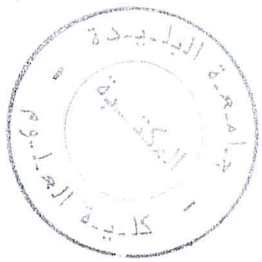
République Algérienne Démocratique et Populaire
Ministère de l'Enseignement supérieur et de la recherche Scientifique
Université de Saad Dahleb
Blida

Faculté des Sciences
Département d'Informatique

Mémoire de fin d'étude
Pour l'obtention du diplôme
D'ingénieur d'état en Génie Informatique

Option : Intelligence Artificielle



Thème

# Indexation géométrique d'une base d'images

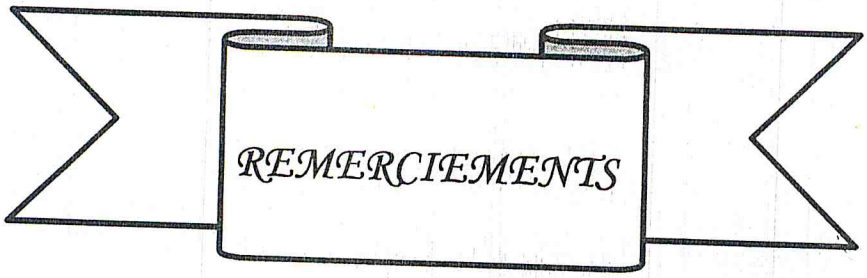
Réalisé par :

*Ghemari Fouzia*
*Benadouane Habiba*

Encadré par :

*M^elle N.Benblidia*
*M^elle F.Reguieg*

Promotion 2003/2004



# REMERCIEMENTS

*Nous tenons tout d'abord à remercier nos promotrices*

*M^elle N.Benblidia et M^elle FZ.Reguieg pour nous avoir encadré tout au long de ce travail, et aussi pour leurs encouragements .*

*Nous profitons de l'occasion pour remercier du fond du cœur*

*Monsieur Laichi Boualem qui nous a aidé beaucoup.*

*Nos remerciements vont également à tout ceux qui nous ont aidé, de manière directe ou indirecte à réaliser ce travail.*

# Dédicaces

*A mes défunts parents qui m'ont donné la vie, symbole de bonté, de fierté, de sagesse et de patience, et qui à ma grande peine ne sont plus la pour apprécier le résultat.*

*A ceux qui sont la source de mon inspiration et mon courage à qui je dois de l'amour et de la reconnaissance.*

*A mes sœurs : Hakima et Djamila.*

*A mes frères : Mohamed, Rabeh, Kamel, Karim et surtout Samir.*

*A mon mari Noureddine et à toute sa charmante famille.*

*A ma belle mère Kheira et mes belles sœurs : Asma, Sihem et Sarah.*

*A Larachi Rabeh et Laichi Boualem*

*A mes neveux et nièces : Anes, AbdelRaouf, SidAli, Anfel, Sahar, Fatiha, Narhimene et Aya.*

*A ma grande-mère, mes oncles, mes tantes et à mes chers voisins et voisines surtout tata Zolikha.*

*A mon binôme Habiba et à toute sa famille*

*A tous mes amis que j'aime et qui m'aiment surtout : Nacéra, Salima, Keltoum, Samira, Akila, Samia, Nabiha.*

*Symbole d'un grand amour éternel que Dieu les gardes, en témoignage de ma grande reconnaissance et mon profond attachement.*

*Fouzia*

# Dédicaces

*Je dédie ce modeste travail à mes chèrs parents surtout ma mère.*

*A Mes frères (Abdelnour, Mahieddine, Achour, Ferhat), je vous réserve toujours une place dans mon cœur et mes pensées ; vous ensuite car j'ai appris avec vous et grâce à vous jusqu'aux moindres notions de la vie.*

*A ma sœur Nacéra, son mari Said et leurs enfants Abdou et Amina.*

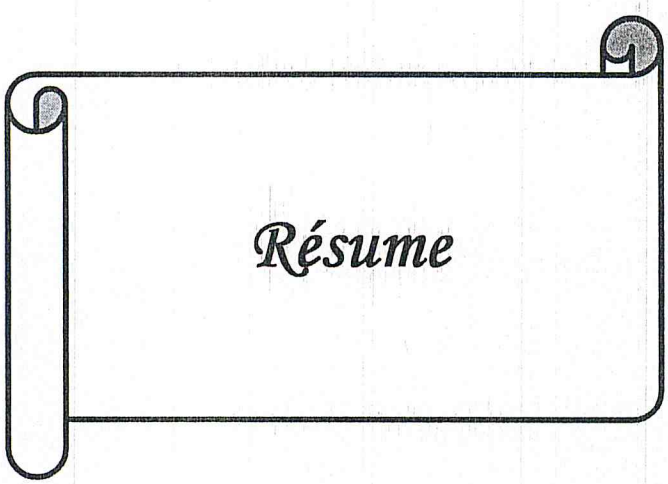
*A mes belles sœurs Saida, Zahia, Khadidja et leurs enfants: Aymen et Bouchera.*

*A mon binôme Fouzia et toute sa famille.*

*A vous aussi mes amis, je tiens à vous dire que jamais je n'oublierai aucun de vous (Fatiha, Ratiba, Nawel, hadjira, Samia, Nabiha, Kelthoum, Akila,Amel, Rachid, Sofiane, Fateh, K. Hamza).*

*A tous mes amis que j'aime et qui m'aiment.*

*Habiba*


Résume

## *Résumé :*

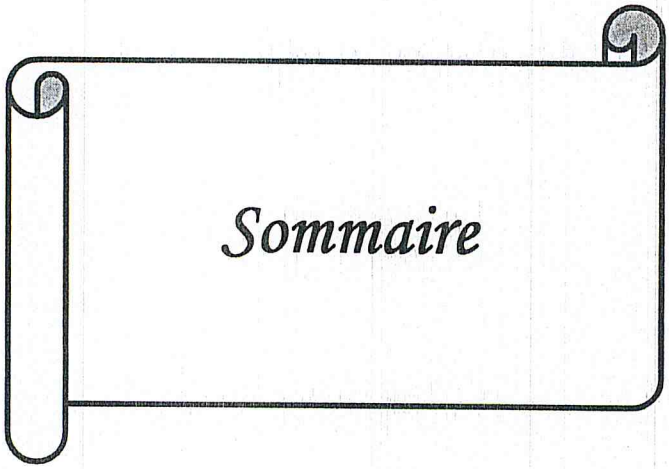
Nous présentons dans ce mémoire un ensemble qui concerne l'appariement, la reconnaissance et l'indexation des images. Cet ensemble de techniques concourt au développement d'un système de reconnaissance automatique d'images.

Dans un premier temps, nous présentons une étude bibliographique comportant les principales techniques d'indexation d'images adaptées spécifiquement aux images segmentées. Puis nous avons proposé un algorithme détaillé pour l'une de ces méthodes puis nous avons vérifié leur validité expérimentalement par la suite.



## *Mots clés :*

Vision par ordinateur, Appariement, Reconnaissance, Indexation, Les quasi invariants géométriques, Descripteurs locaux.



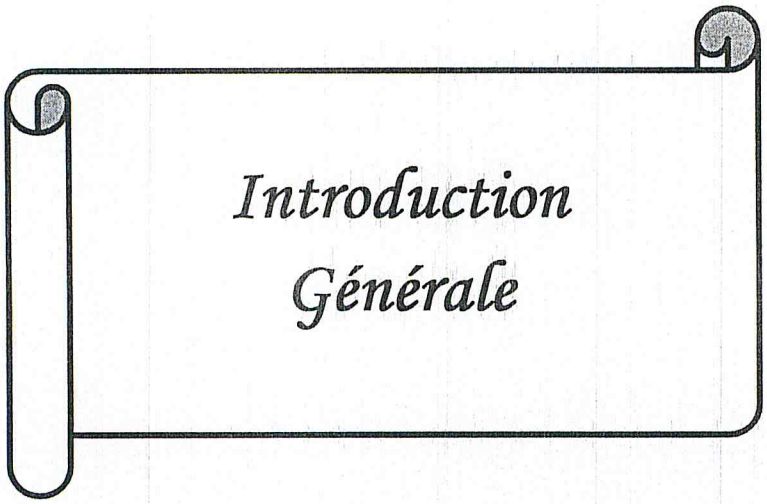
# *Sommaire*

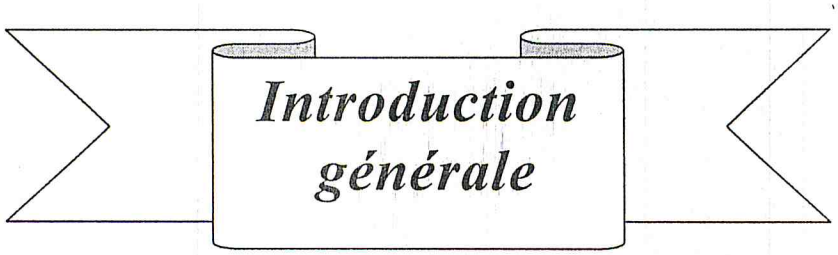
Chapitre II: Structures d'indexation



Chapitre III: Techniques d'indexation

# *Table des figures*


Introduction
Générale




La vision artificielle a pour but de permettre à des systèmes automatiques, informatiques ou robotiques, d'acquérir et de traiter des informations visuelles, en s'affranchissant au maximum des capacités visuelles d'un opérateur humain. En cela, elle diffère de l'imagerie, qui utilise aussi des images, mais qui laisse les opérations de traitement de haut niveau et d'interprétation à un opérateur humain

L'objectif de la vision par ordinateur est l'étude de l'art et la manière d'obtenir des informations sur notre environnement. Les solutions proposées au début tentaient de donner une représentation algorithmique et automatisable de la vision biologique; et puisque cette dernière dépend d'un énorme nombre de facteurs, cette solution est irréalisable. Les plus récents tendent à exploiter des informations extraites des images, qu'elles soient d'ordre géométrique ou d'ordre photométrique.

Le groupe des algorithmes qui nous intéresse, est celui basé sur l'apparence des objets, et qui réduit la tâche de la reconnaissance à un problème d'appariement. Ces algorithmes sollicitent la construction d'un système capable de classer les images par leurs contenus pour les stocker. Ceci nécessite les compétences des domaines des bases de données et du traitement d'images. Ce dernier apporte les techniques d'analyse qui permettent l'extraction des caractéristiques assez discriminantes des images pour pouvoir les décrire au mieux par leur contenu.

Alors que les bases de données sont utilisées dès que la taille de la base d'objets est importante. Donc, elles sont utilisées pour accélérer la vitesse de la recherche et optimiser l'espace mémoire utilisé

Dans ce mémoire, nous allons présenter une étude bibliographique sur les approches principales de la reconnaissance en se basant surtout sur les techniques d'indexations géométriques.

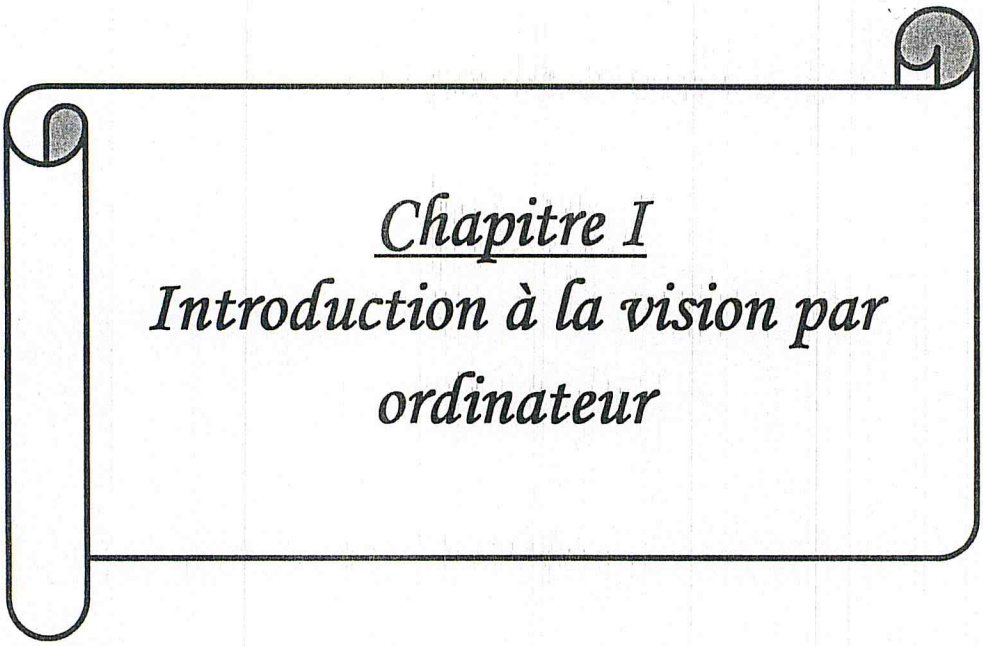
Ensuite, nous allons proposer un algorithme détaillé pour une de ces méthodes qui est ***l'indexation géométrique étendue***.

Cette méthode sert à indexer les images segmentées suivant des ***quasi-invariants*** (descripteurs géométriques locaux) qui sont l'angle formé par deux segments adjacents et leur rapport de longueurs.

Pour l'implémentation de l'algorithme de cette méthode, nous avons utilisé le ***Quad-tree*** pour indexer les données bidimensionnelles. Comme nous avons utilisé la technique d'indexation le ***VA-File*** pour l'étape de vote, puisqu'elles ont pratiquement le même concept. Enfin, nous avons testé l'algorithme proposé pour prouver son efficacité expérimentalement.

Ce mémoire est organisé en 5 chapitres:

- Dans le premier chapitre nous avons présenté quelques notions de base sur la vision artificielle.
- Dans le deuxième chapitre nous avons exposé les déférentes structures de stockage et nous avons opté pour le Quad-tree, une structure de stockage de données bidimensionnelles.
- Dans le troisième chapitre nous avons étudié les déférentes méthodes d'indexation pour la reconnaissance, notamment celles basées sur les propriétés géométriques des objets.
- Dans le quatrième chapitre, nous avons présenté notre méthode de reconnaissance d'objets que nous avons implémenté avec beaucoup détail.
- Dans le dernier chapitre nous avons présenté l'application réalisée, les tests effectués et les statistiques qui nous aident à déterminer les paramètres de l'algorithme de reconnaissance.
- Enfin, dans l'annexe nous avons présenté quelques exemples d'utilisation de l'application.



# *Chapitre I*
# *Introduction à la vision par ordinateur*

# *Chapitre I*

# *Introduction à la vision artificielle*

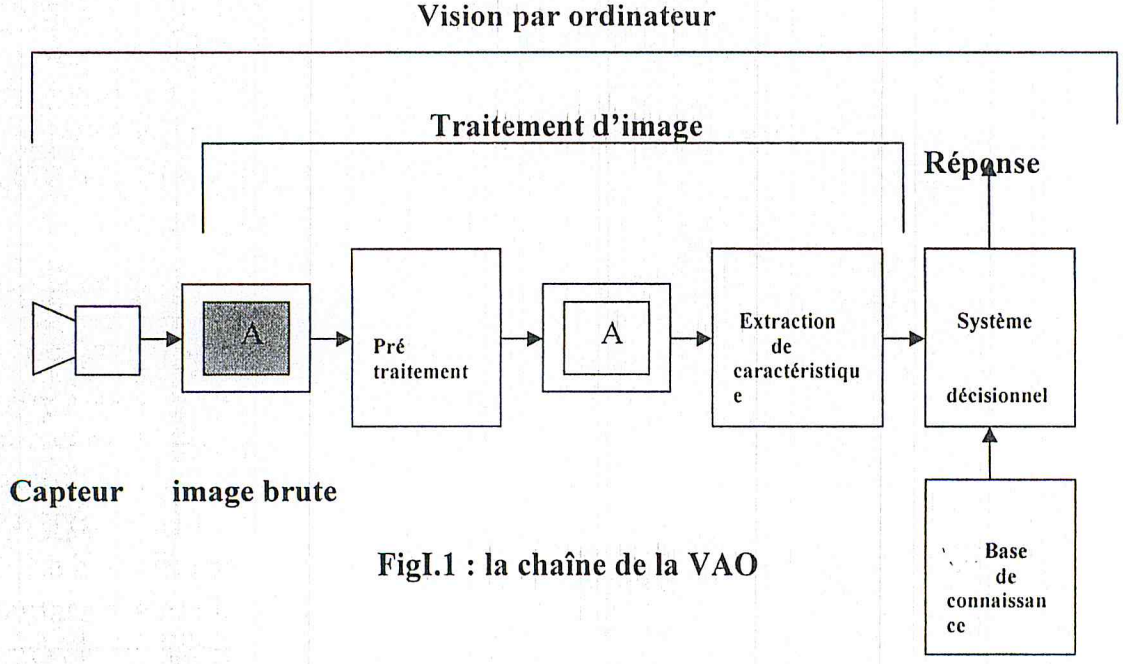
## I) Introduction :

La vision chez l'homme peut être définie comme une fonction qui permet de donner des représentations fiables du monde extérieur.
Le but de la vision par ordinateur est de produire une interprétation de l'image conforme à celle qui serait produite par l'homme, c'est une tache difficile, car l'ensemble des mécanismes qui régissent la vision chez l'homme n'est pas encore vraiment compris.

C'est un nouveau domaine, tirant aussi bien ses outils de l'informatique et de l'intelligence artificielle, que du traitement de signal, des mathématiques et de la géométrie ou de la physique [AUD00].

Le terme V.A.O (la vision assistée par ordinateur) symbolise toute la chaîne visuelle, partant de l'image brute jusqu'à interprétation de son contenu. Il existe donc un aspect décisionnel très important lors de la phase de compréhension, d'où l'interaction avec l'intelligence artificielle.

Le chercheur en vision par ordinateur conçoit et analyse les aspects calculatoires et algorithmiques, des processus d'acquisition, de traitement et d'interprétation des images numériques, son outil est la modélisation mathématique et algorithmique.




**FigI.1 : la chaîne de la VAO**

## II) La vision humaine :

La vision humaine est l'un des sens les plus importants chez l'homme. L'analyse et la compréhension de notre environnement se basent sur la vision.

Le système de la vision humaine possède les aptitudes suivantes:
1- appréciation des formes et des proportions des objets avec une grande précision.
2- détection des défauts d'alignement et de parallélisme.
3- conservation de la taille de l'objet lorsque celui-ci se rapproche ou s'éloigne et cela malgré les variations des dimensions sur les images rétiniennes.
4- invariance de la nature de l'objet lorsque celui-ci subit des homothéties [LAR89].

## III) La vision par ordinateur:

Le rêve d'avoir remplacé l'homme par un robot censé qui se déplace est allé vers les objets, saisir des objets .....etc.; dans des taches où sa présence peut se montrer difficile voir même dangereuse [HOR95].

La vision par ordinateur est une science qui tend à transformer ce rêve en réalité, car il faut le reconnaître, les résultas obtenus jusqu'aujourd'hui aussi importants puissent-ils être paraissent insuffisants en comparaisons avec ce que peut réaliser la vision humaine.

La vision est un processus qui traite l'information. L'entrée d'un système de vision est constituée par une séquence d'images. Le système apporte un certain nombre de connaissance qui interviennent à tous les niveaux et la sortie est une description de l'entrée en terme d'objets et de relation entre les objets.

La vision artificielle permet l'extraction automatique des informations à partir des images, en vue d'effectuer des taches spécifiques.

La vision par ordinateur trouve son application dans différents domaines :

-biomédical (détection de certaines zones dans le corps humain).
-spatial (l'étude et l'analyse des images satéllitaires).
-industriel (travail en milieu hostile et authentification d'objets).
-militaire (détection des cibles et guidage des missiles).

# IV) Système de vision artificielle :

## IV.1) Définition :

On peut définir un système de vision artificielle comme un ensemble de processus liés, opérant des transformations sur des informations acquises par des capteurs (séquences des images).

Ces transformations assurent le passage de l'image physique saisie à une description symbolique (en terme d'objet et des relations entre ces objets), en passant par la segmentation (contour, région ou coopérative) et par l'identification des attributs caractéristiques de la scène observée. Alors, l'objectif d'un système de vision n'est pas seulement d'effectuer des mesures mais d'interpréter automatiquement des scènes plus ou moins complexes, en utilisant des méthodes de l'intelligence artificielle.

Le principe consiste à extraire les caractéristiques primitives (contour et région) d'une ou plusieurs images bidimensionnelles, cette phase est dite « l'apprentissage » (la construction des modèles) .Ensuite ces caractéristiques seront comparées avec celles des différents modèles d'objets présentant dans la scène, cette phase est dite « la reconnaissance » (la comparaison des caractéristiques).

Il existe deux approches possibles pour un système de vision :

***-Ascendante :***
Les approches ascendantes tentent de construire à partir des images la représentation la plus abstraite possible.

***-Descendante :***
Les approches descendantes déduisent à partir des objets connus par le système une description compatible aux primitives extraites de l'image.

## IV.2) Le modèle de MARR pour un système de vision :

D.MARR a proposé une théorie globale des méthodes ascendantes au problème de la reconnaissance (l'approche reconstructioniste).
Il définit la vision comme étant un processus qui produit à partir d'image du monde extérieur une description utile à l'observateur dépouillée de toute information superflue [Mar82].

Le modèle de MARR se décompose en trois étapes successives de traitement :

- ***les traitements de bas niveau :***
  Opérant sur des grandeurs calculées sans compréhension véritable de l'image.
- ***les traitements de niveau intermédiaire :***
  Tentent de décrire les composants de la scène.
- ***les traitements de haut niveau :***
  Correspondent à l'interprétation et la compréhension de l'image.

## IV.3) Traitement informatique d'un système de vision :

### IV.3.1) Les traitements de bas niveau :

A partir d'une image en niveau de gris, les premiers traitements effectués permettent d'extraire des informations de base de type contour ou région qui structurent l'image, sans posséder de connaissances a priori de son contenu.

#### IV.3.1.1) Acquisition de l'image :

L'acquisition correspond à l'opération de conversion numérique du monde physique continu (scène analogique) vers un monde numérique discret (une matrice de valeur numérique manipulable et traitable par l'ordinateur).

Un capteur est un dispositif optique permet d'effectuer des acquisitions d'image.
Il est caractérisé par son rapport (signal sur bruit), sa sensibilité, sa résolution maximale, son temps d'intégration, son seuil de saturation et sa rémanence.
La dernière étape de l'acquisition est dite numérisation, cette opération consiste à donner un format numérique à un support (suite de 1 et de 0) et elle n'est souvent qu'une étape primitive à un processus de dématérialisation qui indique l'indexation pour retrouver l'image ainsi constitue. Puis finalement rematerialisation des données pour l'archivage.

Pour avoir une description du processus d'acquisition d'image, chaque capteur est caractérisé par un modèle géométrique : plusieurs types de modèles sont utilisés :

**a) Modèle sténopé :**

C'est le modèle le plus simplifié pour représenter un capteur [COU94].
Il est formé d'un plan image (P) et d'un centre optique (C), tous les rayons lumineux issus de l'objet observé dans l'espace (M) passent par le centre optique et se propagent en ligne droite (M).

L'axe optique est la perpendiculaire menée du centre optique sur le plan image, le point centrale (Pc) est l'intersection de l'axe optique (c'est la perpendiculaire menée du centre optique sur le plan image) avec le plan image, et la distance focale (F) est la distance du centre optique (C) au plan image (P).
La trace d'un point (M) dans l'espace est (M').

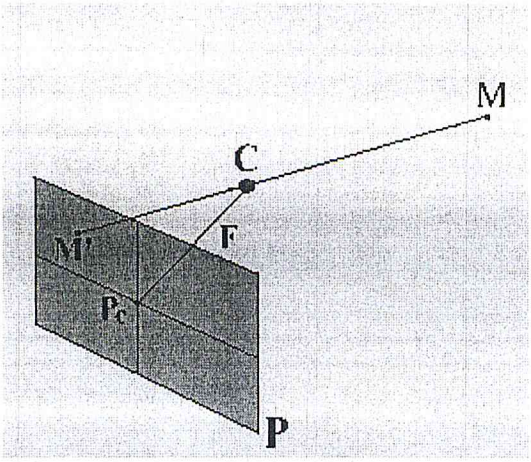



**Fig I.2 : Le modèle sténopé**

*C* : Centre optique
*P* : Plan image
*Pc* : Point central
*F* : Distance focale
*M* : Un point dans l'espace
*M'* : La projection de M sur le plan image

**b) Le modèle projectif parallèle :**

Ce modèle correspond au cas où le centre optique (C) est situé à l'infini (pas d'axe optique).
La projection d'un point (M) est orthogonale lorsque la trace (M') d'un point de l'espace sur le plan image (P) est la base perpendiculaire menée de ce point sur le plan image.
Ce modèle présente une bonne approximation du modèle sténopé si la taille de l'objet observé est faible par rapport à la distance d'observation.

## IV.3.1.2) Segmentation :

### a) Définition :

Elle consiste à séparer les divers éléments de l'image en région connexes ayant les mêmes propriétés, c'est à dire découper un plan (x, y) en région significatives.

Ces régions peuvent être caractérisées par frontières (contours), c'est le cas de la segmentation par extraction de contours, ou bien être caractérisées par les pixels qui les composent, il s'agit alors de la segmentation en région homogènes.

Il est évident que ces deux approches (contour, région) de la segmentation sont duales en ce sens qu'une région définit une ligne par son contour et qu'une ligne fermée définit une région.

Elles amènent à des algorithmes déférents et ne fournissent pas les mêmes résultats [KAB95].

***Définition d'une région :***

C'est un ensemble connexe de pixel ayant les mêmes propriétés (texture, intensité, .....)[COC95].

***Définition d'un contour :***

La notion de contour est associée à une variation d'intensité entre les propriétés de deux ensembles (régions) connexes de pixels.

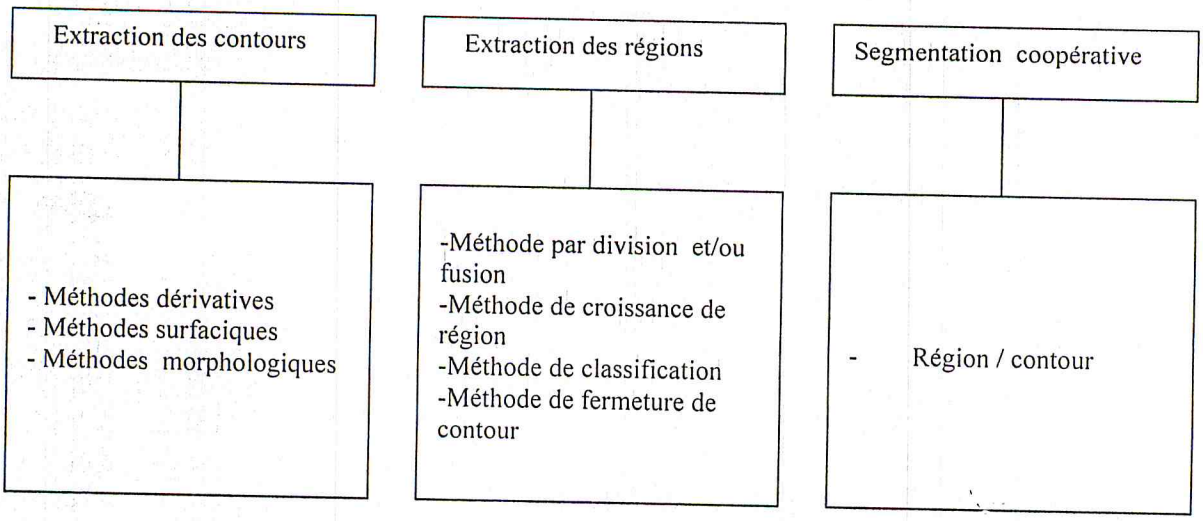
### b) Les techniques de la segmentation :

Il existe trois approches de la segmentation :

-Approche région.
-Approche contour.
-Approche coopérative.

