chaque couple est visualisé par un segment reliant les milieux des deux segments qu'il contient. Ce segment est étiqueté par le numéro du couple. Pour la grille de calibration, ces numéros ne sont pas donnés afin de rendre les figures plus visibles.
- les erreurs sont entourées d'un cercle étiqueté par une lettre.
- les couples manquants sont notés par un arc de cercle étiqueté par une lettre.

On peut constater que le nombre d'erreurs dans la formation des motifs est relativement faible :

- au niveau des groupes directionnels :
  - dans la vue 0, qui contient 106 segments, un seul segment est mal classé (erreur *a* fig. C.6).
  - dans la vue 1, qui contient 97 segments, deux segments sont mal classés (erreurs *a* et *b* fig. D.6)

  Ces erreurs sont dues à une très mauvaise orientation des segments stéréoscopiques qui a plutôt tendance à se produire pour les petits segments.
- au niveau des couples connexes :
  - dans la vue 0, contenant 25 couples connexes, cinq couples sont erronés (erreurs *a*, *b*, *c*, *d* et *e* fig. C.1)
  - dans la vue 1, contenant 28 couples connexes, cinq couples sont erronés (erreurs *a*, *b*, *c*, *d* et *e* fig. D.1).
  - dans la vue de la grille de calibration, contenant 334 couples connexes, sept couples sont faux (fig E.3).
- au niveau des couples colinéaires :

  - dans la vue 0, contenant 33 couples colinéaires, quatre couples sont faux (erreurs *a*, *b*, *c* et *d* fig. C.3).
  - dans la vue 1, contenant 25 couples colinéaires, trois couples sont faux (erreurs *a*, *b* et *c* fig. D.3).
  - dans la vue de la grille de calibration, contenant 385 couples colinéaires, douze couples sont faux (fig. E.4).

Bien entendu, les erreurs dans la formation des couples connexes et colinéaires se répercutent ensuite dans les groupes colinéaires et coplanaires qui s'en déduisent. Par exemple :

- l'erreur *a* dans le modèle MV_0 (fig. C.5), provient de l'erreur *b* dans la formation des couples connexes (fig. C.1) et de l'erreur *b* dans la formation des couples colinéaires (fig. C.3).
- l'erreur *a* dans le modèle MV_1 (fig. D.5), provient de l'erreur *a* dans la formation des couples connexes (fig. D.1).

On peut également observer que tous les motifs que l'on souhaiterait retrouver ne sont pas effectivement retrouvés. Ce phénomène se produit surtout pour les couples connexes. Par exemple, dans les vues 0 et 1, environ un tiers des couples connexes sont perdus (erreurs *f*, *g*, ...*o* fig. C.1 et *f*, *g*, ...*n* fig. D.1).

D'autre part, la formation des groupes coplanaires n'est pas aussi complète qu'on le souhaiterait. Par exemple, dans les vue 0 et 1 (fig. C.5 et fig. D.5), on aimerait pouvoir obtenir un seul groupe coplanaire pour les segments appartenant à la table et de même pour le plan des armoires. En réalité on en obtient plusieurs ou aucun (plan de la table dans la figure D.5).

4.8.2 Correction des segments

Les améliorations apportées par la correction des segments dans les vues 0 à 7 peuvent être observées en comparant les figures se trouvant sur une même page dans l'annexe B. Les repères des vues stéréoscopiques et des modèles correspondants sont visualisés avec une graduation tout les 50 cm. On voit sur ces figures que le repère du modèle est bien aligné sur les murs contrairement au repère stéréoscopique. Les améliorations sur la vue de la grille de calibration sont visibles dans la figure E.2, à comparer avec la figure E.1.

Rappelons que les segments sont corrigés de plusieurs manières :

- lors de la construction des groupes directionnels, les segments sont rendus strictement parallèles à la direction du groupe auquel ils appartiennent. Ceci apporte une correction dans la direction des segments.

- lors de l'étape de correction :
  - les segments appartenant à un même groupe coplanaire sont rendus exactement coplanaires.
  - les segments appartenant à un même groupe colinéaire sont rendus exactement colinéaires.
  - les segments appartenant à un couple connexe sont rendus exactement connexes.

Orientation

Dans les différents modèles créés, on observe que la majorité des segments sont parfaitement orientés. Toutefois, des erreurs importantes peuvent se produire lorsqu'un segment est classé dans un mauvais groupe directionnel.

Connexité

Les effets de la correction de la connexité sont visibles au niveau des vitres (fig. B.2, B.4, B.6), des posters (fig. B.2, B.4, B.6, B.8, B.10, B.12) et des carreaux de la grille de calibration (fig. E.2).

Bien entendu, les erreurs dans la formation des couples connexes se répercutent dans la correction des segments.

Colinéarité

La correction de la colinéarité est apparente dans l'alignement des bords de vitres (fig. B.2, B.4, B.6) et des carreaux de la grille de calibration (fig. E.2).

Les erreurs dans la formation des couples colinéaires se répercutent également dans la correction. Mais les répercussions de ces erreurs sont difficiles à observer dans les figures.

Coplanarité

Deux effets intéressants de la correction de la coplanarité peuvent être notés :

- les carreaux de vitres dans le modèle MV_0 appartiennent tous à un même plan.
- les segments de la grille de calibration appartiennent presque tous à un même plan.

4.9 Conclusion

Dans ce chapitre, nous avons proposé une méthode permettant de déduire d'une vue stéréoscopique des ensembles de segments, appelés motifs, vérifiant certaines propriétés

géométriques fréquentes dans les scènes d'intérieur comme le parallélisme (groupes directionnels et couples parallèles), la coplanarité (groupes coplanaires), la colinéarité (groupes et couples colinéaires) et la connexité (couples connexes).

L'idée essentielle de cette méthode, est de baser la formation des motifs sur les segments bidimensionnels, plus précis que les segments tridimensionnels obtenus par stéréovision :

- la formation des groupes directionnels s'appuie sur les vecteurs du repère du modèle définis à partir des points de fuite. Or les points de fuites sont calculés à partir des segments image associés à la caméra de référence.
- les couples de segments sont extraits des projections des segments stéréoscopiques sur le plan image de la caméra de référence.
- les groupes coplanaires et colinéaires sont simplement déduits des couples colinéaires et connexes.

5
Le module de mise en correspondance

5.1 Principe général

La mise en correspondance de deux modèles a pour but de préparer leur fusion. Elle détermine le déplacement entre leur repères respectifs et établit un ensemble d'hypothèses de correspondance entre leurs segments. Les étapes de mise en correspondance d'un modèle M_1 avec un modèle M_2 sont les suivantes :

1. construction d'une liste de déplacements possibles entre les deux repères.
2. répétition d'une procédure d'appariement et affinement du déplacement pour chaque déplacement de la liste précédente. A chaque itération, cette procédure produit une nouvelle estimation du déplacement ainsi qu'un ensemble de correspondants admissibles pour chaque segment de M_1. Si le déplacement utilisé est suffisamment précis, on obtient une proportion importante de segments appariés (possédant au moins un correspondant) et un déplacement plus précis. L'itération s'arrête donc dès que l'on obtient un appariement satisfaisant, c'est-à-dire lorsque la proportion de segments appariés de M_1 est supérieure à un seuil S_{as}.
3. sélection : si l'étape précédente permet de trouver un déplacement satisfaisant, on choisit les meilleurs correspondants parmi les correspondants admissibles de chaque segment.

5.2 Construction de la liste des déplacements possibles

5.2.1 Principe

Rappelons que, par construction du repère d'un modèle (paragraphe 4.1), il n'y a que quatre rotations possibles entre les repères de M_1 et M_2 :

- R^0 : rotation d'angle nul.
- R^1 : rotation d'angle $\pi/2$ autour d'un axe vertical.
- R^2 : rotation d'angle π autour d'un axe vertical.
- R^3 : rotation d'angle $3\pi/2$ autour d'un axe vertical.

La liste des déplacements est construite en produisant une liste de translations possibles $T_1^i \ldots T_{k_i}^i$ pour chaque rotation R_i (fig. 5.1). Lors de l'estimation des translations possibles pour une rotation donnée, chaque translation T_j^i est associée à un poids p_j^i. La liste des déplacements possibles est l'ensemble des couples (R^i, T_j^i) ordonné de manière décroissante selon p_j^i.

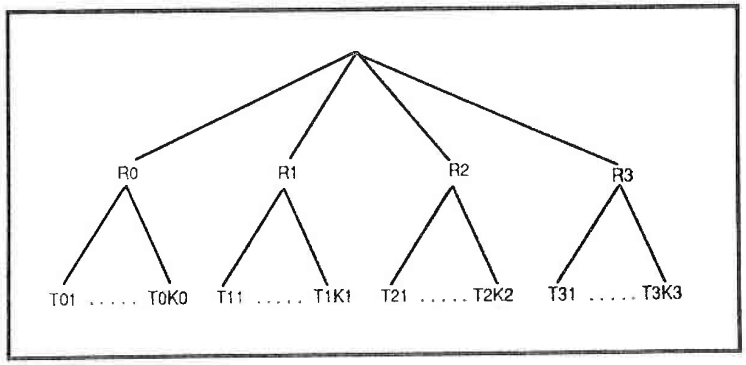


Figure 5.1. Construction de la liste des déplacements possibles

5.2.2 Construction des translations possibles

Principe

Voyons à présent comment est construite la liste des translations possibles pour une rotation donnée. Le principe est le suivant : lorsque la rotation est connue il suffit d'une correspondance entre un point de M_1 et un point de M_2 pour avoir une estimation de la translation. De plus, étant donné que l'environnement est statique (hypothèse 3), on peut supposer que toutes les correspondances exactes donnent *approximativement* (à cause du bruit) la même translation.

Les translations possibles peuvent donc être recherchées par transformée de Hough :

1. établir un ensemble de couples formés d'un point de M_1 et d'un point de M_2 pouvant correspondre à M_1. Ces couples de points sont appelés couples de points admissibles.

2. calculer les translations associées à chaque couple.
3. construire un tableau T à trois dimensions tel que $T(i,j,k)$ contienne toutes les translations (t_x, t_y, t_z) telles que :

$$x_0 + i\,d \leq t_x < x_0 + (i+1)\,d$$
$$y_0 + j\,d \leq t_y < y_0 + (j+1)\,d$$
$$z_0 + z\,d \leq z_0 < z_0 + (k+1)\,d$$

   Chaque élément de T correspond donc à un cube de dimension d. Les dimensions du tableau et les constantes x_0, y_0 et z_0 doivent être définies de telle sorte que toute translation appartienne à un de ses élément.
4. rechercher les points d'accumulation, c'est-à-dire les éléments du tableau contenant le plus de translations.
5. calculer une estimation de la translation pour chaque point d'accumulation à partir des translations qu'il contient.

Construction des couples de points admissibles

Les points considérés sont les centres des couples connexes et les centres des couples parallèles verticaux. Deux couples sont utilisés pour calculer un couple de points admissibles si :

- la différence de leur norme est inférieure à S_{nc}.
- la différence de hauteur entre leurs centres respectifs est inférieure à S_{hc}.

Construction du tableau

Dans notre cas, le déplacement est approximativement horizontal (d'après l'hypothèse 7). On peut donc se ramener à une recherche de translation dans un espace à deux dimensions. On pose $t_z = 0$ pour toutes les translations. T est un tableau à deux dimensions construit à partir des composantes t_x et t_y. Chaque élément correspond à un carré de dimension S_r.

Recherche des points d'accumulation

Pour définir les points d'accumulation, nous calculons le nombre maximum de translations dans un élément de T. Puis nous considérons qu'un élément de T est un point d'accumulation s'il contient un nombre de translations supérieur à une proportion S_{pa} de ce maximum.

Calcul des translations

La translation associée à un point d'accumulation est le centre de gravité de ses éléments. Le poids de cette translation est le nombre d'éléments que contient le point d'accumulation.

5.3 Appariement et affinement du déplacement

5.3.1 Principe

Le principe de fonctionnement de cette procédure est le suivant :

- chaque segment s_1 de M_1 est comparé avec ses *correspondants potentiels* dans M_2.
- si s_1 et s_2 vérifient une certaine *condition d'appariement*, le couple (s_1, s_2) est ajouté à la liste des hypothèses.
- chaque fois que l'on réussit à trouver au moins un correspondant pour s_1, on considère son *meilleur correspondant* s_2^*.
- l'hypothèse (s_1, s_2^*) est utilisée pour *affiner le déplacement*.

Il nous reste donc à préciser :

- comment sont définis les correspondants potentiels d'un segment.
- les critères intervenant dans la condition d'appariement.
- la définition du "meilleur correspondant" d'un segment.
- le principe d'affinement du déplacement à partir d'une hypothèse de correspondance.

5.3.2 Correspondants potentiels

L'estimation courante du déplacement circonscrit les correspondants possibles de chaque segment de M_1. Afin d'accélérer leur recherche, nous divisons l'espace de M_2 en *baquets* selon un principe tiré de l'adressage dispersé [Knu 75] [Asa 85].

Dans notre cas, les baquets sont les éléments de trois tableaux à deux dimensions T_I, T_J et T_K contenant chacun une liste de segments de M_2. T_I (respectivement T_J et T_K) contient tous les segments de M_2 parallèles au vecteur $\vec{I}$ (respectivement $\vec{J}$ et $\vec{K}$) du repère de M_2. Ces tableaux sont construits de la manière suivante :

- soient s, un segment de $S(M_2)$ et (x, y, z) les coordonnées de son milieu dans $R(M_2)$.

- si s est parallèle à $\vec{I}$, il est rangé dans l'élément $T_I(i, j)$ tel que :

$$Y_0 + i\,S_{cp} \leq y < Y_0 + (i+1)\,S_{cp}$$

$$Z_0 + j\,S_{cp} \leq z < Z_0 + (j+1)\,S_{cp}$$

- si s est parallèle à $\vec{J}$, il est rangé dans l'élément $T_J(i, j)$ tel que :

$$X_0 + i\,S_{cp} \leq x < X_0 + (i+1)\,S_{cp}$$

$$Z_0 + j\,S_{cp} \leq z < Z_0 + (j+1)\,S_{cp}$$

- si s est parallèle à $\vec{K}$, il est rangé dans l'élément $T_K(i, j)$ tel que :

$$X_0 + i\,S_{cp} \leq x < X_0 + (i+1)\,S_{cp}$$

$$Y_0 + j\,S_{cp} \leq y < Y_0 + (j+1)\,S_{cp}$$

Les constantes X_0, Y_0 et Z_0 ainsi que les dimensions respectives de chaque tableau sont définies de telle sorte que T contienne tous les segments de M_2.

Nous pouvons à présent définir l'ensemble des correspondants potentiels d'un segment s de M_1. Soient :

- s', le segment obtenu en déplaçant s dans M_2 par la valeur courante du déplacement ;
- (x, y, z), les coordonnées du milieu de s' ;
- $\vec{u}$, la direction de s' ;
- T_u le tableau correspondant à $\vec{u}$.

On calcule les indices i et j de l'élément de T_u correspondant aux coordonnées (x, y, z). Les correspondants potentiels de s sont alors tous les segments de $T_u(i, j)$ ainsi que les segments appartenant aux éléments voisins de $T_u(i, j)$.

5.3.3 Condition d'appariement

Soit :

- s_1, un segment de M_1 ;
- s_2, un segment de M_2 ;
- s_1', l'image de s_1 par la valeur courante du déplacement.

s_1 et s_2 vérifient la condition d'appariement si :

- la distance entre la droite portant s'_1 et la droite portant s_2 est inférieure à S_{ca} ;
- s'_1 et s_2 se recouvrent : $s'_1/s_2 > 0$;
- si s'_1 et s_2 sont verticaux ou bien, s'ils sont horizontaux et que la différence de hauteur entre leurs milieux respectifs est inférieure à S'_{ca}.

5.3.4 Meilleur correspondant

Le meilleur correspondant de s_1, parmi tous ceux qui vérifient la condition d'appariement est celui dont la droite support est à une distance minimale de la droite support de s_1.

5.3.5 Affinement du déplacement

Principe

La composante de rotation du déplacement étant fixée, l'affinement du déplacement modifie le vecteur de translation $\vec{T}$ ainsi que le poids P associé à cette translation. Etant donné une hypothèse de correspondance (s_1, s_2), la mise à jour du déplacement se produit en deux étapes :

1. calcul d'une *mesure de translation* dépendant de s_1 et de s_2. La mesure est un couple $(\vec{T}(s_1, s_2), p)$ où :
   - $\vec{T}(s_1, s_2)$ est un vecteur représentant la translation.
   - p est un poids associé à ce vecteur.
2. mise à jour de l'estimation courante de la translation en fonction de la mesure calculée à l'étape précédente.

Calcul d'une estimation de la translation

Si l'on connaît deux points p_1 et p_2 de M_1 et M_2 qui se correspondent, on peut obtenir une estimation (t_x, t_y, t_z) de la translation (coordonnées de $\vec{T}(s_1, s_2)$) par :

$$t_x = x'_1 - x_2$$

$$t_y = y'_1 - y_2$$

$$t_z = z'_1 - z_2$$

(x_2, y_2, z_2) étant les coordonnées de p_2 et (x'_1, y'_1, z'_1), les coordonnées de l'image de p_1 par la composante de rotation du déplacement.

La précision du résultat dépend du choix des points considérés. Une première idée serait d'utiliser les milieux de chaque segment. Mais cette méthode est peu précise car elle dépend trop de la position des extrémités des segments. Il serait préférable de tenir compte uniquement des droites portant les deux segments. Malheureusement, une correspondance entre deux droites n'est pas suffisante pour déterminer la translation. Nous utilisons donc une correspondance entre couples de droites (fig. 5.2) :

- s_1 est associé à un couple de droites $(D_1(s_1), D_2(s_1))$.
- s_2 est associé à un couple de droites $(D_1(s_2), D_2(s_2))$.

Ces couples de droites sont déterminés de la manière suivante :

- $D_1(s_1)$ est la droite portant s_1 et $D_1(s_2)$, la droite portant s_2.
- $D_2(s_1)$ est la droite portant un autre segment s'_1 déjà apparié et non parallèle à s_1.
- $D_2(s_2)$ est la droite portant le meilleur correspondant s'_2 de s'_1.

On définit alors p_1 (respectivement p_2) comme la pseudo-intersection de $D_1(s_1)$ avec $D_2(s_1)$ (respectivement, la pseudo-intersection de $D_1(s_2)$ avec $D_2(s_2)$).

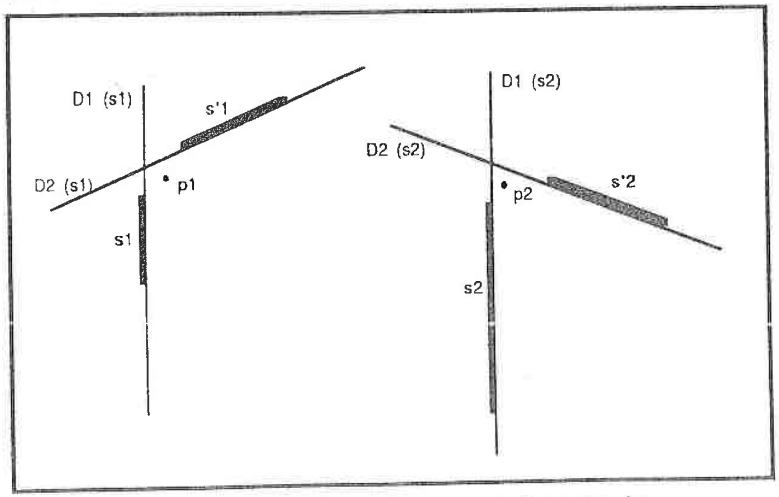


Figure 5.2. Couples de droites considérées pour estimer la translation

Poids associé à la translation

Le poids associé à la translation est fonction de la "qualité de correspondance" des hypothèses utilisées. Celle-ci est estimée en fonction de la distance d_1 entre $D_1(s_1)$ et $D_1(s_2)$ ainsi que la distance d_2 entre $D_2(s_1)$ et $D_2(s_2)$. La formule utilisée est :

$$p = \frac{2\,S_{ca}}{d_1 + d_2}$$

étant que d_1 et d_2 ne peuvent pas dépasser S_{ca} (à cause de la condition d'appariement), p est nécessairement compris entre 0 et 1.

Mise à jour de la translation

Après avoir calculé une nouvelle mesure ($\vec{T}(s_1, s_2)$), la translation courante $\vec{T}$ est remplacée par :

$$\frac{p\vec{T}(s_1, s_2) + P\vec{T}}{p + P}$$

et son poids, par :

$$P = p + P$$

Au départ, le poids P est simplement le poids associé à la l'estimation initiale extraite de la liste des déplacements possibles. Ce poids initial est donc le nombre de translations dans le point d'accumulation correspondant (parapraphe 5.2.2, calcul des translations).

5.4 Sélection finale des meilleures hypothèses

Après la dernière itération de la procédure d'appariement, chaque segment de M_1 est associé à un ensemble de correspondants admissibles. L'étape de sélection permet de retenir le ou les meilleurs correspondants pour chaque segment. Soient :

- s, un segment de M_1 :
- D, la droite portant l'image de s par l'estimation finale du déplacement ;
- $s_1, \ldots s_n$, les correspondants admissibles de s :
- $D_1, \ldots D_n$, les droites portant respectivement $s_1, \ldots s_n$;
- d_i, la distance entre D et D_i ;
- d_{min}, la plus petite distance parmi $d_1, \ldots d_n$.

Les correspondants sélectionnés sont tous les s_i qui se trouvent à un distance minimale de s avec un certain seuil de tolérance S_{sl}, c'est-à-dire tels que :

$$d_i \leq d_{min} + S_{sl}$$

5.5 Importance et détermination des seuils

Les différents seuils utilisés par le module de correspondance ont été déterminés expérimentalement en visualisant, pour différentes valeurs de seuils :

- les déplacements initiaux possibles (pour définir les valeurs de S_{nc}, S_{hc}, S_c et S_{pa}).
- les appariements et le déplacement final obtenu (pour définir les valeurs de S_{as}, S_{cp}, S_{ca}, S'_{ca} et S_{cl}).