Le choix d'une telle technique est lié en premier lieu à la nature d'image (image en niveau de gris, image couleur, image texture ou non) et en second lieu au domaine d'application.

On peut représenter les approches précédentes par l'organigramme suivant :




**FigI.3 : Organigramme de la segmentation d'une image**

## IV.3.2) Les traitements de niveau intermédiaire :

## IV .3.2.1) Systèmes de reconstruction 3D :

La reconstruction permet de trouver la troisième dimension d'une scène observée (l'information perdue lors de l'acquisition) en se basant sur des informations extraites de deux ou plusieurs images.

Il existe plusieurs méthodes pour reconstruire une scène 3D, on peut distinguer deux grandes classes qui sont respectivement la vision passive et la vision active.

### a) La vision active :

On dit qu'une vision est active lorsque les objets sont reconstruits à l'aide d'un éclairage contrôle illuminant la scène d'une manière particulière (ceci permet d'extraire des informations tridimensionnelles sans tenir compte des caractéristiques photométriques propres de la scène puisque celles-ci sont imposées par la source lumineuse).

Une extension de la vision active [BAL91], se base sur la constatation suivante, lorsqu'un observateur ne comprend pas la scène qu'il observe, il cherche à adapter (par exemple en se positionnant différemment), il acquiert ainsi de nouvelles vues de l'objet qui attire son attention et améliore donc sa compréhension de la scène tridimensionnelle.

Parmi les plusieurs techniques qui sont utilisées en vision active, nous citons l'utilisation des rayons lasers et des lumières structurées [ROU99].

**a.1) Lasers :**

Cette méthode consiste à utiliser une source laser, qui crée un plan lumineux qui va éclairer une tranche de la scène
En déplaçant la source (rotation ou translation), on obtient alors une reconstitution complète de la scène.
Cette technique nécessite un calibrage de précision pour la source bien que le capteur.
L'inconvénient principal de cette technique est que souvent on obtient une énorme quantité de données pour une petite surface observée.

**a.2) Lumière structurée :**

Cette méthode consiste à projeter sur la scène un motif structuré (une grille ou des franges d'interférence).
Après, en appariant la grille apparente (celle projetée sur la scène) avec la grille théorique (le motif structurant proprement dit) on obtient une certaine description du relief de la scène.

**b) Vision passive :**

La vision passive n'utilise aucune structuration particulière de la scène (contrairement à la vision active), les images de la scène sont les seules données disponibles.
Il s'agit alors d'extraire les caractéristiques (les attributs) de ses images pour en reconstruire la géométrie. Les plus utilisées sont les discontinuités photométriques qui réalisent l'extraction de points, de contours ou de régions.

**b.1) Photoclinométrie :**

Leur principe est d'extraire une forme unique à partir des variations de lumière observée sur la surface de l'objet.
L'hypothèse de base de la photoclinométrie est que l'intensité lumineuse perçue sur la surface d'un objet dépend de l'orientation de cette surface par rapport à la source lumineuse éclairant la scène.

**b.2) Lignes perspectives :**

A partir d'une image perspective, les techniques de reconstruction se basent sur les frontières des objets.
Ces méthodes consistent à inférer une forme tridimensionnelle à partir de tracés bidimensionnels qui représentent la projection perspective d'un objet.

La plupart de ces méthodes sont orientées vers la construction de facettes planes, d'objet à section conique ou éventuellement de surface réglée [CHA93].

## b.3) Stéréovision :

Leur principe est de reconstruire une forme tridimensionnelle à partir de deux ou plusieurs images de celle-ci, prises simultanément de points de vue différents.
Les étapes de cette technique sont :
- Détermination des primitives sur les images.
- Mise en correspondance (Appariement) des primitives en faisant intervenir des critères (Contraintes) relatifs à celle-ci.
- Détermination de la position dans l'espace de l'antécédent correspondant aux primitives appariées (Triangulation).

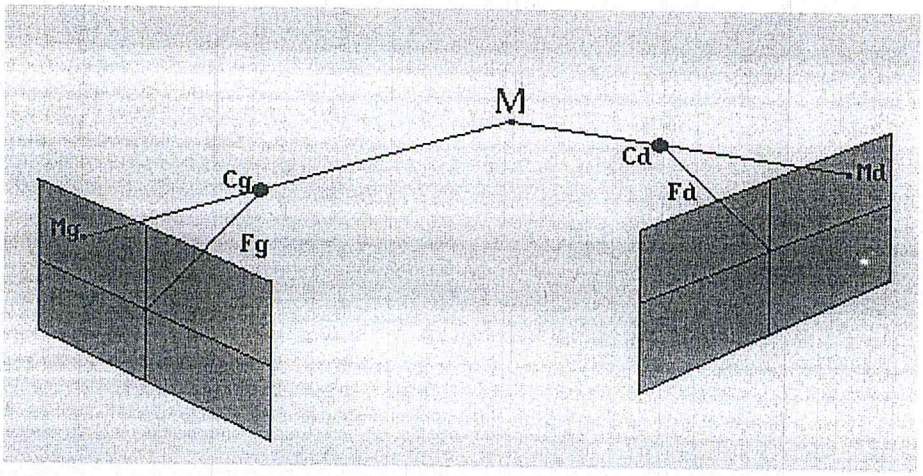



**Fig I.4 : Principe de la stéréovision**

$M$ : Un point dans l'espace.

$C_d$ (resp. $C_g$) : Centre optique droit (resp. gauche).

$M_d$ (resp. $M_g$) : Projection de $M$ sur le plan image droit (resp gauche).

$F_d$ (resp. $F_g$) : Distance focale du capteur droit (resp. gauche).

## IV.3.2.2) Appariement:

L'appariement de points entre deux images est le plus souvent basé sur le principe suivant : un point de la première image représente le même point physique qu'un autre point de la seconde image (on dit aussi que les deux points se correspondent) si les deux points se ressemblent. Cette ressemblance doit

prendre en compte les voisinages des points à cause du bruit dans les images, des occultations, du changement de point de vue de prise d'image…

En absence de toute connaissance a priori, la ressemblance entre deux portions de deux images d'une même scène est quantifiée grâce à la mesure de corrélation qui n'est autre qu'une mesure de ressemblance. Cette mesure de corrélation est calculée entre deux fenêtres, en général carrées et centrées sur les points à mettre en correspondance.

Donc l'appariement de deux images permet de décider quelles sont les primitives dans la paire d'image qui représentent une même primitive dans la scène, en respectant quelques contraintes d'appariement, qui sont utilisées pour limiter l'espace de recherche des correspondants, ou bien pour renforcer ou éliminer certaines correspondances déjà établies [ROU99].
Il existe plusieurs classes de contraintes, mais les plus utilisées celles fondées sur la géométrie.

Il est difficile d'appliquer une contrainte d'ordre géométrique sur un contour ou une région, pour cela ces critères ne peuvent s'appliquer que de manière ponctuelle ou sur des alignements de pixels [ROU99].

**a) Contrainte épipolaire :**

Elle consiste à restreindre l'espace de recherche du point homologue sur une droite appelée « Droite épopilaire » au lieu de le recherche dans toute l'image.
Cette ligne est en fait la projection dans l'autre image de la demi droite engendrée par le point de projection et le point initialement choisi (un point candidat à l'appariement dans des images) dans la première image.

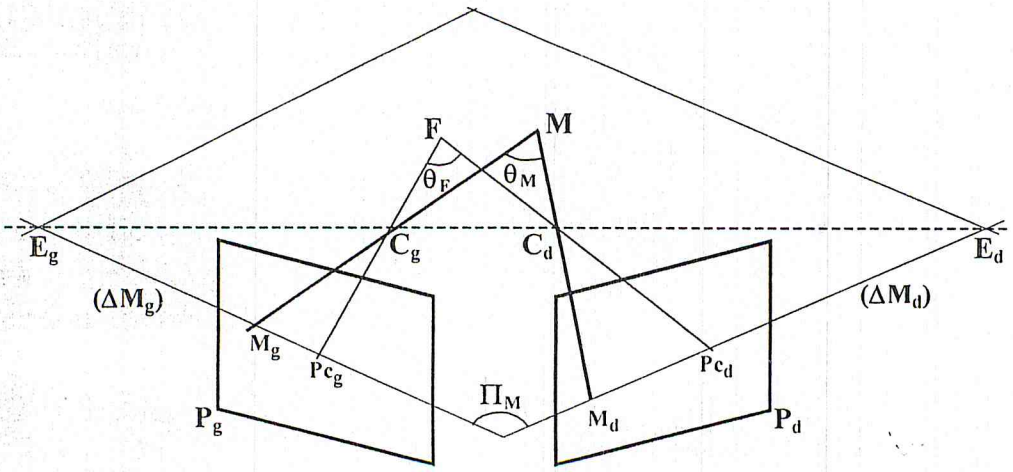



**Fig I.5 : Géométrie épipolaire**

$P_g$ (resp. $P_d$) : Plan image gauche (resp. droit).
$C_g$ (resp. $C_d$) : Centre optique du capteur de gauche (resp. droite).
*Pcg* (resp. $Pc_g$) : Point central du plan gauche (resp. droit).
$M$ : Un point dans l'espace.
$Mg$ (resp. $M_d$) : Projection de $M$ sur $P_g$ (resp. $P_d$).
$F$ : Point de fixation (L'intersection des axes optiques $(Pc_g, C_g)$ et $(Pc_d, C_d)$).
$\Pi_M$ : Plan épipolaire associé à $M$ (Passant par les points : $M$, $C_g$, $C_d$ et aussi $M_g$ et $M_d$).
$\Delta M_g$ (resp. $\Delta M_d$) : Droite épipolaire associé à $M_d$ (resp. $M_g$) : L'intersection des plans $(\Pi_M)$ et $(P_g)$ (resp. $(P_d)$).
$E_g$ (resp. $E_d$) : Epipole gauche (resp. droit) : L'intersection de la droite de base $(C_g, C_d)$ avec le plan $(P_g)$ (resp. $(P_d)$).

## b) Contrainte d'unicité :

Cette contrainte suppose qu'un point (dans une image) n'a qu'un seul correspondant dans l'autre image, mais cette contrainte est violée dans le cas des images transparentes.

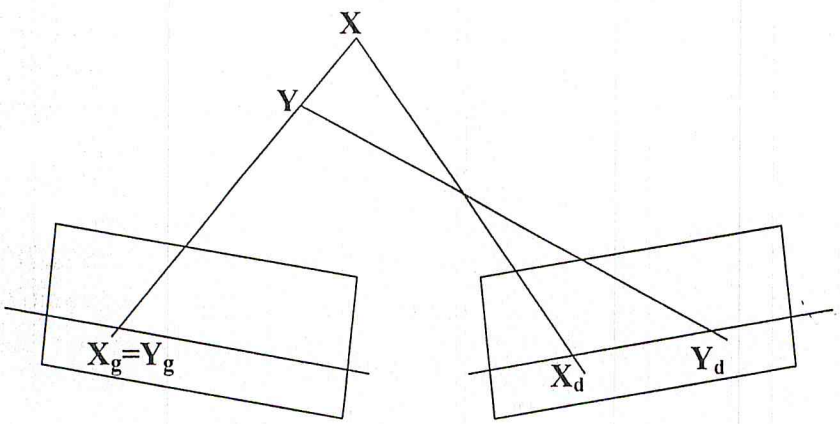



**Fig I.6 : Violation de la contrainte d'unicité**
**(Deux points distincts *X* et *Y* ont la même projection sur l'un des plans image)**

## c) Contrainte d'ordre :

La contrainte d'ordre implique que la projection des objets d'une scène conserve toujours le même ordre dans les deux projections images.
C'est à dire , si un point ( P ) se trouve à gauche d'un point ( Q ) dans l'image de gauche, alors la contrainte d'ordre exige que les points correspondant ( P') et (Q') dans l'image droite soient le même ordre.

**d) Contrainte de continuité :**

Cette contrainte permet à partir d'un appariement initial de prédire d'autres appariements qui viendront confirmer le premier [AIT95].

## IV.3.2.3) Reconstruction 3D :

Cette étape permet de déterminer la structure 3D de la scène à partir des mesures prises sur les images, cette opération s'appelle « l a triangulation ». Donc la triangulation consiste à déterminer le point M (le point physique dans l'espace de la scène) à partir de ses projections sur les plans images des capteurs, en calculant l'intersection des droites reliant chaque projection au centre optique du capteur associé.
Pour obtenir des mesures absolues, il faut déterminer les paramètres des capteurs avec précision, c'est le calibrage.

**a) Calibrage du capteur :**

Chaque modèle de capteur est caractérisé par un certain nombre de paramètres (intrinsèques et extrinsèques) qu'il faut estimer par l'opération d'étalonnage (calibrage).

- Les paramètres intrinsèques (internes) : sont les dimensions horizontales et verticales du pixel, la position du centre de l'image, la distance focale et l'angle entre les lignes et les colonnes du plan image.
- Les paramètres extrinsèques (externes) : sont la position du capteur par rapport à un repère tridimensionnel externe [TOS89].

Puisque les paramètres intrinsèques du capteur restent invariants (sauf la distance focale), alors on peut calibrer les paramètres intrinsèques du capteur à l'avance et calibrer seulement les paramètres extrinsèques et la distance focale durant l'application afin de réduire sa complexité et améliorer son efficacité [WAN91].

Pour éviter le calibrage, une autre opération (qui est très coûteuse en temps de calcul) consiste à utiliser des propriétés géométriques qui restent invariantes par projection.

**b) L'approche des Invariants projectifs :**

Quelques propriétés restent invariantes lors de l'acquisition d'une image ce qui permet la définition de : la connexité, l'alignement, l'adjacence et le birapport .....etc.

Cette approche consiste à utiliser ces propriétés invariantes [MOR93] pour apparier deux ou plusieurs images issues de capteur, sans passer par le calibrage.

La difficulté de calcul des invariants et leur manque de souplesse permettent d'utiliser une nouvelle variante « quasi-invariants » [GRO93]. Les quasi-invariantes ont la caractéristique d'absorber des changements de points de vue modérés [LAM98].

Les travaux développés par BINFORD et LEVITT [BIN93] montrent que certaines valeurs, telles l'angle formé par deux segments ou leur rapport de longueur utilisées par Gros [GRO95A], possèdent des propriétés invariantes lorsque le changement de point de vue entre deux images ne varie que faiblement.

### IV.3.3) Les traitements de haut niveau :

L'objectif de traitement de haut niveau est d'identifier et de localiser des objets qui se trouvent dans la scène observée.

A partir des résultats fournis par le niveau précédent (niveau intermédiaire), chaque objet inconnu est comparé aux modèles mémorisés dans une base de données (pendant la phase d'apprentissage), les groupements de primitives des objets à reconnaître sont mémorisés et constituent des modèles pour ces objets.

Il existe plusieurs techniques de la reconnaissance d'un objet :

#### IV.3.3.1) Reconnaissance par histogrammes :

#### a) Histogramme de couleurs :

SWAIN [SWA91] a démontré que l'histogramme de couleurs peut être utilisé pour l'indexation et la mise en correspondance des objets.
Donc cette technique consiste à représenter chaque objet par un histogramme de couleur.
Les objets sont identifiés par la comparaison de l'histogramme de couleurs d'une région de l'image à des histogrammes de couleurs d'objets pré-calculés, en utilisant des fonctions de comparaison des histogrammes [SCH97].

#### b) Histogramme multidimensionnel de champs réceptifs :

Schiele a utilisé des descripteurs locaux [SCH97], puisque le principe nécessite le calcul d'un vecteur de descripteurs invariants en tout point de l'image, alors une image modèle est représentée par un histogramme multidimensionnel de ces descripteurs.

Pour identifier une image inconnue à l'un des modèles connus, il faut calculer les mêmes types de descripteurs dans un ensemble quelconque de points. On obtient un sous histogramme (pour cette image) que l'on peut comparer aux histogrammes stockés afin de déterminer le modèle le plus semblable.

## IV.3.3.2) Techniques statistiques et probabilistes :

En vision par ordinateur des méthodes statistiques sont fréquemment utilisées : en segmentation d'image [CAN86], en appariement [BRE93] et pour la localisation et la reconnaissance d'objets. Ainsi les objets eux mêmes sont représentés par une densité probabiliste.

### a) Densité de probabilité :

Hornegger et Niemann [HOR95] représentent les objets par une densité probabiliste de caractéristiques, ils utilisent les coordonnés de points comme caractéristiques.
Chaque transformation d'un objet (rotation, translation, changement d'échelle,.....etc.) est représentée par un paramètre de la densité probabiliste.
Alors la reconnaissance passe par une estimation de paramètres par maximum de vraisemblance, à partir d'un modèle probabiliste a priori.

### b) Approche d'images propres :

Cette méthode a été utilisée pour l'identification des visages, mais elle est reprise et adaptée à la reconnaissance d'objets [MUR95].
L'image est représentée par une matrice de dimension n*m de valeurs de niveau de gris.
Une matrice de covariance (nm*mm) est calculée, elle représente la variation entre les différents modèles.

Les vecteurs propres de cette matrice forment une base dans laquelle l'espace défini par les modèles est exprimé (en utilisant l'analyse en composantes principales ACP), on peut réduire cette matrice en ne gardant que les vecteurs propres.
Alors une image est considérée comme un point dans l'espace de ces vecteurs propres, et la reconnaissance consiste à faire une recherche du plus proche voisin dans l'espace considéré.

La représentation d'un objet par un petit nombre de coefficients permet de les stockés et les recherchés facilement, c'est un grand avantage de cette approche, ça veut dire réaliser des indexations efficaces et des recherches rapides.

### IV.3.3.3) reconnaissance par graphes :

Cette méthode est proposée par Skordas [SKO88].elle est basée sur une approche de caractérisation provenant de la théorie des graphes.
Elle utilise des images segmentées en contours (comme entrée), approchés par des lignes polygonales. Les segments (présents dans une image) considérés comme les arcs ou les sommets d'un graphe.
Sossa [SOS92] a amélioré cette méthode en caractérisant le graphe par son polynôme de deuxième immanent.

Cette méthode est très robuste aux occultations des objets, grâce à l'utilisation des sous-graphes qui lui confèrent une certaine localité, mais très sensible aux bruits.

### IV.3.3.4) Approches fondées sur des descripteurs locaux :

Cette approche consiste à mettre en correspondance des configurations des images de l'objet inconnu avec des configurations des modèles d'objets déjà observés [LAM98], les configurations sont caractérisées par des descripteurs.
La comparaison des descripteurs d'une image et ceux d'un modèle permet de faire une mise en correspondance initiale et grossière. Ensuite il faut valider cette mise en correspondance par un simple décompte ou par un processus plus complexe.
Les descripteurs retenus (pour caractériser les objets) doivent être facilement comparables et la distance entre deux descripteurs doit être en rapport avec le degré de ressemblance entre les images qu'ils représentent.

#### a) Invariants locaux de luminance :

Schmid a proposé cette méthode d'indexation [SCH96], leur principe est d'extraire un ensemble de points d'intérêts des images modèles, qui sont à leur tour caractérisés par un vecteur d'invariants de luminance.
Elles sont invariantes à la rotation et la translation. La comparaison d'une image inconnue et les modèles à reconnaître passe par une indexation des descripteurs et un vote parmi les modèles.

#### b) reconnaissance fondée sur les invariants projectifs :

P.Gros [GRO93] [GRO98] introduit une méthode de reconnaissance d'objets en utilisant les quasi invariants projectifs.
Cette méthode consiste à dire que le mouvement apparent entre deux images qui représentent un même objet, peut être approché par une similitude quand la différence de point de vue n'est pas très grande [GRO95B].

La richesse des invariants calculés issus des images en niveau de gris, permet d'utiliser les méthodes d'appariements directement pour la reconnaissance.
Les modèles sont construits à partir d'une sélection d'images, on calcule les invariants de toutes les images des modèles, puis on les compare avec les invariants de l'image de l'objet inconnu, et enfin on choisit le modèle dont l'appariement avec l'image de l'objet paraît le plus vraisemblable.

## V) Conclusion :

Nous pouvons conclure que la vision artificielle constitue l'outil qui permet de rendre la machine plus compréhensible du monde qui l'entoure.

Alors cette machine doit être doté d'un ensemble de processus permettant le passage par plusieurs étapes, depuis l'acquisition des images jusqu'à l'interprétation finale de la scène, c'est à dire reconnaître l'objet observé grâce à une information a priori concernant cet objet.

Pour stocker cette information, il faut que la machine dispose d'une base de données.

La mise en correspondance entre un modèle stocké dans cette base et la description d'objet fournie par le niveau intermédiaire permet d'étiqueter l'objet observé.

Après cette tache une indexation doit être établie pour prendre des décisions (en temps optimal).

Cette indexation revient à indexer des vecteurs de données de grande dimension.

Dans le chapitre suivant, nous allons présenter quelques structures de l'indexation.

# *Chapitre II*
# *Structures d'indexation*

# *Chapitre II*

# *Structures d'indexation*

## I) Introduction :

Pour établir la reconnaissance, une approche très pratique est utilisée. Elle consiste à représenter une image par des vecteurs des descripteurs. Cependant, lorsque la recherche de similarité est faite dans des espaces de données de grande dimension, l'appréhension du domaine devient très difficile.

Il existe plusieurs structures d'indexations, chacune utilise des concepts différents pour indexer les données [MAT99].

## II) Les structures d'indexation :

Les structures d'indexation sont classées en deux grandes catégories:

- Celles qui regroupent les données en fonction de leur proximité (*data partitioning index methods*).
- Celles qui découpent a priori l'espace multidimensionnel et qui ensuite stockent les données selon ce découpage (*space partitioning index methods*) [AMS00].

### II-1) B-arbre (arbre balancé) :

Cette structure a été proposée en 1970 par R.Bayer [BAY71], elle est la plus utilisée dans les bases de données unidimensionnelles. Elle est très efficace dans le cas d'une nouvelle insertion ou d'une suppression, car elle évite d'effectuer des réorganisations fréquentes des données.

#### II-1-1) Principe :

Un B-arbre est un arbre balancé (équilibré) car tous les chemins possibles à partir d'une racine à une feuille quelconque sont de même longueur, chaque nœud peut avoir entre 0 et m enfants, et contenir entre 0 et (m-1) valeurs où m est le nombre de descendants directs d'un nœud interne.

Un B-arbre d'ordre m et de profondeur p est un arbre dont:

- La racine a au moins 2 descendants, sauf si c'est une feuille.
- Chaque nœud interne a au maximum *m* descendants.
- Chaque nœud interne a au minimum de $\lfloor m/2 \rfloor$ descendants (l'arrondi supérieur ou inférieur de $\lfloor m/2 \rfloor$ est utilisé).
- Chaque nœud contient *k* clés triées, avec $m-1 \le k \le 2m-1$ sauf la racine pour laquelle *k* vérifie $1 \le k \le 2m-1$.
- Pour tout nœud non feuille, le *i*-ème fils contient des valeurs comprises entre la i-ème et (*i*+1)-ème clé de ce nœud.
- L'arbre est équilibré, c'est à dire, toutes les feuilles apparaissent au même niveau : *p*, la profondeur de l'arbre.

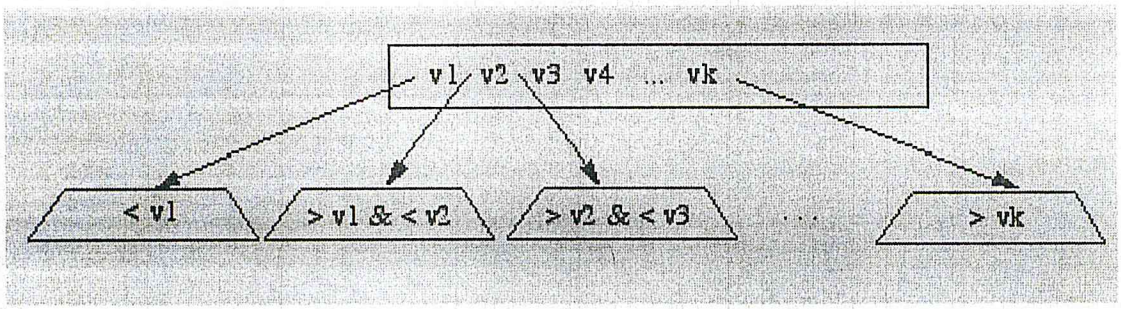



**Fig II.1 : Exemple d'un nœud d'un B-arbre**

## II-1-2) Recherche :

Pour rechercher un élément d'une clé donnée, tout d'abord, on compare sa clé avec la racine et on détermine le fils correspondant.
Alors s'il est un nœud, on continue sur le fils correspondant de même façon.
Sinon, si c'est une feuille un accès séquentiel aux éléments de cette feuille va déterminer si cet élément existe ou non, si on rencontre un pointeur vide (NIL) alors cet élément n'existe pas.

## II-1-3) Insertion :

La procédure d'insertion d'un nouvel élément dans un B-arbre est simple, on effectue les mêmes étapes que la recherche jusqu'à ce que l'on arrive à une page feuille où l'élément à insérer aurait pu se trouver s'il était présent dans la structure alors on essayera de l'insérer dans la feuille en question.

Plusieurs cas possibles peuvent être rencontré lors de l'insertion:
- si un élément doit être inséré dans une feuille contenant au moins un emplacement vide, l'insertion s'effectuera facilement à l'intérieur de cette feuille.
- si la page est pleine et qu'une feuille adjacente possède une place libre, alors on effectuera un transfert entre les deux feuilles (en tenant compte de l'élément charnière qui se trouve dans le nœud parente).
- l'insertion d'un nouvel élément dans une feuille pleine avec des pages adjacentes pleines, oblige d'allouer un espace pour une nouvelle feuille, qui sera attachée au nœud père.

Si la capacité de ce dernier est dépassée, on procèdera de la même façon avec ses nœuds frères, sinon on procèdera à un nouvel éclatement.

## II-2) Quad-tree :

Le Quad-tree est un arbre à précision infinie où chaque nœud peut avoir au maximum quatre fils, il est utilisé pour le stockage d'un ensemble de données bidimensionnelles [LAM98].

### II-2-1) Principe :

L'indexation par un Quad-tree revient à faire un découpage (cf. figure II.2) de l'espace de données bidimensionnelles en quatre rectangles, chacun de ces rectangles est découpé à son tour en quatre rectangles, ce qui veut dire que chaque niveau de l'arbre correspond à un découpage.

L'espace est découpé en plusieurs régions (rectangles) de tailles différentes, ce qui est pratique pour des données non uniformes si elles sont connues a priori.

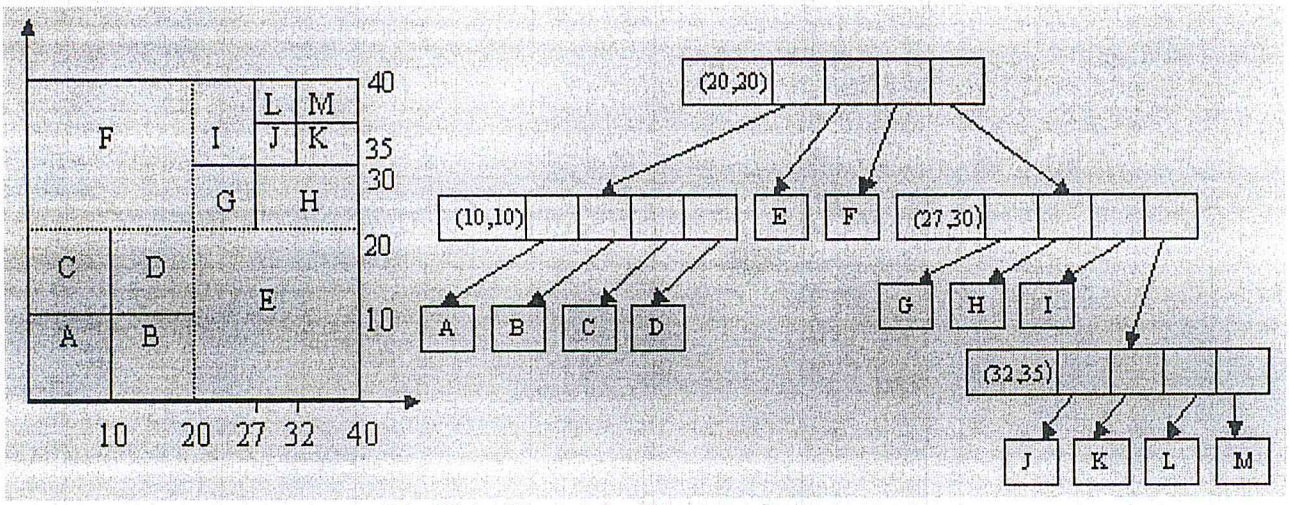



**Fig II.2: Exemple d'un quad -tree**

Dans un Quad-tree, chaque nœud contient un point de l'espace $(x_0,y_0)$ obtenu par l'intersection des deux droites qui découpent le rectangle associé à ce nœud. Il contient aussi quatre pointeurs vers ses fils où le premier fils est la racine du sous arbre dont les valeurs de tous les nœuds qu'il contient ont leurs abscisses inférieures à $x_0(x< x_0)$, et leurs ordonnées inférieures à $y_0(y< y_0)$. Le second contient les nœuds ayant $(x>= x_0)$ et $(y< y_0)$, le troisième contient les nœuds ayant $(x< x_0)$ et $(y>= y_0)$ et le dernier contient les nœuds ayant $(x>= x_0)$ et $(y>y_0)$.

L'avantage de cette structure s'il s'agit d'une recherche exacte, est que les points stockés dans les nœuds (qui sont utilisés pour le découpage) peuvent appartenir à l'ensemble des données, ce qui peut diminuer le temps de réponse d'une requête, mais s'il s'agit d'une recherche de vecteurs similaires, cet avantage ne compte pas et si les données sont stockées indépendamment de l'index, ceci donne une plus grande souplesse pour les opérations d'insertion et de suppression.

## II-2-2) Recherche de points :

### II-2-2-1) Recherche d'un point précis :

Pour rechercher un point précis dans un Quad-tree, on suit le parcours suivant: Comparer ses coordonnées avec celles de la racine, après avoir déterminer le fils correspondant au rectangle qui contient la requête, on compare de nouveau les coordonnées requête avec celles du nœud fils, et ainsi de suite jusqu'à la localisation de la requête. Si on rencontre un pointeur vide alors le point requête n'appartient pas à la base.

Si on dispose d'un ensemble de données contenu dans $4^n$ feuilles (réparties uniformément), on n'aura besoin que de (n+2) accès pour la requête.
Pour cela, on peut dire que la recherche d'un point précis dans un Quad-tree est très simple et efficace.

### II-2-2-2) Recherche de points similaires :

La recherche de points similaires à un point donné, où les points se trouvent dans un rectangle dont le centre est ce point consiste à déterminer le nœud correspondant au plus petit rectangle englobant la zone de recherche.

Si ce nœud pointe sur une feuille alors on détermine les points contenus dans la zone requête [MAT99].

Sinon, pour chacun de ses fils dont le rectangle correspondant se chevauche avec la zone requête, et on lance récursivement cette procédure de recherche.

### II-2-3) Insertion :

L'opération de l'insertion dans un Quad-tree est difficile, car on peut rencontrer des cas spéciaux. L'insertion d'un nouveau point revient à faire rechercher le nœud dont le rectangle correspondant est le plus petit rectangle contenant ce point.
Il existe deux cas possibles:

- si ce nœud pointe sur une feuille déjà existante, alors le point y est inséré, et si elle est pleine, on divise le rectangle auquel elle correspond en quatre nouveaux rectangles, ce qui est connu par l'éclatement, cette feuille va alors être remplacée par un nœud pointant sur quatre nouvelles feuilles et ses vecteurs seront insérés dans ces feuilles.
- si ce nœud ne pointe pas sur une feuille correspondant à ce rectangle, alors une nouvelle feuille est allouée et le point y est inséré.

## II-3) R-tree:

Le R-tree est la première méthode d'indexation multidimensionnelle basée sur les B-arbre, elle est proposée par A.GUTTMAN [GUT84].
Le R-tree permet de consulter des données qui peuvent être représentées sous forme de points ou de régions dans un espace de grande dimension.

### II-3-1) Principe :

Cette méthode est basée sur l'approximation des objets complexes multidimensionnels par des RMEs (Rectangle Minimum Englobant les données).

L'information est une donnée caractérisée par un sous-espace, celui-ci étant représenté par un ensemble de points du système. La sélection d'une information correspond à trouver le RME englobant l'information dans les n dimensions ; dans ce cas le RME sera un hyperrectangle à n dimensions (cf. figure II.3).

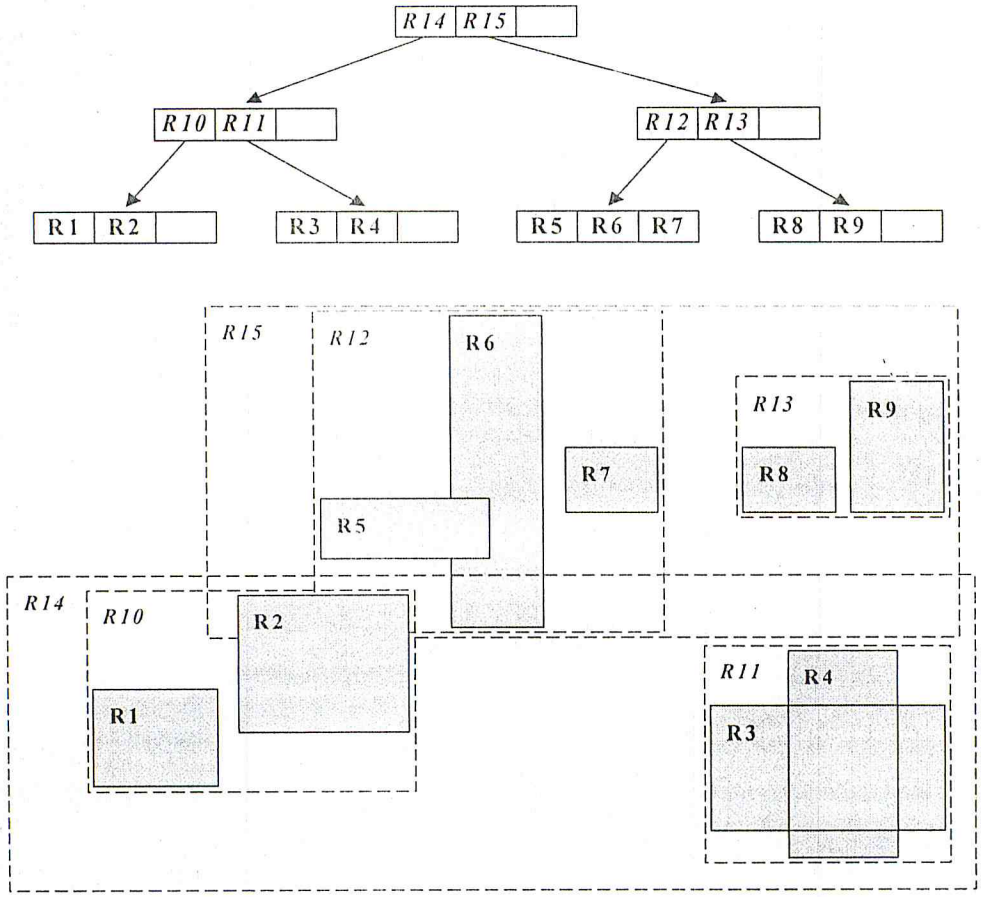



**Fig II.3: Exemple d'un R-tree**

Dans un R-tree, les objets ne sont contenus que dans les feuillès, les nœuds internes ne servant qu'au parcours de l'arbre. Les feuilles ont plusieurs entrées contenant un RME et un pointeur vers l'objet qu'il englobe. La racine et les nœuds internes ont plusieurs entrées chacune contenant un RME et un pointeur vers le fils correspondant à ce RME, qui est le plus petit rectangle englobant tous les objets décrits par ce fils.

Un R-tree a deux paramètres, m et M qui sont respectivement le nombre minimum et le nombre maximum d'entrées dans les nœuds et les feuilles avec $2 \leq m \leq M/2$.

Les nœuds internes et les feuilles possèdent entre m et M entrées, et la racine contient entre 2 et M fils.

Le principe de l'indexation de points multidimensionnels consiste à les grouper, chaque groupe sera englobé par un RME auquel sera associé une valeur permettant de définir sa position et ses dimensions, l'indexation se fera grâce à ces clés. Ces RMEs sont groupés pour former leur RMEs pères jusqu'à obtenir un seul RME correspondant à la racine.

## II-3-2) Recherche :

La procédure de recherche dans un R-tree est la même que les B-arbres sauf que pour les R-tree, on compare les coordonnées des rectangles, pour voir s'il existe une intersection ou non,

La recherche sera optimale lorsque le chevauchement est minimum, évitant ainsi de décomposer la recherche en plusieurs sous-recherches.

Si le rectangle de recherche est chevauché par plusieurs RMEs, alors on ne peut pas se prononcer sur la recherche, puisqu'il faudra la continuer sur les fils correspondants. En arrivant au nœud associé au plus petit RME qui contient l'intégralité du rectangle de recherche, et pour chaque entrée de ce nœud, on peut citer deux cas possibles:

- si une entrée est une feuille alors on examine chacune de ses entrées pour déterminer si son RME chevauche le RME requête, alors envoyer ce RME comme une solution de la recherche sur toute cette entrée.
- si une entrée n'est pas une feuille, on examine chacune de ses entrées pour déterminer si son RME chevauche le RME requête, alors on continue la recherche sur chaque sous-arbre pointe sur ces entrées.

## II-3-3) Insertion :

L'algorithme d'insertion doit déterminer la feuille hôte, car l'insertion d'un objet dans un R-tree se faite au niveau des feuilles, en utilisant l'algorithme de recherche précèdent, Plusieurs cas peuvent se présenter :

- si l'objet à insérer appartient à une seule feuille, alors on l'insère dans cette feuille.
- si le contraire (cet objet appartient aux plusieurs feuilles), alors on l'insère dans une feuille de plus petite surface.
- si l'objet à insérer n'appartient à aucune feuille, alors on l'insère dans la feuille qui aura subi après agrandissement (de façon à l'englober) la plus petite augmentation du surface (ceci entraîner la redéfinition des tailles des RMEs pères de cette feuille).

## II.3.4) Les variantes de R-tree:

Le R-tree a été repris dans de nombreuses variantes telles le packed-tree, le $R^+$-tree, le $R^*$-tree, et la plus récentes: le X-tree.

### II.3.4.1) Packed R-tree(1985):

Cette structure est développée par N.Roussopoulos et Leifker D (Université de Maryland). Son principe consiste à minimiser l'espace non utilisé de la structure d'index ce qui permet la réduction de la mémoire nécessaire, d'où le terme de Packed R-tree (arbre comprimé).
Cette structure est utilisée pour les bases de données relativement statiques c'est à dire qui sont très peu modifiées telles que les données géométriques d'une carte.

### II.3.4.2) $R^+$-tree(1987):

Le $R^+$-tree est proposé par N.Roussopoulos, T.sellis, C.Faloutsos (Université de Maryland). Il consiste à éviter le chevauchement des régions en splittant si nécessaire les régions de niveau inférieur, donc il réduit efficacement le nombre de parcours lors de la recherche.

### II.3.4.3) $R^*$-tree(1990):

Le $R^*$-tree est proposé par Beckmann, N.Kriegel, Schneider R (unversité de Bremen). Il est apparu après de nombreuses études et expériences (utilisation des différentes stratégies dans les algorithmes d'insertion et de split).

### II.3.4.4) X-tree(1996):

La structure d'indexation qui supporte la recherche de similarité dans un espace de données multidimensionnelles est appelée le X-tree (eXtented tree), c'est la variante la plus récente du R-tree [GUT84].

Le chevauchement entre les zones contenant les données (RMEs) est le problème majeur qui apparaît dans les bases de données structurées par le R-tree, et les RMEs à différents niveaux de l'arbre, ce qui indique une détérioration des performances avec l'augmentation de la dimension (cf. figure II.4).

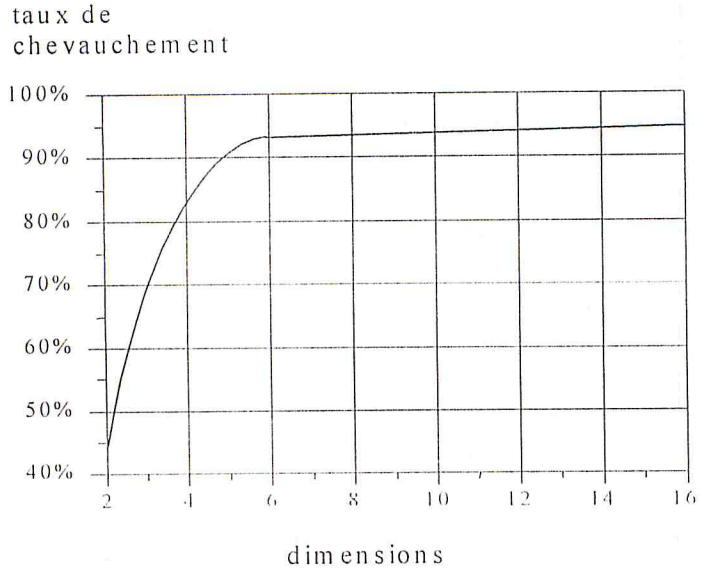



**Fig II.4 : Evolution du taux de chevauchement en fonction de la dimension des données [BER96]**

## II.3.4.4.1) Principe :

Le concept d'un X-tree (cf. figure II.5) est le même d'un R-tree, la seule différence est que le X-tree peut éviter les chevauchements grâce à l'utilisation d'une structure hétérogène. Les nœuds et les feuilles du X-tree correspondent à des RMEs (comme dans les R-tree).
Le X-tree comporte les super-nœuds qui sont des nœuds de taille variable (Le multiple de la taille d'un nœud normale).

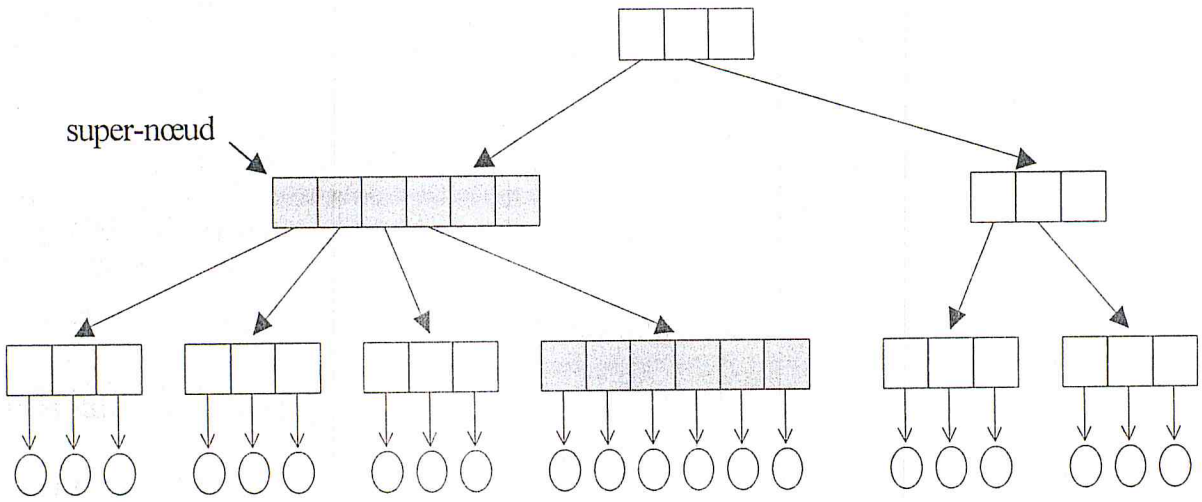



**Fig II.5: Exemple d'un X-tree**

L'utilité des super-nœuds est d'avoir un seul super-nœud au lieu d'avoir deux ou plusieurs nœuds dont les RMEs se chevauchent, ce super-nœud pointe sur tous les fils de ces nœuds.

**Remarque:**

Dans un X-tree, la recherche s'effectue de façon identique que dans le R-tree, pour cela, on s'intéresse à l'insertion.

#### II.3.4.4.2) Insertion :

L'insertion dans un X-tree s'effectue de même façon que dans un R-tree, sauf dans le cas d'un éclatement.

L'insertion dans un X-tree consiste à éviter les éclatements qui génèrent des taux de chevauchements très élevés (comme l'exemple la figure II.5).

Pour l'insertion, on commence par déterminer le RME hôte qui va abriter l'objet à insérer, l'objet est inséré dans cette feuille si aucun éclatement ne peut être provoqué par l'insertion, et si un éclatement s'effectue à un nœud de niveau supérieur, alors ce nœud est éclaté de telle manière à avoir des RMEs avec des propriétés optimales (taille minimale, chevauchement réduit……). Si cet éclatement provoque un taux de chevauchement très élevé, alors ce nœud doit être remplacé par un super-nœud qui va contenir tous les RMEs de ce nœud, en plus du nouveau RME crée par l'insertion.

## II.4) Kd-tree:

Le Kd-tree est une structure d'indexation simple de sa mise en œuvre pour indexer des espaces multidimensionnels, cette structure est un arbre binaire.

### II.4.1) Principe :

Dans un Kd-tree (cf. figure II.6), l'indexation des données consiste à découper chaque niveau de l'arbre suivant une dimension donnée et chaque dimension est partagée en plusieurs intervalles [GRO98].

La profondeur de l'arbre sera égale à la dimension de l'espace, pour cela le choix de l'ordre des dimensions est très important, car il peut influencer sur le temps de réponse dans le cas de la recherche des voisins.

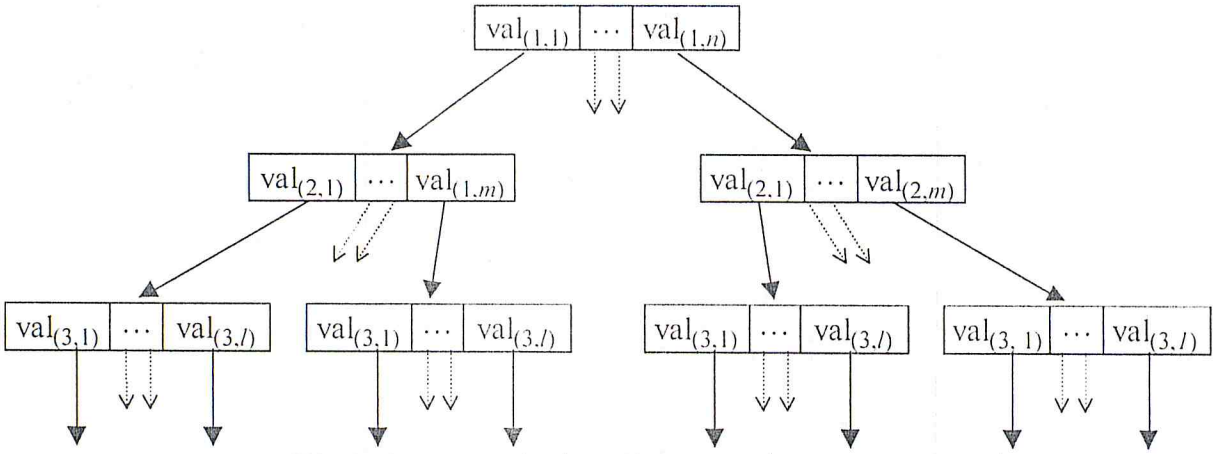



**Fig II.6 : Exemple d'un Kd-tree à 3 dimensions**

Le nombre des nœuds dans un Kd-tree accroît exponentiellement, si on prend par exemple un Kd-tree complet de dimension n dont toutes les branches sont développées et si on suppose que chacune des n dimensions est découpée en k intervalles, alors le nombre des nœuds est: $K^n-1/K-1$

## II.4.2) Recherche :

La recherche s'effectue à partir de la racine. On affecte une comparaison entre les valeurs du nœud et la composante du vecteur requête associée au niveau de ce nœud, et on détermine le nœud prochain du niveau suivant et ainsi de suite.

L'opération de recherche devient moins efficace lorsque la recherche de vecteurs se trouve à une distance donnée du vecteur requête car elle peut s'étendre facilement à la majeure partie des nœuds de l'arbre.
A chaque niveau de l'arbre, la zone de recherche est projetée sur la dimension associée à ce niveau , et la recherche est étendue vers les fils de ce nœud qui sont touchés par la projection .

## II.4.3) Insertion :

L'objectif principal de l'insertion dans le Kd-tree est d'économiser l'espace mémoire, en limitant le nombre de nœuds, c'est-à-dire, en ne développant que les branches qu'elles contiennent au moins un vecteur.

L'insertion consiste à rechercher la feuille qui va contenir l'objet à insérer, si elle est localisée, alors on insère cet objet dans cette feuille, sinon, si l'arbre n'est pas entièrement développé, alors on crée des nouveaux nœuds à chacun des niveaux restants pour aboutir à la feuille qui va contenir l'objet à insérer.

Dans un Kd-tree, pas d'éclatement, Pour cela les feuilles doivent avoir une structure qui leur permet de pointer vers d'autres feuilles (si elles sont pleines).

Dans un Kd-tree, en cas de recherche de vecteurs similaires, une amélioration peut être provoquée. Son principe est d'ajouter à chaque feuille des pointeurs vers ses feuilles voisines, alors, la recherche va commencer par déterminer la feuille qui contient le vecteur requête, puis à partir de cette feuille, on passe vers les feuilles voisines qui peuvent contenir les vecteurs proches de la requête sans remonter aux nœuds père ou provoquer des sous-recherches.
Mais l'inconvénient de cette amélioration est que dans un espace à n dimension, le nombre exorbitant de feuilles voisines est $(3^n-1)$.

## II.5) M-tree :

Le M-tree est une structure d'indexation qui permet la recherche des similarités dans des espaces de grande dimension dotées d'une métrique [ZEZ96][CIA97].

### II.5.1) Principe :

Le M-tree consiste à partitionner les vecteurs de la base selon leurs distances relatives calculées grâce à la métrique de l'espace utilisé.
Les vecteurs sont stockés dans un fichier à part et pas dans le M-tree, ce qui permet d'insérer, rechercher et supprimer les vecteurs sans contrainte d'ordre.
Seuls les informations liées au calcul des distances sont stockées dans les nœuds.

Le principe de cette structure est de grouper les vecteurs dans des sous ensembles (hypersphères). Comme dans les R-tree, les petites hypersphères seront groupées dans des hypersphères plus grandes et ainsi de suite jusqu'à la racine.

Chaque nœud contient un pointeur vers son père (sauf la racine) et des pointeurs vers des racines de sous-arbres. Tous les fils d'un nœud sont à une distance inférieure de la distance de recouvrement de ce nœud.
Les feuilles aussi contiennent un pointeur vers le nœud père, et ainsi que l'identificateur de l'objet.

### II.5.2) Recherche :

La recherche dans un M-tree commence par le nœud racine et consulte toutes les branches qui ne peuvent pas être exclues.
On détermine pour chaque nœud parmi ses fils ceux dont l'hypersphère associée s'intersecte avec la zone de recherche et pour chacun de ces fils la procédure relance récursivement jusqu'à atteindre les feuilles.

### II.5.3) Insertion :

Dans le M-tree, pour localiser la feuille la plus adéquate à contenir l'objet à insérer, la procédure d'insertion descend récursivement.
Le choix de la feuille revient à choisir à chaque niveau les nœuds dont l'hypersphère associée couvre le nouvel objet.
Si plusieurs nœuds vérifient cette propriété, on prend celui dont le centre de l'hypersphère associée est le plus proche à l'objet à insérer.
Si aucun nœud ne peut contenir le nouvel objet, on prend celui dont l'hypersphère correspond va présenter après l'insertion la plus petite augmentation dans son rayon.

Le M-tree est une structure d'indexation plus efficace que d'autres (par exemple le R-tree), elle donne de bons résultats, mais lorsque la dimension de l'espace augmente, elle dégénère rapidement et le coût du calcul des distances reste très élevé.

## III) Conclusion :

Nous avons présenté dans ce chapitre plusieurs structures d'indexations, quelques unes donnent de bons résultats par rapport aux autres ; mais malgré ça, elles restent loin des objectifs espérés, car elles (sauf le Quad-tree) voient leurs performances dégradées avec l'augmentation de la dimension.

Les structures d'indexation que nous avons présenté dans ce chapitre, vont servir comme des outils de base efficaces pour certaines techniques d'indexation dans le domaine de la reconnaissance d'objet.

Dans le chapitre suivant, nous allons exposer les différentes techniques d'indexation.

# Chapitre III
# Techniques d'indexation

# *Chapitre III*

# *Techniques d'indexation*

## I) Introduction :

La façon la plus simple pour rechercher l'image la plus ressemblante à une image donnée dans un ensemble d'images, revient à apparier cette image avec toutes celles de l'ensemble, qui veut dire que le problème de l'indexation est une suite naturelle à celui d'appariement.
Mais le problème est de réaliser cette opération d'une manière plus rapide, ce qui nécessite de la faire d'une manière parallèle.
Pour résoudre ce problème, plusieurs recherches sont faites, chacune propose une technique d'indexation, qui est généralement basée soit sur une description géométrique des images ( donnant lieu aux techniques de hachage géométrique[LAM88]) , soit sur une description structurelle [SOS92].

L'interrogation de base d'images par le contenu a été rendue possible par le mariage de techniques de base de données et de traitement d'images.
Puisque la grande variété des classes d'objets à reconnaître implique une grande variété dans les modèles de représentation possibles, d'où la nécessité d'un stockage dans une mémoire secondaire, et l'utilisation des compétences en bases de données (BD) pour la recherche d'information.

Les SGBD actuels sont orientés vers l'indexation et la recherche de données exactes, mais en cas des vecteurs imprécis de grande dimension, ils ne sont pas mieux adaptés. Ce qui nécessite de la mise au point de méthodes d'indexation spécifiques à la reconnaissance.

## II) Les techniques d'indexation :

Plusieurs techniques d'indexation sont utilisées en base de données pour la reconnaissance d'objets.