Voici un récapitulatif des différents seuils, précisant les influences négatives d'une valeur trop faible ou trop élevée et la valeur obtenue expérimentalement :

- S_{as} : proportion minimale de segments devant être appariés pour valider un déplacement (cf. paragraphe 5.1).
  - Influence sur les résultats :
    - si la valeur de S_{as} est trop faible, des déplacements erronés peuvent être validés.
    - si la valeur de S_{as} est trop importante, un déplacement satisfaisant peut être invalidé.
  - Valeur fixée à : 0.3
- S_{nc} : différence maximale entre les normes de deux couples connexes ou parallèles dans la détermination des couples de points admissibles (cf. paragraphe 5.2.2).
  - Influence sur les résultats :
    - si la valeur de S_{nc} est trop faible, le nombre de couples de points admissibles est insuffisant.
    - si la valeur de S_{nc} est trop importante, de nombreux points qui ne se correspondent pas sont tolérés. Ceci augmente la proportion de mauvaises estimations de translation dans T. Une deuxième conséquence est l'accroissement du temps de calcul dans la construction des déplacements possibles.
  - Valeur fixée à : 30 cm.

- S_{hc} : différence maximale de hauteur entre les centres de deux couples connexes ou parallèles dans la détermination des couples de points admissibles (cf. paragraphe 5.2.2).
  - Influence sur les résultats : identique à celle de S_{nc}.
  - Valeur fixée à : 20 cm.
- S_c : dimension des carrés correspondant aux éléments de T (cf. paragraphe 5.2.2, construction du tableau).
  - Influence sur les résultats :
    * une valeur de S_c trop faible conduit à un tableau T de dimension importante avec peu de translations dans chaque élément. Les points d'accumulations deviennent nombreux et le poids associé à chaque translation possible devient peu fiable.
    * une valeur de S_c trop importante implique une faible précision des translations possibles calculées.
  - Valeur fixée à : 15 cm.
- S_{pa} : rapport minimal entre le nombre de translations contenues dans un élément e de T et le nombre maximal de translations contenues dans un élément de T, pour que e soit considéré comme un point d'accumulation.
  - Influence sur les résultats :
    * si la valeur de S_{pa} est trop faible, on retient un grand nombre de déplacements.
    * si la valeur de S_{pa} est trop importante, la liste des déplacements possibles est très réduite et le risque que le meilleur déplacement initial ne soit pas contenu dans cette liste est grand.
  - Valeur fixée à : 0.6
- S_{cp} : dimension d'un baquet définissant les correspondants potentiels d'un segment (cf. paragraphe 5.3.2).
  - Influence sur les résultats :
    * une valeur trop faible de S_{cp} implique que de nombreux correspondants ne seront pas retrouvés.
    * une valeur trop elevée de S_{cp} implique un temps de calcul important.
  - Valeur fixée à : 50 cm.

- S_{ca} : distance maximale entre segments, dans la condition d'appariement (cf. paragraphe 5.3.3).
  - Influence sur les résultats :
    * avec une valeur trop faible de S_{ca}, beaucoup de segments possédant un correspondant ne sont pas appariés.
    * avec une valeur trop élevée, de mauvais appariements peuvent perturber la mise à jour du déplacement.
  - Valeur fixée à : 20 cm.
- S'_{ca} : différence maximale de hauteur entre deux segments horizontaux, dans la condition d'appariement (cf. paragraphe 5.3.3).
  - Influence sur les résultats : identique à celle de S_{ca}.
  - Valeur fixée à : 10 cm.
- S'_{cl} : tolérance dans la sélection finale des correspondants (cf. paragraphe 5.4).
  - Influence sur les résultats :
    * une valeur élevée augmente le risque d'erreur.
    * avec une valeur trop faible certains correspondants (au cas où le segment en possède plusieurs) risquent de ne pas être retenus.
  - Valeur fixée à : 0.5 cm.

5.6 Résultats

Nous donnons ici les résultats de la mise en correspondance entre les modèles associés à des vues successives (MV_i et MV_{i-1} pour i allant de 1 à 7).

L'annexe G montre les correspondances et les déplacements obtenus dans chaque mise en correspondance entre MV_i et MV_{i-1} :

- les correspondances entre segments sont visualisées de la manière suivante :
  - les segments des deux modèles sont représentés en projection perspective sur le plan image de la caméra de référence.
  - la figure de gauche montre les segments de MV_i et celle de droite, ceux de MV_{i-1}.
  - les segments avec correspondants sont visualisés en gras. Ceux de MV_i sont étiquetés par le numéro de leur correspondant dans MV_{i-1}. Ceux de MV_{i-1} sont étiquetés par leur numéro.

- les segments sans correspondant sont étiquetés par un "S".
- les erreurs sont entourées d'un cercle étiqueté par une lettre.

- le déplacement entre MV_i et MV_{i-1} est visualisé par l'union des segments de MV_{i-1} projetés sur le plan image de la caméra de référence et des segments de MV_i (en gras), également déplacés et projetés sur le plan image de la caméra de référence. En principe, un déplacement parfait devrait correspondre à une superposition parfaite des segments.

Les tableaux H.1, ...H.7 (annexe H) donnent les différentes estimations initiales du déplacement :

- colonne t_x : translation en x (en cm).
- colonne t_y : translation en y (en cm).
- colonne α : angle de rotation (en radians).
- colonne p : poids du déplacement ou, autrement dit, le nombre d'éléments du point d'accumulation utilisé pour calculer ce déplacement.

Le tableau 5.1 donne pour chaque correspondance d'un modèle MV_i avec un modèle MV_{i-1} :

- le nombre d'itérations de la procédure d'appariement (colonne I).
- les composantes x et y du déplacement initial (colonnes t_x^0 et t_y^0),
- les composantes x et y du déplacement final (colonnes t_x^F et t_y^F).
- le nombre de segments de MV_i (colonne N).
- le nombre de segments de MV_i pour lesquels aucun correspondant n'a été trouvé bien qu'il en existe un (colonne S).
- le nombre d'hypothèses de correspondance entre les segments de MV_i et ceux de MV_{i-1} (colonne H).
- le nombre d'hypothèses de correspondance erronées (colonne E).

D'après les résultats obtenus on constate que :

- le nombre de déplacements possibles est toujours très faible (colonne D du tableau 5.1).

i	D	I	t_x^0	t_y^0	t_x^F	t_y^F	N	S	H	E
1	13	1	-36	36	-30	37	97	11	53	4
2	10	2	4	19	4	4	71	17	35	3
3	8	4	-20	9	-17	17	65	11	25	1
4	9	1	-17	20	-14	20	59	15	26	2
5	10	1	54	10	59	6	55	11	28	8
6	4	1	36	4	36	7	47	11	23	3
7	6	4	-8	5	-5	11	40	11	15	5

Tableau 5.1. Résultats de mise en correspondance de MV_i avec MV_{i-1}

- le premier déplacement de la liste est souvent le meilleur (4 cas sur 7) et dans le pire des cas, quatre déplacements sont essayés avant de trouver le bon (colonne I du tableau 5.1).
- la proportion de mauvaises hypothèses de correspondance (somme des éléments de la colonne E divisée par la somme des éléments de la colonne H) est assez faible (13% environ).

En revanche :

- dans certains cas, la précision du déplacement final n'est pas suffisante. On peut constater une mauvaise estimation du déplacement dans les figures G.6, G.4 et G.2. Les segments de la table (en bas de la figure) sont mal surposés. Ce défaut s'explique en partie par une erreur systématique dans les vues stéréoscopiques distribuées par l'INRIA (due à des problèmes matériels sur le système d'aquisition) les segments verticaux sont toujours décalés par rapport au segments horizontaux (voir par exemple les segments des armoires figure B.1). Lors de l'exploitation des contraintes géométriques (surtout la correction de la coplanarité), ce décalage provoque des distorsions. Les positions relatives des segments ne sont pas conservées d'une vue à l'autre, ce qui empêche une bonne superposition.

  Pour confirmer ceci, nous avons effectué une mise en correspondance sur les modèles M_0 et M_1 non corrigés (à part l'orientation des segments). On constate que l'estimation du déplacement donne une meilleure superposition des segments (comparer la figure 5.4 avec la figure 5.3).

- la proportion de segments non appariés alors qu'ils devraient l'être (somme des éléments de la colonne S du tableau 5.1 divisée par la somme des éléments de la colonne N du même tableau) est importante. Elle représente 20% des segments.

Une proportion importante (88%) des erreurs de correspondance est due à des ambiguïtés mal résolues. Ces ambiguïtés apparaissent généralement dans les appariements entre des segments appartenant à un couple parallèle. Les erreurs suivantes peuvent alors se produire :

- plusieurs segments sont appariés à tort avec un même segment (erreurs a, h de la figure G.1, erreur d de la figure G.9, erreur o de la figure G.13).
- un segment est apparié avec un mauvais segment (erreurs i et j fig. G.1, erreurs m, n et e fig. G.3, erreurs e et f fig. G.5. ...).

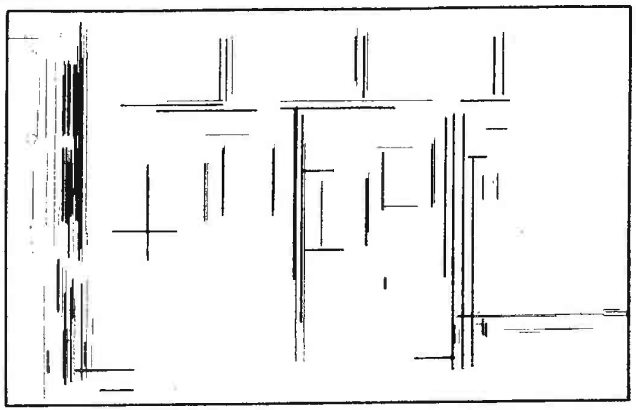


Figure 5.3. Position relative de MV_0 par rapport à MV_1 visualisée en projection horizontale (sans correction).

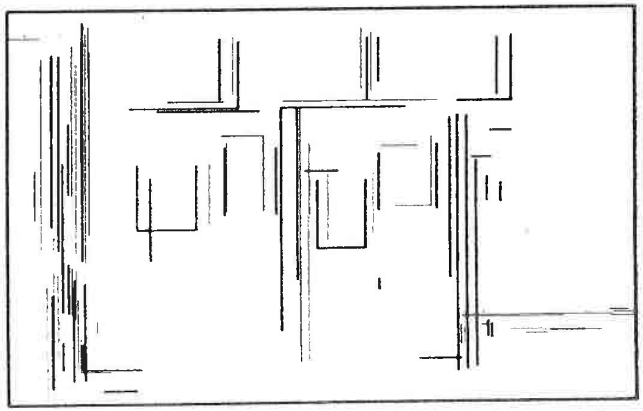


Figure 5.4. Position relative de MV_0 par rapport à MV_1 visualisée en projection horizontale (avec correction).

6
Le module de fusion

6.1 Fonction

Le module de fusion permet de construire un modèle M_3 en combinant les entités de deux modèles M_1 et M_2. Il utilise pour cela les résultats de la mise en correspondance des deux modèles, c'est-à-dire :

- l'ensemble $H(M_1, M_2)$ des hypothèses de correspondance entre les segments de M_1 et ceux de M_2. Chaque hypothèse est un couple de segments (s_1, s_2) où s_1 est un segment de M_1 et s_2 un segment de M_2.
- le déplacement $D(M_1, M_2)$ entre le repère de M_1 et celui de M_2.

Pour simplifier, le repère de M_3 est identique à celui de M_2. On peut donc considérer que les attributs géométriques des entités de M_2 sont déjà exprimés dans $R(M_3)$. Quant aux entités de M_1, il suffit de leurs appliquer le déplacement $D(M_1, M_2)$, pour connaître leurs attributs géométriques dans $R(M_3)$.

6.2 Etapes de la fusion de deux modèles

Les opérations intervenant successivement dans la fusion de deux modèles sont :

1. la suppression des appariements multiples. Cette opération ramène le problème initial à un problème plus simple. Elle transforme M_1, M_2 et $H(M_1, M_2)$ de telle sorte que chaque segment de M_1 ait au plus un correspondant dans M_2.
2. construction des segments.
3. construction des groupes directionnels.
4. construction des couples.
5. construction des groupes colinéaires et coplanaires.
6. correction des segments et calcul des attributs des motifs.

6.3 Suppression des appariement multiples

La suppression des appariements multiples consiste à remplacer les segments de M_1 qui correspondent à un même segment de M_2, par un segment unique. Cette substitution implique une mise à jour des motifs de M_1. La même opération est répétée pour M_2, puis la liste des hypothèses de correspondance est réactualisée en conséquence.

6.3.1 Transformation des modèles

Chaque modèle M (M_1 ou M_2) subit les opérations suivantes :

Transformation des segments

Chaque ensemble de segments $E(s) = \{s_1, \ldots, s_k\}$ correspondant à un même segment s est remplacé par un segment unique s_0.

Les attributs géométriques de ce nouveau segment sont définis de la manière suivante :

- soit C, le centre de gravité des milieux des segmentde $E(s)$:
- soit D, la droite passant par C et dirigée selon le vecteur $\uparrow(s_1)$ (identique à $\uparrow(s_2)$, $\ldots$, $\uparrow(s_k)$).

Les extrémités de s_0 sont les deux points p_1 et p_2 les plus éloignés entre eux, parmi les projections des extrémités des segments de $E(s)$ sur D (fig 6.1).

Mise à jour des couples

Chaque couple (s_1, s_2) de M est :

- supprimé, si $s_1 \in E(s)$ et $s_2 \in E_s$.
- remplacé par (s_0, s_2), si $s_1 \in E(s)$ et si s_2 n'appartient pas à $E(s)$.
- remplacé par (s_1, s_0) dans le cas inverse.
- inchangé sinon.

Mise à jour des groupes

Les éléments de chaque groupe g contenant des segments de $E(s)$ sont remplacés par $(S(g) - E(s)) \cup \{s_0\}$.

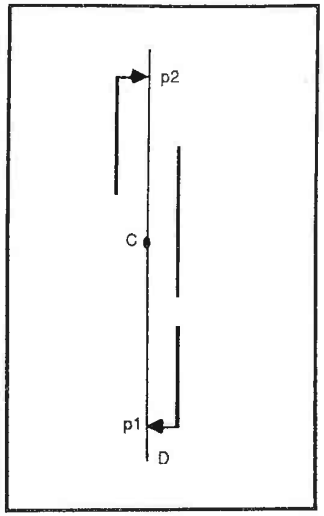


Figure 6.1. Fusion de segments appariés avec un même segment

6.3.2 Mise à jour des hypothèses

L'ensemble $H(M_1, M_2)$ des hypothèses de correspondance est remplacé par

$$H(M_1, M_2) = (H(M_1, M_2) - \{(s, s_1), \ldots, (s, s_k)\}) \cup \{(s, s_0)\}$$

si s est un segment de M_1 et par :

$$H(M_1, M_2) = (H(M_1, M_2) - \{(s_1, s), \ldots, (s_k, s)\}) \cup \{(s_0, s)\}$$

si s est un segment de M_2.

6.4 Construction des segments

L'ensemble des segments de M_3 est formé de l'union de deux ensembles de segments :

- S_1' : l'ensemble des segments de M_1 sans correspondants.
- S_2 : l'ensemble des segments de M_2.

Les attributs géométriques d'un segment s de M_3 sont redéfinis dans le repère de M_3 de la manière suivante :

- si s appartient à S'_1, ses attributs géométriques sont transformés par $D(M_1, M_2)$.
- si s appartient à S_2 :
  - s'il n'a pas de correspondant dans M_1, ses attributs sont inchangés.
  - sinon, soit :
    * s', son correspondant dans M_1 déplacé dans $R(M_3)$:
    * D, la droite passant par le centre C des milieux de s' et s et dirigée selon le vecteur $\uparrow(s)$.

    Les attributs de s se déduisent de ses nouvelles extrémités, qui sont les deux points p_1 et p_2 les plus éloignés parmi les extrémités de s et s' projetées sur D (fig. 6.2).

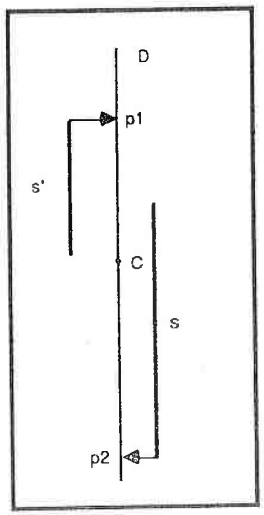


Figure 6.2. Mise à jour des coordonnées d'un segment.

6.5 Construction des groupes directionnels

Les groupes directionnels de M_3 sont les groupes directionnels V, H_1 et H_2 de M_2.

Les éléments de chacun de ces groupes sont redéfinis en fonction des éléments de leur correspondant dans M_1 :

- si l'angle de rotation est π ou 0, le correspondant de H_1 dans M_2 est H_1 dans M_1 et le correspondant de H_2 dans M_2 est H_2 dans M_1.
- dans le cas contraire (angle de $\pi/2$ ou $3\pi/2$), le correspondant de H_1 dans M_2 est H_2 dans M_1 et le correspondant de H_2 dans M_2 est H_1 dans M_1.
- Le correspondant de V dans M_2 est V dans M_1 indépendamment de la rotation.

A chaque groupe directionnel de M_2, on ajoute les segments de S'_1 appartenant à son correspondant dans M_1

6.6 Construction des couples

Les couples de M_3 sont obtenus par union des couples de M_2 (inchangés) et de certains couples de M_1.

Un couple de M_1 est ajouté à M_3, si un de ses éléments n'a pas de correspondant ou bien, si les deux ont un correspondant et qu'il n'existe pas de couple de M_2 du même type formé de ces deux correspondants. Dans chaque cas les éléments du couple possédant un correspondant sont remplacés par celui-ci.

6.7 Construction des groupes colinéaires et coplanaires

Les groupes colinéaires et coplanaires sont formés à partir des couples de M_3 selon la méthode présentée au chapitre 4.

6.8 Correction des segments et calcul des attributs des motifs

La méthode de correction des segments et de calcul des attributs des motifs est la même que celle du chapitre 4 à un détail près : la coplanarité n'est pas corrigée par projection perspective inverse, mais par projection orthogonale. Autrement dit chaque segment appartenant à un groupe coplanaire est projeté orthogonalement sur le plan défini par celui ci.

6.9 Résultats

6.9.1 Résultats obtenus dans la fusion de MV_1 et ME_0

6.9.2 Mise à jour des motifs

Dans l'annexe F, nous montrons les résultats de la mise à jour des motifs pour la fusion du modèle MV_1 avec le modèle ME_0 (où MV_0). Les motifs du modèle ME_1 résultant, sont visualisés en projection orthogonale. Les conventions adoptées pour la visualisation sont les mêmes que dans les figures présentant les résultats de la structuration de MV_0 et de MV_1 (annexe C et D).

6.9.3 Mise à jour des segments

Les figures 6.3, 6.4, 6.5 et 6.6 présentent différentes projections orthogonales de ME_1.

Les effets des erreurs de correspondance (fig. G.1) sont visibles dans les figures 6.4 et 6.5 :

- des segments peuvent disparaître. Lorsque deux segments sont appariés à tort avec un même segment, l'un des deux disparait lors de la fusion (erreur *c* fig. 6.4 et erreur *a* fig. 6.5 , provenant respectivement des erreurs de correspondance *a* et *h* fig. G.1).
- des segments parasites peuvent être ajoutés. Lorsqu'un segment n'est pas apparié et que son correspondant existe, il est ajouté au nouveau modèle alors qu'il ne devrait pas l'être. L'erreur *b* de la figure 6.4 est un exemple d'erreur de ce type. Elle provient de l'erreur *b* dans la mise en correspondance. Cette erreur se produit fréquemment.

6.9.4 Résultats obtenus dans la fusion de huit modèles

La figure 6.7 montre le modèle ME_7 obtenu après fusion des huit modèles $MV_0, \ldots, MV_7$. Cette figure peut être comparée avec la figure 6.8 qui montre la même pièce reproduite en simulation avec un angle de vue similaire.

Le tableau 6.1 donne les temps d'exécution (en secondes) des trois modules (écrits en Le lisp version 15.21 et non compilés) avec un SUN3/260 (4 Mips). Les temps sont de l'ordre de une à deux minutes pour chaque opération (structuration, mise en correspondance ou fusion). Chaque ligne i donne :

- le nombre de segments de MV_i (colonne ns_i).
- le nombre de segments de ME_{i-1} (colonne ns_e).
- les temps de calcul pour la structuration de la vue i (colonne t_s).

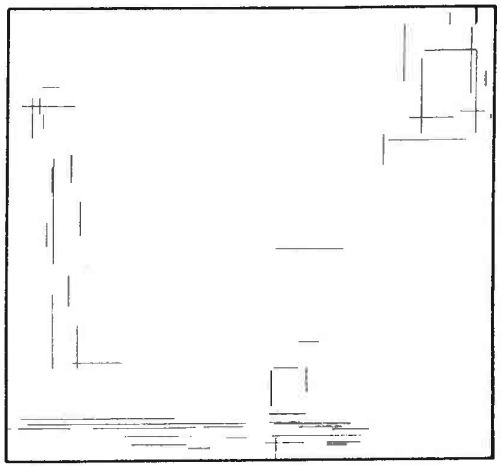


Figure 6.3. Projection verticale de ME_1.

- le temps de calcul de la mise en correspondance du modèle MV_i résultant avec le modèle ME_{i-1} (colonne t_m).
- le temps de calcul de l'estimation du déplacement dans cette mise en correspondance (colonne t_e).
- le temps de calcul de la fusion de MV_i avec ME_{i-1} (colonne t_f).
- le nombre d'itétations de la procédure d'appariement (colonne N).

On remarquera qu'une part importante du temps de mise en correspondance est occupé par l'estimation initiale du déplacement (50% du temps environ). Le temps total (incluant la structuration de chaque vue, la mise en correspondance et la fusion) est approximativement de 35 minutes. La structuration des vues correspond à 30% de ce temps, la mise en correspondance à 50% et la fusion 20%.

En général la première estimation du déplacement est la bonne, sauf pour les vues 4 et 6.

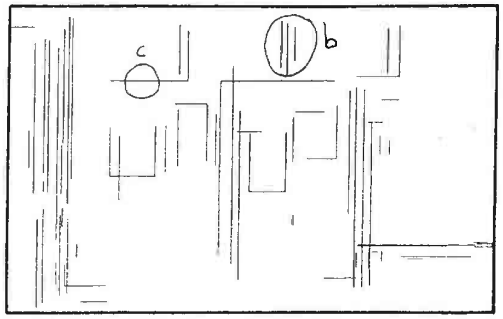


Figure 6.4. Projection horizontale de ME_1.