Elles structurent les données en index pour accélérer les recherches de similarités entre l'image inconnue (l'image observée) et les modèles de la base.

Dans ce chapitre, on s'intéresse aux techniques d'indexation fondées sur l'utilisation des caractéristiques géométriques des images.

Il existe une autre classe de méthodes d'indexation qui utilise les caractéristiques photométriques des images qu'elles traitent. Ces descripteurs peuvent être globaux (un seul descripteur pour toute l'image) ou locaux (décrivant la variation du signal sur des régions ou bien autour de point d'intérêt).

L'utilisation directe du signal des images en niveaux de gris permet de développer des systèmes puissants d'appariement, d'indexation et de reconnaissance d'images [GRO97].

D'autres techniques tentent d'utiliser des images en couleur au lieu de celles en niveaux de gris. Et pour caractériser l'image en fonction de la couleur, il faut utiliser des descripteurs invariants aux changements d'illumination (qu'ils soient en intensité ou en couleur). Pour cela, certains travaux proposent de normaliser les images, ce qui consiste en les transformer de telle sorte que le résultat de cette transformation soit indépendant des paramètres de l'illumination [SCH96]. La plupart de ces méthodes d'indexation (qui utilisent la couleur) sont basées sur une approche globale de l'image. C'est à dire l'image entière est décrite par un seul vecteur. Et on peut classer ces méthodes en trois catégories: la comparaison d'histogramme, le regroupement, et les moment statiques.

Mais d'autres travaux récents proposent d'adapter ces méthodes pour des approches plus locales.

Dans les sections suivantes de ce chapitre, nous présentons les techniques les plus importantes d'indexation qui sont basées sur les descripteurs d'ordre géométrique.

## II.1) Le hachage:

### II.1.1) Le hachage géométrique :

C'est une technique de stockage et de recherche de l'information dans une table de hachage en utilisant les indexes entiers issus d'une fonction de hachage à partir des clés définies par les invariants.

Le hachage consiste à calculer l'adresse entière (nommée index), de l'emplacement d'un élément dans la table de hachage à partir de sa clé, définie par une fonction de hachage.

[CAR97] a démontré que la recherche est plus optimale quand la fonction de hachage tend vers la fonction uniforme.

Une application du hachage géométrique a été proposée par LAMDAN et WOLFSON [LAM88] dans le domaine de la reconnaissance ; où les objets sont modélisés par une représentation des ensembles de points d'intérêt 2D, invariants à une transformation affine.

L'étape de l'apprentissage (pré-traitement) consiste à extraire les coordonnés de *N* points d'intérêt (un point d'intérêt correspond à un changement bidimensionnel du signal. Des exemples sont les coins et les jonctions en T, et aussi les endroits où la texture varie fortement). Un triplet de points non colinéaires est choisi pour former une base affine du plan. Les coordonnées des *N*-3 autres points d'intérêt sont considérées comme invariantes. Dans un second temps, trois autres points non colinéaires sont pris pour former une nouvelle base, de façon à définir les descriptions invariantes de *N*-3 points restant ; et ainsi de suite.

Ensuite, la méthode génère toutes les combinaisons possibles de trois points, parmi *N*. Ces combinaisons sont ensuite caractérisées par leurs coordonnées exprimées dans les repères affines. Ces coordonnées constituent les clés d'indexation. Une fonction de hachage à deux dimensions est mise en place, recevant pour tout point caractérisé une référence au modèle auquel il appartient, ainsi que l'indication du repère affine. On peut se contenter d'utiliser un seul triplet, mais pour réduire la complexité de la reconnaissance, toutes les

combinaisons de trois points non colinéaires sont retenues pour former des bases affines du plan.

Pour la reconnaissance d'objets, une table de vote est établie, et le même opérateur de points d'intérêt est appliqué pour l'extraction des points d'intérêt d'une image test. Un triplet arbitraire, non colinéaire, est employé comme base affine et les coordonnées des autres points sont utilisées pour voter pour les objets (ou plus précisément pour un triplet particulier et son modèle d'objet). Pour chaque caractérisation, la base d'indexation est consultée et un modèle voit son compteur incrémenté chaque fois qu'une de ses références est trouvée dans la table. Si aucun objet n'a obtenu un nombre suffisant de vote après une étape, il est possible de choisir un autre triplet non colinéaire pour une autre étape de vote. Si un modèle d'objet a accumulé un nombre suffisant de vote, il peut être vérifié par une mise en correspondance de l'image test et du modèle d'objet.

### II.1.2) Hachage géométrique par les invariants affines décrivant les morceaux des contours :

Elle est proposée par [MAT99], et basée sur le hachage géométrique .Elle consiste à utiliser la description des contours par les invariants locaux $I_1$, $I_2$ comme clés d'indexation.

Pour un contour fermé, composé de $N_i$ points caractéristiques est représenté par $N_i$ couples d'invariants $\left(I_{1_{(n)}}, I_{2_{(n)}}\right)_{n=1\ldots N_i}$. Ces invariants sont donnés par les relations suivantes :

$$I_1 = \frac{\mathrm{Det}(P_1 - P_0, P_3 - P_0)}{\mathrm{Det}(P_1 - P_0, P_2 - P_0)} \quad I_2 = \frac{\mathrm{Det}(P_2 - P_0, P_3 - P_0)}{\mathrm{Det}(P_1 - P_0, P_2 - P_0)}$$

Avec $P_0$, $P_1$, $P_2$, $P_3$ : sont quatre point successifs d'une courbe.

#### II.1.2.1) Pré-traitement :

Le principe de cette étape est de calculer pour chaque modèle $m_i$ de la base d'objets, les invariants $\left(I^{m_i}_{1(n)}, I^{m_i}_{2(n)}\right)_{n=1\ldots N_i}$. Par l'utilisation d'une fonction de

hachage, ces invariants nous fournissent les indexes $\left(i_{(n)}^{m_i}, j_{(n)}^{m_i}\right)_{n=1...N_i}$ indiquant les entrées de la table de hachage $T$. A l'endroit indiqué par les indexes $\left(i_{(n)}^{m_i}, j_{(n)}^{m_i}\right)_{n=1...N_i}$ dans la table $T$, on stocke sous la forme d'une liste chaînée, le modèle $m_i$ (représenté par son numéro) et les invariants $\left(I_{1(n)}^{m_i}, I_{2(n)}^{m_i}\right)$ correspondants. Le triplet $\left(m_i, \left(I_{1(n)}^{m_i}, I_{2(n)}^{m_i}\right)\right)$ constituant un élément de la liste chaînée.

Tous les modèles de la base, traités de cette façon, construisent la table de hachage, autrement dit la base d'indexation.

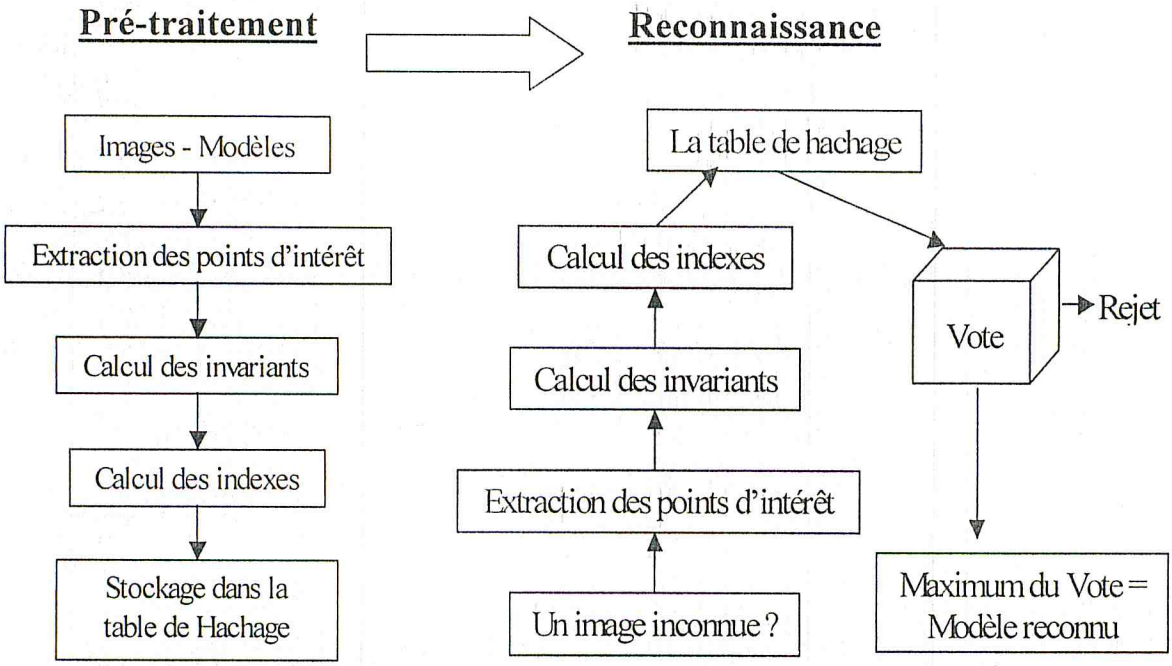



**Fig III.1 : Diagramme résumant les deux étapes de la méthode**

## II.1.2.2) Reconnaissance :

A partir d'une image requête $R$ (inconnu), il faut calculer les $N_R$ couples d'invariants $\left(I_{1(n)}^{R}, I_{2(n)}^{R}\right)_{n=1...N_R}$ et les $N_R$ indexes $\left(i_{(n)}^{R}, j_{(n)}^{R}\right)_{n=1...N_R}$ correspondants à cette image, afin de trouver le modèle d'objet qu'il correspond à elle.

Pour chaque couple d'invariants $\left(I^R_{1(n)}, I^R_{2(n)}\right)$ de la requête, nous accédons à la case indiquée par les indexes $\left(i^R_{(n)}, j^R_{(n)}\right)$ dans la table de hachage, en récoltant les informations concernant les modèles $m_i$ et les invariants correspondants $\left(I^{m_i}_{1(n)}, I^{m_i}_{2(n)}\right)$ se trouvant dans cette case.

Dans le cas où plusieurs triplets sont présents dans cette case, nous pouvons filtrer les réponses en calculant la distance entre les invariants $\left(I^R_{1(n)}, I^R_{2(n)}\right)$ de la requête et les invariants $\left(I^{m_i}_{1(n)}, I^{m_i}_{2(n)}\right)$ de la base, trouvés dans la case. Si cette distance est inférieure à un seuil, le compteur VOTE ($m_i$) est incrémenté (représenté par une table unidimensionnelle), sinon le modèle est rejeté. A la fin de cette procédure, le compteur VOTE contient le résultat pour les modèles qui ont été trouvés. Le plus grand nombre d'occurrences de vote indique les modèles de la base, semblable à la requête.

L'étape de vote peut être réalisée par un simple compteur, si la fonction de hachage est uniforme, car toutes les valeurs des indexes sont équiprobables. Dans le cas contraire, le vote est complexe et doit être réalisé par les méthodes probabilistes [MOR93] [SCH96].

## II.2) Indexation géométrique étendue :

GROS LAMIROY[LAM96][LAM98] ont introduit une méthode de mise en correspondance d'images segmentées, qu'ils la considèrent comme proche de la technique du Hachage Géométrique, Sauf que celle-là, utilise seulement deux quasi-invariants qui sont: l'angle ($\theta$) formé par deux segments adjacents, et leur rapport de longueur ($\rho$).

Cette technique s'appelle « l'Indexation Géométrique Etendue », puisqu'elle est basée sur des données géométriques pour caractériser les images, et elle peut être étendue vers d'autres descripteurs.

Le principe de l'algorithme de reconnaissance par indexation géométrique étendue est de comparer l'image inconnue à chacun des modèles de la base et d'offrir un classement de ces modèles, et puisque le coût d'exécution ne permet pas de parcourir tous les modèles et de les comparer un à un à l'image inconnue, alors il faut organiser les quasi-invariants ($\rho$,$\theta$) pour réduire la reconnaissance à une mise en correspondance entre deux images.

Pour chaque image, on énumère toutes les configurations de deux segments concourants. Après on calcule les quasi-invariants associés, formés par l'angle et le rapport de longueurs des deux segments intervenants. On peut considérer que les couples de configurations qui ont des quasi-invariants associés proches forment des appariements plausibles.

Ensuite, on établie une table d'indexation bidimensionnelle qui a comme clés les quasi-invariants précédemment calculés. Chaque cellule de la table pointe vers une liste de couples (vecteur d'invariants ($\rho$,$\theta$), modèle correspondant). Pour la reconnaissance, l'image inconnue est segmentée et ses descripteurs sont comparés à ceux stockés dans la table. Chaque mise en correspondance ainsi obtenue est classée selon le modèle auquel appartient le deuxième descripteur.

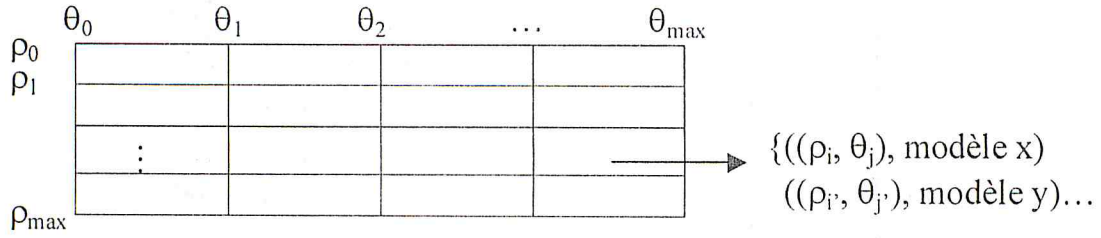



**Fig III.2 : La table d'indexation**

Au lieu de procéder à un simple vote majoritaire, cette technique fait recours à la transformée de HOUGH pour chaque mise en correspondance trouvée. En effet, l'hypothèse de base permet de dire que le mouvement apparent entre deux images peut être approché par une similitude. Donc, chaque appariement fournit individuellement une transformation entre les deux images. Alors on peut déduire que des appariements corrects définiront des transformations identiques, tandis que les mises en correspondance erronées définiront des transformations différentes.

La vérification par mouvement apparent global vient du fait qu'à partir d'une certaine complexité d'images et d'un certain nombre de modèles, les erreurs liées à l'appariement deviennent suffisamment nombreuses pour qu'un simple vote comptage des mises en correspondance ne puisse plus donner la réponse exacte [GRI90].

Le contexte de reconnaissance et de mise en correspondance étant défini, quelques améliorations peuvent être apportées. Il est clair que des méthodes d'indexation ne sont optimales que lorsque les clés ont une distribution uniforme [CAR97]. Puisque la distribution de l'angle et du rapport de longueur n'est pas uniforme, [LAM98] propose d'opérer des modifications sur les clés d'indexation $\rho$ et $\theta$ :

$$f(\rho,\theta)=\left(\left|\frac{\ln(\rho)+\ln(\rho_{max})}{2\ln(\rho_{max})}\,k_\rho\right|\ ,\ \left|\frac{\theta}{\theta_{max}}\,k_\theta\right|\right)$$

$$\text{avec : } \rho\in\left]\frac{1}{\rho_{max}},\rho_{max}\right[\ ,\quad \theta\in[0,\theta_{max}[$$

où : $k_\rho$ et $k_\theta$ représentent le nombre de paniers disponibles en chaque variable.

$\rho_{max}$ (resp. $\theta_{max}$) : valeur maximale de $\rho$ (resp. $\theta$).

Pour traiter le problème de l'incertitude, on peut utiliser les solutions présentées dans les méthodes précédentes. Aussi, il existe une autre solution [LAM96] consiste à utiliser deux tables identiques, mais découpées différemment, c'est à dire n'ayant pas leur frontières aux mêmes endroits. On peut alors déterminer la table à utiliser pour que le descripteur recherché soit le plus éloigné possible d'une frontière, ceci évite de regarder les cases voisines. Le stockage en mémoire secondaire des vecteurs de descripteurs peut être réalisé en utilisant un Quad-tree [LAM98]. La technique de l'indexation géométrique étendue peut être étendue vers d'autres descripteurs, mais ceci augmente la complexité de la recherche des correspondants.

L'inconvénient majeur de cette technique est son extrême sensibilité envers la précision de la segmentation, et son incapacité de gérer de très grands volumes de données. Par contre, elle est capable dans des situations complexes, et en absence de tout contrôle d'éclairage, d'étalonnage de procéder efficacement à des tâches d'identification.

## II.3) Techniques d'indexation évoluées :

[Web98] montre que le nombre de dimension au-dessus duquel une recherche séquentielle devient plus performante que toute navigation dans un index se situe autour de 10.

Deux approches ont été récemment proposées et conçues spécifiquement pour éviter le problème de la dimensionnalité des données, ce sont : le Pyramid-Tree [Ber98] et le VA-File [Web98].

Elles ont , en effet , été conçues dès le départ de façon à ce que leur comportement ne dégénère pas trop rapidement lorsque sont indexées des données ayant de nombreuses dimensions ; et ça en améliorant les performances de la recherche séquentielle, puisque celle-ci s'avère la plus efficace en grandes dimensions.

La technique du VA-File utilise une première phase d'approximation permettant de diminuer le nombre de descripteurs à comparer, et le Pyramid-Tree utilise une méthode dont la complexité croit linéairement (et non exponentiellement) avec la dimension. Elles sont actuellement les plus performantes.

## II.3.1) VA-File « Vector Approximation File »:

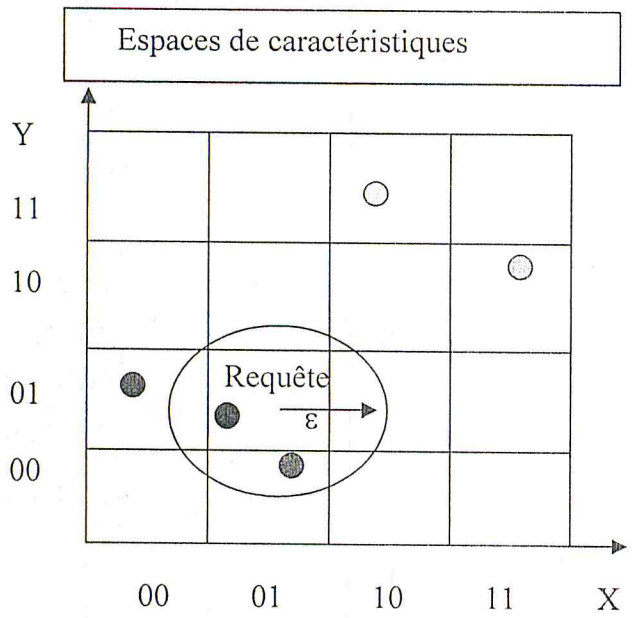
Weber , Schek et Blott développent une méthode qui vise à améliorer les performances de la recherche séquentielle, puisque celle-ci s'avère la plus efficace en grandes dimensions, c'est la technique <<VA-File>> qui a été présenté par [Web 98].

Cette technique utilise une première phase d'approximation permettant de diminuer le nombre de descripteurs à comparer. Elle gère deux ensembles de données : un fichier contenant les descripteurs et un autre contenant une approximation géométrique de ces descripteurs. Ce dernier fichier doit tenir en mémoire centrale pour garantir de bonnes performances.

Lors de la création de l'index, pour réaliser l'approximation géométrique, le VA-File découpe chaque dimension $d_i$ en 2 puissance $b_i$ cases (ce découpage est codé sur $b_i$ bits).

On découpe de même chacune des $d$ dimensions de l'espace, ce qui revient à découper l'espace en $2^b$ hypercubes (où $b$ est la somme de tous les $b_i$). Il existe ainsi $2^b$ cases, qui sont numérotées de 0 à $2^{b-1}$. Les descripteurs de la base sont alors lus les uns après les autres pendant la phase du pré-traitement, et l'approximation d'un descripteur est déterminée par la case entre les limites de laquelle il tombe. L'approximation est égale au numéro de la case identifiée.

Le fichier d'approximation (VA-File) est ainsi composé des approximations de paires (identifiant de descripteur, numéro de case). Seules les informations liées aux cases contenant au moins un descripteur sont stockées, évitant ainsi le problème d'avoir à gérer un très grand nombre de cases vides.




Vecteurs

| | ID | Coordonnées | |
|---|---|---|---|
| ○ | 1 | 0.48 | 0.50 |
| ◎ | 2 | 0.72 | 0.58 |
| ● | 3 | 0.09 | 0.34 |
| ● | 4 | 0.22 | 0.29 |
| ● | 5 | 0.37 | 0.19 |

Approximations

| | |
|---|---|
| 1 | 1011 |
| 2 | 1110 |
| 3 | 0001 |
| 4 | 0101 |
| 5 | 0100 |

**Fig III.3 : Création du fichier d'Approximations et Principe de la recherche**

Lors de la recherche, le descripteur requête subit le même processus d'approximation. L'algorithme détermine alors les cases (hypercubes) qui se trouvent à une certaine distance ε. Un accès séquentiel aux vecteurs contenus dans ces hypercubes permet d'éliminer les vecteurs dont la distance du vecteur requête dépasse ε.

Cette étape agit donc comme un filtre qui élimine de la recherche les hypercubes (et donc les descripteurs) n'ayant aucune chance de faire partie de la réponse, limitant le nombre de comparaisons à faire par rapport à la recherche séquentielle.

L'objectif principal de l'approche VA-File est:
– Le filtrage des données par approximation en mémoire centrale
– La recherche séquentielle et exhaustive dans l'espace de recherche approché

## II.3.2) Pyramid-Tree :

Le pyramid-tree a été proposé par BERCHTOLD, BOHM et KRIEGEL [BER98], c'est une technique d'indexation de vecteurs imprécis dans un espace de données de grande dimension.

Elle consiste à transformer l'espace de données vers l'espace canonique de données multidimensionnel $[0,1]^d$. Puis cet espace est divisé en deux étapes :

- Au début, il est découpé en $2d$ pyramides. Chaque pyramide est numéroté et a son sommet placé au centre de l'espace (0.5, 0.5, ...,0.5), et sa base est l'hyperplan de dimension ($d$-1).
- En second lieu, chaque pyramide est divisée en plusieurs tranches parallèles à sa base, chacune de ses tranches correspond à une page de données d'un $B^+$-tree.

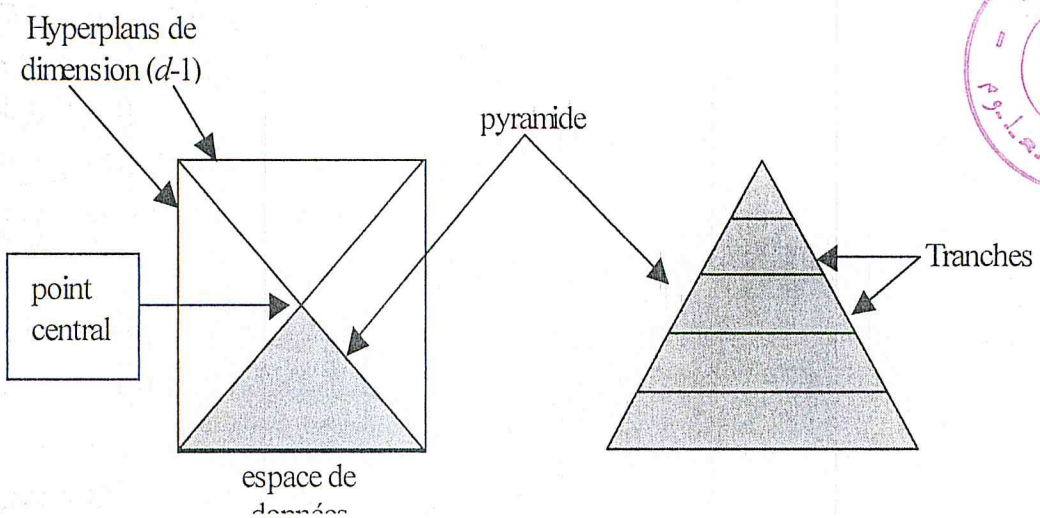


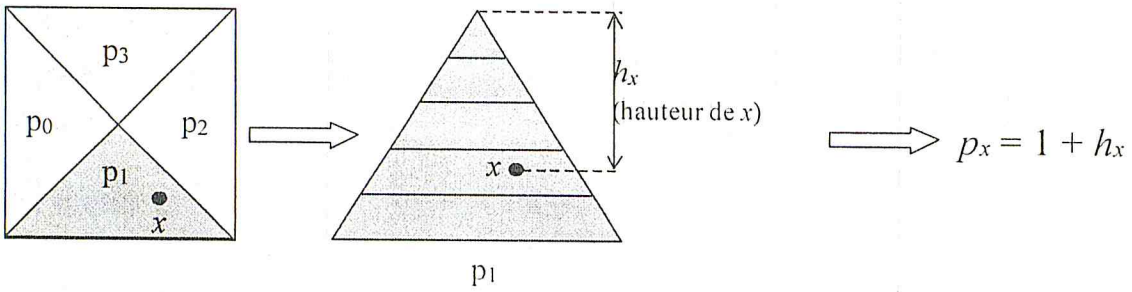



**Fig III.4 : Principe du découpage du Pyramid-tree (espace de 2 dimensions)**

Ce double découpage permet à tout point $x$ de l'espace d'être exprimé comme une paire (i , $h_x$ ) où i est le numéro de la pyramide à laquelle appartient le point $x$, et $h_x$ la hauteur de ce point dans cette pyramide (distance entre x et le hyperplan contenant le centre de l'espace).

Donc pour tout point x de l'espace on calcule sa *valeur pyramidale* $p_x = (i + h_x)$. Les valeurs de i sont entières et celles de la hauteur sont des réels

parcourant l'intervalle [0, 0.5]. Par conséquent, chaque pyramide couvre un intervalle de [$i$, $i$+0.5] valeurs pyramidales, et les ensembles des valeurs pyramidales de deux pyramides différentes sont disjoint.




avec 1 : numéro de pyramide
$h_x$ : hauteur dans la pyramide

**Fig III.5 : Calcul de la valeur pyramidale d'un point**

Ayant la transformation déterminant la valeur pyramidale d'un point, l'étape de l'indexation devient aisée. Pour chaque point, sa valeur pyramidale est calculée, puis il est inséré dans un B-tree en utilisant cette valeur comme clé. Cette valeur est stockée avec les points correspondant puisque la transformation vers la valeur pyramidale n'est pas injective. Donc, le problème de stockage de vecteurs d-dimensionnels se réduit à un stockage de valeurs unidimensionnelles.

Contrairement à l'étape de la création de l'index, la recherche est très complexe. Pour trouver les vecteurs qui sont à une certaine distance d'un vecteur requête $x\,(x_1, x_2, \ldots, x_d)$, on recherche les tranches des pyramides de l'espace qui contiennent l'hyperrectangle :

$$(x_{1\min}, x_{1\max}), (x_{2\min}, x_{2\max}), \ldots, (x_{d\min}, x_{d\max})$$

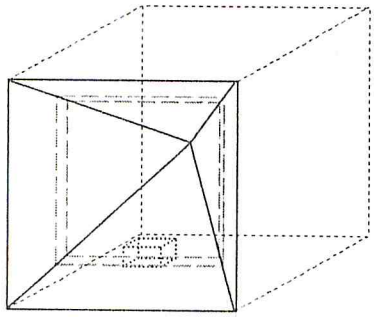


**Fig III.6 : Modélisation de la requête (Espace 3D)**

Donc, les vecteurs contenus dans ces tranches sont candidats à une possible mise en correspondance avec le vecteur requête. Le calcul des tranches traversées par l'hyperrectangle de la requête s'avère être une opération très compliquée, et cet hyperrectangle peut couper plusieurs pyramides, ce qui rend la recherche des voisins très complexe. En plus, en cas de données réelles, seules quelques pyramides vont contenir la majorité des données, ce qui va engendrer un arbre non équilibré.