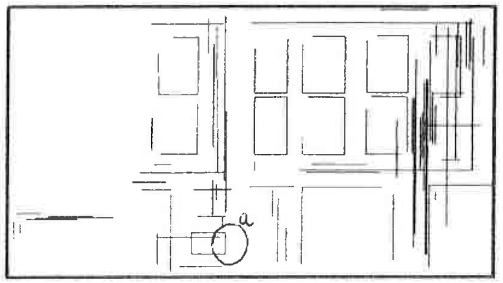


Figure 6.5. Projection horizontale de ME_1.

i	ns_i	ns_ϵ	$t_\circ$	t_m	t_c	t_f	N
0	106		145				
1	97	106	120	150	70	45	1
2	71	159	90	130	70	40	1
3	65	184	60	140	65	45	1
4	59	219	55	210	75	50	4
5	55	250	40	170	110	60	1
6	47	269	45	175	105	65	2
7	40	287	30	100	60	70	1

Tableau 6.1. Temps pour la fusion des huit vues

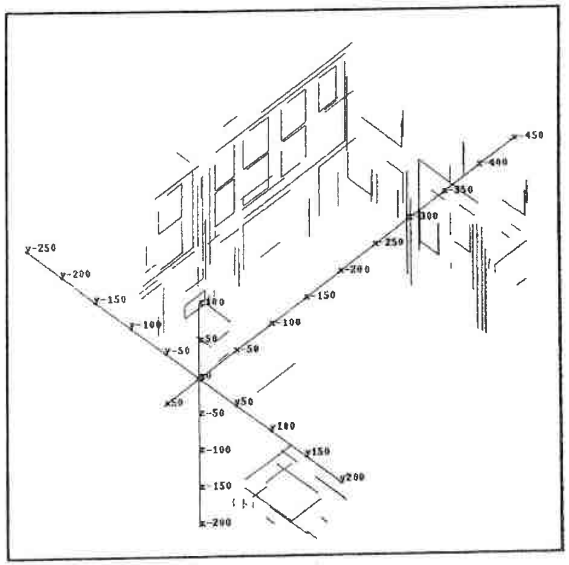


Figure 6.6. ME_1 vu de biais.

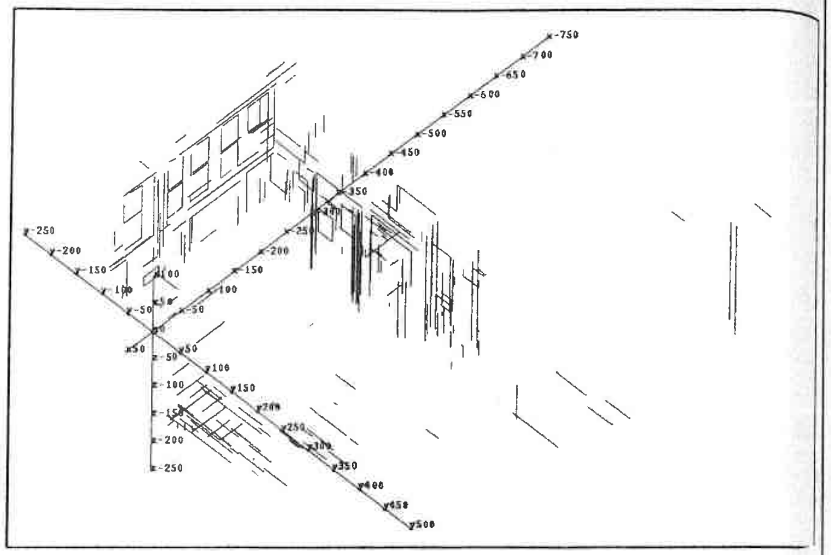


Figure 6.7. Modèle obtenu par fusion des huit vues

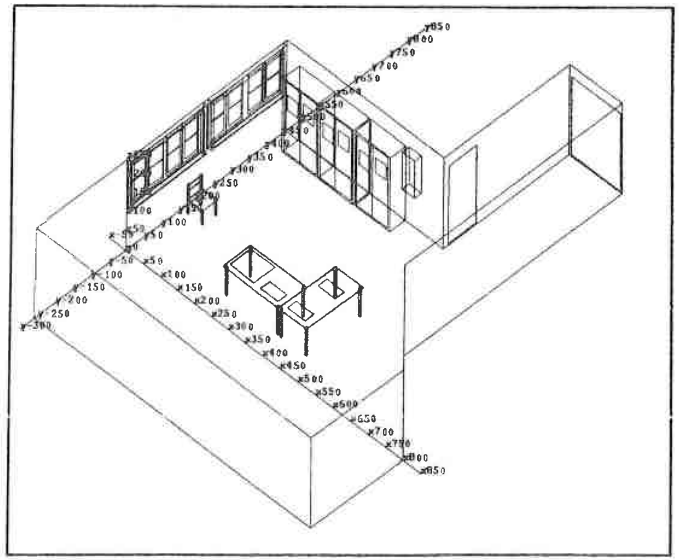


Figure 6.8. Modèle synthétique observé selon le même angle

7
Conclusion

7.1 Travaux connexes

7.1.1 Organisation perceptuelle

Dans notre approche, la représentation de l'environnement est fondée sur des relations entre segments et des groupes de segments définis à partir de ces relations.

A ce niveau, nous rejoignons les principes de l'*organisation perceptuelle*, étudiés en psychologie dans la "Gestalt Theory" [Wer 38] et en vision [Low 83] [Low 87] [Wit 83] [Zuc 78] [Zuc 83] [Tuc 87]. L'idée essentielle est le groupement d'indices visuels, selon certaines relations de proximité ou de similatité. Il est intéressant de remarquer que la vision humaine détecte rapidement les regroupements d'indices de ce type. Dans la figure 7.1, par exemple, on différencie immédiatement les groupes qui résultent des relations de connexité, de colinéarité et de parallélisme.

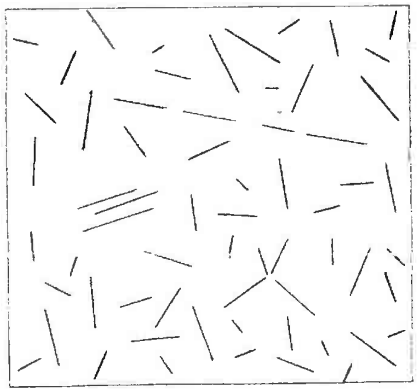


Figure 7.1. Illustration de la faculté de regroupement de la vision humaine

D. Lowe, un des premiers investigateurs de l'organisation perceptuelle en vision, a récemment montré l'intérêt des groupes perceptuels pour identifier des objets tridimensionnels dans une image [Low 87]. Son système rejoint le notre à plusieurs points de vue :

- les relations entre segments sont les mêmes que celles que nous avons utilisées : la colinéarité, la connexité et le parallélisme (fig. 7.2)
- ces relations sont utilisées pour définir des groupes qui sont ensuite utilisés dans la mise en correspondance entre les segments de l'image et ceux du modèle.

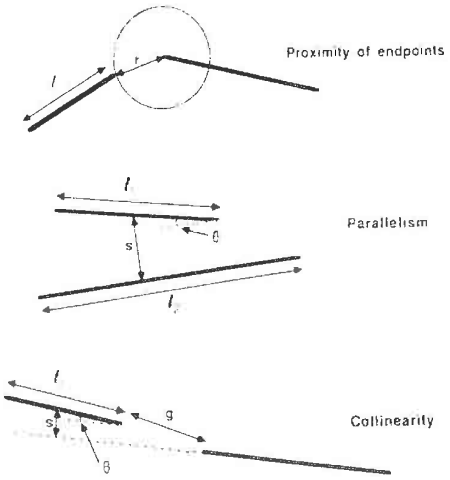


Figure 7.2. Relations utilisées par Lowe [Low 87]

7.1.2 Mise en correspondance

Rappelons brièvement la méthode que nous employons pour effectuer une mise en correspondance entre les segments de deux modèles. Nous commençons tout d'abord par construire une liste de déplacements possibles entre les repères respectifs de ces deux modèles. Ensuite, une procédure d'appariement et d'affinement du déplacement est appliquée à chaque déplacement de la liste jusqu'à réussir à apparier suffisamment de segments.

Différents auteurs ont utilisé une méthode semblable pour localiser des objets rigides [Sou 83] [Fau 83] [Aya 86]. Cette approche, permet une réduction considérable de l'espace de recherche. La transformation à estimer et à affiner n'est pas nécessairement un déplacement dans l'espace. Par exemple, dans le cas d'une mise en correspondance entre une image et un modèle 3D, il s'agit d'une projection perspective.

Notre méthode de mise en correspondance est partiellement inspirée de celle du système HYPER de N. Ayache. Ce système permet de localiser des objets plats dans une image (fig. 7.3) à niveaux de gris. Les segments de droites, obtenus par approximation polygonale des contours de l'image, sont appariés avec des segments 2D approximant les bords des objets.

Dans le cas bidimensionnel, il suffit d'une correspondance entre un segment de l'image et un segment du modèle pour avoir une estimation de la transformation (similitude). HYPER utilise donc les appariements entre les plus longs segments de l'image et les plus longs segments du modèle pour construire la liste des transformations possibles. Ensuite, le système essaie chacune de ces transformations en appariant les segments et en affinant la transformation à l'aide du filtre de Kalman [Jaz 70].

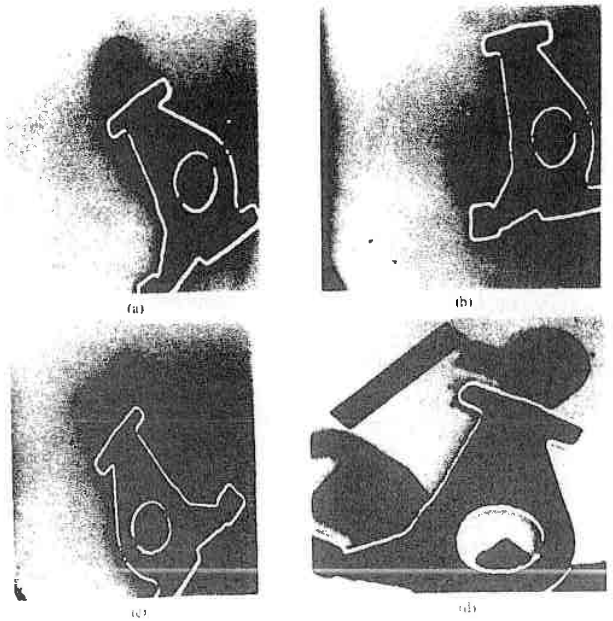


Figure 7.3. Objets localisés par HYPER. Facteur d'homothétie : 0.68 en (a)(b)(c) et 1.06 en (d)

N. Ayache a récemment utilisé la même méthode pour résoudre le problème d'appren-

tissage géométrique défini dans le cadre du projet ORASIS. Une comparaison avec notre approche s'impose donc ici.

La représentation est fondée sur trois primitives géométriques : le point, la droite et le plan. Une matrice de covariance attachée à chaque primitive du modèle, représente l'incertitude sur sa position. De même, une matrice de covariance est associée à chaque segment stéréoscopique d'une vue donnée. Le filtre de Kalman est utilisé à plusieurs niveaux :

- durant la mise en correspondance, il permet :
  - de connaître l'incertitude δD sur l'estimation courante du déplacement D.
  - de connaître l'incertitude sur un segment déplacé par D, en fonction de δD et de l'incertitude associée à ce segment. Ceci permet d'adapter la condition d'appariement entre deux segments en fonction de leurs incertitudes de position.
- lors de la fusion de deux vues, il permet de calculer l'incertitude sur la fusion de deux segments connaissant l'incertitude sur leur position et l'incertitude sur le déplacement.

La figure 7.4 montre une projection verticale du modèle obtenu par cette méthode sur les six vues correspondant aux positions p_0, p_3, p_6, p_9, p_{12}. La figure 7.5 visualise une projection verticale du modèle obtenu par notre méthode sur les vues correspondant aux positions $p_0, \ldots, p_7$.

On peut constater que les tables sont mieux reconstruites avec la méthode de N. Ayache. Nous pensons que cette différence provient de la prise en compte des incertitudes associées à chaque segment. Le fait de tenir compte des incertitudes permet d'une part, d'avoir une meilleure précision dans la mise à jour du déplacement et d'autre part, d'obtenir de meilleures hypothèses de correspondance car la condition d'appariement s'adapte localement aux incertitudes. De plus, la méthode de N. Ayache est moins sensible que la nôtre au décalage systématique entre les verticales et les horizontales car ces segments sont traités indépendamment.

Les différences essentielles entre l'approche de Ayache et la nôtre sont les suivantes :

1. Contrairement à N. Ayache, nous ne tenons pas compte des incertitudes associées aux segments et au déplacement.
2. Bien qu'il soit moins précis que celui de N. Ayache, notre modèle est plus régulier géométriquement :
   - les segments appartenant à un même groupe directionnel sont parfaitement parallèles.

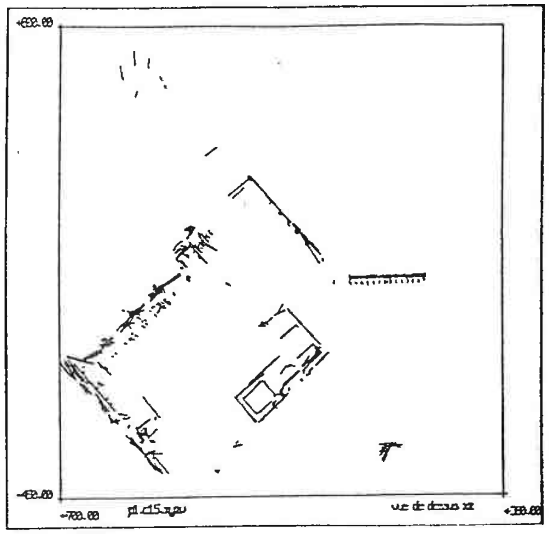


Figure 7.4. Résultats obtenus par N. Ayache [Aya 88, p286]

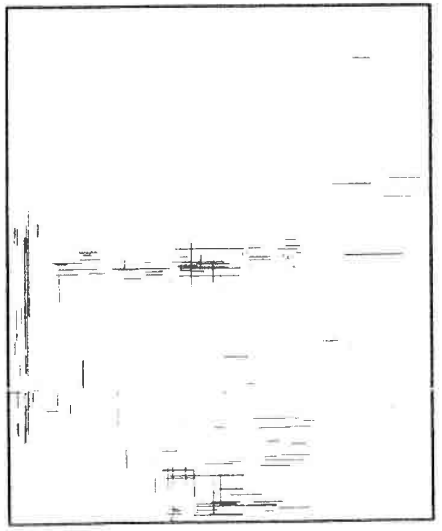


Figure 7.5. Résultats obtenus par notre méthode

- les segments appartenant à même groupe coplanaire sont parfaitement coplanaires.
- les segments appartenant à même groupe colinéaire sont parfaitement colinéaires.
- les segments connexes ont effectivement une extrémité commune.

En contre partie une erreur de structuration (classement d'un segment dans un mauvais groupe directionnel, coplanaires, colinéaires ...) provoque une erreur de position qui ne peut plus être corrigée par la suite.

3. Nous donnons une solution au problème de l'estimation initiale du déplacement :
   - grâce aux points de fuites, le repère d'un modèle est toujours aligné sur les trois directions principales du bâtiment. De ce fait, il n'existe toujours que quatre rotations possibles entre les repères respectifs de deux modèles. De plus, la valeur de ces quatre rotations est connue de manière exacte.
   - la rotation étant connue, les correspondances entre couples de segments de deux modèles permettent d'obtenir, par transformée de Hough, un ensemble de translations possibles fiables et peu nombreuses.
4. Nous proposons une méthode assez peu sensible au bruit pour retrouver des informations géométriques intéressantes comme le parallélisme, la coplanarité, la colinéarité et la connexité.

7.1.3 Exploitation de contraintes géométriques

Nous avons montré que les motifs représentent des contraintes géométriques sur les segments qu'ils contiennent. Ces contraintes peuvent être exploitées de manière intéressante pour corriger les coordonnées des segments. Cette idée se retrouve dans les travaux de N. Ayache [Aya 88] et M. Herman [Her 86].

N. Ayache donne un moyen de réduire l'incertitude attachée à des segments de droites par l'intermédiaire du filtre de Kalman, lorsque certaines relations géométriques entre ces segments, comme la colinéarité et la perpendicularité, sont connues.

Dans le système MOSAIC de M. Herman, présenté en introduction, les contraintes géométriques sont essentiellement des contraintes de coplanarité, de colinéarité et de connexité. Elles sont représentées par des liens entre les entités topologiques (coins, segments, faces) et des entités géométriques (points, droites et plans). Le calcul des attributs géométriques se fait par la méthode des moindres carrés et les contraintes sont propagées automatiquement à travers le modèle par l'intermédiaire d'une méthode inspirée de la théorie de Doyle [Doy 79].

7.2 Propositions pour améliorer le système

7.2.1 Les problèmes

Nous pensons que les résultats obtenus ne sont pas encore satisfaisants à plusieurs niveaux :

1. L'information contenue dans un modèle n'est pas suffisante :
   (a) il n'y a pas suffisamment de segments.
   (b) beaucoup de couples connexes ne sont pas retrouvés.
   (c) les groupes coplanaires sont trop incomplets.
   (d) pour des tâches de navigation, il manque une information de surface (existence d'une surface matérielle entre deux segments).
2. La qualité de l'information contenue dans un modèle n'est pas suffisante :
   (a) la position des segments n'est pas suffisamment précise.
   (b) les modèles construits par fusion contiennent trop de segments parasites.
   (c) les erreurs dans la formation des motifs, bien que relativement peu fréquentes, pourraient encore être réduites.

7.2.2 Solutions

Pour corriger ces différents défauts, nous proposons les solutions suivantes :

Utiliser les régions

L'utilisation des régions permettrait :

- D'avoir une information de surface.
- D'obtenir d'avantage de couples connexes en supposant que deux segments consécutifs (et non colinéaires) appartenant à un même contour de région, sont nécessairement connexes. Comparons, par exemple, les régions de la figure 7.6 (extraites par une méthode de division-fusion par B. Wrobel [Wro]) avec les couples connexes extraits de la même image (fig. 7.7). On constate que les couples connexes manquant, noté g, k, h, l, m, n et o, auraient pu être facilement obtenus à l'aide des régions.
- d'obtenir des groupes coplanaires plus complets en utilisant l'appartenance de deux segments à un même bord de région comme une indication de coplanarité.

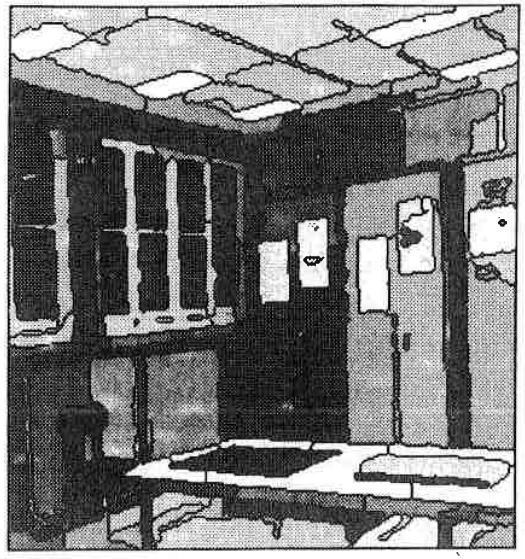


Figure 7.6. Régions associées à la caméra de référence dans la vue 0 [Wro]

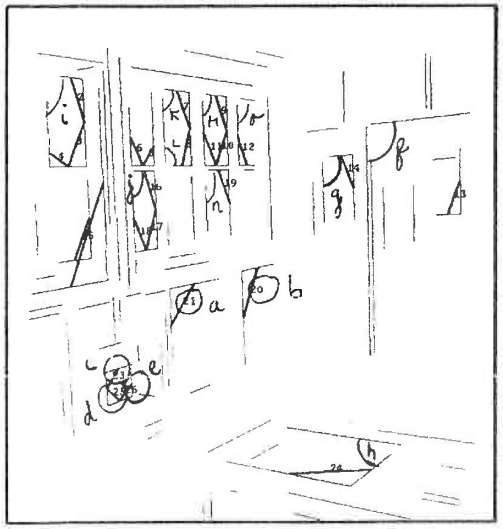


Figure 7.7. Couples connexes extraits de la vue 0.

Utiliser les segments image

L'utilisation conjuguée des segments stéréoscopiques et des segments image associés à la caméra de référence, permettrait de "combler les trous" dus aux segments non appariés durant la stéréovision. La figure 7.8 montre que cette perte de segments est importante. Elle présente les segments image associés à la caméra de référence dans la vue 0, et en superposition, les segments stéréoscopiques projetés sur le plan image de cette même caméra (en gras). Pour avoir plus de segments tridimensionnels, on pourrait déduire les coordonnées spatiales de certains segments image, sachant qu'ils sont colinéaires, connexes ou coplanaires à un segment stéréoscopique.

Figure 7.8. Segments image et segment stéréoscopique projetés, en superposition.

Tolérer des segments n'appartenant à aucun groupe directionnel

Les segments stéréoscopiques qui n'appartiennent à aucun groupe directionnel sont ignorés (cf. paragraphe 4.3). Avec cette restriction les objets non parallèles au murs peuvent être totalement ignorés. En fait ceci est surtout une simplification plutôt qu'une restriction fondamentale de la méthode. Pour accepter des segments non parallèles aux directions majoritaires il suffirait d'effectuer les modifications suivantes :

1. définir la direction d'un segment par :
   - la direction du groupe directionnel auquel il appartient, si ce dernier existe ;
   - sinon, par un vecteur unitaire dirigé parallèlement au segment et dans le sens du vecteur du repère le plus proche (angle minimal).
2. définir la direction des groupes colinéaires, des couples colinéaires et parallèles par une sorte de "moyenne" des directions des segments qu'ils contiennent.
3. définir la direction des couples connexes et des groupes coplanaires par la normale du plan approximant au mieux les segments qu'ils contiennent. Orienter la normale dans le sens du vecteur du repère le plus proche.
4. dans la formation des couples comparer les directions des segments au lieu de comparer leurs groupes directionnels (cette comparaison pourrait être effectuée aussi bien dans le repère stéréoscopique que dans le repère image, après projection).