# III) Etude comparative :

Une étude a été réalisée par R.MEZHOUD, P. GROS et L.AMSALEG [AMS00] pour comparer les performances de trois techniques d'indexation : le VA-File, le Pyramid-tree et l'accès séquentiel, puisque cette dernière s'avère être très compétitive lorsque la dimension de l'espace accroît.

Dans une première partie, cette étude montre l'évolution des performances des trois techniques en fonction de la dimension des descripteurs retenus pour la reconnaissance, puis elle montre l'influence de la taille de la base sur les performances, et enfin, l'influence du nombre de descripteurs formant la requête.

Pour l'expérimentation, un descripteur local de dimension 24 est utilisé, et qui tolère des occultations partielles et des changements de composition de la scène observée, et qui reste invariant aux translations et aux rotations 2D de l'image, ainsi qu'aux variations affines de l'illumination. L'objet de la requête est de retourner les 10 plus proches voisins. Les tests ont été effectués sur des données tirées aléatoirement selon une loi uniforme dans l'intervalle [0,1], puis sur des données réelles qui consistent en 610 images, et en 1206 images constituant une cinquantaine de secondes consécutive d'une séquence vidéo, ce qui donne en tout 1816 images décrites par 413412 descripteurs.

## III.1) Influence de la dimension :

### III.1.1) Test :

Pour étudier l'influence de la dimension, les descripteurs sont tronqués à 2, 4, 7, 10, 15, 20 et 24 dimensions.

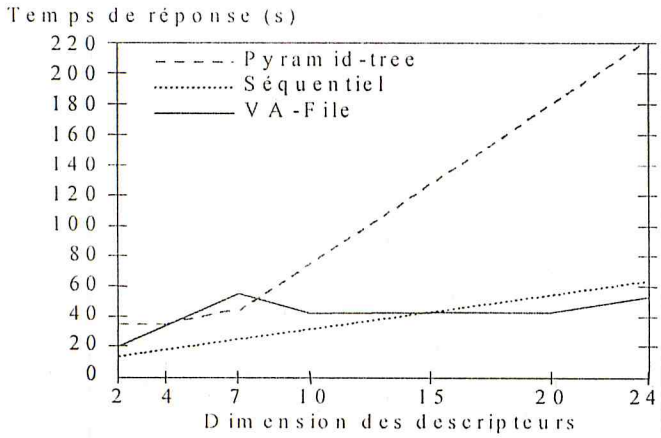



**Fig III.7 : Influence de la dimension des descripteurs sur le temps de réponse (Base de 413.412 descripteurs, 150 descripteurs par requête)**

### III.1.2) Résultat :

Le Pyramid-tree est la technique la moins performante, quelque soit la dimension. En dessous de 15 dimensions, le séquentiel est meilleur que le VA-File dont le temps de réponse reste autours d'une minute.

## III.2) Influence de la taille de la base :

### III.2.1) Test :

Pour étudier l'influence de la taille de la base, de nouvelles bases ont été générées en ne conservant que 100.000, 200.000, 300.000 et 400.000 de descripteurs contenus dans la base initiale qui en contient 413.412.

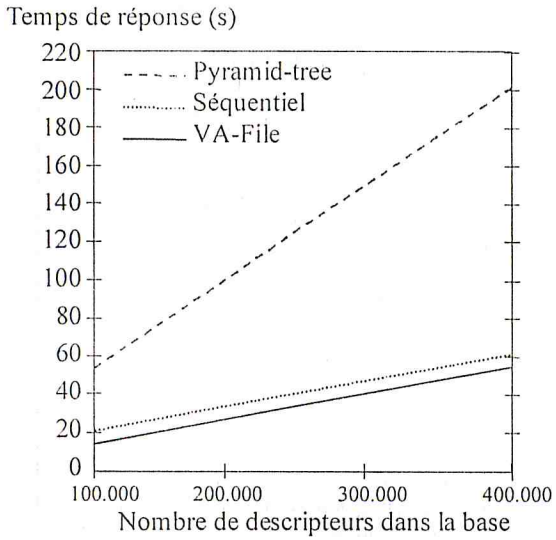



**Fig III.8 : Influence de la taille de la base sur le temps de réponse (Descripteurs de 24 dimensions, 150 descripteurs par requête)**

### III.2.2) Résultat :

Les constatations sont du même ordre que pour le cas précédent, à savoir que le Pyramid-tree a un coût nettement supérieur aux autres techniques, et que le VA-File est légèrement plus performant que la recherche séquentielle.

## III.3) Influence du nombre de descripteurs dans la requête :

### III.3.1) Test :

Pour étudier l'influence du nombre de descripteurs dans la requête, tous les descripteurs sont utilisés et sont de dimension 24. Pour former les requêtes, des images ont été cherchées dans la base, et pour lesquelles 150, 200, 300 et 400 descripteurs étaient calculés. Aussi, une requête artificielle ne contenant qu'un seul descripteur a été testée.

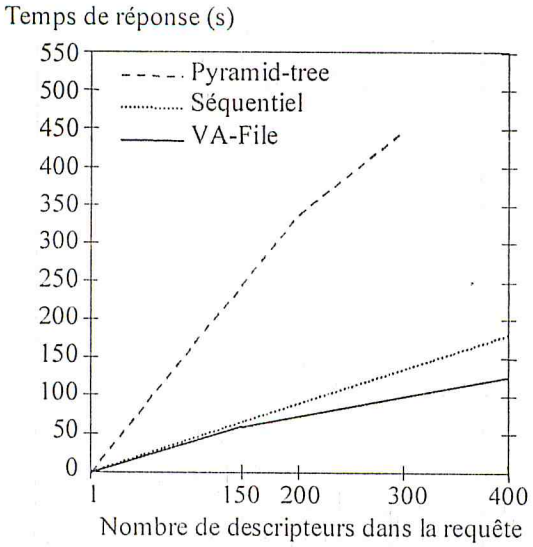



**Fig III.9 : Influence du nombre de descripteurs dans la requête sur le temps de réponse (Base de 413.412 descripteurs, descripteurs de 24 dimensions)**

### III.3.2) Résultat :

Les résultats de cette expérience montrent de nouveau que seuls le VA-File et la recherche séquentielle restent compétitifs. Malgré cela, les temps de réponse augmentent fortement à mesure que le nombre de descripteurs dans une requête croît.

## III.4) Interprétation :

Ces évaluations montrent que l'accroissement de l'un des facteurs allonge fortement le temps de réponse.

Le VA-File résiste le mieux à l'augmentation d'un facteur, mais son temps de réponse reste très sensible à ces variations. Et ceci en sachant que les paramètres utilisés et variés ont des valeurs moyennes et non pas extrêmes, ce qui va entraîner une dégradation de performances encore plus importante.

## IV) Conclusion :

Dans ce chapitre, nous avons présenté les différentes techniques d'indexation pour la reconnaissance par l'apparence, en se basant sur les techniques d'indexation fondées sur l'utilisation des caractéristiques géométriques des images.

Parmi ces techniques, on s'intéresse à l'indexation géométrique étendue, Puisque elle supporte mieux les différentes transformations (rotation, translation, homothétie) des objets observés, et elle permet l'utilisation d'images obtenues par des moyens différents, puisqu'elle n'a aucun besoin pour le calibrage des capteurs, ou bien de connaître les conditions de prise de vue.

Dans le chapitre suivant, nous allons exposer cette technique d'indexation avec plus de détails, pour pouvoir l'implémenter sur machine et tester par la suite son efficacité.

# Chapitre IV
# Indexation géométrique étendue

# Chapitre IV

# Indexation géométrique étendue

## I) Introduction :

Plusieurs méthodes d'appariement entre une image et une base d'images utilisent des données géométriques (telles que segments, jonctions et ellipses) qui sont en général locales et invariantes aux différentes transformations.
Ces méthodes permettent de traiter uniquement des objets relativement simples, dans le cas d'objet plus complexe, le calcul de grandeurs géométriques devient instable.

Dans ce chapitre, nous allons présenter la méthode de reconnaissance que nous allons utiliser, et qui est basée sur les caractéristiques géométriques locales des objets. Le principe est de mettre en correspondance une image de l'objet inconnu avec les images appartenant aux modèles stockés dans la base de connaissance, et à la fin, classer ces modèles suivant les résultats de la mise en correspondance.

## II) Indexation géométrique étendue :

### II.1) Description de la méthode utilisée :

GROS a introduit une méthode de mise en correspondance d'images segmentées fondées sur le calcul d'invariants basée sur des transformations affines ou des similitudes (rotation, translation et homothétie).

L'hypothèse générale de cette technique est que le mouvement apparent entre deux images qui représentent le même objet, peut être approché par une similitude, ceci suppose que le changement de point de vue soit un peu modéré.

Ce qui permet de caractériser les images par des quasi-invariants qui sont l'angle ($\theta$) formé par deux segments adjacents et leur rapport de longueur ($\rho$).

L'utilisation des quasi-invariants vient remplacer celle des invariants projectifs qui furent utilisés auparavant pour la reconnaissance [LAR91][MOR93], et ce, malgré qu'il n'existe pas d'invariant formel lors d'une transformation projective de R3 vers R2 [LAM98]. Mais leur fragilité par rapport au bruit et leur pouvoir descriptif un peu faible [AST94] les rendent moins favorisés que les quasi-invariants pour leur utilisation à des fins de reconnaissance.

Dans notre travail, nous nous sommes basées sur la technique d'indexation géométrique étendue, en ajoutant quelques améliorations au niveau du stockage des données et du vote. Nous allons utiliser le Quad-tree comme une structure de stockage pour indexer les descripteurs afin d'optimiser la phase de recherche, au lieu de la table de hachage bidimensionnelle utilisée à l'origine. Aussi, nous nous sommes inspirées de la technique du VA-File pour l'étape du vote.

## II.2) Quasi-invariants :

### II.2.1) Définition :

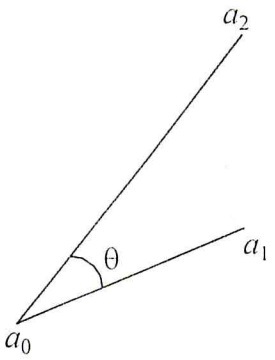
L'apparition des quasi-invariants dans la communauté de vision remonte à la fin des années 1960 et est due à Binford [Bin93] , Binford reprend et précise la définition des quasi-invariants. Il démontre que les invariants (un invariant est une propriété qui est constante pour un ensemble de fonctions) au groupe des similitudes image sont des quasi-invariants pour une transformation perspective.

### II.2.2) Calcul des quasi-invariants :

L'utilisation des quasi-invariants est mieux que l'utilisation des invariants projectifs qui passe nécessairement par des hypothèses (telle la coplanarité pour le calcul du birapport [AOU93]), car ils sont plus facile à extraire d'une image 2D, et moins coûteux en temps de calcul.

Deux quasi-invariants seront utilisés:

L'angle $\theta$ formé par deux segment adjacents et leur rapport de longueurs $\rho$.




$$\rho = \frac{\overrightarrow{a_0a_1}}{\overrightarrow{a_0a_2}} \qquad \theta = \arccos\frac{\overrightarrow{a_0a_1}.\overrightarrow{a_0a_2}}{\|\overrightarrow{a_0a_1}\|.\|\overrightarrow{a_0a_2}\|}$$

**Fig IV.1 : Le calcul des quasi-invariants (l'angle et le rapport de longueurs) à partir d'une configuration de segment adjacents [GRO95A]**

Par convention, l'ordre des segments est choisi tel que l'angle θ soit toujours positif et minimum.

Alors pour chaque configuration, les deux valeurs θ et ρ extraites seront toujours les mêmes quelque que soit la position et la direction de cette configuration.

## II.3) Définition d'une similitude :

Une similitude est une bijection définie de $\mathbb{R}^2$ vers $\mathbb{R}^2$. Elle est composée de trois transformations bijectives: une translation, une rotation et une homothétie.

### II.3.1) Translation :

On définie une translation de vecteur $\overrightarrow{T}$ comme étant une application affine de $\mathbb{R}^2$ vers $\mathbb{R}^2$ qui associe à tout point $M$ le point $M'$ tel que $\overrightarrow{MM'} = \overrightarrow{T}$ .

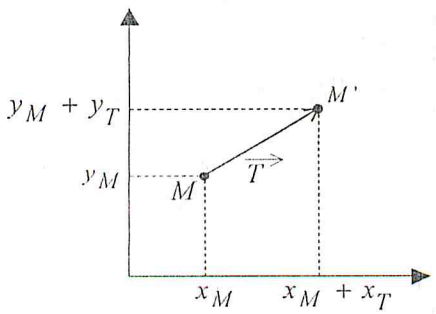



**Fig IV.2 : Translation d'un point *M***

## II.3.2) Rotation :

Une rotation d'angle $\alpha$ est un automorphisme de $\mathbb{R}^2$ qui associe à chaque point $M$ le point $M'$ tel que l'angle compris entre les deux segments $[OM]$ et $[OM']$ soit égal à $\alpha$ ($M\hat{O}M' = \alpha$).

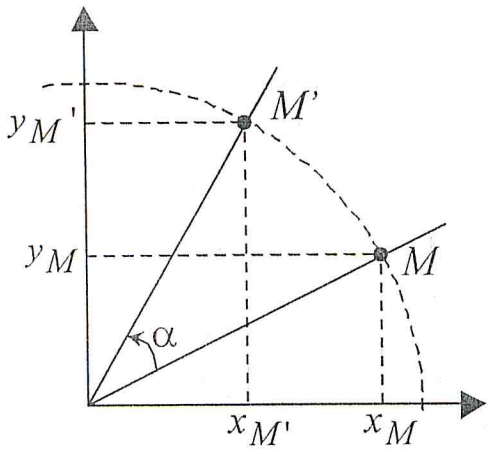



**Fig IV.3 : Rotation d'un point $M$ d'un angle $\alpha$**

Le point $M'$ est obtenu par la relation :

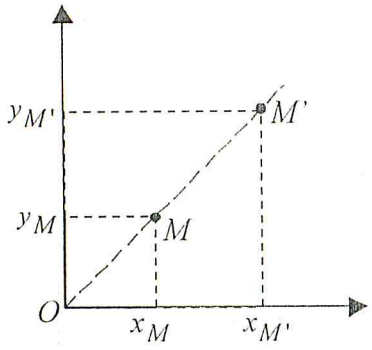
$$\begin{pmatrix} x_{M'} \\ y_{M'} \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \times \begin{pmatrix} x_M \\ y_M \end{pmatrix}$$

d'où :

$$\begin{cases} x_{M'} = x_m.\cos\alpha - y_M.\sin\alpha \\ y_{M'} = x_m.\sin\alpha + y_M.\cos\alpha \end{cases}$$

## II.3.3) Homothétie :

Une homothétie de paramètre $k$ est une application définie de $\mathbb{R}^2$ vers $\mathbb{R}^2$, qui associe à chaque point $M$ le point $M'$ tel que $\overrightarrow{OM'} = k\ \overrightarrow{OM}$, où $k$ est un scalaire.




$$\overrightarrow{OM'} = k\ \overrightarrow{OM}$$

**Fig IV.4 : Homothétie de paramètre $k > 1$ d'un point $M$**

Les cordonnées du point $M'$ sont données par les relations suivantes :

$$\begin{cases} x_{M'} = k\ x_m \\ y_{M'} = k\ y_M \end{cases}$$

## II.2.4) Calcul des paramètres d'une similitude :

Une similitude de paramètres $(k, \alpha, \overrightarrow{T})$ est composée d'une homothétie d'un scalaire $k$, d'une rotation d'angle $\alpha$ et d'une translation de vecteur $\overrightarrow{T}$.

Étant donné deux configurations de deux segments adjacents $(a, a_1, a_2)$.et $(b, b_1, b_2)$ :

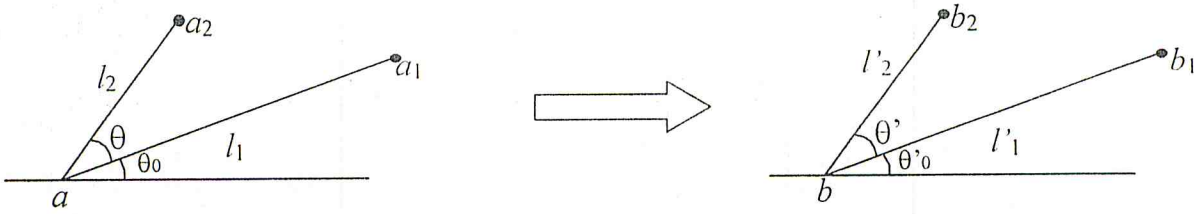



**Fig IV.5 : Similitude entre deux configurations**

$(a, a_1, a_2)$ et $(b, b_1, b_2)$ : configurations de deux segments adjacents.
$l_1$ (resp. $l_2$) : longueur du segment $[a\ a_1]$ (resp. $[a\ a_2]$).
$l'_1$ (resp. $l'_2$) : longueur du segment $[b\ b_1]$ (resp. $[b\ b_2]$).
$\theta$ (resp. $\theta'$) : angle compris entre les segments $[a\ a_1]$ et $[a\ a_2]$ (resp. $[b\ b_1]$ et $[b\ b_2]$).
$\theta_0$ (resp. $\theta_0'$) : angle compris entre l'horizontale et le segment $[a\ a_1]$ (resp. $[b\ b_1]$).

Les paramètres de la similitude entre ces deux configurations sont donnés par [GRO95A] :

$$k = \frac{1}{2}\left(\frac{\|\overrightarrow{b\,b_1}\|}{\|\overrightarrow{a\,a_1}\|} + \frac{\|\overrightarrow{b\,b_2}\|}{\|\overrightarrow{a\,a_2}\|}\right)$$

$$\alpha = \left(\widehat{\overrightarrow{a\,a_1}\ \overrightarrow{b\,b_1}}\right) + \frac{1}{2}\left[\left(\widehat{\overrightarrow{b\,b_1}\ \overrightarrow{b\,b_2}}\right) - \left(\widehat{\overrightarrow{a\,a_1}\ \overrightarrow{a\,a_2}}\right)\right]$$

$$T = \overrightarrow{a'b} \quad \text{où} \quad a' = H_k \circ R_\alpha(a)$$

$R_\alpha$ représente une rotation d'angle $\alpha$ dont le centre est l'origine, et $H_k$ une homothétie d'echelle $k$ dont le centre est l'origine aussi.

D'où :

$$k = \frac{1}{2}\left(\frac{l'_1}{l_1} + \frac{l'_2}{l_2}\right)$$

$$\alpha = |\theta'_0 - \theta_0| + \frac{1}{2}(\theta' - \theta)$$

$$T = \begin{pmatrix} x_b - k(x_a \cos\alpha - y_a \sin\alpha) \\ y_b - k(x_a \sin\alpha + y_a \cos\alpha) \end{pmatrix}$$

## II.4) Les descripteurs :

### II.4.1) Définition :

Un descripteur est un vecteur qui porte les informations de l'image, il peut être global lorsque un seul descripteur représente toute l'image, comme il peut être local lorsque l'image porte plusieurs descripteurs.

### II.4.2) Composition des descripteurs utilisés :

Les attributs qui seront retenus pour former un descripteur doivent être vérifié les deux conditions suivantes :

- Deux configurations différentes de deux segments adjacents ne peuvent pas avoir le même descripteur.

- Ces attributs permettent le calcul des paramètres d'une similitude existante entre deux configurations.

Donc les attributs formant un descripteur sont :

- L'angle θ formé par les deux segments.
- L'angle $\theta_0$ formé par le premier segment et la semi-droite $[OX$.
- Les longueurs des deux segments : $l_1$ et $l_2$.
- Les coordonnées du point d'intersection des deux segment $x_0$, $y_0$.

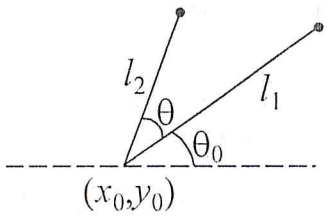



**Fig IV.6: Les éléments formants un descripteur**

Les descripteurs seront indexés suivant les valeurs de l'angle θ et du rapport $\rho = l_1/l_2$. Leur choix comme clé de stockage vient du fait qu'elles sont invariantes aux similitudes. Donc, une condition nécessaire de la validité de la mise en correspondance entre deux configurations est que leurs angles et rapports soient relativement proches.

Pour ne pas calculer ρ à chaque fois à partir de $l_1$ et $l_2$, il vaut mieux de l'intégrer dans la structure du descripteur pour optimiser le temps de recherche.

## II.4.3) Étude Statistique sur les éléments de l'index:

Les méthodes d'indexation ne sont optimales que quand les clés ont une distribution uniforme dans leur espace d'indexation.

Pour cela et afin de bien mener l'implémentation de la phase d'indexation des descripteurs, nous avons réalisé une étude statistique sur la distribution des éléments de l'index (l'angle θ et le rapportρ).

Cette étude a porté sur 31957 descripteurs extraits de 697 images (la manière dont ces données ont été obtenues est reprise en détail dans le chapitre V).

### II.4.3.1) L'angle (θ) formé par les deux segments:

L'angle θ a déjà les bonnes propriétés , mais il est difficile de comparer deux valeurs dans l'intervalle [0 , 2π[, puisqu'elles sont définies modulo π . Par contre, il est évident que l'on peut toujours considérer comme le plus petit angle formé par les deux segments, on obtient ainsi des valeurs dans l'intervalle [0, π[. Celle-ci conservant la propriété de la distribution uniforme et éliminent la contrainte sur la comparaison.

Nous avons étudié la distribution des descripteurs suivant l'angle θ, et nous avons obtenu le graphe ci-dessous.

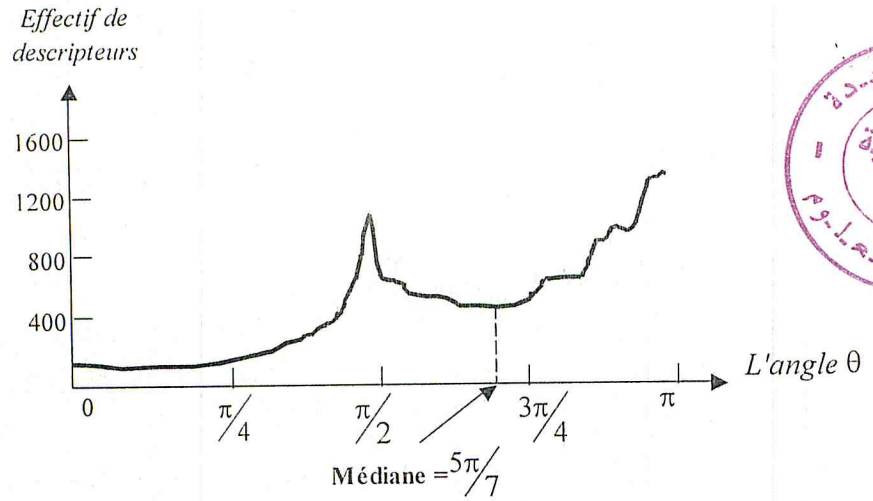





**Fig IV.7: Distribution des descripteurs suivant l'angle θ**

Nous remarquons que la quantité de l'information (nombre de descripteurs) dans l'intervalle [0,5π/7[ est la même que dans [5π/7,+∞[ , donc la médiane est égale à 5π/7 .Aussi il y a un pic qui se situe autours de $\pi/2$, un angle qui apparaît souvent dans les configurations. Nous remarquons aussi qu'il y a une concentration se situant autours des grands angles, à cause de la mauvaise qualité de la segmentation qui, pour certaines configurations, transforme un angle assez petit, en deux angles de plus grande envergure (cf. Figure VI.8).

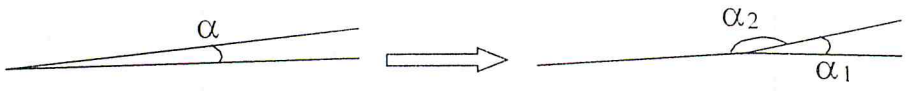



**Fig VI.8: Exemple d'une transformation engendrée par la segmentation**

## II.4.3.2) Le rapport (ρ) des longueurs des deux segments:

Dans une configuration de deux segments adjacents, la probabilité pour que le 1er segment soit plus long que le 2nd segment est égale à celle où il est moins long. Alors, le nombre de configurations dont le rapport $\rho \geq 1$ $(\rho \in [1,+\infty[)$ est égal à celui des configurations dont le $\rho \leq 1$ $(\rho \in ]0,1])$ ; donc une densité très importante dans la distribution des descripteurs dans cette région de l'espace de données $(\rho \in ]0,1])$ . Et le graphe ci-dessous confirme cette information.

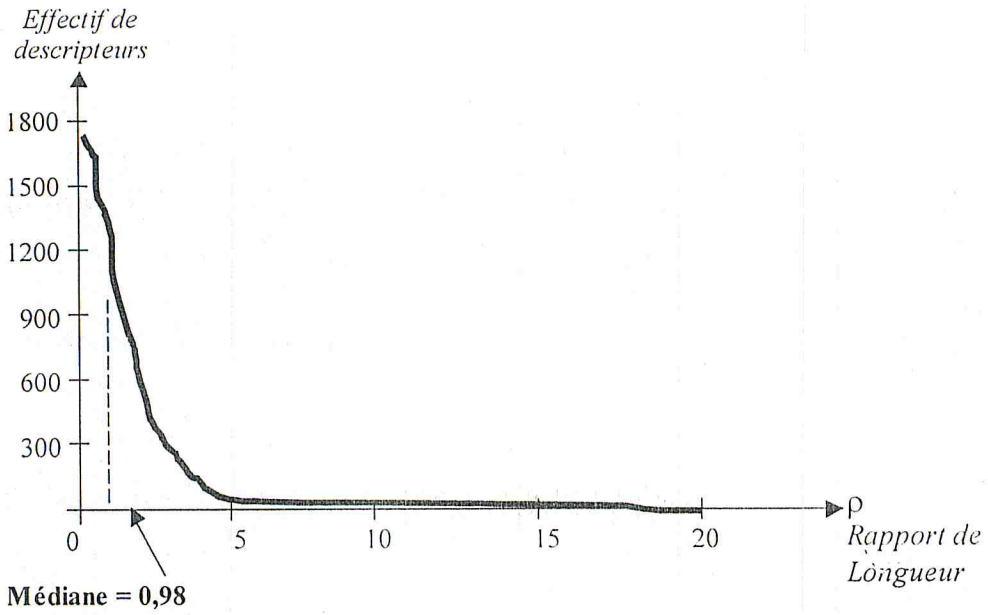



**Fig IV.9: Distribution des descripteurs suivant le rapport de longueur ρ**

La forte densité dans l'intervalle ]0,1] apparaît clairement dans cette courbe. Les deux intervalles ]0 , 0,98] et [0,98 , +∞[ contiennent la même quantité de descripteurs , d'où la médiane étant égale à 0,98.

Pour remédier un peu à ce problème, afin de rendre la distribution des descripteurs plus uniforme, il suffit de remplacer le rapport ρ par son logarithme népérien $\ln(\rho)$.