Utiliser l'incertitude

Il serait intéressant d'associer une matrice de covariance aux centres des différentes entités d'un modèle afin de représenter leur incertitude de position. L'outil développé par N. Ayache dans sa thèse pourrait alors être utilisé pour gérer ces incertitudes à tous les niveaux :

- dans la formation des motifs, les conditions portant sur les segments stéréoscopiques pourraient être pondérées en fonction de leur incertitude. Cela permettrait d'éviter des erreurs dues à un seuil de tolérance trop important (causant par exemple la formation de couples connexes erronés) ou trop faible (causant la perte de certains motifs).
- pour calculer les centres des motifs en fonction des incertitudes associées à chacun de leur segment. Les centres obtenus de cette manière seraient certainement plus précis car ils seraient moins influencés par les segments bruités. Les effets de la correction seraient donc également améliorés.

- pour avoir une estimation de l'erreur sur un segment déplacé et tenir compte de cette erreur dans la condition d'appariement de deux segments. Une condition d'appariement dépendant des incertitudes améliorerait certainement les résultats de mise en correspondance :
  - il y aurait moins d'erreurs d'appariement sur les segments de faible incertitude
  - les segments non appariés à cause d'une erreur de position trop importante (20% des segments dans notre cas), seraient moins fréquents.
- pour obtenir un déplacement plus précis, car plus influencé par les segments de faible incertitude, que par les segments ayant une incertitude importante.
- pour fusionner des segments appariés en fonction des incertitudes sur le déplacement et des incertitudes associées aux segments. Cette méthode permet d'obtenir une meilleure précision sur la position des segments obtenus par fusion.

Effectuer une mise en correspondance par projection perspective

Etant donné que les segments stéréoscopiques sont moins bruités, lorsqu'ils sont projetés sur le plan image d'une des caméras, il serait intéressant d'effectuer une mise en correspondance, non pas sur les segments stéréoscopiques, mais sur leur projection. On pourrait alors utiliser un principe semblable au principe de formation des couples, dans la condition d'appariement de deux segments : utiliser une condition peu restrictive sur la position spatiale et beaucoup plus restrictive, sur la position des segments projetés.

Contrôler géométriquement la formation des groupes

Les groupes colinéaires et coplanaires sont construit par un mécanisme de déduction s'appuyant sur les couples connexes et colinéaires. Il serait souhaitable, pour éviter les erreurs, de contrôler géométriquement l'adjonction d'un segment à un groupe de ce type. Par exemple, pour un groupe colinéaire, il suffirait de vérifier que le segment est suffisamment proche de la droite définie par le groupe courant.

Contrôler géométriquement la correction

Les erreurs dans la formation des motifs provoquent des erreurs importantes lors de la correction des segments. De plus, ces erreurs ne peuvent plus être corrigées par la suite. Pour éviter cela, il serait judicieux de vérifier, avant de corriger un segment, que l'effet de la correction ne le déplace pas trop. Dans ce cas, on pourrait ne pas effectuer la correction si elle implique un déplacement trop important du segment.

7.3 Limitations du système

Les limitations du système décrit dans cette seconde partie sont :

1. les limitations sur le moyen de perception :

   (a) étant donné que tout le système est basé sur les segments de droite (aussi bien au niveau de la représentation qu'au niveau des traitements effectués), chaque vue doit être un ensemble de segments tridimensionnels.

   (b) ces segments tridimensionnels doivent être obtenus par stéréovision. En effet, la méthode de structuration suppose la connaissance de la matrice de projection d'une caméra c définie dans le repère associés aux segments 3D. De plus, elle suppose que l'erreur sur la position d'un segment 3D se produit essentiellement dans le plan passant par ce segment et le centre focal de c. Ceci est vrai pour la stéréovision, mais serait certainement faux pour un autre moyen de perception.

2. les limitations sur la position du repère stéréoscopique :

   (a) sa hauteur doit être approximativement constante. Cette restriction permet d'approximer le déplacement en z et de simplifier ainsi l'estimation de la translation (paragraphe 5.2.2, construction du tableau). Pour supprimer cette restriction, il suffirait d'estimer la translation dans un espace à trois dimensions au lieu de deux. Il ne s'agit donc pas d'une limitation fondamentale.

   (b) l'axe des Y doit être approximativement vertical et dirigé vers le bas. Cette hypothèse sert uniquement à distinguer le point de fuite associé à la vertical définir un repère dont l'axe des Z est dirigé vers le haut (paragraphe 4.1.2). En la supprimant, le repère d'un modèle aurait vingt-quatre positions possibles au lieu de quatre et il y aurait donc vingt-quatre rotations possibles entre deux modèles. Mais cela ne changerait rien au principe d'estimation du déplacement.

3. les limitations sur la manière d'observer l'environnement :

   (a) la partie de l'environnement observée à un moment donné doit toujours avoir un recouvrement minimal avec la partie de l'environnement déjà modélisé. Sans cela, le principe d'estimation de la translation, basé sur la correspondance d'un grand nombre de points, ne serait pas applicable. D'autre part, sans un recouvrement minimal, il ne serait pas possible de valider un déplacement par la proportion de segments appariés (seuil S_{ar}). Pour que cette condition soit réalisée il suffit d'observer l'environnement avec un déplacement suffisament faible entre chaque vue, ce qui n'est pas très gênant dans une phase d'apprentissage.

   (b) la partie de l'environnement observée doit toujours contenir un nombre suffisant de couples connexes et parallèles. Sans cela, le nombre d'estimations de translation (paragraphe 5.2.2) serait insuffisant pour déterminer la translation par transformée de Hough. Il est donc préférable d'ignorer les vues dans lesquelles il n'y a pas suffisament de couples connexes et parallèles (ceci peut par exemple se produire lorsqu'on observe un mur ou une surface uniforme de près).

4. les limitations sur l'environnement :

   (a) les objets doivent être polyédriques en majorité. Avec des objets non polyédriques, la représentation par segments de droite deviendrait instable. Par exemple, une sphère aurait une représentation variable selon la position d'observation (les segments stéréoscopiques approximant le bord de la sphère se déplaceraient sur sa surface). D'autre part, avec des objets courbes les relations de connexité, colinéarité et parallélisme qui sont fondamentales dans notre approche, seraient très rares.

   (b) l'environnement doit être statique. Cette hypothèse est fondamentale. Sans elle, le principe d'estimation initiale du déplacement n'est plus valable. Des points qui se correspondent ne sont plus nécessairement associés à une même translation (après rotation du repère) et l'estimation de la translation par transformée de Hough n'est donc plus valable. Cette hypothèse est également indispensable pour le bon fonctionnement de la procédure d'appariement et affinement du déplacement. Sans l'hypothèse de rigidité, une correspondance entre segments ne contraint pas nécessairement le déplacement et réciproquement.

   (c) l'environnement doit contenir trois directions majoritaires dont l'une verticale et les deux autres horizontales et perpendiculaires entre elles. Cette hypothèse est en effet indispensable pour le système actuel mais elle pourrait être un peu moins restrictive sans rien changer au principe. Son intérêt essentiel est de simplifier l'estimation du déplacement en réduisant l'ensemble des rotations possibles à un ensemble fini. Or pour estimer le déplacement entre le repère stéréoscopique d'une vue V et celui du modèle courant M_c , il suffit de deux hypothèses :

      - la vue doit contenir au moins deux points de fuite p_1 et p_2 correspondant à deux directions (non nécessairement orthogonales).
      - ces deux directions sont déjà représentées dans le modèle par deux groupes directionnels (non nécessairement parallèles aux axes du repère de M_c).

   La première condition permet de définir deux vecteurs $\vec{V_1}$ et $\vec{V_2}$. $\vec{V_1}$ est un des deux vecteurs unitaires correspondants à p_1 et $\vec{V_2}$, un des deux vecteurs

unitaires correspondants à p_2. D'après la deuxième hypothèse $\vec{V_1}$ et $\vec{V_2}$ correspondent chacun à un groupe directionnel de M_c. Or, si le modèle contient n groupes directionnels, il y a au pire $n(n-1)$ correspondances possibles entre le couple $(\vec{V_1}, \vec{V_2})$ et un couple de groupes directionnels de M_c (si les angles entre les directions des groupes sont inégaux il y en a moins). Et pour chaque correspondance, il y a au pire quatre rotations possibles (quatre si les deux directions correspondantes sont perpendiculaires et deux dans le cas général). Il y a donc au pire $4n(n-1)$ rotations possibles dans tous les cas.

7.4 Idées à retenir

Nous pensons que les principales idées à retenir de ce travail sont :

- la méthode de structuration d'une vue géométrique, qui permet d'extraire des contraintes géométriques (parallélisme, colinéarité, connexité, coplanarité) relativement fiables d'une vue stéréoscopique très bruitée.
- l'idée d'exploiter ces contraintes géométriques pour corriger les segments. Par la correction, les positions des segments deviennent dépendantes et leur position peuvent être améliorées.
- l'exploitation des hypothèses sur l'environnement pour estimer le déplacement :
  - l'existence de directions majoritaires simplifie le problème de l'estimation de la rotation. Elle permet de construire des repères "alignés sur l'environnement" entre lesquels il n'existe qu'un nombre fini de rotations possibles.
  - l'hypothèse de rigidité est exploitée avantageusement par la transformée de Hough pour estimer la translation associée à une rotation donnée.

PARTIE III

Perspectives

Dans cette thèse, nous avons décrit deux systèmes : le premier permet d'interpréter des vues tridimensionnelles synthétiques avec un modèle générique et le second construit automatiquement un modèle d'un environnement d'intérieur à partir de plusieurs vues stéréoscopiques. Pour l'instant, la connexion entre ces deux systèmes est encore loin de pouvoir être réalisée. D'un côté, le système d'apprentissage est capable de construire un modèle à partir de données réelles extrêmement bruitées, mais incapable d'en extraire des informations sémantiques (où sont les tables, les chaises, etc). D'un autre côté, le système d'interprétation a la possibilité de reconnaître des objets très variés en utilisant des connaissances sémantiques, mais il est incapable de gèrer les erreurs présentes dans les données réelles.

D'une manière générale, il reste des problèmes très difficiles à résoudre pour effectuer une reconnaissance d'objets génériques dans des images réelles avec un modèle construit à partir de données réelles :

- trouver une représentation suffisamment générale, générique, et contenant une information sémantique. Un modèle basé sur cette représentation devrait pouvoir être construit automatiquement à partir de données réelles. Par conséquent, la représentation devrait également tenir compte des erreurs.
- être capable de reconnaître les objets représentés dans ce modèle dans des données bruitées.

Une autre possibilité, envisageable à plus brève échéance, serait de réaliser un système interprétant des vues stéréoscopiques avec un modèle parfait construit par l'homme, représentant un univers d'intérieur. Pour être plus facilement interprétable, la vue stéréoscopique pourrait être structurée selon la méthode proposée dans la seconde partie, en intégrant en plus les régions. Quelques idées de TRIDENT pourraient être retenues pour aller dans ce sens :

- représenter les valeurs possibles des attributs de manière discrète, c'est à dire par des ensembles finis de symboles. Nous pensons que cette approche permet une certaine généricité et en même temps, une certaine tolérance au bruit. D'autre part elle permet une propagation rapide des informations durant l'interprétation.
- interpréter la vue par plusieurs analyses fonctionnant en parallèle. Comme nous l'avons vu, ce principe permet de réduire l'indéterminisme. De plus, pour interpréter des données bruitées, il est préférable que la description courante soit confimée par plusieurs interprétations locales indépendantes.

Annexe A
Modèle utilisé par TRIDENT

A.1 Caractéristiques globales du modèle

Le modèle présenté dans cet annexe représente un univers d'intérieur. Il comprend 113 règles décrivant soixante objets différents. Les objets constitués d'un bloc comportant deux faces verticales différentes (par exemple le coté d'un bureau, l'accoudoir ...) sont toujours définis par deux règles. Ceci permet d'éviter la distinction inutile entre la première et la seconde face verticale. Enfin, les relations utilisées sont définies par les contraintes suivantes :

same att (o_1, o_2) :

$$o_1.xmin = o_2.xmin$$

$$o_1.ymin = o_2.ymin$$

$$o_1.zmin = o_2.zmin$$

$$o_1.xmax = o_2.xmax$$

$$o_1.ymax = o_2.ymax$$

$$o_1.zmax = o_2.zmax$$

Dans le cas d'une règle de la forme $o_1 \rightarrow o_2$, cette relation sert à exprimer le fait que les coordonnées extrèmes de o_1 et de o_2 sont nécessairement égales.

same RF (o_1, o_2) :

$$o_1.l = o_2.l$$

$$o_1.L = o_2.L$$

$$o_1.phi = o_2.phi$$

$$o_1.alpha = o_2.alpha$$

Idem, mais dans le cas d'une règle $o_1 \rightarrow o_2$ avec $o_2 = RF$.

same-plane (o_1, o_2) :

$$o_1.phi = o_2.phi$$

vertical (o) :

$$o.phi = (90)$$

horizontal (o) :

$$o.phi = (0)$$

same-z (o_1, o_2) :

$$o_1.zmin = o_2.zmin$$

$$o_1.zmax = o_2.zmax$$

xy-inside (o_1, o_2) :

$$o_2.xmin \leq o_1.xmin$$

$$o_1.xmax \leq o_2.xmax$$

$$o_2.ymin \leq o_1.ymin$$

$$o_1.ymax \leq o_2.ymax$$

Cette relation exprime le fait que o_1 se trouve à "l'intérieur" de o_2 vu du dessus.

A.2 Les quatre scènes possibles

(MG scene MD (drawing-room))

(MG scene MD (kitchen))

(MG scene MD (office))

(MG scene MD (empty-room))

A.3 Cuisine

(MG kitchen MD (room lamp* picture* table* chair* shelves*))

(MG kitchen MD (room lamp* table* chair*))

(MG kitchen MD (room lamp* table*))

A.4 Pièce vide

(MG empty-room MD (room office-lamp*))

(MG empty-room MD (room lamp*))

A.5 Pièce de bureau

(MG office MD (room office-lamp* desk* writing-pad* chair* cupboard*))

(MG office MD (room office-lamp* desk* writing-pad* chair*))

(MG office MD (room office-lamp* desk* writing-pad* cupboard*))

(MG office MD (room office-lamp* desk* writing-pad*))

A.6 Salon

(MG drawing-room MD (room lamp* picture* sofa* armchair* shelves* low-table*))

(MG drawing-room MD (room lamp* sofa* low-table*))

(MG drawing-room MD (room lamp* sofa* shelves* low-table*))

(MG drawing-room MD (room lamp* armchair* low-table*))

(MG drawing-room MD (room lamp* armchair* shelves* low-table*))

(MG drawing-room MD (room lamp* sofa* picture* low-table*))

(MG drawing-room MD (room lamp* sofa* picture* shelves* low-table*))

(MG drawing-room MD (room lamp* armchair* picture* low-table*))

(MG drawing-room MD (room lamp* armchair* picture* shelves* low-table*))

(MG drawing-room MD (room lamp* armchair* sofa* low-table*))

(MG drawing-room MD (room lamp* armchair* sofa* shelves* low-table*))

A.7 Ensemble de tables basses

(MG low-table* MD (low-table))

A.8 Ensemble de dessous de main

(MG writing-pad* MD (writing-pad))

(MG writing-pad* MD (writing-pad writing-pad))

(MG writing-pad* MD (writing-pad writing-pad writing-pad))

A.9 Ensemble de divans

(MG sofa* MD (sofa))

A.10 Ensemble de tables

(MG table* MD (table))

A.11 Ensemble de chaises

(MG chair* MD (chair))

(MG chair* MD (chair chair))

A.12 Ensemble de fauteuils

(MG armchair* MD (armchair))

(MG armchair* MD (armchair armchair))

A.13 Ensemble de tableaux

(MG picture* MD (picture))

(MG picture* MD (picture picture))

A.14 Ensemble de lampes

(MG lamp* MD (lamp))

(MG lamp* MD (lamp lamp))

(MG lamp* MD (lamp lamp lamp))

A.15 Ensemble de lampes de bureau

(MG office-lamp* MD (office-lamp))

(MG office-lamp* MD (office-lamp office-lamp))

(MG office-lamp* MD (office-lamp office-lamp office-lamp))

A.16 Ensemble d'étagères

(MG shelves* MD (shelves)
ATT (shelves (l D L D h D))

(MG shelves* MD (shelves shelves))

(MG shelves* MD (shelves shelves shelves))

A.17 Ensemble d'armoires

(MG cupboard* MD (cupboard))

(MG cupboard* MD (cupboard cupboard))

(MG cupboard* MD (cupboard cupboard cupboard))

A.18 Ensemble de bureaux

(MG desk* MD (desk))

(MG desk* MD (desk desk))

(MG desk* MD (desk desk desk))

A.19 Première face verticale d'un bloc

(MG v-face1 MD (RF)
ATT (v-face1 (L D l D phi A alpha A))
REL (same att (v-face1 RF) same-plane (v-face1 RF)
same-RF (v-face1 RF) vertical (RF)))

A.20 Deuxième face verticale d'un bloc

(MG v-face2 MD (RF)
ATT (v-face2 (L D l D phi A alpha A))
REL (same-att (v-face2 RF) same-plane (v-face2 RF)
same-RF (v-face2 RF) vertical (RF)))

A.21 Face horizontale d'un bloc

(MG h-face MD (RF)
ATT (h-face (L D l D phi A alpha A))
REL (same-att (h-face RF) same-plane (h-face RF)
same-RF (h-face RF) horizontal (RF)))

A.22 Bloc

Les attributs d'un bloc sont :

- hfw : largeur de la face horizontale.
- hfl : longueur de la face horizontale.
- vfw1 : largeur de la face verticale 1.
- vfl1 : longueur de la face verticale 1.
- vfw2 : largeur de la face verticale 2.
- vfl2 : longueur de la face verticale 2.
- alpha1 : angle du plus grand cote de v-face1 avec la verticale.
- alpha2 : meme chose pour v-face2

(MG block MD (v-face1 v-face2 v-face1 v-face2 h-face h-face)
ATT (block (hfw D hfl D vfw1 D vfl1 D vfw2 D vfl2 D alpha1 A alpha2 A))
REL (same-z (v-face1 v-face2))
CTR ((block.hfw = h-face.l)
(block.hfl = h-face.L)
(block.vfw1 = v-face1.l)
(block.vfl1 = v-face1.L)
(block.vfw2 = v-face2.l)
(block.vfl2 = v-face2.L)

(block.alpha1 = v-face1.alpha)
(block.alpha2 = v-face2.alpha)
(block.zmin = v-face1.zmin)
(block.zmax = v-face1.zmax)
(block.zmin = v-face2.zmin)
(block.zmax = v-face2.zmax)
(block.zmin ≤ h-face.zmin)
(block.zmax ≥ h-face.zmin)
(block.xmin = h-face.xmin)
(block.ymin = h-face.ymin)
(block.xmax = h-face.xmax)
(block.ymax = h-face.ymax)
(block.xmin ≤ v-face1.xmin)
(block.xmin ≤ v-face2.xmin)
(block.ymin ≤ v-face1.ymin)
(block.ymin ≤ v-face2.ymin)
(block.xmax ≥ v-face1.xmax)
(block.xmax ≥ v-face2.xmax)
(block.ymax ≥ v-face1.ymax)
(block.ymax ≥ v-face2.ymax)))

A.23 Cylindre

(MG cylinder MD (horizontal-NRF horizontal-NRF RF RF RF RF RF RF)
ATT (cylinder (l D L D alpha A))
CTR ((cylinder.zmin = RF.zmin)
(cylinder.zmax = RF.zmax)
(cylinder.zmin ≤ horizontal-NRF.zmin)
(cylinder.zmax ≥ horizontal-NRF.zmax)
(cylinder.xmax ≥ RF.xmax)
(cylinder.ymax ≥ RF.ymax)
(cylinder.xmin ≤ RF.xmin)
(cylinder.ymin ≤ RF.ymin)
(cylinder.xmax = horizontal-NRF.xmax)
(cylinder.ymax = horizontal-NRF.ymax)
(cylinder.xmin = horizontal-NRF.xmin)
(cylinder.ymin = horizontal-NRF.ymin)
(cylinder.l = RF.l)

(cylinder.L = RF.L)
(cylinder.alpha = RF.alpha)
(RF.phi = 90))
REL (xy-inside (RF horizontal-NRF)))

A.24 Cône

(MG cone MD (bent-NRF bent-NRF bent-NRF bent-NRF
bent-NRF bent-NRF bent-NRF bent-NRF
bent-NRF bent-NRF bent-NRF bent-NRF horizontal-NRF horizontal-NRF)
CTR ((bent-NRF.zmin = cone.zmin)
(bent-NRF.zmax = cone.zmax)
(cone.zmin ≤ horizontal-NRF.zmin)
(cone.zmax ≤ horizontal-NRF.zmax)))

A.25 Face non rectangulaire inclinée

(MG bent-NRF MD (NRF)
REL (same-att (NRF bent-NRF))
CTR ((NRF.phi = 45)))

A.26 Face non rectangulaire horizontale

(MG horizontal-NRF MD (NRF)
REL (same-att (NRF horizontal-NRF))
CTR ((NRF.phi = 0)))

A.27 Pied de table

(MG table-leg MD (block)
REL (same-att (table-leg block))
CTR ((block.zmax = 80)
(block.zmin = 0)
(block.vfl1 = 80)
(block.vfl2 = 80)
(block.vfw1 ≤ 10)
(block.vfw2 ≤ 10)
(block.hfl = 0)

(block.hfw = 0)
(block.alpha1 = 0)
(block.alpha2 = 0)))

(MG table-leg MD (cylinder)
REL (same-att (table-leg cylinder))
CTR ((cylinder.zmin = 0)
(cylinder.zmax = 80)
(cylinder.l = 0)
(cylinder.L = 80)
(cylinder.alpha = 0)))

A.28 Plateau de table

(MG table-top MD (block)
REL (same-att (table-top block))
CTR ((table-top.zmax = 80)
(table-top.zmin = 80)
(block.vfw1 = 0)
(block.vfw2 = 0)
(block.vfl1 = 80 160)
(block.vfl2 = 80 160)
(block.hfw = 80 160)
(block.hfl = 80 160)
(block.alpha1 = 90)
(block.alpha2 = 90)))

A.29 Plateau de table hexagonale

(MG hexa-table-top MD (cylinder)
REL (same-att (hexa-table-top cylinder))
CTR ((cylinder.zmax = 80)
(cylinder.zmin = 80)
(cylinder.l = 0)
(cylinder.L = 40 80)
(cylinder.alpha = 90)))

A.30 Table

(MG table MD (rec-table))

(MG table MD (hexa-table))

A.31 Table rectangulaire

(MG rec-table MD (table-top table-leg table-leg table-leg table-leg)
REL (xy-inside (table-leg table-top)))

A.32 Table hexagonale

(MG hexa-table
MD (hexa-table-top table-leg table-leg table-leg table-leg table-leg
table-leg)
REL (xy-inside (table-leg hexa-table-top)))

A.33 Table basse

(MG low-table
MD (low-table-top
low-table-leg low-table-leg low-table-leg low-table-leg)
REL (xy-inside (low-table-leg low-table-top)))

A.34 Pied de table basse

(MG low-table-leg MD (block)
REL (same-att (low-table-leg block))
CTR ((block.zmin = 0)
(block.zmax = 40)
(block.hfw = 0)
(block.hfl = 0)
(block.vfw1 = 0)
(block.vfl1 = 40)
(block.vfw2 = 0)
(block.vfl2 = 40)
(block.alpha1 = 0)
(block.alpha2 = 0)))

A.35 Plateau de table basse

(MG low-table-top MD (block)
REL (same-att (low-table-top block))
CTR ((block.zmin = 40)
(block.zmax = 40)
(block.hfw = 40 80)
(block.hfl = 80 160)
(block.vfw1 = 0)
(block.vfw2 = 0)
(block.alpha1 = 90)
(block.alpha2 = 90)
(block.vfl1 = 40 80)
(block.vfl2 = 80 160)))

(MG low-table-top MD (block)
REL (same-att (low-table-top block))
CTR ((block.zmin = 40)
(block.zmax = 40)
(block.hfw = 40 80)
(block.hfl = 80 160)
(block.vfw1 = 0)
(block.vfw2 = 0)
(block.alpha1 = 90)
(block.alpha2 = 90)
(block.vfl2 = 40 80)
(block.vfl1 = 80 160)))

A.36 Dessous de main

(MG writing-pad MD (block)
REL (same-att (writing-pad block))
CTR ((block.zmax = 80)
(block.zmin = 80)
(block.hfw = 40)
(block.hfl = 80)
(block.vfl1 = 40)
(block.vfw1 = 0)
(block.vfl2 = 80)

(block.vfw2 = 0)
(block.alpha1 = 90)
(block.alpha2 = 90)))

(MG writing-pad MD (block)
REL (same-att (writing-pad block))
CTR ((block.zmax = 80)
(block.zmin = 80)
(block.hfw = 40)
(block.hfl = 80)
(block.vfl2 = 40)
(block.vfw2 = 0)
(block.vfl1 = 80)
(block.vfw1 = 0)
(block.alpha1 = 90)
(block.alpha2 = 90)))

A.37 Plateau de bureau

(MG desk-top MD (block)
REL (same-att (desk-top block))
CTR((desk-top.zmax = 80)
(desk-top.zmin = 80)
(block.vfw1 = 0)
(block.vfw2 = 0)
(block.vfl1 = 80 160)
(block.vfl2 = 80 160)
(block.hfw = 80 160)
(block.hfl = 80 160)
(block.alpha1 = 90)
(block.alpha2 = 90)))

A.38 Coté de bureau

(MG desk-side MD (block)
REL (same-att (desk-side block))
CTR ((block.zmin = 0)
(block.zmax = 80)
(block.vfl1 = 80)

(block.vfw1 = 40)
(block.vfw2 = 80)
(block.vfl2 = 80)
(block.hfw = 40 80)
(block.hfl = 80)))

(MG desk-side MD (block)
REL (same-att (desk-side block))
CTR ((block.zmin = 0)
(block.zmax = 80)
(block.vfl2 = 80)
(block.vfw2 = 40)
(block.vfw1 = 80)
(block.vfl1 = 80)
(block.hfw = 40 80)
(block.hfl = 80)))

A.39 Bureau

(MG desk MD (desk-top desk-side desk-side)
REL (xy-inside (desk-side desk-top)
xy-inside (desk-side desk-top)))

A.40 Plateau de chaise

(MG chair-top MD (block)
REL (same-att (block chair-top))
CTR ((block.zmin = 40)
(block.zmax = 40)
(block.vfw1 = 0)
(block.vfw2 = 0)
(block.vfl1 = 40)
(block.vfl2 = 40)
(block.hfw = 40)
(block.hfl = 40)
(block.alpha1 = 90)
(block.alpha2 = 90)))

A.41 Barre verticale de dossier de chaise

(MG chair-back-vp MD (block)
REL (same-att (chair-back-vp block))
CTR ((block.zmin = 40)
(block.zmax = 80)
(block.vfw1 = 0)
(block.vfw2 = 0)
(block.vfl1 = 40)
(block.vfl2 = 40)
(block.hfw = 0)
(block.hfl = 0)
(block.alpha1 = 0)
(block.alpha2 = 0)))

A.42 Pied de chaise

(MG chair-leg MD (block)
REL (same-att (chair-leg block))
CTR ((block.zmin = 0)
(block.zmax = 40)
(block.vfw1 = 0)
(block.vfw2 = 0)
(block.vfl1 = 40)
(block.vfl2 = 40)
(block.hfw = 0)
(block.hfl = 0)
(block.alpha1 = 0)
(block.alpha2 = 0)))

A.43 Dossier de chaise

Dossier de chaise composé d'une seule partie

(MG chair-back MD (block)
REL (same-att (chair-back block))
CTR ((block.zmin = 40 80)
(block.zmax = 40 80)
(block.vfl1 = 40)

(block.vfw1 = 20 40)
(block.vfl2 = 20)
(block.vfw2 = 0)
(block.hfw = 0)
(block.hfl = 40)
(block.alpha1 = 90)
(block.alpha2 = 0)))

(MG chair-back MD (block)
REL (same-att (chair-back block))
CTR ((block.zmin = 40 80)
(block.zmax = 40 80)
(block.vfl2 = 40)
(block.vfw2 = 20 40)
(block.vfl1 = 20)
(block.vfw1 = 0)
(block.hfw = 0)
(block.hfl = 40)
(block.alpha1 = 0)
(block.alpha2 = 90)))

Dossier de chaise composé de deux parties

(MG chair-back MD (chair-back-hp chair-back-hp))

A.44 Partie horizontale d'un dossier de chaise

(MG chair-back-hp MD (block)
REL (same-att (chair-back-hp block))
CTR ((block.zmin = 40 80)
(block.zmax = 40 80)
(block.vfw1 $\leq$ 10)
(block.vfl1 = 40)
(block.vfw2 $\leq$ 10)
(block.vfl2 = 0)
(block.hfw = 0)
(block.hfl = 40)
(block.alpha1 = 90)
(block.alpha2 = 0)))

(MG chair-back-hp MD (block)
REL (same-att (chair-back-hp block))
CTR ((block.zmin = 40 80)
(block.zmax = 40 80)
(block.vfw2 ≤ 10)
(block.vfl2 = 40)
(block.vfw1 ≤ 10)
(block.vfl1 = 0)
(block.hfw = 0)
(block.hfl = 40)
(block.alpha1 = 0)
(block.alpha2 = 90)))

Partie horizontale d'un pied de chaise

(MG h-chair-leg MD (block)
REL (same-att (h-chair-leg block))
CTR ((block.zmin = 0)
(block.zmax = 0)
(block.hfw = 0)
(block.hfl = 20)
(block.vfw1 = 0)
(block.vfl1 = 0)
(block.vfw2 = 0)
(block.vfl2 = 20)
(block.alpha2 = 90)))

(MG h-chair-leg MD (block)
REL (same-att (h-chair-leg block))
CTR ((block.zmin = 0)
(block.zmax = 0)
(block.hfw = 0)
(block.hfl = 20)
(block.vfw2 = 0)
(block.vfl2 = 0)
(block.vfw1 = 0)
(block.vfl1 = 20)
(block.alpha1 = 90)))

A.45 Chaise

(MG chair
MD (chair-top chair-leg chair-leg chair-leg chair-leg
chair-back-vp chair-back-vp chair-back)
REL (xy-inside (chair-leg chair-top)
xy-inside (chair-back-vp chair-top)
xy-inside (chair-back chair-top)))

(MG chair
MD (chair-top chair-leg chair-back-vp chair-back-vp chair-back
h-chair-leg h-chair-leg h-chair-leg h-chair-leg)
REL (xy-inside (chair-leg chair-top)
xy-inside (chair-leg chair-top)
xy-inside (chair-back-vp chair-top)
xy-inside (chair-back chair-top)
xy-inside (h-chair-leg chair-top)))

A.46 Fauteuil

(MG armchair MD (armchair-bottom elbow elbow armchair-back)
REL (xy-inside (armchair-back armchair-bottom)))

A.47 Base d'un fauteuil

(MG armchair-bottom MD (block)
REL (same-att (armchair-bottom block))
CTR ((block.zmin = 0)
(block.zmax = 40)
(block.vfl1 = 80)
(block.vfw1 = 40)
(block.vfl2 = 80)
(block.vfw2 = 40)
(block.hfw = 80)
(block.hfl = 80)
(block.alpha1 = 90)
(block.alpha2 = 90)))

A.48 Accoudoir

```
(MG elbow MD (block)
REL (same-att (elbow block))
CTR ((block.zmin = 40)
     (block.zmax = 40)
     (block.hfw = 10 20)
     (block.hfl = 40 80)
     (block.vfw1 = 10 20)
     (block.vfl1 = 10 20)
     (block.vfw2 = 10 20)
     (block.vfl2 = 40 80)
     (block.alpha2 = 90)))

(MG elbow MD (block)
REL (same-att (elbow block))
CTR ((block.zmin = 40)
     (block.zmax = 40)
     (block.hfw = 10 20)
     (block.hfl = 40 80)
     (block.vfw2 = 10 20)
     (block.vfl2 = 10 20)
     (block.vfw1 = 10 20)
     (block.vfl1 = 40 80)
     (block.alpha1 = 90)))
```

A.49 Dossier de fauteuil

```
(MG armchair-back MD (block)
REL (same-att (armchair-back block))
CTR ((block.zmin = 40)
     (block.zmax = 80)
     (block.vfw1 = 40)
     (block.vfl1 = 80)
     (block.vfw2 = 10 20)
     (block.vfl2 = 40)
     (block.hfw = 10 20)
     (block.hfl = 80)
     (block.alpha1 = 90)
```



```
     (block.alpha2 = 0)))

(MG armchair-back MD (block)
REL (same-att (armchair-back block))
CTR ((block.zmin = 40)
     (block.zmax = 80)
     (block.vfw2 = 40)
     (block.vfl2 = 80)
     (block.vfw1 = 10 20)
     (block.vfl1 = 40)
     (block.hfw = 10 20)
     (block.hfl = 80)
     (block.alpha1 = 0)
     (block.alpha2 = 90)))
```

A.50 Divan

```
(MG sofa MD (sofa bottom sofa-back elbow elbow)
REL (xy inside (sofa-back sofa-bottom)))
```

A.51 Base d'un divan

```
(MG sofa-bottom MD (block)
REL (same-att (sofa-bottom block))
CTR ((block.zmin = 0)
     (block.zmax = 40)
     (block.vfl1 = 80)
     (block.vfw1 = 40)
     (block.vfl2 = 160)
     (block.vfw2 = 40)
     (block.hfw = 80)
     (block.hfl = 160)
     (block.alpha1 = 90)
     (block.alpha2 = 90)))

(MG sofa bottom MD (block)
REL (same att (sofa bottom block))
CTR ((block.zmin = 0)
     (block.zmax = 40)
```

(block.vfl2 = 80)
(block.vfw2 = 40)
(block.vfl1 = 160)
(block.vfw1 = 40)
(block.hfw = 80)
(block.hfl = 160)
(block.alpha1 = 90)
(block.alpha2 = 90)))

A.52 Dossier de divan

(MG sofa-back MD (block)
REL (same-att (sofa-back block))
CTR ((block.zmin = 40)
(block.zmax = 80)
(block.vfl2 = 40)
(block.vfw2 = 10 20)
(block.vfl1 = 160)
(block.vfw1 = 40)
(block.hfw = 10 20)
(block.hfl = 160)
(block.alpha1 = 90)
(block.alpha2 = 0)))

(MG sofa-back MD (block)
REL (same-att (sofa-back block))
CTR ((block.zmin = 40)
(block.zmax = 80)
(block.vfl1 = 40)
(block.vfw1 = 10 20)
(block.vfl2 = 160)
(block.vfw2 = 40)
(block.hfw = 10 20)
(block.hfl = 160)
(block.alpha1 = 0)
(block.alpha2 = 90)))

A.53 Pièce

(MG room MD (block)
REL (same-att (room block))
CTR ((block.zmin = 0)
(block.zmax = 320)
(block.hfw = 320 640)
(block.hfl = 320 640)
(block.vfw1 = 320)
(block.vfl1 = 320 640)
(block.vfw2 = 320)
(block.vfl2 = 320 640)))

A.54 Lampe de bureau

(MG office-lamp MD (block)
REL (same-att (office-lamp block))
CTR ((block.zmin = 320)
(block.zmax = 320)
(block.vfw1 = 0)
(block.vfl1 = 20 40 80)
(block.vfw2 = 0)
(block.vfl2 = 20 40 80)
(block.hfw = 20 40 80)
(block.hfl = 20 40 80)
(block.alpha1 = 90)
(block.alpha2 = 90)))

A.55 Lampe conique

(MG lamp MD (cone)
REL (same-att (lamp cone))
CTR ((cone.zmin $\geq$ 160)
(cone.zmax $\geq$ 160)))

A.56 Tableau

(MG picture MD (block)
REL (same-att (picture block))

CTR ((block.zmin = 80 160)
(block.zmax ≥ 80)
(block.vfw1 = 40 80)
(block.vfl1 = 40 80)
(block.vfl2 = 40 80)
(block.vfw2 = 0)
(block.hfw = 0)
(block.hfl = 40 80)
(block.alpha2 = 0)))

(MG picture MD (block)
REL (same-att (picture block))
CTR ((block.zmin = 80 160)
(block.zmax ≥ 80)
(block.vfw2 = 40 80)
(block.vfl2 = 40 80)
(block.vfl1 = 40 80)
(block.vfw1 = 0)
(block.hfw = 0)
(block.hfl = 40 80)
(block.alpha1 = 0)))

A.57 Armoire

(MG cupboard MD (block)
REL (same-att (cupboard block))
CTR ((block.zmin = 0)
(block.zmax ≥ 160)
(block.vfl1 ≥ 160)
(block.vfl2 ≥ 160)
(block.vfw1 = 40)
(block.vfw2 = 80 160)
(block.hfw = 40)
(block.hfl = 80 160)
(block.alpha1 = 0)
(block.alpha2 = 0)))

(MG cupboard MD (block)
REL (same-att (cupboard block))

CTR ((block.zmin = 0)
(block.zmax ≥ 160)
(block.vfl2 ≥ 160)
(block.vfl1 ≥ 160)
(block.vfw2 = 40)
(block.vfw1 = 80 160)
(block.hfw = 40)
(block.hfl = 80 160)
(block.alpha1 = 0)
(block.alpha2 = 0)))

A.58 Coté d'étagère

(MG shelves-side MD (block)
ATT(shelves-side (l D L D h D))
REL (same-att (shelves-side block))
CTR ((block.zmin ≥ 80)
(block.zmax ≥ 160)
(block.hfw = 0)
(block.hfl = 20 40)
(block.vfw1 = 0)
(block.vfl1 = 80 160)
(block.vfw2 = 20 40)
(block.vfl2 = 80 160)
(block.alpha1 = 0)
(block.alpha2 = 0)
(shelves-side.l = block.hfw)
(shelves-side.L = block.hfl)
(shelves-side.h = block.vfl1)
(shelves-side.h = block.vfl2)))

(MG shelves-side MD (block)
REL (same-att (shelves-side block))
CTR ((block.zmin ≥ 80)
(block.zmax ≥ 160)
(block.hfw = 0)
(block.hfl = 20 40)
(block.vfw2 = 0)
(block.vfl2 = 80 160)

(block.vfw1 = 20 40)
(block.vfl1 = 80 160)
(block.alpha1 = 0)
(block.alpha2 = 0)
(shelves-side.l = block.hfw)
(shelves-side.L = block.hfl)
(shelves-side.h = block.vfl1)
(shelves-side.h = block.vfl2)))

A.59 Planche d'étagère

(MG shelves-board MD (block)
REL (same-att (shelves board block))
CTR ((block.zmin ≥ 80)
(block.zmax ≥ 80)
(block.hfw = 20 40)
(block.hfl = 80 160)
(block.vfw1 = 0)
(block.vfl1 = 80 160)
(block.vfw2 = 0)
(block.vfl2 = 20 40)
(block.alpha1 = 90)
(block.alpha2 = 90)))

(MG shelves-board MD (block)
REL (same-att (shelves board block))
CTR ((block.zmin ≥ 80)
(block.zmax ≥ 80)
(block.hfw = 20 40)
(block.hfl = 80 160)
(block.vfw2 = 0)
(block.vfl2 = 80 160)
(block.vfw1 = 0)
(block.vfl1 = 20 40)
(block.alpha1 = 90)
(block.alpha2 = 90)))

A.60 Étagère

(MG shelves
MD (shelves-side shelves-side shelves-board shelves-board shelves-board)
CTR ((shelves.l = shelves-side.l)
(shelves.L = shelves-side.L)
(shelves.h = shelves-side.h)))

(MG shelves
MD (shelves-side shelves-side shelves-board shelves-board shelves-board shelves-board)
CTR ((shelves.l = shelves-side.l)
(shelves.L = shelves-side.L)
(shelves.h = shelves-side.h)))