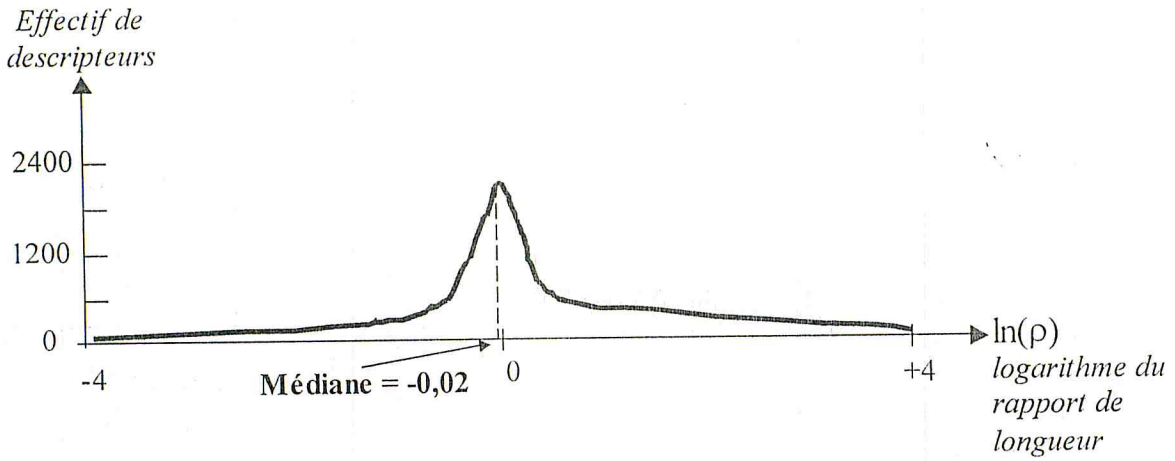



**Fig IV.10: Distribution des descripteurs suivant le logarithme népérien du rapport de longueur ($\ln(\rho)$)**

Puisque plus de 98% des descripteurs sont contenus dans l'intervalle [-4,+4] ; nous avons exclu les descripteurs dont la valeur absolue du logarithme du rapport dépasse 4 ($|\ln(\rho)| > 4$),et en dépassant cette valeur, le quasi-invariant n'aura pratiquement aucune signification à cause du bruit.

## II.4.4) La forme finale d'un descripteur :

Au début nous avons représenté un descripteur par les attributs suivants :

$\theta$ , $\theta_0$, $l_1$ , $l_2$, $x_0$ , $y_0$ .

Pour faciliter et accélérer le calcul de (ρ), c'est à dire éviter de l'évaluer à chaque fois à partir de $l_1$ et $l_2$, on l'ajoute dans la structure de descripteur, mais après l'étude statistique établie, nous le remplaçons par ($\ln(\rho)$) qui assure une meilleure répartition que (ρ).

Donc, la forme d'un descripteur sera comme suit :

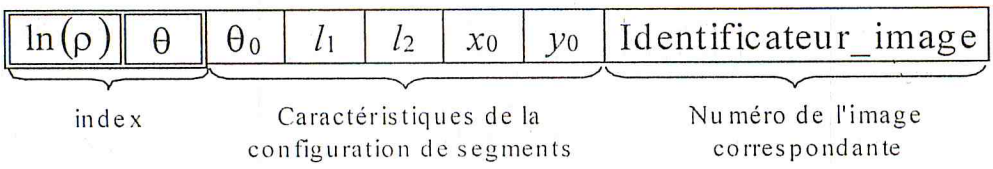



**Fig IV.11 : la forme d'un descripteur**

## III) Étapes de l'algorithme:

Les images utilisées pour implémenter cet algorithme sont sous forme d'ensembles de segments de droite, l'extraction des contours et leurs approches polygonales sont supposées être faites par d'autres algorithmes.

L'algorithme sera réparti en deux grandes étapes :

- Le pré-traitement : la construction de la base (la création et l'insertion).
- La reconnaissance : (la recherche et le vote), qui nous intéresse plus, puisque la première étape ne sera effectuée qu'une seule fois ; en plus, l'efficacité d'un algorithme repose sur la rapidité de la reconnaissance. Donc, la priorité sera donnée à l'opération de recherche au détriment de celle de l'insertion.

### III.1) Pré-traitement :

#### III.1.1) Structure de l'indexation :

La structure d'indexation que nous avons utilisé comme base est le Quad-tree, qui est un arbre à précision infinie destiné à l'indexation de données bidimensionnelles.

Pour stocker les feuilles et les nœuds d'un Quad-tree, nous avons utilisé deux fichiers, un fichier de descripteurs (qui contient les feuilles) et un fichier d'index (qui contient les nœuds).

Puisque les descripteurs seront stockés dans un fichier indépendamment de l'index, ceci donne une grande liberté pour les opérations d'insertion; et aussi, puisque les nœuds ne contiennent pas de descripteurs, ceci donne la possibilité de maintenir tout l'index dans la mémoire centrale lors de l'étape de recherche en raison de sa taille relativement petite, et ce qui permet d'optimiser le temps de réponse

**Le fichier de descripteurs :** contient les feuilles, chaque feuille contient ***Taille_Feuille*** descripteurs, et un compteur qui donne le nombre de descripteurs contenus.

```
Feuille = Enregistrement
            Compteur : entier ;
            Case : Tableau[Taille_Feuille] de Descripteur ;
          Fin ;
```

**FigIV.12: Structure d'une feuille**

**Le fichier d'index :** est composé de nœuds, Les nœuds contiennent les coordonnées (X,Y) du point qui divise le rectangle qui lui est associé en quatre parties, et quatre pointeurs sur ses fils associés à ces parties (un pointeur peut être vide ou pointer vers un autre nœud ou une feuille) .

Ce fichier doit être maintenu dans la mémoire centrale pendant les opérations d'insertion et de recherche. Ceci est fait en copiant tous les nœuds dans un tableau de taille dynamique, donc tous les pointeurs des nœuds pointent vers les entrées de leurs nœuds fils dans ce tableau.

```
Pointeur = Enregistrement
             Type : char ;    { vide, nœud ou feuille }
             Position : entier ;
           Fin ;
```

**FigIV.13: Structure d'un pointeur**

```
Nœud = Enregistrement
         X,Y : Réel ;
         Fils : Tableau[4] de Pointeur ;
```

**FigIV.14: Structure d'un nœud**

## III.1.2) Création de la base:

Cette procédure permet de créer une base d'objets, qui est formée de quatre fichiers :

- Un fichier (*.dsc) qui va contenir les feuilles du quad-tree.
- Un ficher (*.idx) qui va contenir l'index, c'est à dire les nœuds de l'arbre. A sa création, ce fichier contient la racine seulement avec les valeurs $\ln(\rho)=0$ et $\theta=5\pi/7$ considérées comme les valeurs médianes de l'espace de données.
- Un fichier (*.img) qui va contenir les couples (numéro image, numéro objet) pour faire le lien entre une image et l'objet qu'elle représente.
- Un fichier (*.obj) qui va contenir les couples (numéro de l'objet, identificateur de l'objet).

```
Algorithme Création_ Base(nom_base);

Début
  Créer les fichiers : nom_base.dsc
                       nom_base.idx
                       nom_base.img
                       nom_base.obj

  Insérer dans le fichier "nom_base.idx" le nœud racine
  avec les valeurs : ln(ρ)=0 et θ = 5π/7 ;
Fin ;
```

**FigIV.15: Création de la base**

### III.1.3) Insertion d'un objet dans la base :

Pour insérer une image d'un objet dans la base, tout d'abord, il faut la passer sur un traitement de bas niveau qui consiste en l'extraction des contours de ces images (la segmentation) puis, l'approximation polygonale de ces contours. Ensuite, toutes les configurations existantes de segments adjacents sont énumérées et pour chacune d'elles le descripteur associé est calculé (voir §II.4).Et à la fin, on l'insère dans la base.

Cette opération peut être représentée par l'organigramme suivant :

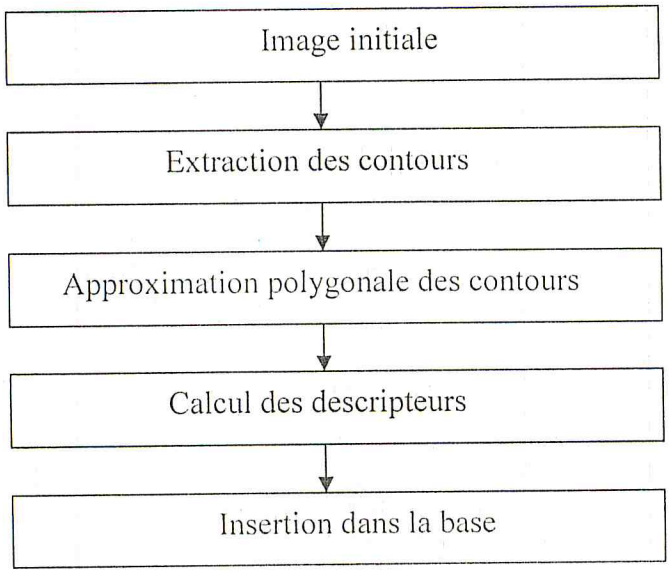



**Fig IV.16 : Traitement de l'image avant l'insertion**

La procédure d'insertion d'un nouveau descripteur dans la base débute par la racine, et descend à travers les niveaux de l'arbre.

- Si à la fin, le nœud ne pointe vers aucune feuille, une feuille est créée et le descripteur y est inséré.
- Si le nœud pointe vers une feuille, le descripteur y est inséré.
- Si cette feuille est pleine, on procède à son éclatement : le nœud père de cette feuille va pointer vers un nouveau nœud, et tout les éléments de cette feuille seront insérés à partir de ce nouveau nœud. Une propriété intéressante du Quad-tree, est que les éclatements ne sont pas récursifs, une feuille pleine sera remplacée par nœud pointant vers quatre nouvelles feuilles sans toucher au nœud père de l'ancienne feuille.

```
Algorithme Insertion_Objet_Base;

Début
  Pour chaque objet faire
  Debut
     Insérer dans le fichier "nom_base.obj" le couple
       (nom_objet,numéro_objet) ;
     pour chaque image Im faire
     début
          Insérer dans le fichier "nom_base.img" le couple
             (numéro_image,numéro_objet) ;
          Segmenter (Im);
          Σ=Ensemble des configurations de segments adjacents de
             Im ;
          pour chaque configuration c∈Σ faire
          début
               Calcul_Descripteur(c) ;
               Insérer dans l'arbre le descripteur calculé ;
          fin ;
     fin ;
  fin ;
Fin ;
```

FigIV.17: Insertion d'objet dans la base

```
Algorithme Insertion_Arbre(Descripteur);

Début
  Nœud:= Racine ;
  Tant que((Nœud ≠ NIL)et(Nœud ≠ Feuille))faire Avancer dans
                                                  l'arbre ;
  Si (Nœud = Feuille) alors
    Début
          si (Feuille non pleine) alors Insérer(Descripteur) ;
          sinon
               début
               Eclatement(Feuille) ;
               Insérer(Descripteur)
               fin ;
    fin
  sinon          /* Nœud = NIL */
        début
             Allouer_Espace(Nouvelle_Feuille) ;
             Insérer(Descripteur) ;
        Fin ;
Fin ;
```

FigIV.18: Insertion des descripteurs

En cas d'éclatement d'une feuille, pour diviser le rectangle associé à elle, la manière la plus simple est de le diviser en quatre rectangles de même surface, mais ceci engendre une mauvaise distribution des descripteurs qui peut même provoquer un nouveau éclatement pour un de ces nouveaux rectangles si les données y sont fortement concentrés. Donc, il faut découper le rectangle de telle manière à avoir une distribution assez équilibrée, pour cela on a opté pour l'utilisation du centre de gravité des données comme centre du découpage.

L'efficacité de ce choix est visible dans le graphe suivant, où nous avons calculé le nombre de nœuds de la base en variant le nombre total de descripteurs, et ceci pour les deux types de découpage. Le nombre de feuilles de la base varie pratiquement de la même manière.

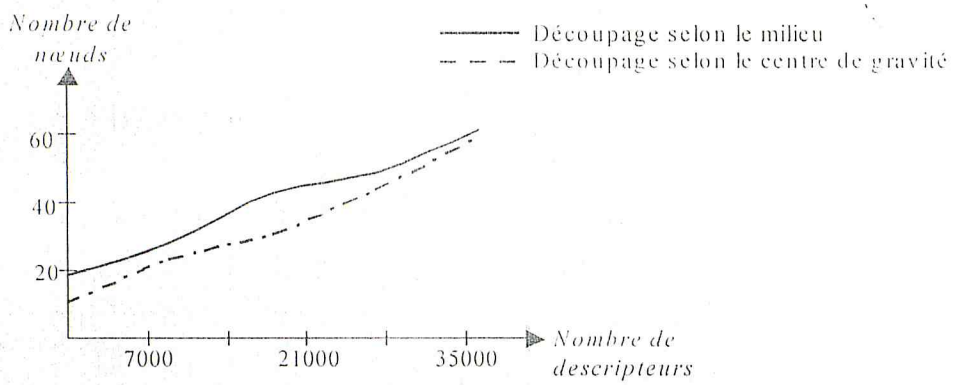



**Fig IV.19 : Variation du nombre de nœuds pour les deux types de découpage des rectangles du quad-tree**

## III.2) Reconnaissance :

L'objectif de l'algorithme de la reconnaissance est de pouvoir comparer une image inconnue à un ensemble d'images, et puisque cet ensemble peut être volumineux, la comparaison ne peut pas s'effectuer image par image, alors il faut trouver un moyen qui permet de traiter les images de la base toutes en même temps, et non les unes après les autres.

Tout d'abord l'image inconnue subit le même traitement qu'ont subi les images de la base, c'est à dire l'extraction des contours puis leurs approximation par des segments, suivies par le calcul des descripteurs de paires de segments adjacents (voir §II.4).

Ensuite, pour chaque descripteur de l'image requête, on détermine leurs descripteurs voisins, c'est à dire les descripteurs appartenant à la base se trouvant dans un rayon donné (L'algorithme de recherche sera présenté dans la prochaine section). Les images auxquelles appartiennent ces descripteurs sont toutes candidates pour former des appariements plausibles avec l'image inconnue. Pour trancher, on calcule les paramètres des similitudes existantes entre chaque descripteur de l'image inconnue et ses voisins appartenant à la base, les bons appariements vont donner pratiquement les mêmes similitudes. Cette étape de vote permet de trouver les objets les plus semblables de l'image requête (cette étape de vote sera reprise en détails dans §III.2.2).

```
Algorithme Reconnaissance(Image_Inconnue);

Début
  Segmenter(Image_Inconnue);
  Σ=Ensemble des configurations de segments adjacents de
     Image_Inconnue ; /* étapes effectuées au préalable */
  pour chaque configuration c∈Σ faire
     début
          D1=Calcul_Descripteur(c) ;
          Liste = Recherche_Voisins(D1) ;
          pour chaque descripteur D2∈Liste faire
               début
                    Calculer les paramètres de la similitude
                       qui projette D1 sur D2 ;
               fin ;
     fin
  Choisir l'image dont les paramètres des similitudes associées
     à elle, forment l'amas le plus dense ;  /* l'étape de vote
     */
Fin ;
```

FigIV.20: Reconnaissance

### III.2.1) Recherche des voisins dans le Quad-tree :

La recherche des voisins du descripteur requête se rapproche à celle de la recherche des K plus proche voisins (kppv) d'un point dans un espace multidimensionnel. Donc nous ne cherchons pas un nombre fixe de voisins, mais pour ceux qui se trouvent à une certaine distance du descripteur requête (La valeur de ce rayon de recherche sera déterminée expérimentalement dans le chapitre V).

Logiquement, la zone de recherche devrait être une ellipse, l'algorithme parcourt l'arbre et détermine les rectangles englobés totalement ou partiellement par cet ellipse, et à la fin, un accès séquentiel aux feuilles correspondantes déterminera les descripteurs répondant à la zone requête. Mais puisque la recherche des rectangles englobés s'avère être plus simple et surtout plus rapide avec un rectangle qu'avec une ellipse, alors l'ellipse sera remplacée par un rectangle de recherche ; et à la fin, on détermine parmi les descripteurs retournés ceux qui se trouvent dans cette ellipse (cf. figure IV.16).

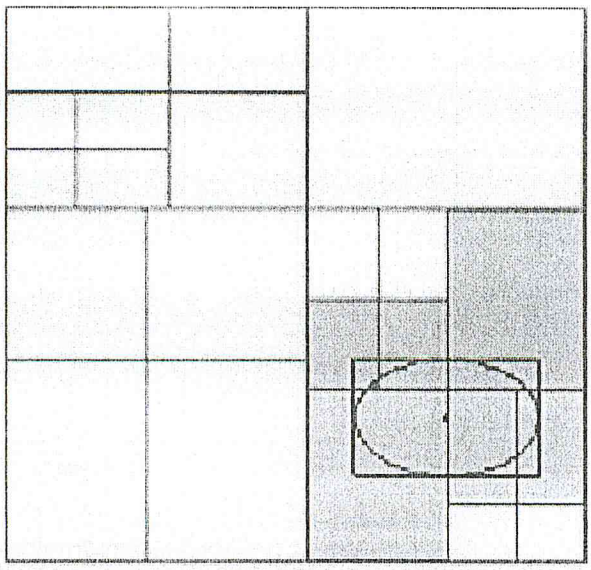


**Fig IV.21 : Le remplacement de l'ellipse de recherche par un rectangle**

```
Algorithme Recherche_Voisins(Nœud ; MinX, MinY, MaxX, MaxY) ;

   Entrée :
    Nœud : Le nœud à partir duquel commencer la recherche
    MinX, MinY, MaxX, MaxY : Les coordonnées du rectangle de
       recherche
   Sortie : Les voisins trouvés seront insérés en fin de la Liste

Début
   Tant que ((Nœud ≠ NIL)et(Nœud ≠ Feuille)et
   (le rectangle est inclus totalement dans un seul fils)) faire
   Avancer dans l'arbre ;

   Si (Nœud = Feuille) alors Inclure dans Liste tout les
                             descripteurs de Nœud se trouvant à
                             rayon donné du descripteur requête ;
   Sinon
        Si (Nœud = NIL) alors Fin de la procédure
        Sinon    /* le rectangle est réparti entre plusieurs fils
        */
                 Début
                 Pour chaque intersection du rectangle de recherche
                 avec un fils faire lancer la procédure
                 Recherche_Voisins avec comme paramètres les cordonnées
                 de l'intersection et ce fils comme nœud ;
                 fin ;
Fin ;
```

**FigIV.22: Recherche des voisins d'un descripteur dans un Quad-tree**

Cette procédure est lancée au départ avec comme paramètre la racine (nœud) et les coordonnées du rectangle de recherche dont le centre est le descripteur requête. Cette procédure descend dans les niveaux de l'arbre tant que le rectangle de recherche est inclu totalement dans le rectangle associé au nœud. De là, si ce rectangle est réparti entre plusieurs fils la procédure est relancée pour chacun de ces fils. En arrivant aux feuilles, les descripteurs se trouvant dans les feuilles concernées vont subir une comparaison pour déterminer ceux qui se trouvent dans l'ellipse de recherche.

Après avoir déterminé les descripteurs voisins au descripteur requête, on calcule les similitudes existantes (voir §II.2.4).

Ainsi, on obtient la liste de toutes les similitudes existantes entre les configurations de l'image inconnue avec celles de quelques images de la base.

**III.2.2) Vote :**

La phase de vote est la partie la plus importante de l'algorithme de reconnaissance. Elle permet d'éliminer les mises en correspondance erronées parmi celles trouvées lors de l'étape précédente.
Elle s'appuie sur le fait que la théorie des quasi-invariants permet d'affirmer que localement, et pour des changements de points de vue modérés, le mouvement apparent entre deux images s'approche raisonnablement d'une similitude.

A la fin de l'étape précédente, une liste contenant des identificateurs d'images et les similitudes existantes entre chacune de ces images et l'image inconnue, a été établie. Pour sélectionner les images des objets contenus dans l'image requête, on recherche pour chacune l'amas le plus dense de paramètres de similitudes associées à elle dans l'espace de représentation de ces paramètres, et on établie un ordre décroissant de la densité de l'amas trouvé. Les modèles élus seront évidemment ceux associés aux amas les plus denses.

Nous avons choisi la technique d'indexation qui est le VA-File (Voir Chapitre III §II.6) pour implémenter cette étape, puisque dans le vote, des vecteurs de réels imprécis (paramètres de similitudes) sont distribués sur des hypercubes. Donc, on associe à chaque image un espace de représentation des paramètres des similitudes (inclus dans $\mathbb{R}^4$, où chaque dimension correspond à un des quatre paramètres de similitude ((**k**) l'échelle de l'homothétie, ($\alpha$) l'angle de rotation, ($x$ et $y$) Les composantes de la translation). Cet espace sera découpé en $2^b$ hypercubes, où chacune des quatre dimensions de cet espace est découpée en $2^{b_i}$ intervalles, avec $b = \sum_{i=1}^{4} b_i$ (Ces $b_i$ seront déterminés dans le chapitre V).

```
Algorithme Vote();

Début
  Découper l'espace des similitudes en 2^b hypercubes;

  Pour chaque image Im faire
      Début
      Liste:= vide;
      Pour chaque similitude S de Im faire
          Début
          Déterminer l'hypercube auquel appartient S;
          Pour cet hypercube et chacun de ses voisins faire
              Début
              Si le numéro de l'hypercube appartient à Liste
                  Alors
                      incrémenter le compteur associé à ce numéro
                  Sinon
                      insérer ce numéro dans Liste avec 1 comme
                      valeur de compteur;
              fin;
          fin;
      Associer à Im la valeur maximum des compteurs;
      fin;
  Classer les images suivant l'ordre décroissant de leurs valeurs
  maximums;
Fin.
```

FigIV.23: Vote

Pour chaque image, on associe une liste initialement vide qui va contenir les numéros des hypercubes et leurs compteurs correspondants.

Pour chaque similitude, on déterminera le numéro de l'intervalle de la $i^{ème}$ dimension contenant sa $i^{ème}$ composante, et en concaténant ces numéros (en binaire), on obtient le numéro de l'hypercube qui englobe les paramètres de cette similitude. Si ce numéro appartient déjà à la liste associée au début à l'image, on incrémente son compteur, sinon on l'insère en initialisant son compteur à 1.

La mise à jour des compteurs qui est effectuée à chaque vote permet de connaître à chaque instant le point d'accumulation maximale dans chacun des espaces de vote des images de la base, ainsi que l'image ayant la plus grosse accumulation et qui est donc l'image la plus proche de l'image inconnue.

On utilisera encore une table de hachage pour les numéros d'hypercubes et leurs compteurs, pour éviter d'y accéder d'une manière séquentielle, afin d'accélérer l'accès. Ainsi ces couples seront distribués sur plusieurs listes.

A la fin, cet algorithme retourne comme résultat une liste des images ayant des similitudes avec la requête, triée selon l'ordre décroissant du nombre maximum de vote.

## IV) Conclusion :

Dans ce chapitre, nous avons présenté les différentes étapes de l'algorithme de reconnaissance que nous avons choisi (la structure d'indexation, la création de la base, l'insertion d'objet dans la base, la recherche des voisins d'un descripteur, et enfin le vote).

Cet algorithme nécessite l'utilisation des images segmentées en contours, les contours eux mêmes étant approchés par des segments de droite.

Puis ces images sont modélisées par un ensemble de configurations de couples de segments adjacents.

Ces configurations sont caractérisées par des quasi-invariants, qui sont l'angle formé par les deux segments ($\theta$) et leurs rapport du langueur ($\rho$). Ce qui leur permet d'être indexées d'une manière à donner la possibilité de déterminer les images qui peuvent être des candidates pour la reconnaissance.

Après ce premier tri qui donne comme résultat les images candidates pour la reconnaissance, d'autre étape permet de filtrer les appariements erronés de ces images, c'est l'étape de vote, qui ressemble à la technique d'indexation le VA-File, et qui consiste à éliminer les mises en correspondance initiales erronées, puis classe les objets en fonction de leur ressemblance avec l'image requête.

Dans le chapitre suivant, nous allons réaliser des tests, pour montrer l'efficacité de cet algorithme.

# *Chapitre V*

# *Evaluation et application développée*

# *Chapitre V*

# *Evaluations et application développée*

## I) Introduction

Le développement des logiciels a beaucoup évolué et dispose à présent d'outils performants. Notamment, depuis l'apparition et l'expansion de l'environnement Windows, qui offre à l'utilisateur une interface conviviale et facile d'emploi.

Avant de présenter notre logiciel, nous allons tout d'abord exposer les résultats des évaluations de l'algorithme de reconnaissance étudié dans le chapitre précédent; ainsi que les statistiques qui nous ont permis de déterminer les valeurs des constantes et des paramètres de cet algorithme.

## II) Nature des données utilisées

Pour le développement de notre application nous avons utilisé un micro-ordinateur de type Pentium 4 doté d'une RAM de capacité 128 Méga octets, et d'un disque dur de 40 Giga octets.

### II.1) Choix du langage de programmation

Nous avons réalisé notre logiciel en utilisant le compilateur C++Builder (version5) sous Widows XP.

Plusieurs raisons justifient notre choix , parmi lesquelles nous citons :

- La puissance du langage C++ pour les applications scientifiques ;
- La facilité de réaliser une interface graphique interactive, contenant des menus, des boîtes de dialogue, ...etc... Ce qui permet d'utiliser le logiciel de maniere très simple .

- Il offre des possibilités de la programmation orientée objet (POO) donnant l'accés à l'environnement Windows grâce à la librairie OWL (Object Windows Library) qu'il intégre . Cette bibliothèque de classes facilite grandement la programmation Windows .

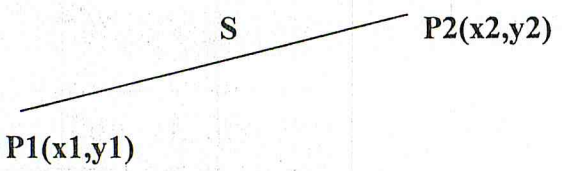
## II.2) Nature des images utilisées

Les évaluations ont porté sur 697 images représentant 24 objets. Ces images ont été générées par un logiciel d'animation 3D **RayDream®**. Les images générées par ce logiciel ont une apparence quasi réelle (une des caractéristiques importantes de ce logiciel) c'est à dire irrégularité des surfaces, des contours pas très nets.

Pour chaque objet, des vues sont prises à des intervalles de 20° sur deux axes de rotation différents. Nous avons supprimé manuellement l'information redondante (à cause de la symétrie de certains objets).

Afin d'uniformiser les couleurs des surfaces (changer les textures en couleurs). Les images utilisées, sont traitées par le logiciel **CorelTrace®** Enfin, les images ainsi obtenues sont traitées par une application qui permet d'extraire les segments de droites suivant les approches d'extraction de contours (les frontières ) .

Le résultat de ce traitement (de bas niveau) est un ensemble de coordonnées des segments (figure V.1) formant l'image d'origine.




avec P1(x1,y1) et P2 (x2,y2) sont les coordonnées d'un segment S .

**Figure V.1 : Forme d'un segment de droite**

# III) Détermination des paramètres de l'algorithme

Dans cette partie, nous avons fait des études statistiques afin de déterminer certains paramètres de l'algorithme de reconnaissance, qui n'ont pas été déterminés dans le chapitre précédent.