Annexe B
Les huit vues et les modèles correspondants

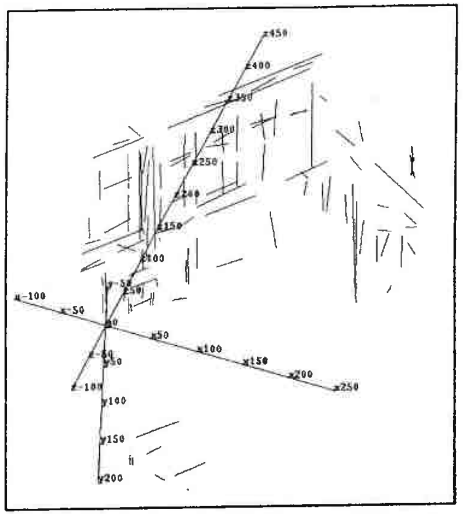


Figure B.1. Vue 0

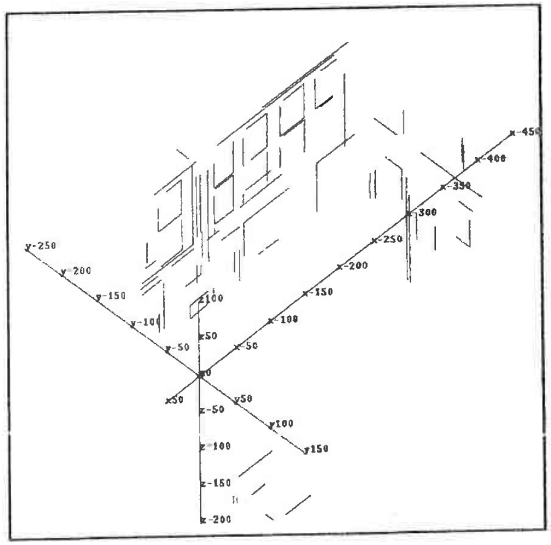


Figure B.2. MV_0

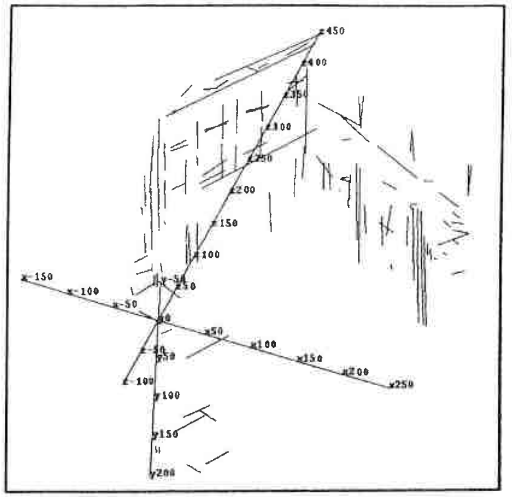


Figure B.3. Vue 1

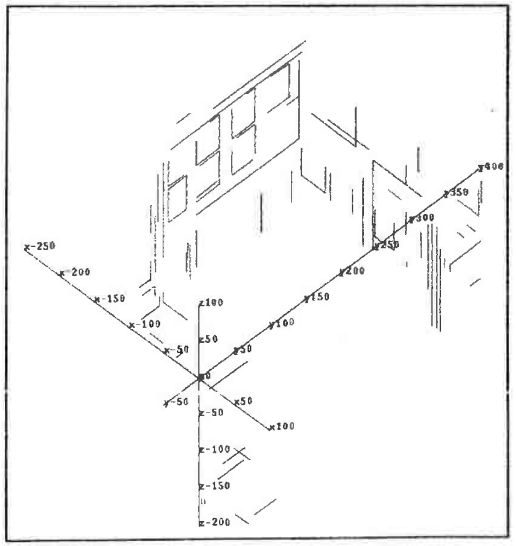


Figure B.4. MV_1

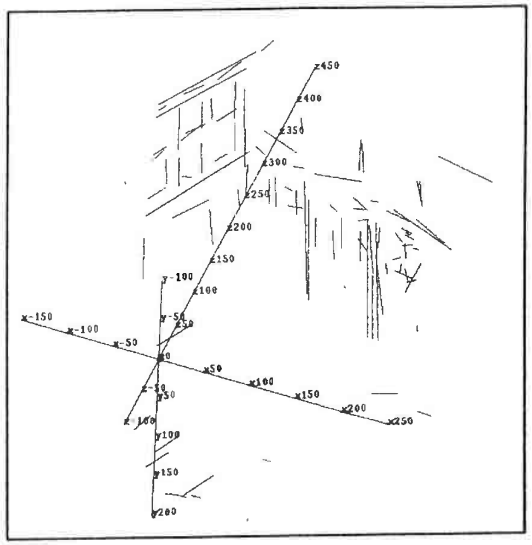


Figure B.5. Vue 2

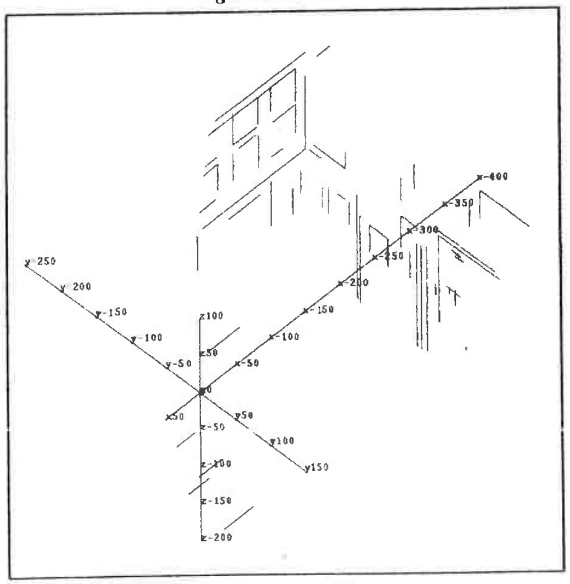


Figure B.6. MV_2

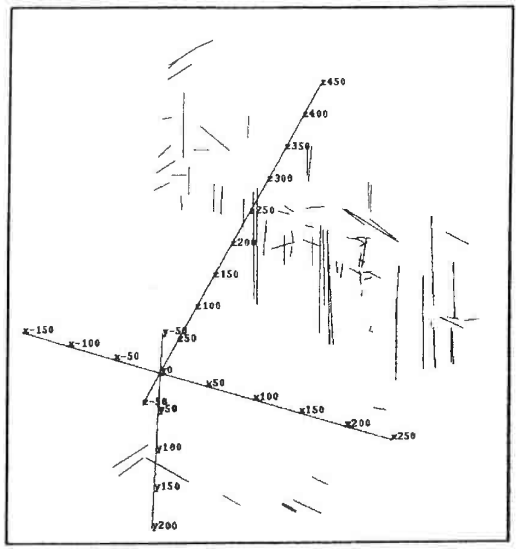


Figure B.7. Vue 3

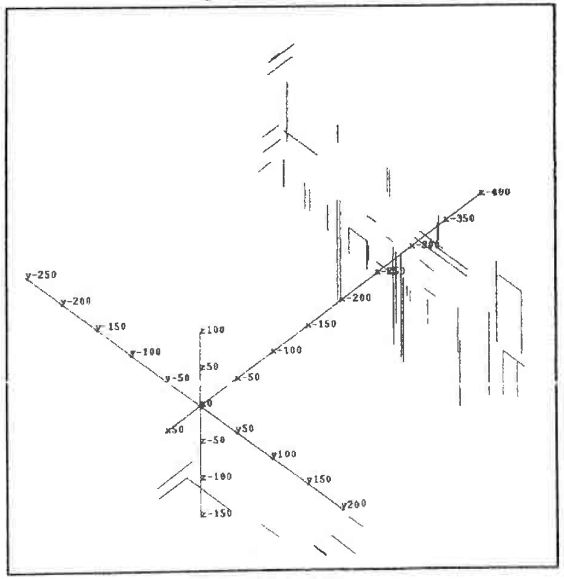


Figure B.8. MV_3

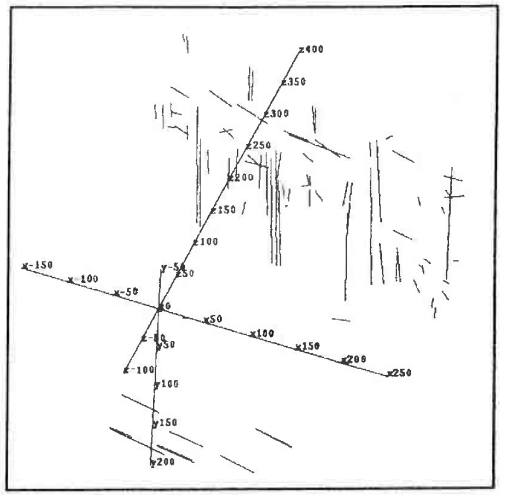


Figure B.9. Vue 4

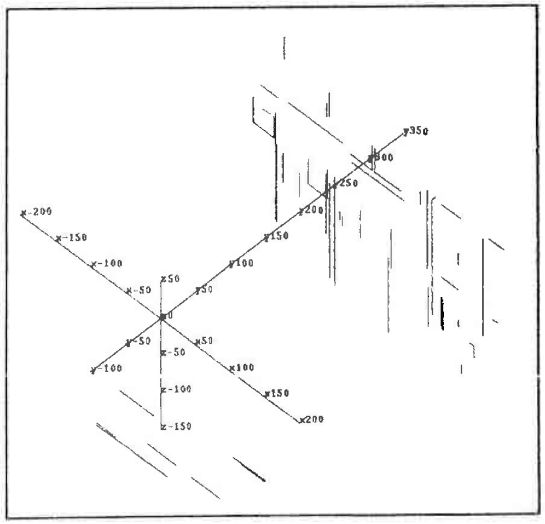


Figure B.10. MV_4

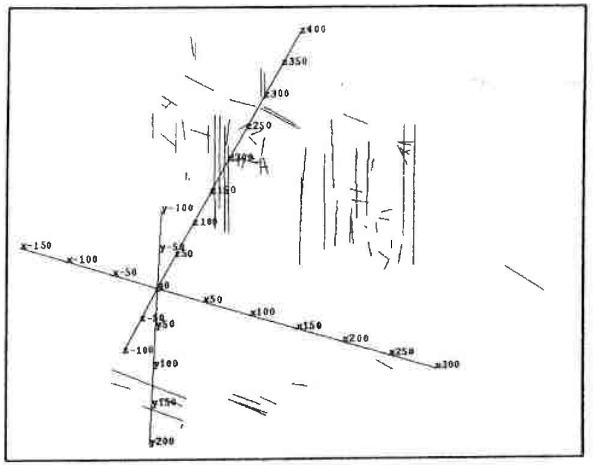


Figure B.11. Vue 5

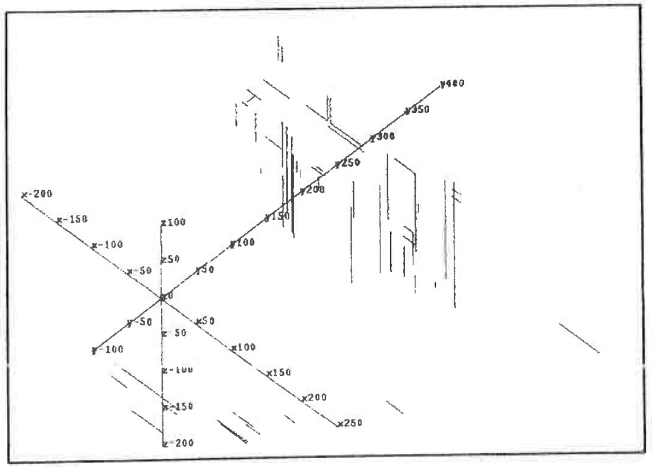


Figure B.12 MV_5

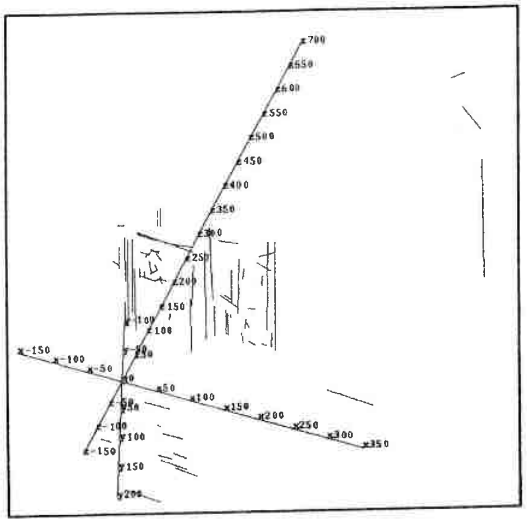


Figure B.13. Vue 6

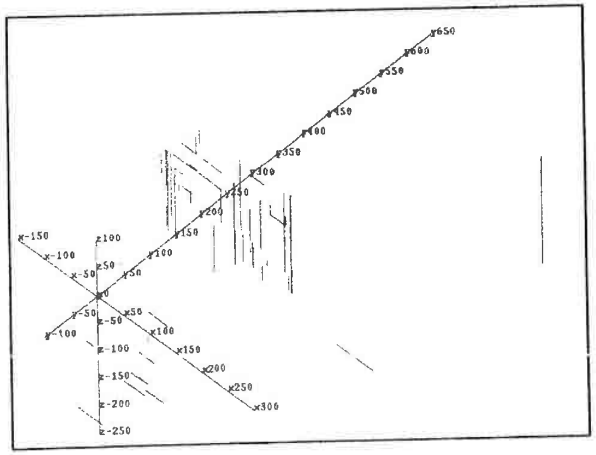


Figure B.14 MV_6

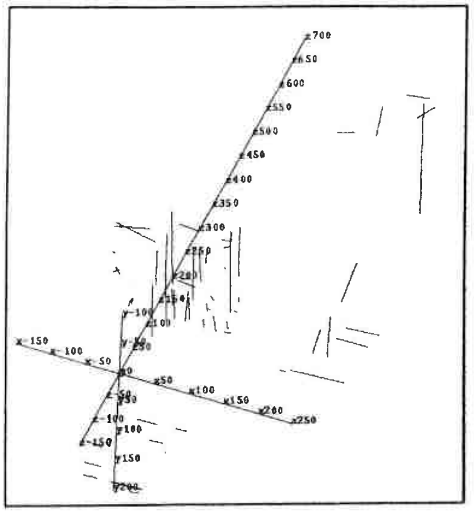


Figure B.15. Vue 7

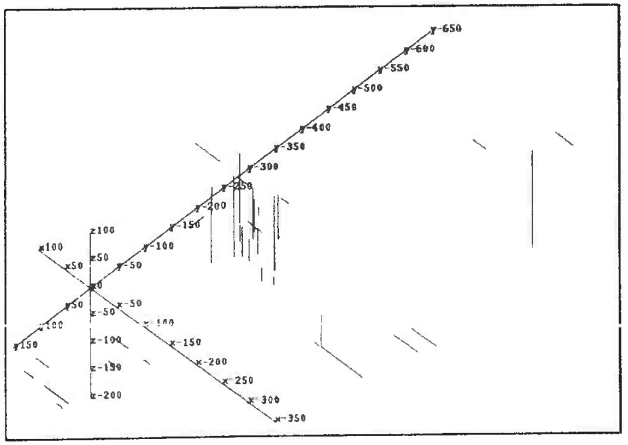


Figure B.16. MV_7

Annexe C
Motifs extraits de la vue 0

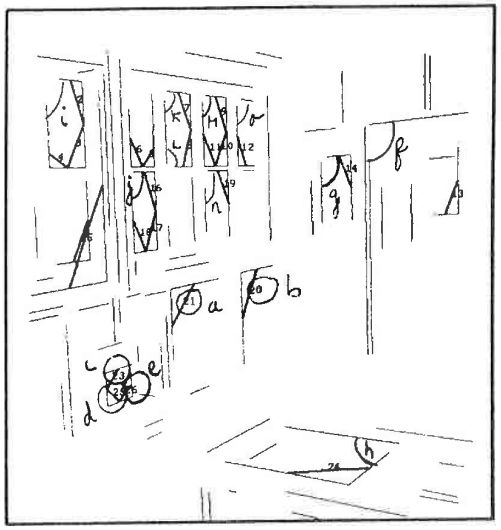


Figure C.1. Vue 0 : couples connexes

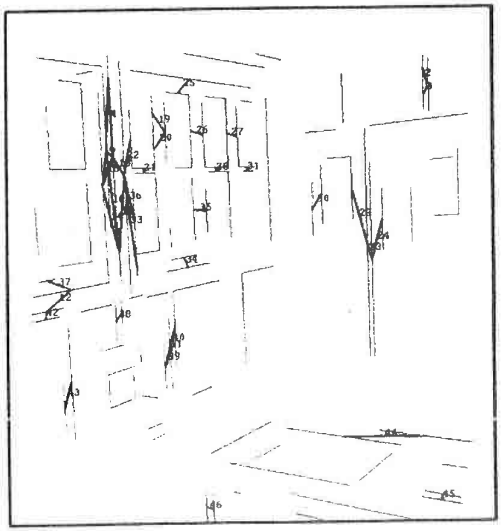


Figure C.2. Vue 0 : couples parallèles

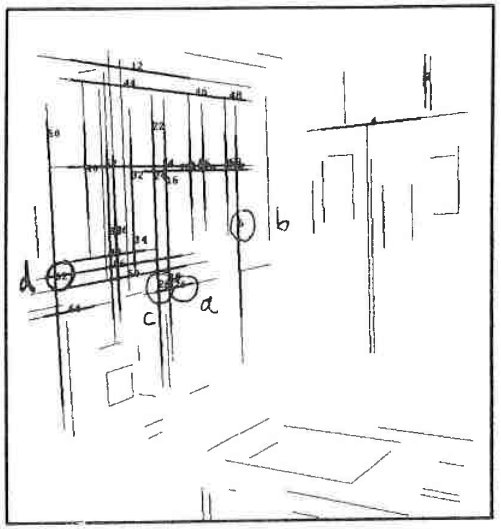


Figure C.3. Vue 0 : couples colinéaires

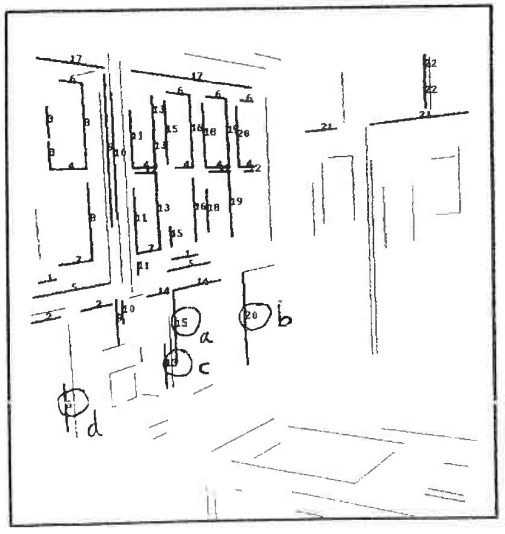


Figure C.4. Vue 0 : groupes colinéaires

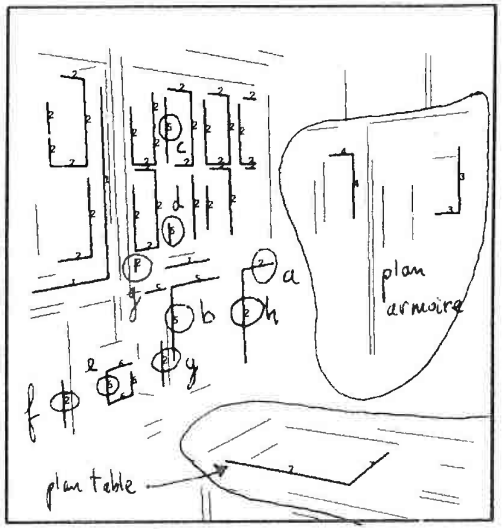


Figure C.5. Vue 0 : groupes coplanaires

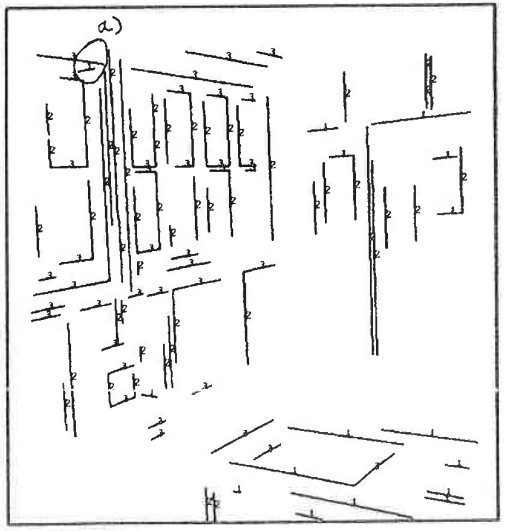


Figure C.6. Vue 0 : groupes directionnels

Annexe D
Motifs extraits de la vue 1

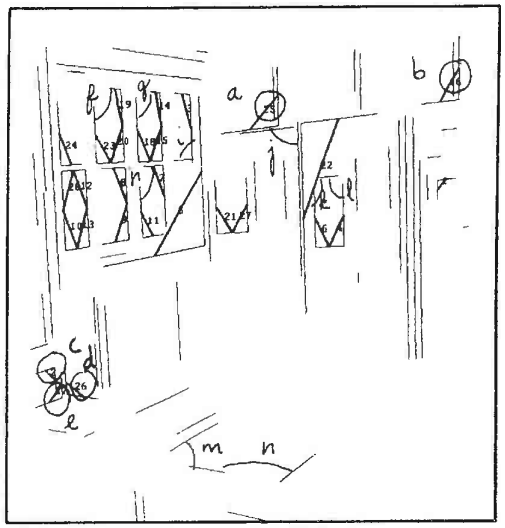


Figure D.1. Vue 1 : couples connexes

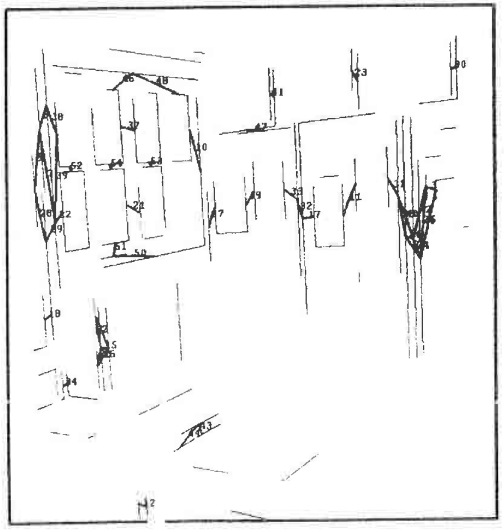


Figure D.2. Vue 1 : couples parallèles

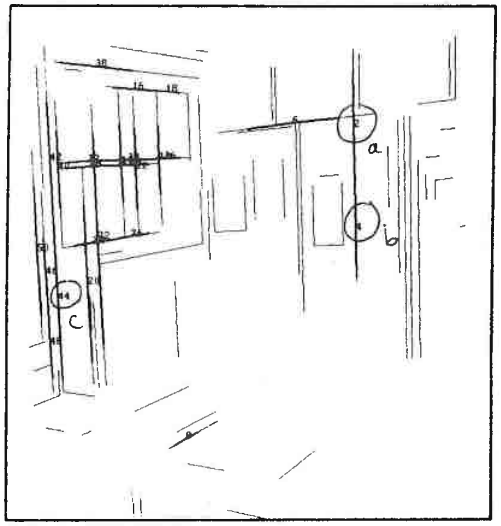


Figure D.3. Vue 1 : couples colinéaires

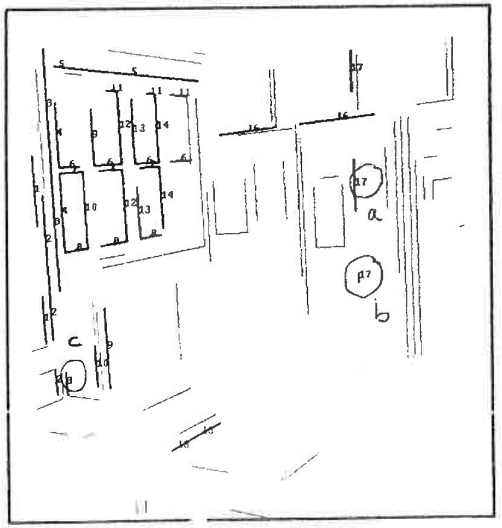


Figure D.4. Vue 1 : groupes colinéaires

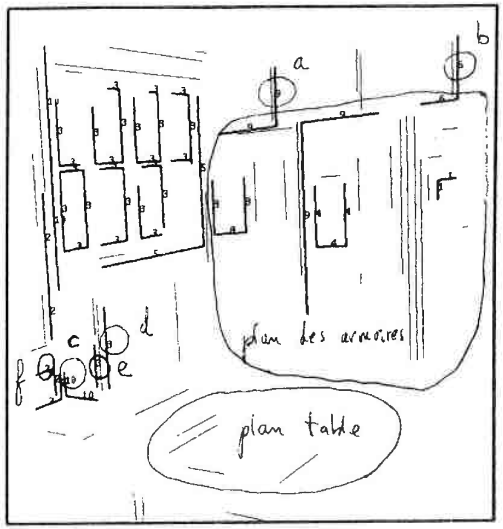


Figure D.5. Vue 1 : groupes coplanaires

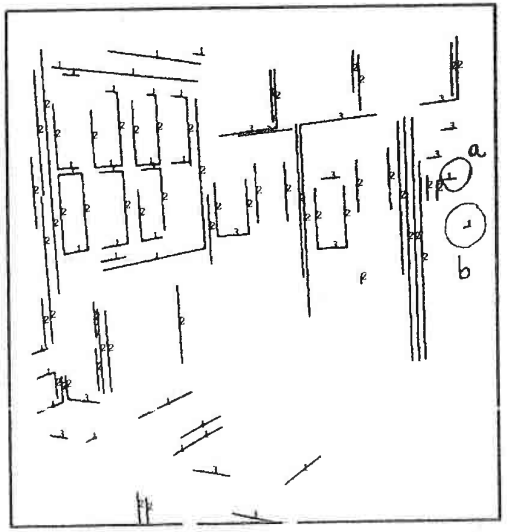


Figure D.6. Vue 1 : groupes directionnels

Annexe E

Résultats concernant la grille de calibration

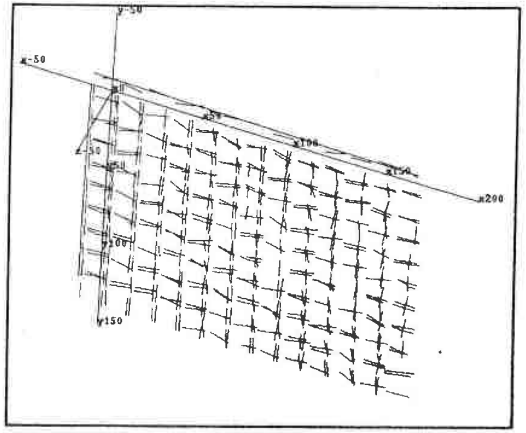


Figure E.1. Grille de calibration : segments stéréoscopiques

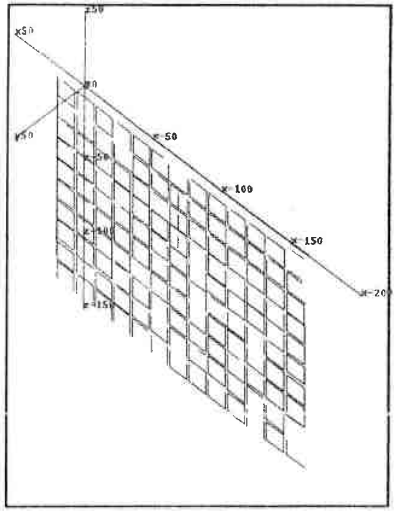


Figure E.2. Grille de calibration : modèle

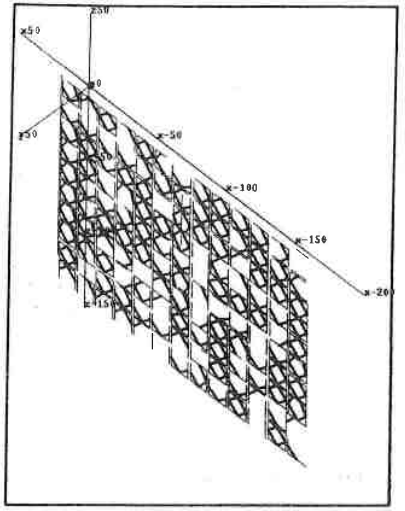


Figure E.3. Grille de calibration : couples connexes

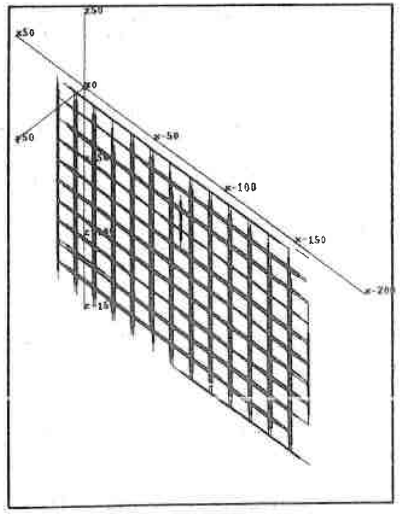


Figure E.4. Grille de calibration : couples colinéaires

Annexe F

Motifs appartenant à la fusion de $_{M}V_0$ et $_{M}V_1$

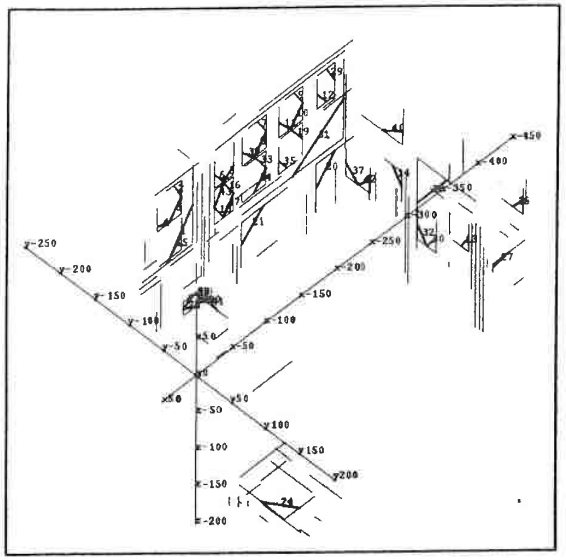


Figure F.1. Modèle 1 : couples connexes

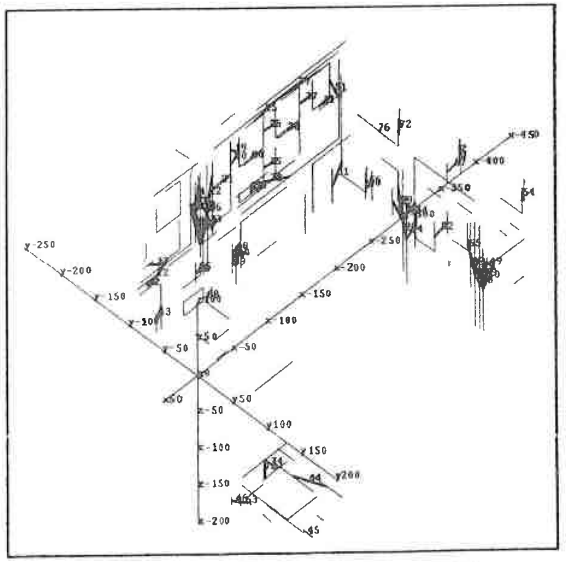


Figure F.2. Modèle 1 : couples parallèles

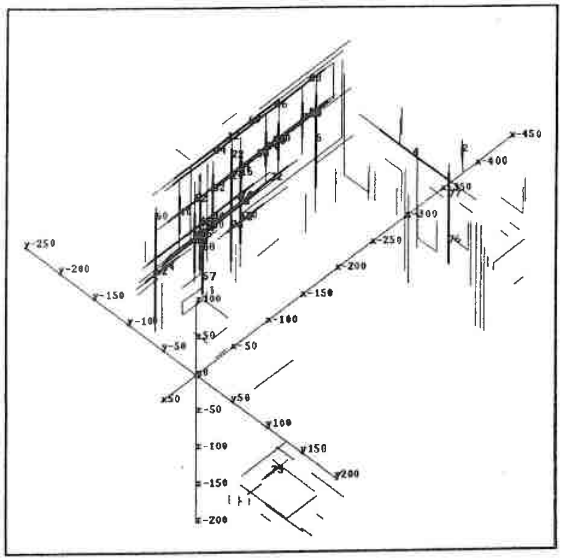


Figure F.3. Modèle 1 : couples colinéaires

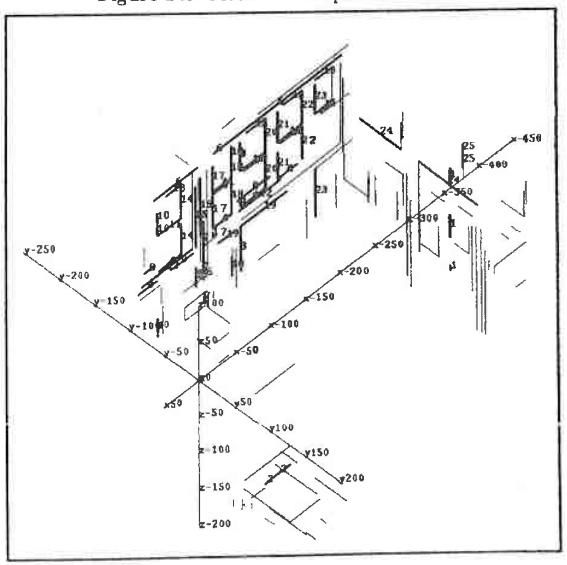


Figure F.4. Modèle 1 : groupes colinéaires

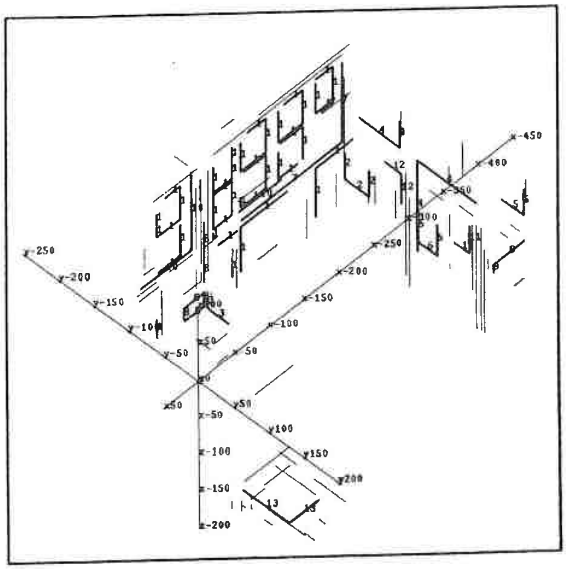


Figure F.5. Modèle 1 : groupes coplanaires

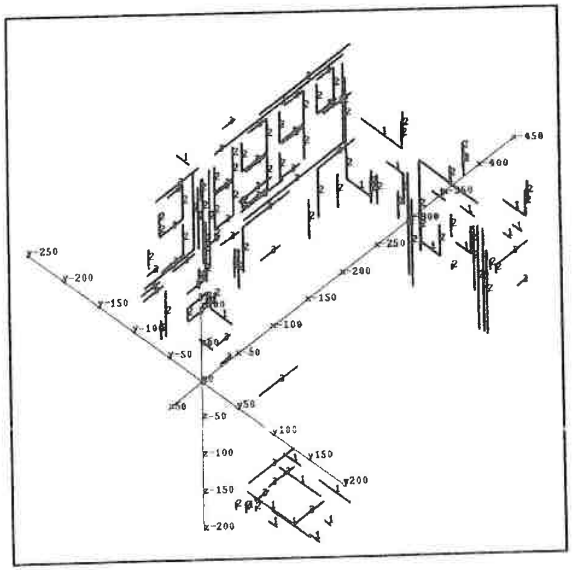


Figure F.6. Modèle 1 : groupes directionnels

Annexe G

Mise en correspondance des modèles associés à des vues successives

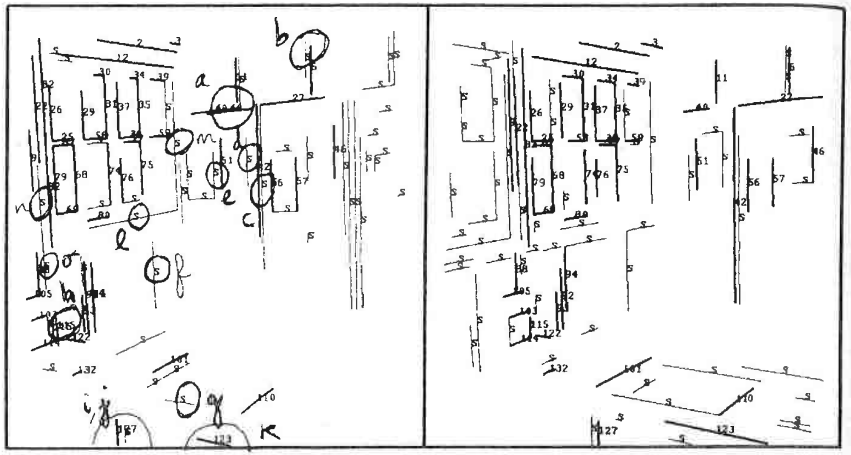


Figure G.1. Correspondances entre MV_1 et MV_0

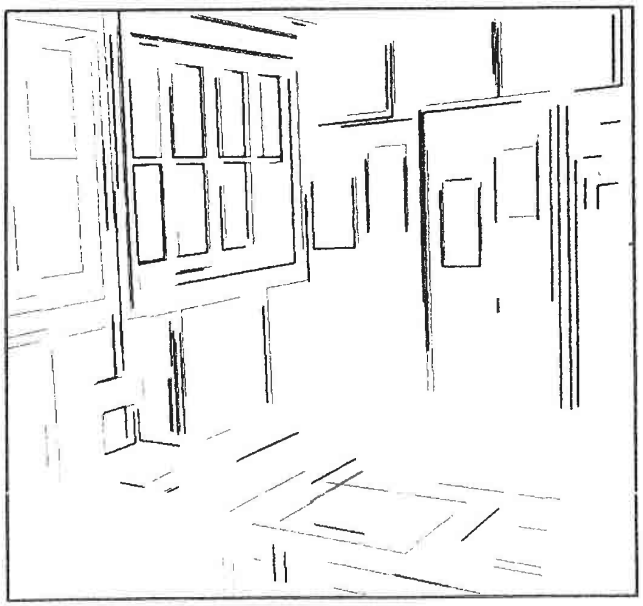


Figure G.2. Déplacement final entre MV_1 et MV_0

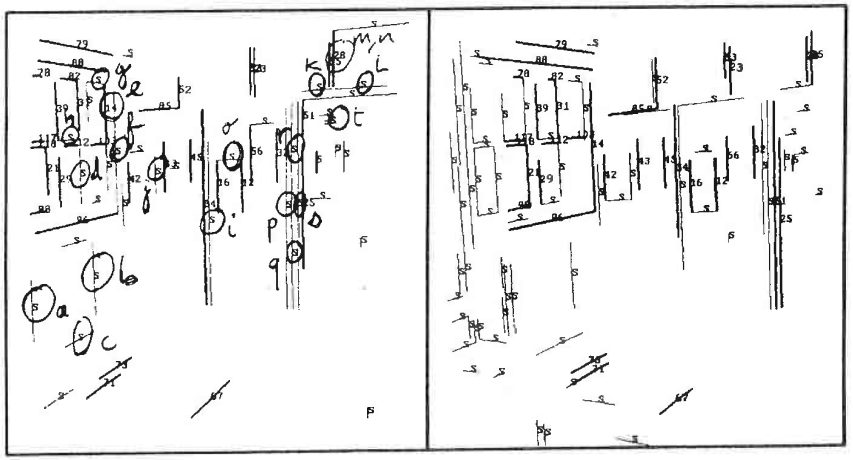


Figure G.3. Correspondances entre MV_2 et MV_1

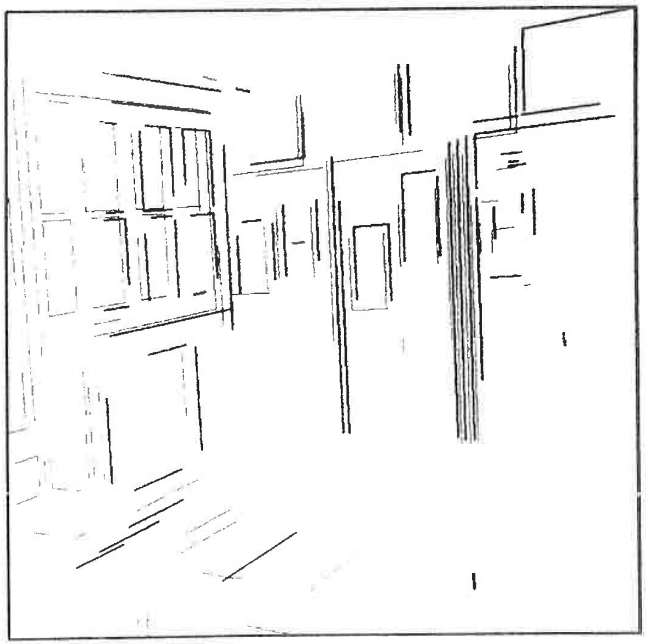


Figure G.4. Déplacement final entre MV_2 et MV_1

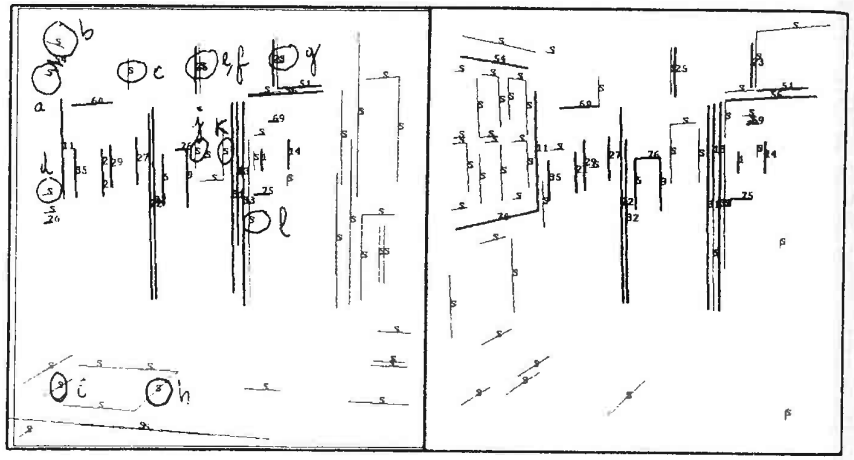


Figure G.5. Correspondances entre MV_3 et MV_2

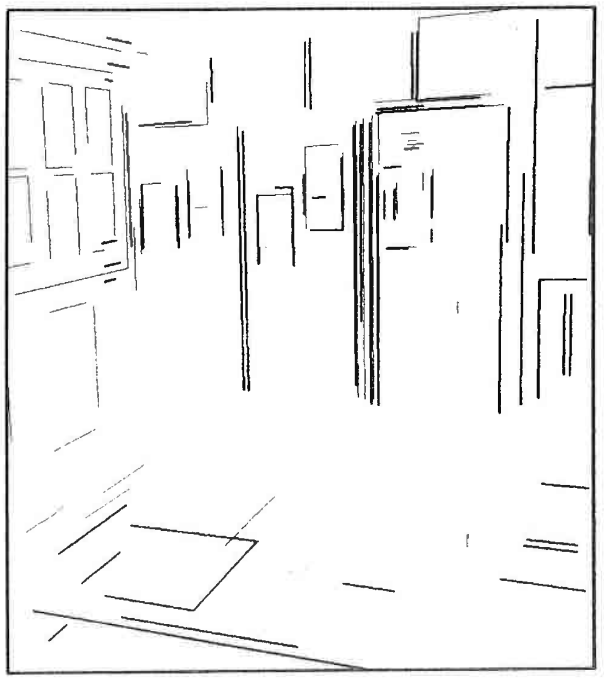


Figure G.6. Déplacement final entre MV_3 et MV_2

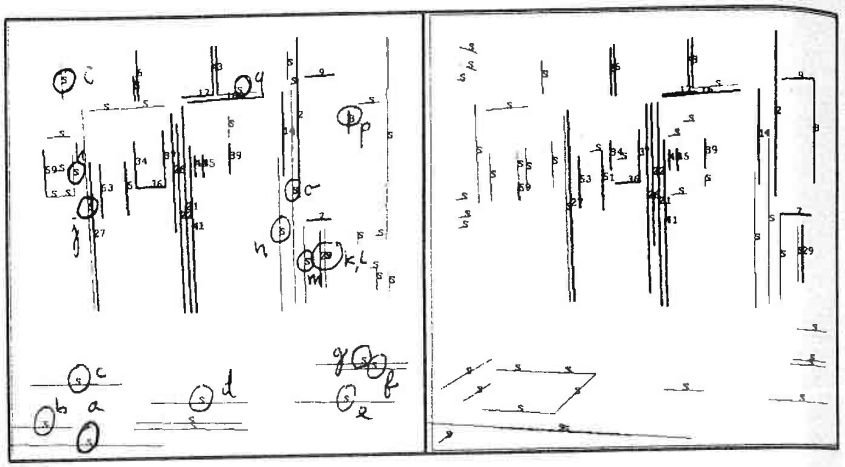


Figure G.7. Correspondances entre MV_4 et MV_3

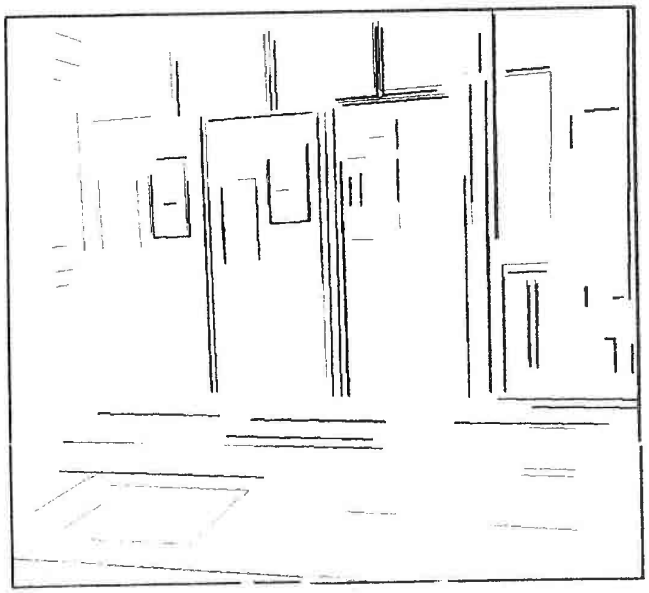


Figure G.8. Déplacement final entre MV_4 et MV_3

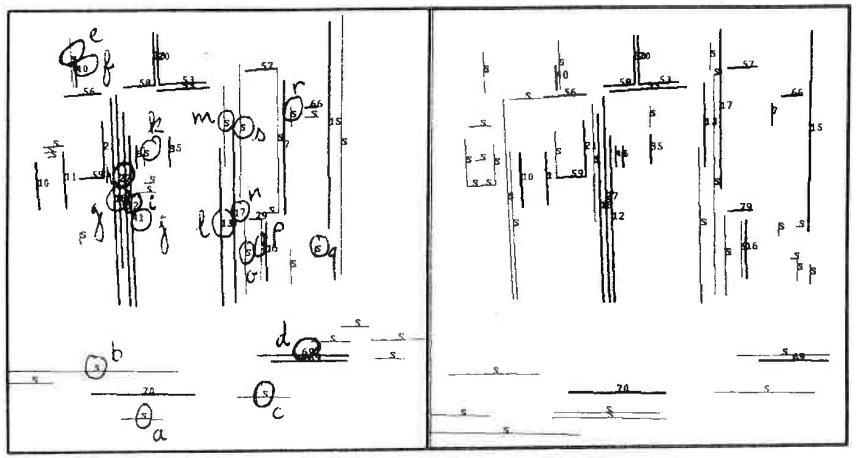


Figure G.9. Correspondances entre MV_3 et MV_4

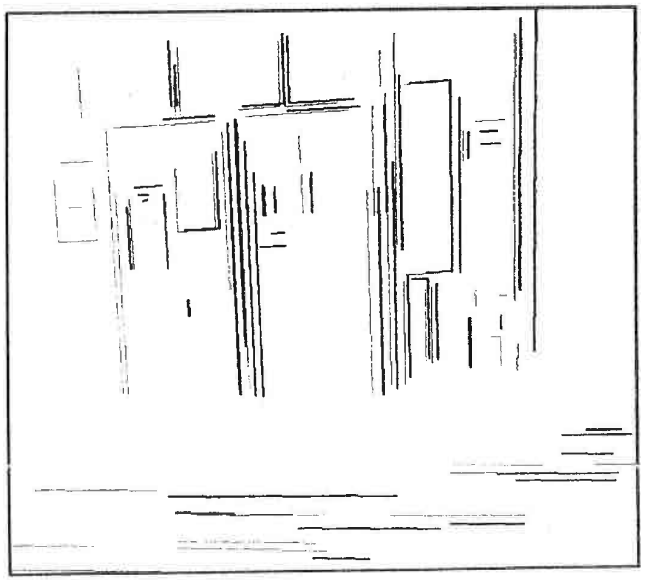


Figure G.10. Déplacement final entre MV_3 et MV_4

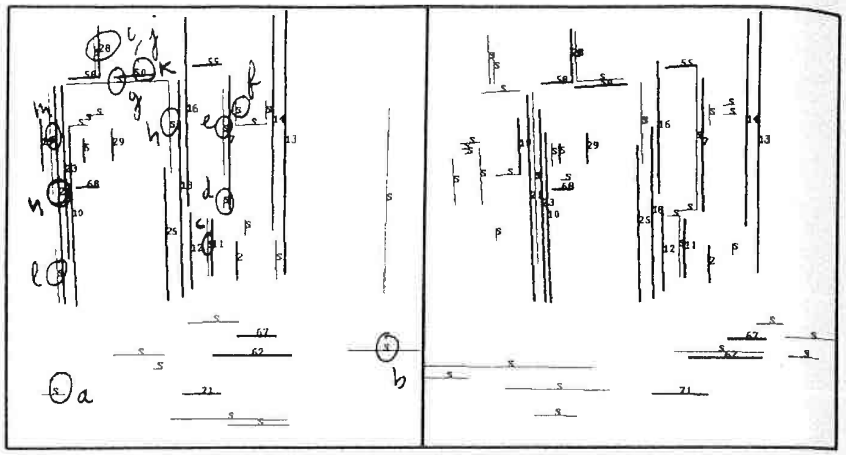


Figure G.11. Correspondances entre MV_6 et MV_5

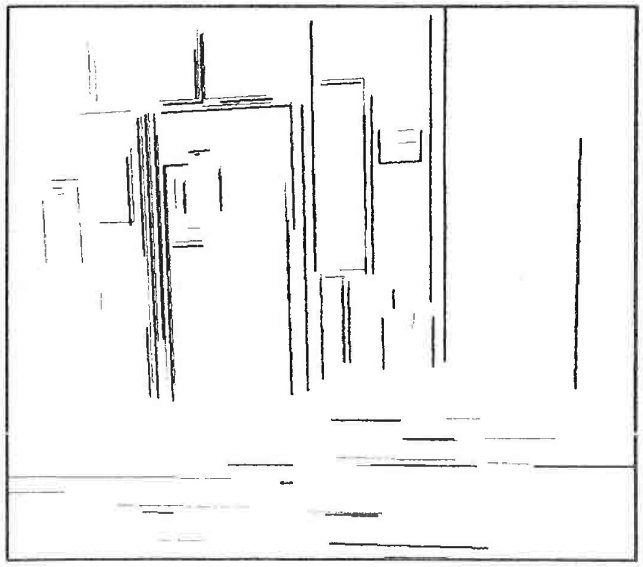


Figure G.12. Déplacement final entre MV_6 et MV_5

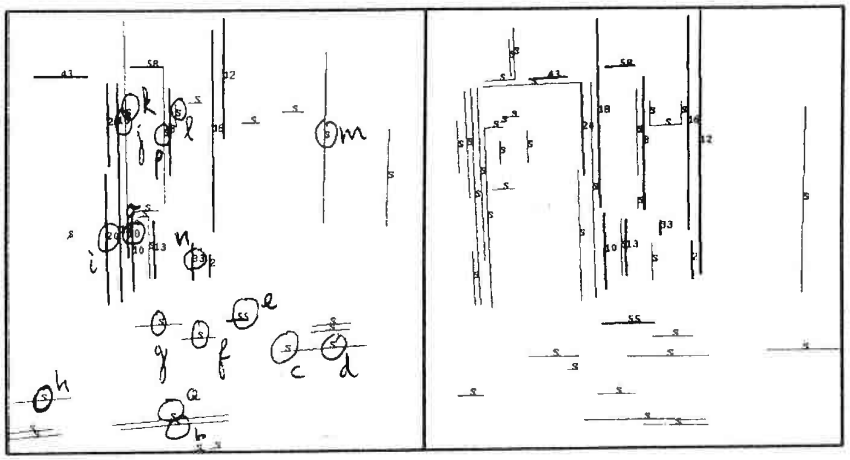


Figure G.13. Correspondances entre MV_7 et MV_6

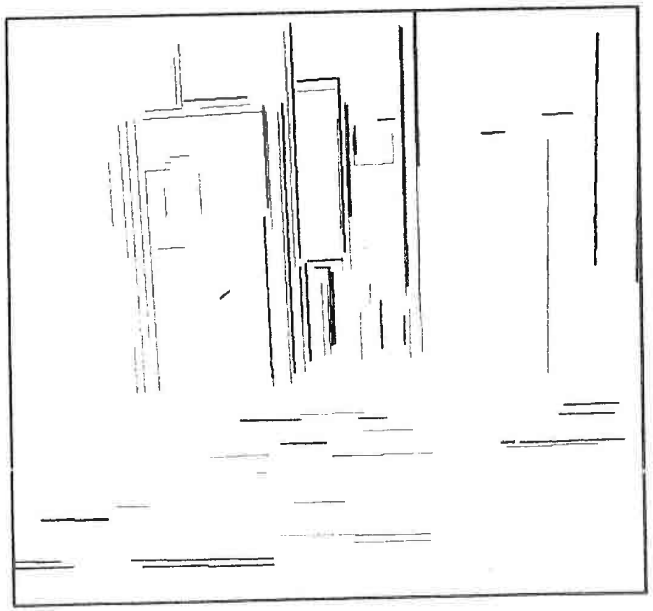


Figure G.14. Déplacement final entre MV_7 et MV_6

Annexe H

Déplacements initiaux

t_x (cm)	$t_y(cm)$	α (radians)	p
-36	36	$\pi/2$	32
-25	38	$\pi/2$	27
-427	-380	$3\pi/2$	14
-324	-368	$3\pi/2$	14
35	-306	0	12
-368	-22	π	12
-370	-7	π	11
-396	-391	$3\pi/2$	11
-444	-391	$3\pi/2$	10
-521	-391	$3\pi/2$	9
-308	-367	$3\pi/2$	9
-442	23	π	9
39	-321	0	9

Tableau H.1. Déplacements initiaux entre MV_1 et MV_0.

t_x (cm)	$t_y(cm)$	α (radians)	p
-276	339	π	16
4	19	$3\pi/2$	15
-6	8	$3\pi/2$	14
-7	18	$3\pi/2$	14
-52	612	$\pi/2$	13
50	610	$\pi/2$	13
322	291	0	12
39	608	$\pi/2$	9
54	619	$\pi/2$	8
311	293	0	8

Tableau H.2. Déplacements initiaux entre MV_2 et MV_1.

t_x (cm)	$t_y(cm)$	α (radians)	p
-308	-250	$3\pi/2$	17
-576	38	π	17
-291	303	$\pi/2$	15
-20	9	0	15
-26	-83	0	14
-23	20	0	13
-33	10	0	10
-280	295	$\pi/2$	10

Tableau H.3. Déplacements initiaux entre MV_3 et MV_2.

t_x (cm)	$t_y(cm)$	α (radians)	p
-17	20	$\pi/2$	24
-536	-6	$3\pi/2$	13
-262	-247	0	12
-276	275	π	10
-533	4	$3\pi/2$	10
-544	-96	$3\pi/2$	9
-528	22	$3\pi/2$	9
-275	-234	0	9
-291	275	π	7

Tableau H.4. Déplacements initiaux entre MV_4 et MV_3.

t_x (cm)	$t_y(cm)$	α (radians)	p
54	10	0	38
236	324	$\pi/2$	13
-79	508	π	12
-71	507	π	12
-261	191	$3\pi/2$	11
-247	202	$3\pi/2$	10
243	322	$\pi/2$	10
248	307	$\pi/2$	9
-52	502	π	8
-260	198	$3\pi/2$	7

Tableau H.5. Déplacements initiaux entre MV_5 et MV_4.

t_x (cm)	$t_y(cm)$	α (radians)	p
36	4	0	24
185	351	$\pi/2$	12
-171	498	π	12
-310	145	$3\pi/2$	11

Tableau H.6. Déplacements initiaux entre MV_6 et MV_5

t_x (cm)	$t_y(cm)$	α (radians)	p
158	236	$3\pi/2$	6
-96	516	0	6
-123	-18	π	5
-8	5	π	4
-378	262	$\pi/2$	4
-371	261	$\pi/2$	3

Tableau H.7. Déplacements initiaux entre MV_7 et MV_6.

Bibliographie

[Asa 85] T. Asano, M. Edahiro, H. Imai, M. Iri, et K. Murota. *Practical Use of Bucketing Techniques in Computational Geometry.* in *Computational Geometry*, G.T. Toussaint Ed., Elsevier Science Publishers, North Holland, 1985.

[Aya 86] N. Ayache et O.D. Faugeras. Hyper : A New Approach for the Recognition and Positioning of Two-Dimensional Objects. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 8(1):44-54, 1986.

[Aya 87] N. Ayache et F. Lustman. Fast and reliable passive trinocular stereovision. *Proceeding of the First International Conference on Computer Vision*, pages 422-427. London. 1987.

[Aya 88] N. Ayache. *Construction et fusion de représentations visuelles 3D.* Thèse d'Etat. Université de Paris-Sud. 1988.

[Bah 81] B. Bahnu. *Shape Matching and Image Segmentation using Stochastic Labeling.* Ph.D Thesis. University of Southern California, Image Processing Insitute. 1981.

[Bar 83] S.T. Barnard. Interpreting Perspective Images. *Artificial Intelligence*. 21:435-462. 1983.

[Bel 84] Y. Belaid. *Détection de surfaces par transformée de Hough en vision tridimensionnelle.* Thèse de troisième cycle. Université de Nancy 1. 1984.

[Bel 86] Y. Belaid, M. Camonin, G. Masini, R. Mohr, et E. Thirion. *Le système de vision TRIDENT. Rapport final du contrat ADI No 83.601.* Rapport CRIN no. 86-R-012. Centre de Recherche en Informatique de Nancy, 1986.

[Ble 75] W.W. Bledsoe. A new method for proving certain presburger formulas. *Advanced papers 4th International Joint Conference on Artificial Intelligence*. Tbilissi. 1975.

[Bob 77] D.G. Bobrow. Natural Language Input for a Computer Problem Solving System. *Cognitive Science*. 1:3-46. 1977.

[Bri 83] J.P. Briot. L'instanciation dans les langages objets. *Journées d'Etudes sur les Langages Orientés objets*, pages 173-209. 1983.

[Bro 78] R.A. Brooks, R. Greiner, et T.O. Binford. The acronym model-based vision system. *Proceedings of the ARPA Image Understanding Workshop*, Pittsburgh. 1978.

[Bro 81] R.A. Brooks. Symbolic Reasoning Among 3-D Models and 2-D Images. *Artificial Intelligence*, 17(1-3):285-348. 1981.

[Cam 87] M. Camonin. *MEPHISTO : un outil de validation de modèle tridimensionnel*. Thèse de troisième cycle, Université de Nancy 1, 1987.

[Cha 86] J. Chailloux, M. Devin, F. Dupont, J.M. Hullot, B. Serpette, et J. Vuillemin. *Le-lisp de L'INRIA, version 15.2. Le manuel de référence*. INRIA Rocquencourt. 1986.

[Con 87] J.H. Connell et M. Brady. Generating and Generalizing Models of Visual Objects. *Artificial Intelligence*. 1987.

[Cro 85] J.L. Crowley. Dynamic world modelling for an intelligent mobile robot unsing a rotating ultrasonic ranging device. *IEEE conference on Robotics and Automation*. St Louis. 1985.

[Doy 79] J. Doyle. A Truth Maintenance System. *Artificial Intelligence*, 12(12):231-272. 1979.

[Ede 62] M. Eden. Handwriting and Pattern Recognition. *IRE Trans. Inf. Theory*. 2. 1962.

[Fau 83] O. Faugeras et M. Hebert. A 3-d recognition and position algorithm using geometric matching between primitive surfaces. *Proc. IJCAI83*, pages 996-1002. Karlsruhe. 1983.

[Fol 82] R.M. Foley et A. Van Dam. *Fundamentals of Interactive Computer*. Addison-Wesley, Reading Massacusetts. 1982.

[Fu 74] K.S. Fu. *Syntactic Methods in Pattern Recognition*. Academic Press, New York. 1974.

[Gol 83] A. Goldberg et D. Robson. *Smalltak 80 The language and its implementation*. Addison Wesley Publishing Company. 1983.

[Gon 78] R.C. Gonzalez et M.G. Thomason. *Syntatic Pattern Recognition*. Addison-Wesley. 1978.

[Gri 59] R.L. Grimsdale. Automatic Pattern Recognition - New Morphological System Using a Digital Computer. *Wireless World*. (65):499-501. 1959.

[Han 78] A.R. Hanson et E.M. Riseman. Visions: a Computer System for Interpreting Scenes. in *Computer Vision Systems*. Hanson and Riseman editors. Academic Press, New York, pages 303-334. 1978.

[Har 78] R. Haralick, L.Davis, A. Rosenfeld, et D. Milgram. Reduction Operations for Constraint Satisfaction. *Information Science*. 8. 1978.

[Hen 81] T.C. Henderson et L. Davis. Hierarchical Models and Analysis of Shape. *Pattern Recognition*. 14(1-6):197-204. 1981.

[Her 86] M. Herman et T. Kanade. Incremental Construction of 3D Scenes from Multiple. Complex Images. *Artificial Intelligence*. 30. 1986.

[Hor 75] S.L. Horowitz. A Syntactic Algorithm for Peak Detection in Waveforms with Application to Cardiography. *Communications of the ACM*. 18(5). 1975.

[Hul 84] J.M. Hullot. *CEYX Version 4 Le manuel de référence*. Rapport INRIA. 1984.

[Jaz 70] A.M. Jazwinsky. *Stochastic Processing and Filtering Theory*. Academic Press. 1970.

[Knu 71] D.E. Knuth. *Semantics of context-free languages*. Math. Syst. Theory 5. 1. 1971.

[Knu 75] D.E. Knuth. *Sorting and Searching*. in *The Art of Computer Programming*. Vol. 3. Addison Welsey. Reading. Ma.. 1975.

[Lau 82] J.L. Laurière. Utilisation et représentation des connaissances. *TSI*. 1982.

[Led 64] R.S. Ledley. High-Speed Automatic Analysis of Biomedical Pictures. *Science*. 1964.

[Low 83] D.G. Lowe et T.O. Binford. Perceptual organization as a basis for visual recognition. *in Proc. of the National Conference on Artificial Intelligence*. pages 255-260. Washington. 1983.

[Low 87] D.G. Lowe. Three-Dimensionnal Object Recognition from Single Two-Dimensional Images. *Artificial Intelligence*. 31. 1987.

[Lux 85] A. Lux. *Algorithmique et controle en vision par ordinateur*. Thèse d'Etat. Institut National Polytechnique de Grenoble. 1985.

[Mac 85] A.K. Mackworth et E.C. Freuder. The Complexity of some Polynomial Network Consistency Alogorithms for Constraint Satisfaction Problems. *Artificial Intelligence*. 25. 1985.

[Mar 82] D. Marr. *Vision*. Freeman, San Francisco. 1982.

[Mar 83] W.N. Martin et J.K. Aggarval. Volumetric Descriptions of Objetcs from Multiple Views. *IEEE Transaction on Pattern Analysis and Machine Intelligence*. 5(2):150–158. 1983.

[Mas 78] G. Masini. *Réalisation d'un système de reconnaissance structurelle et d'interprétation de dessins*. Thèse de troisième cycle. Université de Nancy I. 1978.

[Mas 83] G. Masini et R. Mohr. MIRABELLE: A System for Structural Analysis of Drawings. *Pattern Recognition*. 16(4):363–372. 1983.