## III.1) Nombre de descripteurs par feuille

Pour déterminer cette constante qui a été désignée par ***Taille_Feuille*** dans la présentation de la structure de la feuille (voir Chapitre IV §III.1.1) , nous avons calculé le taux de remplissage des feuilles de la base en fonction du nombre de descripteurs contenus dans une feuille, en variant ce nombre entre 100 et 1000, car au dessous de 100 descripteurs par feuille, la taille de l'index devient trop importante pour la maintenir en mémoire centrale, et au dessus de 1000 descripteurs par feuille, la taille de celle-ci devient énorme par rapport à notre base.

### III.1.1) Influence sur le taux de remplissage

Au début de notre étude, nous avons commencé par insérer toutes les images dans une base (ce qui nous a donné 31957 descripteurs en tout). Nous avons fait varier le nombre de descripteurs contenus dans une feuille entre 100 et 1000 descripteurs, et nous avons déterminé le taux de remplissage de cette base. Le même test est effectué avec la moitié des images (15.000 descripteurs environ) et enfin le dernier test est fait en ne gardant que le tiers (10.000 descripteurs environ).

Les résultats obtenus peuvent être représentés par le schéma suivant:

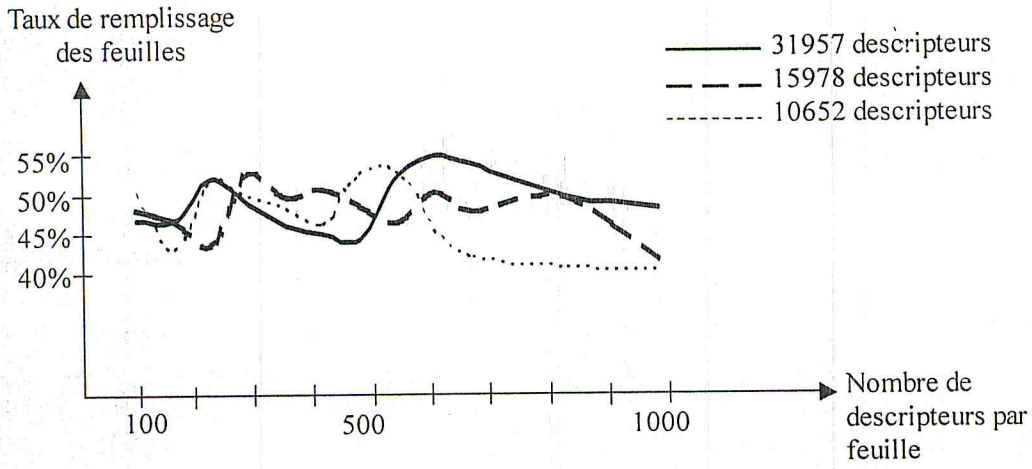



**Figure V.2 : Taux de remplissage des feuilles en fonction du nombre de descripteur par feuille**

Nous remarquons que le taux de remplissage reste moyen entre 45% et 55%. Il commence à décroître quand la taille de la feuille est très grande (avoisine les 1000 descripteurs), surtout lorsque la moitié de la base est utilisée; ce qui veut dire que la taille de la feuille devient trop importante pour le nombre total des descripteurs.

Malgré ceci, la variation de la taille de la feuille n'influe pas clairement sur le taux de remplissage; nous avons donc étudié leur influence sur le temps de réponse d'une recherche.

### III.1.2) Influence sur le temps de réponse d'une recherche

Nous avons refait le même test comme dans l'évaluation précédente, mais cette fois, nous avons testé l'influence de la taille de la feuille sur le temps de réponse (en la variant de 100 à 1000).

Nous avons pris au hasard 100 images (contenant 5627 descripteurs environs) comme requêtes, pour avoir plus de précision dans la mesure de la durée de la recherche. Le test s'est déroulé en trois étapes : utiliser toute la base puis la moitié et enfin le tiers (figure V.3).

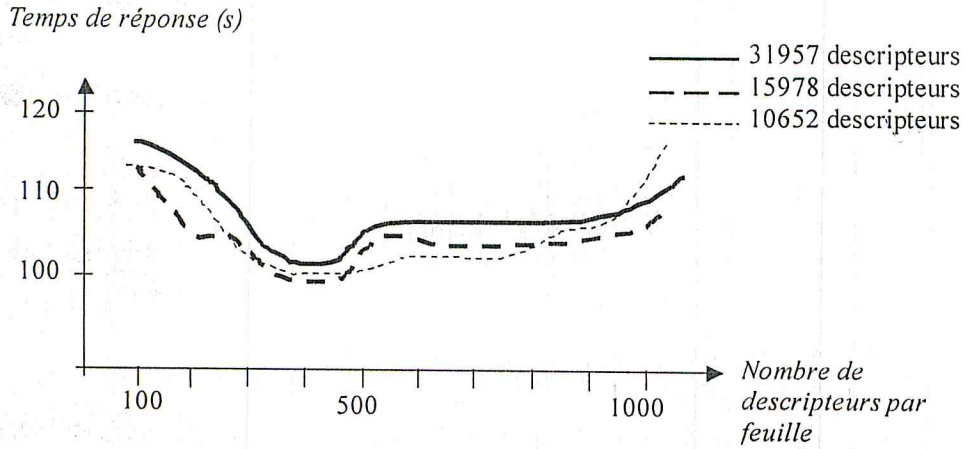



**Figure V.3 : Temps de réponse de la recherche en fonction du nombre de descripteur par feuille**

A partir de ce graphe, nous pouvons remarqué que les trois courbes ont en général un comportement assez pareil.

Nous avons remarqué que le temps de réponse est très grand quand la taille des feuilles est petite (aux environs de 100 descripteurs), il atteint son minimum aux environs de 400, puis entame une montée quand la taille de la feuille atteint de grandes valeurs puisque ici, la recherche s'approche du séquentiel.

Puisque pour des feuilles contenant 400 descripteurs, le temps de réponse atteint son minimum, nous avons donc choisi 400 comme la taille idéale des feuilles pour la recherche.

## III.2) Seuils de comparaison θ et ln(ρ) pour la recherche

Lors de la recherche des voisins du descripteur requête, l'algorithme détermine parmi les descripteurs de la base, ceux qui se trouvent à un certain rayon du descripteur requête (voir Chapitre IV §III.2).

La zone de recherche est un rectangle (au lieu d'une l'ellipse) caractérisé par deux paramètres: un seuil de recherche pour l'angle de la configuration de deux segments adjacents (θ), et un autre pour le logarithme de leur rapport de longueur ($\ln(\rho)$).

Pour déterminer ces deux paramètres, nous avons calculé la différence entre les deux quasi-invariants des mêmes configurations appartenant à des couples d'images successives prises du même d'objet.

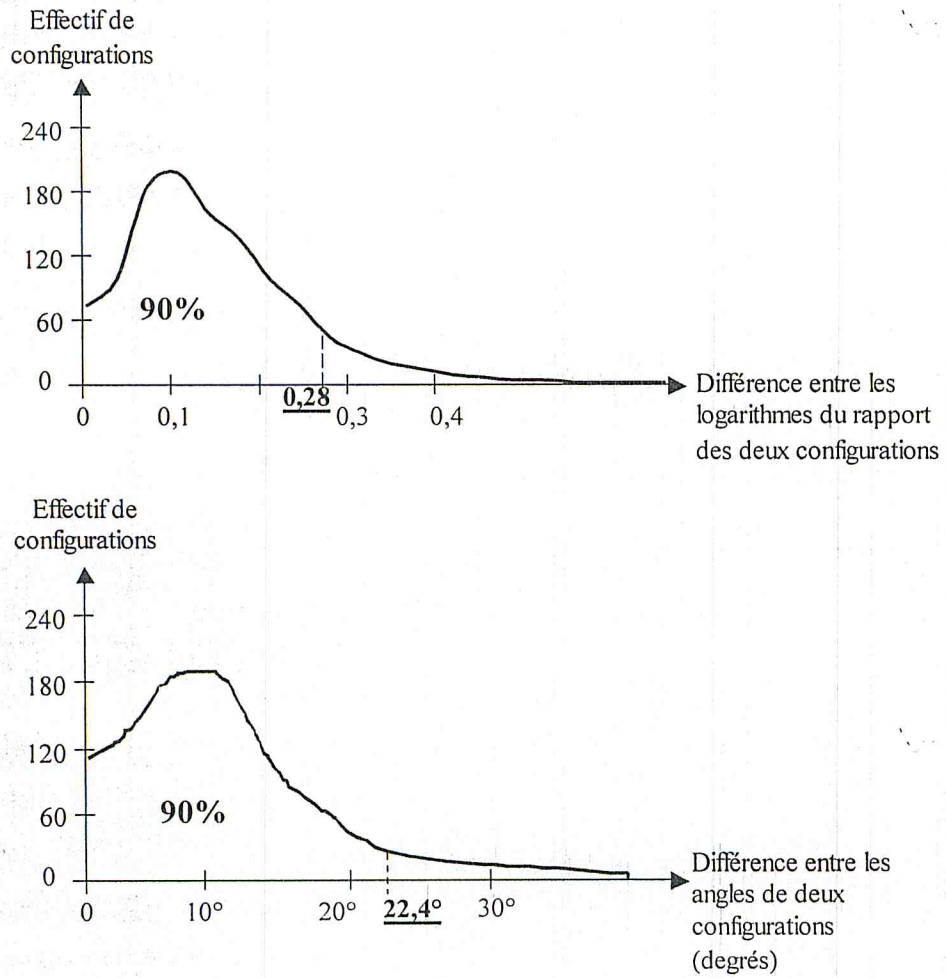



**Figure V.4 : Différence entre les quasi-invariants de même configurations apparentant à des couples d'images successives**

Nous avons pris 20 couples d'images successives; pour chaque couple, nous avons pris les couples de configurations identiques et nous avons calculé la différence entre les deux quasi-invariants (les résultats sont montrés dans les deux graphes ci-dessus). La distance entre les quasi-invariants de 90% des couples de configurations est inférieure à ($\ln(\rho)$=0.28, $\theta$ =22.4).Puisque une image requête va se situer entre deux images successives de la base, alors le seuil de recherche peut être pris comme étant la moitié de la distance entre deux images.

Donc nous avons :

Le seuil de recherche pour $\ln(\rho)=0.28*0.5=0.14$,

Le seuil de recherche pour $\theta=22.4*0.5=11.2$.

## III.3) Paramètres du découpage de l'espace des similitudes

Nous avons déjà vu dans l'algorithme de vote (voir Chapitre IV §III.2.2) que l'espace de représentation des similitudes était découpé en $2^b$ hypercubes, où chacune des quatre dimensions de cet espace était découpé en $2^{b_i}$ intervalles avec $b = \sum_{i=1}^{4} b_i$ .

L'hypothèse sur laquelle est fondée l'algorithme, prévoyait que le mouvement apparent entre deux images peut être assimilé à une similitude si le changement du point de vue reste relativement modéré; ce qui implique que les appariements corrects définiront des similitudes assez proches, tandis que les mises en correspondance erronées définiront des similitudes différentes.

Pour déterminer les valeurs de ces paramètres $b_i$, ainsi que les bornes des intervalles, nous avons fait des études statistiques :

### III.3.1) Détermination des bornes des intervalles de variation

Nous avons étudié la distribution des similitudes existantes entre des images prises au hasard parmi la totalité des images de la base, nous avons alors obtenu 482463 similitudes.

Les distributions de leurs paramètres ((**k**) d'échelle d'homothétie (**α**) d'angle de rotation, des composantes de la translation ($x$ et $y$)) sont données dans les graphes ci-dessous :

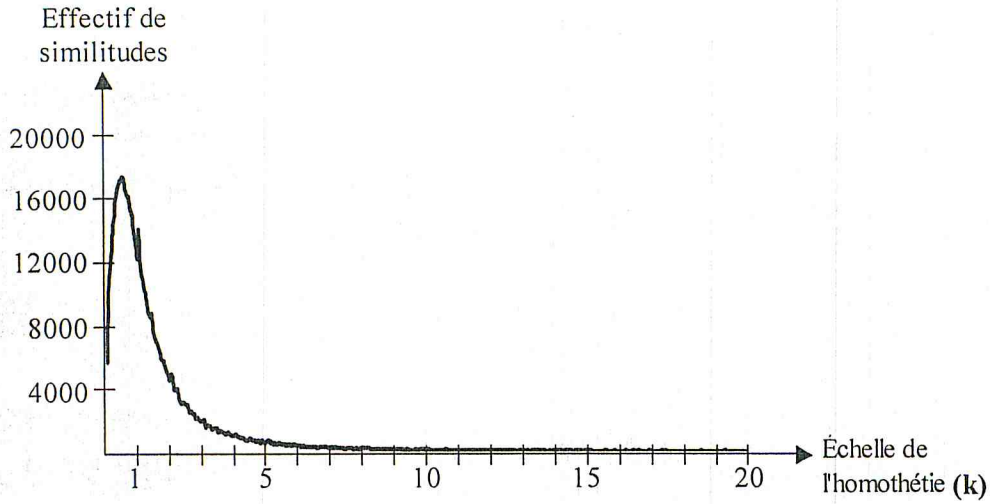



**Figure V.5 : Distribution des similitudes suivant l'échelle de l'homothétie (*k*)**

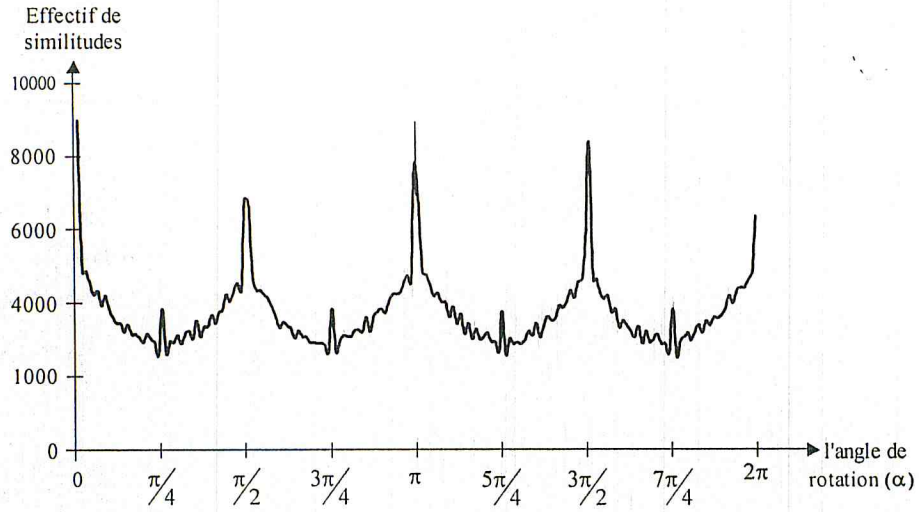



**Figure V.6 : Distribution des similitudes suivant l'angle de rotation (α)**

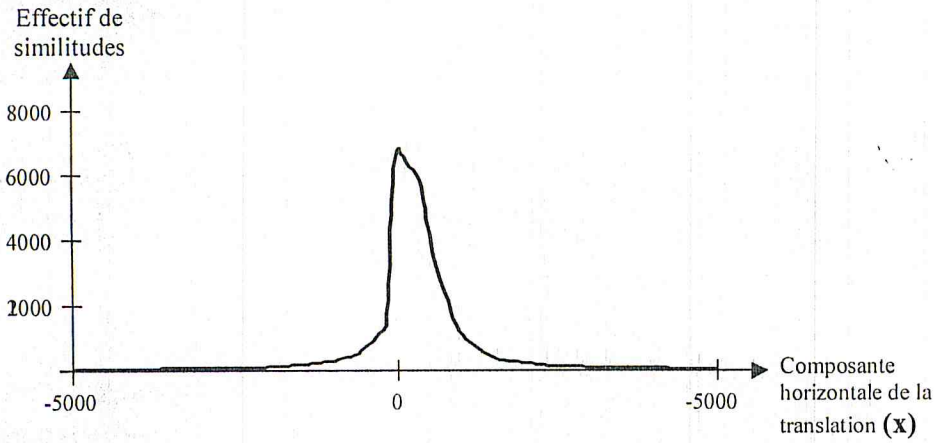



**Figure V.7 : Distribution des similitudes suivant la composante horizontale de la translation (*x*)**

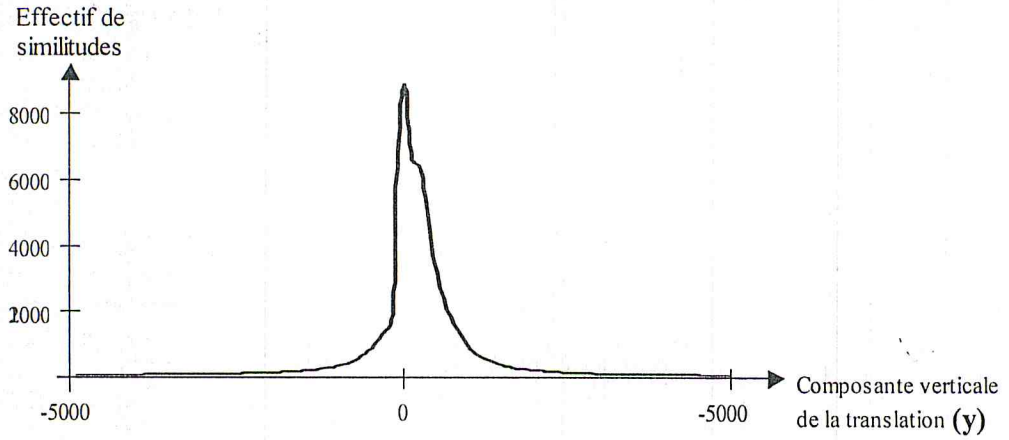



**Figure V.8 : Distribution des similitudes suivant la composante verticale de la translation ($y$)**

A partir de ces graphes, nous remarquons que la distribution des paramètres de similitude n'est pas uniforme.

L'échelle de l'homothétie a le même comportement que le rapport de longueurs des segments $\ln(\rho)$ d'une configuration (voir Chapitre IV §II.3), ce paramètre est alors remplacé par son logarithme népérien, pour rendre la distribution assez symétrique , ce qui va faciliter le découpage en intervalles.

Les composantes de la translation ($x$ et $y$) ont été bornées par les valeurs [-5000, +5000] puisque plus de 99% des valeurs sont contenues dans cet intervalle au-delà alors de ces valeurs, la similitude est assurément fausse.

Le logarithme de l'échelle de l'homothétie a été aussi borné dans l'intervalle [-3, +3] pour les mêmes raisons.

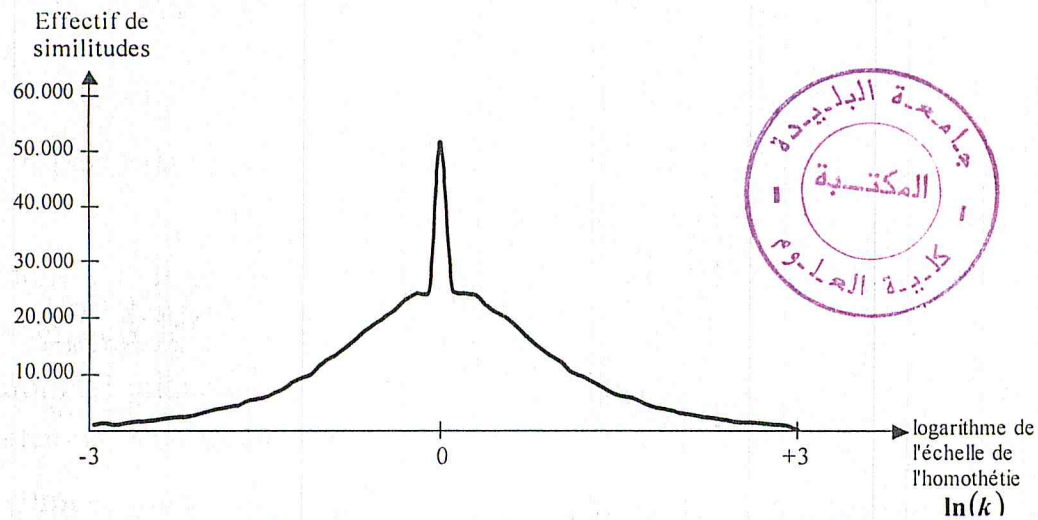





Figure V.9 : Distribution des similitudes suivant le logarithme népérien de l'échelle de l'homothétie ($\ln(k)$)

## III.3.2) Détermination des valeurs de $b_i$

Afin de déterminer le nombre d'intervalles ($2^{b_i}$) pour chaque dimension (pour k, α et x, y), nous avons calculé les similitudes existantes entre les mêmes configurations appartenant à des couples d'images successives; nous avons constaté que les similitudes obtenues pour chaque couple d'images étaient assez proches.

Pour chaque paramètre, nous avons calculé l'écart maximum existant entre deux images successives, et nous avons déterminé les moyennes de ces différences.

En fin de traitement, nous avons calculé le nombre d'intervalles pour chaque paramètre suivant le tableau ci-dessous:

| | Intervalle de variation | Écart moyen | Calcul du $b_i$ | |
|---|---|---|---|---|
| $k$ | [-3, +3] → 6 | 0,11 | $6 / 0.11 = 51.26 = 2^{6,68}$ | $b_k = 6$ |
| α | [0°,360°[ → 360 | 5,10 | $360 / 5,10 = 70.52 = 2^{6,14}$ | $b_\alpha = 6$ |
| $X$ | [-5000,+5000] → 10.000 | 47,10 | $10000/47,10 = 212,30 = 2^{7,73}$ | $b_x = 8$ |
| $y$ | [-5000,+5000] → 10.000 | 48,42 | $10000/48,42 = 206,50 = 2^{7,69}$ | $b_y = 8$ |

**Paramètres statistiques**

## III.3.3) Découpage

Après la détermination des valeurs de $b_i$, et les bornes des intervalles de variation, nous découpons l'intervalle de variation de chaque paramètre de similitude en $2^{b_i}$ intervalles de largeurs égales.

## IV) Description et présentation du logiciel

Le logiciel réalisé, dont la fenêtre principale est représentée par la figure (V.10), utilise les moyens offerts par Windows à savoir :

- Un menu donnant accès à toutes les fonctions du système ;
- Une utilisation indifférente de la souris ou de clavier pour les différentes manipulations;
- Des boîtes de dialogues permettant une interaction facile avec les commandes;
- Une barre de boutons qui reprend sous forme visuelle les opérations les plus utilisées dans le menu.

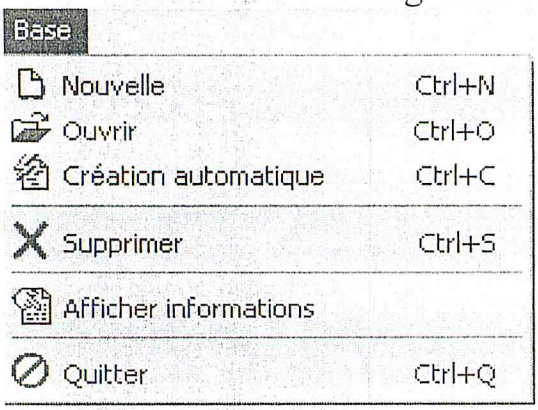

**Figure V.10 : Fenêtre principale du logiciel**

## IV.1) Menus :

Les menus au nombre de cinq, sont présentés dans ce qui suit :

***Base :***

Ce menu concerne la gestion de la base, il porte les commandes suivantes :




**Ouvrir**: lance une boîte de dialogue d'ouverture de fichiers pour ouvrir une base déjà existante;

**Nouvelle**: lance une boite de dialogue de sauvegarde de fichiers pour créer une nouvelle base, si une base déjà existante est sélectionnée, elle sera écrasée.

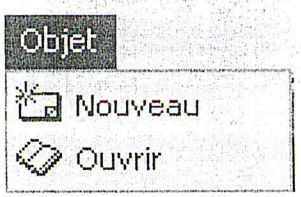
**Création Automatique:** cette commande permet de créer automatiquement une base, son utilité apparaît lors de la création de grandes bases; à partir d'un répertoire sélectionné, tout ses sous-repertoires sont considérés comme des objets, les fichiers contenus par ces derniers formeront les images qui les représentent ;

**Supprimer:** Supprime les fichiers de la base choisie ;

**Afficher Informations:** affiche les informations concernant la base ouverte: le nombre d'objets, le nombre d'images, le nombre de descripteurs, le nombre de nœuds, le nombre de feuilles et le taux de remplissage de la base (le contenu des nœuds et des feuilles peut être rendu sous forme d'un fichier texte).

***Objet :***

Il contient deux commandes seulement :

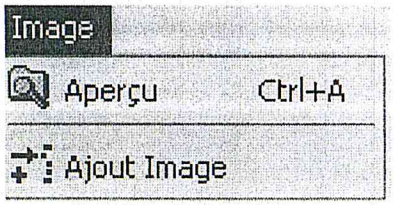


**Nouveau**: crée un nouvel objet dans la base ouverte.

**Ouvrir**: ouvre un objet déjà créé pour y insérer des images.

***Image :***

Il contient aussi deux commandes seulement :

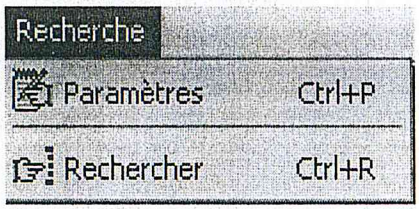



**Aperçu**: permet d'avoir l'aperçu d'une image sélectionnée, en la dessinant à partir de ses coordonnées ;

**Ajout image**: permet d'insérer des images de l'objet ouvert.

***Recherche :***

Il contient les commandes suivantes :




**Paramètres:** Permet de modifier les paramètres de recherche:

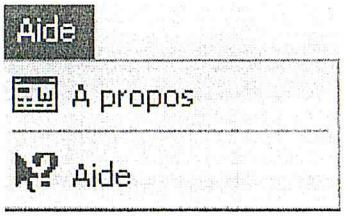
-Les seuils de recherche (pour l'angle et le logarithme du rapport) ;

-Le rapport minimum entre le nombre de descripteurs de la requête et le nombre de similitudes d'une image de la base pour qu'elle soit concernée par le vote ;

-Les composantes de la distance maximum pour l'adjacence de deux segments.

**Rechercher**: recherche l'image sélectionnée, et retourne comme résultat un tableau d'images ayant des similitudes avec cette image, avec les objets auxquels elles appartiennent, et le nombre de vote obtenu.

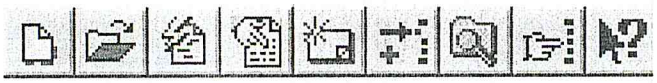
***Aide :*** il contient les deux commandes suivantes :




**Aide** : Permet d'accéder au fichier d'aide ;
**A propos de** : Permet d'afficher la boite de dialogue « A propos ».

## IV.2) Barre des boutons :

La barre des boutons contient les boutons de raccourcis vers les traitements utilisés.