[Mas 85] G. Masini, R. Mohr, et E. Thirion. Stratégie de perception pour un modèle hiérarchique. *Actes du 5ème congrès AFCET Reconnaissance des formes et Intelligence Artificielle*. pages 631-640. Grenoble, 1985.

[Moh 79] R. Mohr. *Descriptions structurées et analyse de formes complexes*. Thèse de Doctorat d'Etat. Université de Nancy 1. CRIN 79-T-033. 1979.

[Moh 85] R. Mohr et T.C. Henderson. Arc and Path Consistency Revisited. *Artificial Intelligence*. 28(2):225–233. 1985.

[Mon 74] U. Montanari. Networks of Constraints : Fundamental Properties and Applications to Picture Processing. *Information Science*. 7. 1974.

[Mon 87] O. Monga et B. Wrobel. Segmentation d'images : vers une méthodologie. *Revue Traitement du Signal (TS). Numéro spécial : Vision par ordinateur*. 4:169-193. 1987.

[Nar 66] R. Narasimhan. Syntax-Directed Interpretation of Classes of Pictures. *Communication of the ACM*. 9(3):166–173. 1966.

[New 73] A. Newell. *Production systems : models of controls structures*. Academic Press, New York. 1973.

[Qua 87] L. Quan et R. Mohr. Location of a mobile robot by natural frames from a monocular image. *Proc. International Symposium on Technologies for Optoelectronics*. pages 114-120. Cannes. 1987.

[Qua 88] L. Quan et R. Mohr. *Determining Perspective Structures Using Hierarchical Hough Transform*. Rapport CRIN no. 88-R-063. CRIN, Université de Nancy 1. 1988.

[Ros 76] A. Rosenfeld, R. Hummel, et S. Zucker. Scene Labelling by Relaxation Operations. *IEEE Transaction on Systems, Man and Cybernectics*. 6(6):420–433. 1976.

[Sha 69] A.C Shaw. A Formal Picture Description Description Scheme as a Basis for Picture Processing Systems. *Information and Control*. 14(1):9–52. 1969.

[Sho 77] R.E. Shostak. On the Sup-inf Method for Proving Pressburger Formulas. *Journal of the ACM*. 24(4). 1977.

[Sou 83] V. Souvignier. *PVV - Un système d'interprétation d'images par prédiction-vérification*. Thèse de $3^{ème}$ cycle. INP Grenoble. 1983.

[Thi 84] E. Thirion. *Description de l'interpréteur du système de vision Trident*. Rapport de DEA no. 84-R-076. CRIN. Université de Nancy 1. 1984.

[Tuc 87] M. Tuceryan, A.K. Jain, et Y. Lee. Extracting perceptual structures in line patterns. *Proc. on the 5th Scandinavian Conference on Image Analysis*. pages 531-538. Stockholm. 1987.

[Vam 73] T. Vamos et Z. Vassy. Industrial pattern recognition experiment - a syntax aided approach. *International Join Conference on Pattern Recognition*. pages 445-452. Washington. 1973.

[Wer 38] M. Wertheimer. *Laws of Organisation in Perceptual Forms*. in *A source Book of Gestalt Psychology*. W.D. Ellis editor. Harcourt, Brace. New York. pages 71-88. 1938.

[Wey 86] T.E. Weymouth. *Using Object Description in a Schema Network for Machine Vision*. COINS Technical Report no. 86-24. University of Massachusetts. 1986.

[Win 84] P.H. Winston. *Artificial Intelligence*. Addison-Wesley. 1984.

[Wit 83] A.P. Witkin et J.M. Tenenbaum. *On the Role of Structure in Vision*. in *Human and Machine Vision*. J. Beck, B. Hope and A. Rosenfeld ed., Academic Press. New York. pp 481-543. 1983.

[Wro 88] B. Wrobel-Dautcourt. *Perception de la distance par mise en correspondance de régions entre des images stéréoscopiques*. Thèse de l'Institut National Polytechnique de Lorraine. 1988.

[Zar 83] F. Zaroli. *Réalisation d'un système d'analyse et d'interprétation d'images tridimensionnelles.* Thèse de troisième cycle, Université de Nancy 1, 1983.

[Zuc 78] S.W. Zucker et J.L. Mohammed. A hierarchical relaxation system for line labeling and grouping. *Proceedings IEEE Conference on Pattern Recognition and Image Processing*, pages 410–415, 1978.

[Zuc 83] S.W. Zucker. *Computational and Psychophysical Experiments in Grouping : Early Orientation Selection.* in *Human and Machine Vision.* A. Rosenfeld, B. Hope and J. Beck editors, Academic Press, New York, 1983, 1983.

Table des matières

Liste des Figures

Résumé

Cette thèse présente le résultat de notre travail dans deux projets de vision par ordinateur : ORASIS et TRIDENT. L'objectif du premier projet est de permettre à un robot mobile de se déplacer dans un bâtiment qu'il ne connaît pas *a priori*. Pour cela, le robot doit être capable d'acquérir automatiquement une représentation de son environnement par "apprentissage visuel", c'est-à-dire par simple observation. Dans le projet ORASIS, nous avons réalisé un système permettant cette acquisition automatique par fusion de plusieurs vues stéréoscopiques. Avant d'intégrer une nouvelle vue dans le modèle, une analyse monoculaire permet d'extraire des relations entre les segments perçus ainsi que des groupes de segments. Ces relations et ces groupes ont deux intérêts. Ils servent tout d'abord à corriger les coordonnées des segments et ainsi à améliorer la qualité des données stéréoscopiques. Ensuite, ils sont utilisés pour estimer de manière fiable, la position des segments perçus par rapport aux segments du modèle déjà construit.

Le second projet est orienté vers la reconnaissance et la représentation d'objets génériques (objets pouvant prendre plusieurs formes). Le modèle, donné par l'utilisateur, est un ensemble de règles décrivant les différentes formes possibles que peuvent prendre les objets. Les données interprétées sont des vues synthétiques tridimensionnelles obtenues par un système de simulation. A partir d'un tel modèle et d'une telle vue, le système réalisé fournit une description structurée de la scène en faisant coopérer plusieurs analyses ascendantes/descendantes appliquées à différentes parties de la vue. Les analyses communiquent par propagation des connaissances ce qui permet de réduire l'indéterminisme des choix en exploitant le contexte. Les résultats obtenus montrent que la meilleure stratégie consiste à effectuer en premier lieu les interprétations les moins ambigües sans imposer de contraintes sur la manière de construire la description (nombre d'analyses en parallèles, construction ascendante ou descendante).

Mots clés :

- vision par ordinateur - stéréovision - appariement
- organisation perceptuelle - modélisation - interprétation -
- approche syntaxique -