 Pour créer une nouvelle base ;

 Pour ouvrir une base déjà existante ;

 Pour créer automatiquement une base ;

 Pour afficher les informations concernant une base ouverte ;

 Pour créer un nouvel objet dans la base ouverte ;

 Pour insérer des images de l'objet ouvert ;

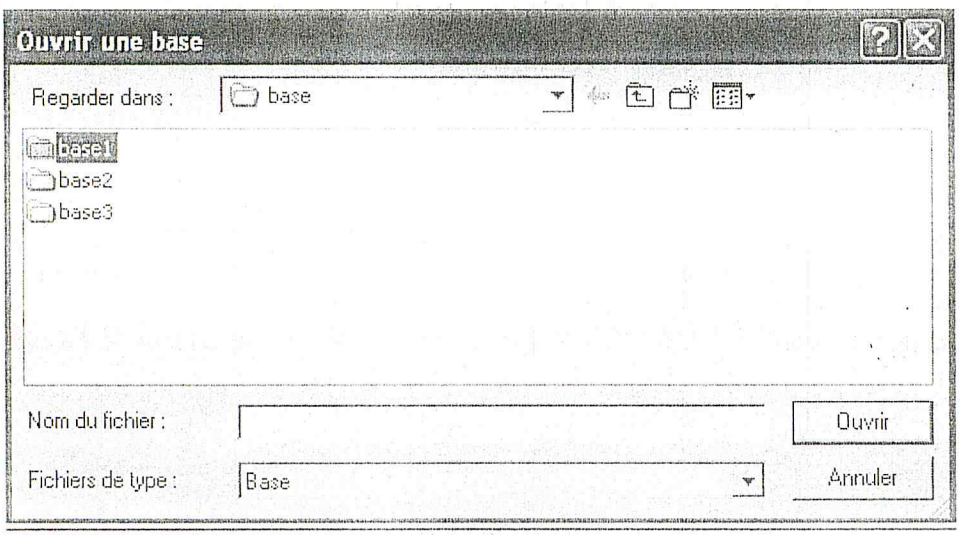
 Pour apercevoir une ou plusieurs images sélectionnées;

 Pour rechercher une image sélectionnée ;

 Pour afficher l'aide.

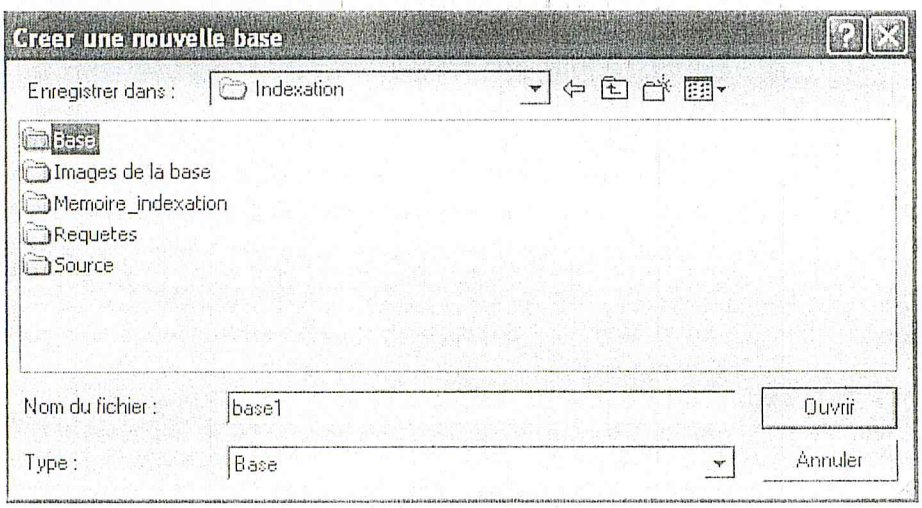
## IV.3) Les boîtes de dialogues

### * Ouvrir une base



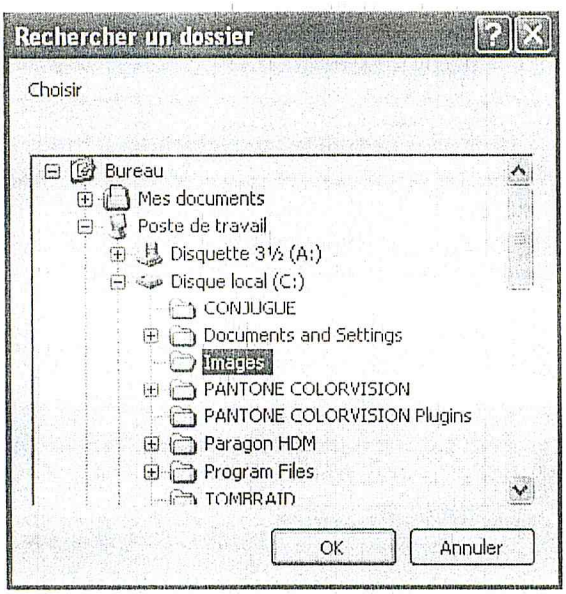

FigV.11 : Ouverture d'une base

### * Créer une base



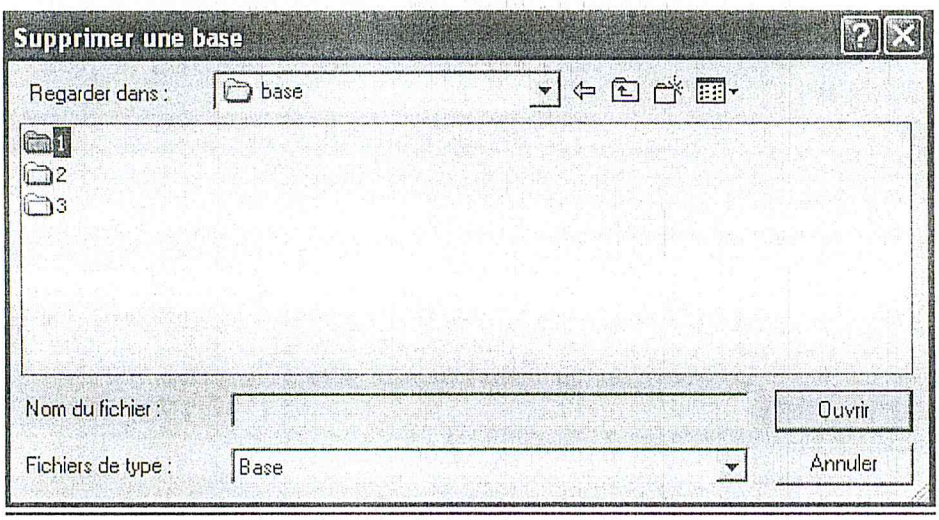

FigV.12 : Création d'une base

## * *Créer une base automatiquement*



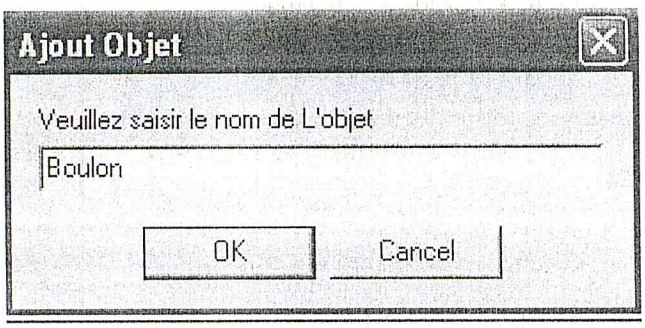

FigV.13 : Création automatique d'une base

## * *Supprimer une base*



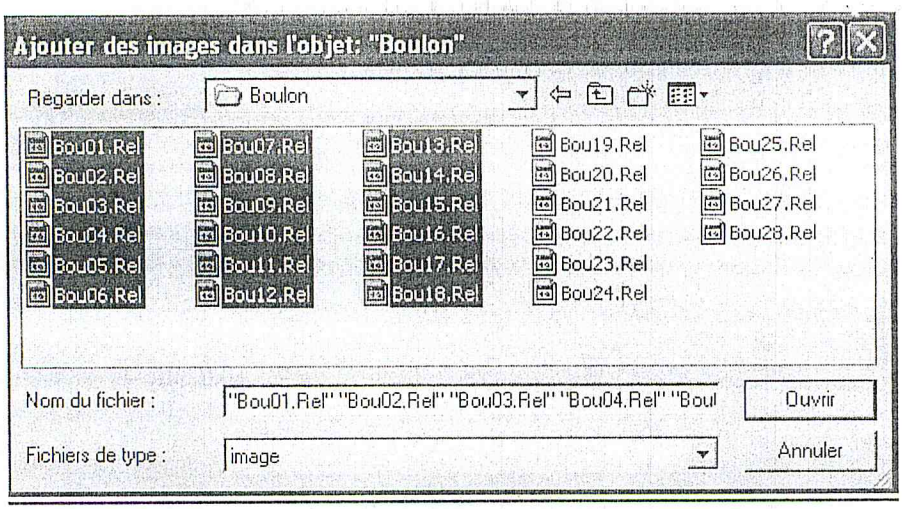

FigV.14 : Suppression d'une base

## * Ajouter un objet



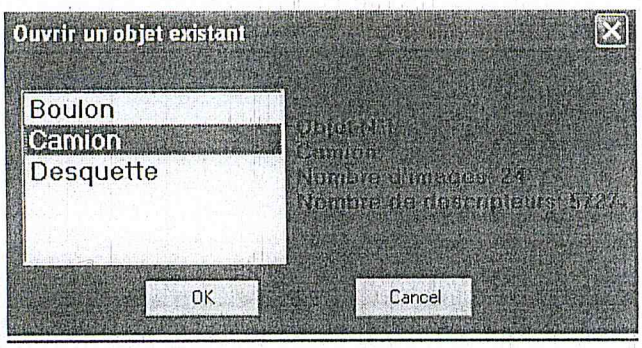

FigV.15 : Ajout d'un objet

## * Ajouter des images dans l'objet ouvert



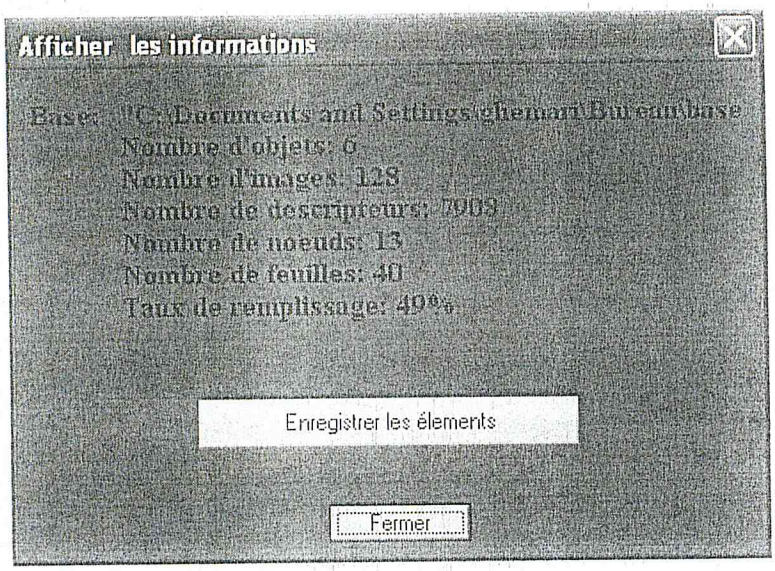

FigV.16 : Ajout des images de l'objet ouvert

## * Ouvrir un objet



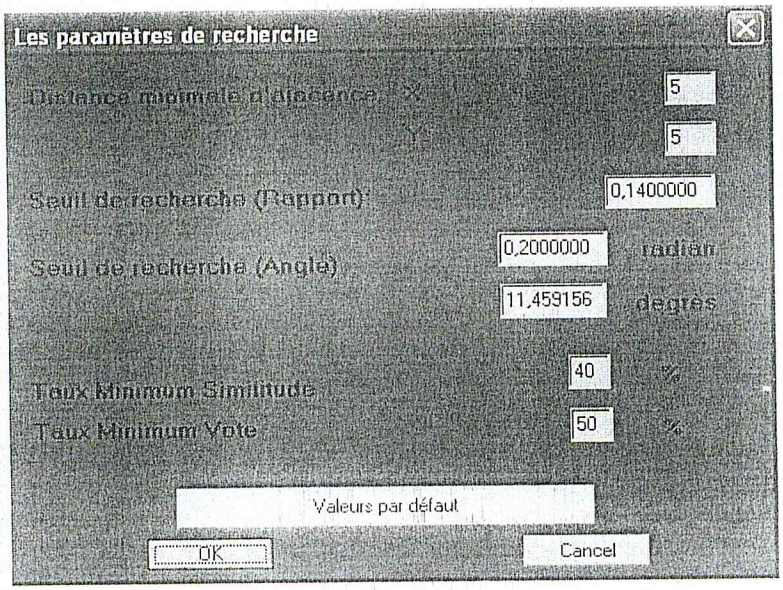

FigV.17 : Ouverture d'un objet

## * Afficher les informations



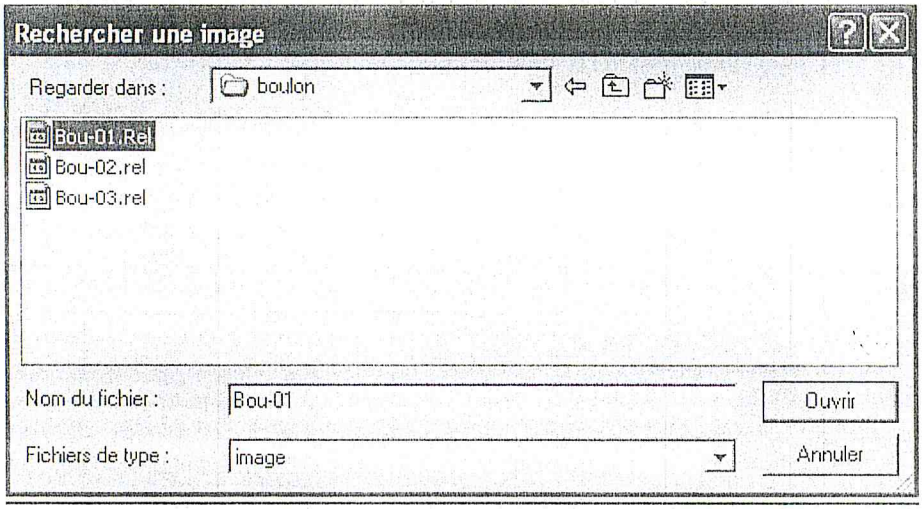

FigV.18 : Affichage d'informations

## *Paramètres de recherche



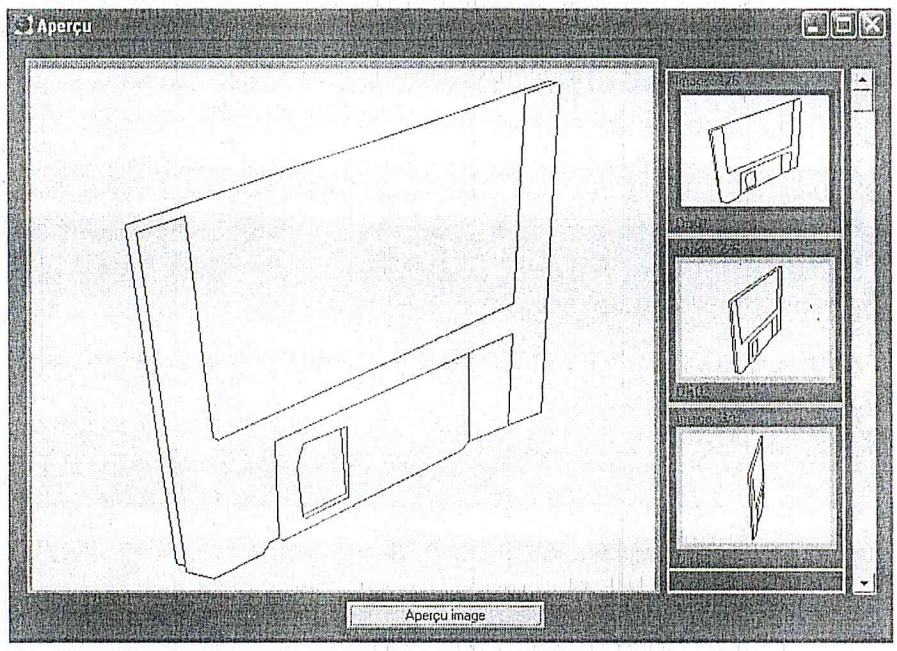

FigV.19 :Paramètres de recherche

## * Rechercher une image




FigV.20 : Recherche d'une image

## * Aperçu




FigV.21 : Aperçu d'une ou plusieurs images

# V) Résultats des tests

Dans cette section, nous avons testé notre application avec des images requêtes qui contiennent en moyenne 50 descripteurs. La recherche dans la base fournissait en moyenne 210 similitudes pour chaque descripteur de la requête.

Tout d'abord, nous avons étudié le taux de reconnaissance, ensuite nous avons étudié le temps de réponse pour la recherche.

## V.1) Taux de reconnaissance

Deux catégories d'images étaient utilisées pour effectuer les tests d'évaluation des taux de reconnaissance: des images faisant partie de la base, et d'autres images caractérisant les objets de la base ne faisant pas partie d'elle.

### V.1.1) Taux de reconnaissance pour les images de la base

Pour tester le taux de reconnaissance des images apparentant à la base. Nous avons crée une base qui contient la totalité des images. Puis nous avons choisi au hasard pour chaque objet, quatre images comme requête.

Les résultats obtenus pour cette catégorie d'images, montrent que la quasi totalité des images ont été reconnues (taux de reconnaissance estimé à 98 %). Les rares cas d'échec sont expliqués par le fait que les images requêtes contenaient des vues critiques des objets qu'elles représentent. Et pour l'objet qui détient la deuxième position dans le vote, nous obtenons en moyenne 18 % du nombre de vote de l'objet reconnu.

### V.1.2) Taux de reconnaissance pour les images exclues de la base

Tout d'abord, nous avons commencé par construire pour chaque objet, deux nouvelles images à partir de nouvelles prises de vue (de la façon même qu'ont été construites les images de la base, voir §II), nous nous sommes assurées que ces prises de vue sont différentes de celles utilisées pour modéliser l'objet ; ce dernier subit de plus une rotation et un changement de la taille (effectué par le logiciel d'animation 3D).

Pour cette catégorie d'images, le taux de reconnaissance est estimé à 62%.

En cas d'échec, l'objet auquel appartient l'image requête réellement, engendre un nombre de vote équivalent à 58% de celui obtenu par l'objet reconnu, et en cas de succès de la reconnaissance, le second objet dans le classement génère 22% du nombre de vote de l'objet reconnu.

En cas de reconnaissance, nous avons remarqué que la majorité des objets reconnus (90%) obtiennent un vote supérieur à 50% du nombre de descripteurs de l'image requête, et la majorité des objets qui détiennent la seconde place (98%), obtiennent un nombre de vote inférieur à 50% du nombre de descripteurs de la requête. Pour cela, nous avons considéré ce pourcentage comme un seuil minimal pour qu'un objet soit reconnu, c'est à dire, si l'image la plus ressemblante à la requête obtient un nombre de vote inférieur à 50% du nombre de descripteurs de la requête, alors l'algorithme considère qu'aucun objet n'a été reconnu.

Mais dans notre application nous avons affiché tous les résultats de la recherche d'une image requête même pour un nombre de vote inférieure à 50% (le seuil de recherche), pour montrer comment l'opération de vote classe les images.

## V.2) Temps de réponse

Dans cette section, nous allons présenter les temps de réponse à une requête donnée. Pour ce test, le fait que l'image appartient ou pas à la base n'est pas important, car l'algorithme procède de la même façon.

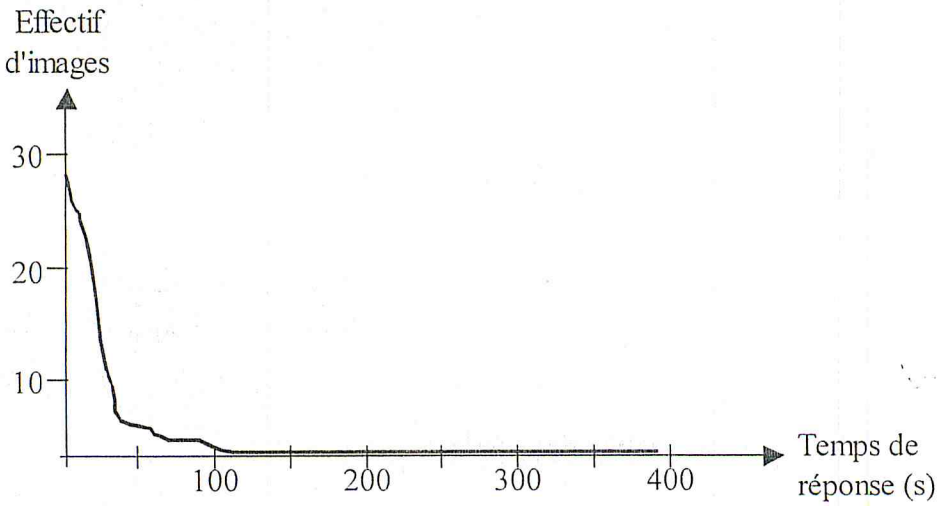



**Figure V.22: Distribution des images suivant leurs temps de réponse**

Nous avons trouvé que les temps de réponse obtenus varient entre moins d'une seconde et 360 secondes. Plus de 45% des images engendrent un temps inférieur à 10s. Le temps moyen est estimé à 22,68s. La quasi totalité de ce temps (93,8%) est utilisée par l'étape de vote (en moyenne 22,09s). Le calcul de descripteurs de la requête, la recherche dans la base et le calcul des similitudes ne durent en moyenne que 1,28s (6% du temps total); et ce temps-là, est utilisé dans sa majorité pour le calcul des similitudes; le calcul des descripteurs et la recherche ne durent qu'une fraction de seconde (l'estimation de la durée était impossible à cause de son infime valeur).

Nous avons remarqué aussi que le temps de vote est en rapport avec le nombre de similitudes trouvées avec les images de la base. Nous avons estimé ce rapport à 0,144% (0,144s pour 100 similitudes).

## VI) Conclusion

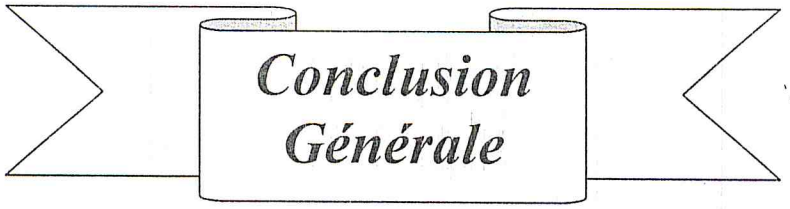
Dans ce chapitre, nous avons présenté les résultats des tests de l'algorithme de reconnaissance, ainsi que les statistiques qui nous ont permis de déterminer les valeurs des constantes et des paramètres de cet algorithme.

Ces valeurs restent propres aux images que nous avons utilisé.

Les tests effectués ont montré que la méthode d'indexation que nous avons utilisé est acceptable et exploitable dans la mesure où la segmentation est suffisamment précise.

Nous avons remarqué que dans la majorité des cas, la reconnaissance était formelle puisque le second objet obtient un vote de 23% en moyenne du vote de l'objet reconnu, et que pour des objets complexes, les temps de réponses augmentent très nettement. Nous avons remarqué aussi que l'étape de vote détient la majeure partie de la durée de recherche malgré sa simplicité.

# *Conclusion Générale*




Dans ce mémoire, nous avons fait une étude bibliographique sur les principales méthodes d'indexation existantes destinées au domaine de la reconnaissance d'objets par ordinateur.

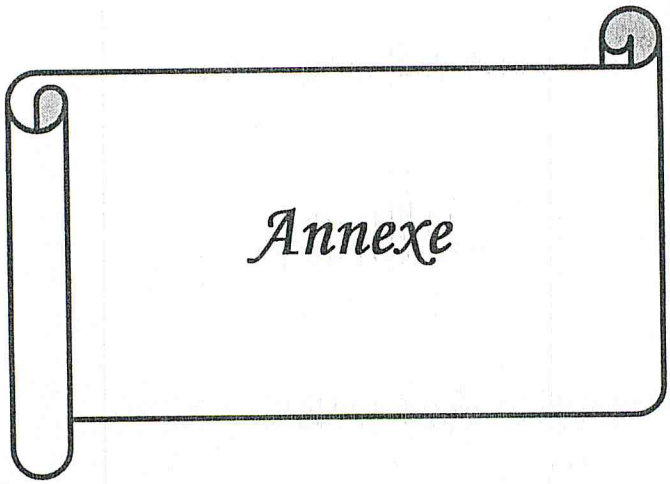
Ensuite, nous avons traité la méthode ***d'indexation géométrique étendue*** avec beaucoup détails. Cette méthode nécessite l'utilisation des images segmentées qui sont modélisées par des configurations géométriques (couples de segments adjacents). Ces configurations ont été caractérisées par des quasi-invariants (l'angle formé par un couple de segments adjacents et leur rapport de longueurs). Ces derniers représentent les descripteurs locaux qui caractérisent les images en les constituant dans la structure d'indexation ***quad-tree*** puisqu'elle est la mieux adaptée aux index bidimensionnels , car elle permet un accès rapide aux données.

La seule hypothèse émise est que chaque objet doit être modélisé par un ensemble d'images dont le changement de point de vue doit être assez faible. La recherche des ***quasi-invariants*** de la base proches à ceux de la requête permet de faire une mise en correspondance initiale assez grossière entre l'image requête et celles de la base. Ces mises en correspondance sont ensuite validées ou rejetées par l'algorithme de vote, dans lequel nous avons une technique d'indexation basée sur le *VA-File*, puisqu'elle permet d'éliminer les mises en correspondance initiales erronés . et enfin de classer les objets en fonction de leur ressemblance avec l'image requête (triée selon l'ordre décroissant du nombre maximum de vote).

Nous avons implémenté notre algorithme, et nous avons vérifié son efficacité par des tests effectués sur une base d'objets de taille moyenne. Les résultats montrent que le taux de reconnaissance était assez acceptable, mais le temps de réponse augmentait très sensiblement pour des objets complexes.

Comme perspectives à notre travail, nous proposons d'améliorer les algorithmes de reconnaissance, que ce soit dans le domaine des bases données ou bien dans le domaine du traitement des images, par exemple:

- L'amélioration de la qualité de la segmentation des images, puisque c'est sur cette base que les algorithmes débutent.
- L'utilisation d'autres types de descripteur, que se soit des descripteurs géométriques ou photométriques.
- L'amélioration des structures de données destinées aux stockages de grands vecteurs imprécis, pour garantir l'accès rapide aux données .
- L'utilisation des objets plus complexes.


Annexe

# *Annexe :*

## *Exemples de l'application*

Dans cet annexe, nous allons exposer quelques exemples d'utilisation de l'application pour montrer leur efficacité.

## I) Etapes de l'application :

Pour utiliser notre application, il faut suivre les étapes suivantes :

### I.1) La création d'une base :

Une base peut être créée soit manuellement, ou bien automatiquement :

#### I.1.1) Manuellement :

En exécutent dans le menu principal l'option « ***Base / Nouvelle*** » ou bien à partir du clavier en lancent la commande « ***Ctrl*** + *N* », une boite de dialogue de sauvegarde de fichiers est ouverte , elle permet de saisir le nom de la base . Ensuite, il faut insérer de nouveau objets , en lançant « ***Objet / Nouveau*** ». Le nom de l'objet est saisi dans un champs, et après validation, une boite de dialogue permet de sélectionner les images de cet objet.
Et pour ajouter des images à un objet déjà existant, il suffit de l'ouvrir « ***Objet / Ouvrir*** », puis de sélectionner les images à ajouter en exécutant « ***Image / Ajouter image*** ».

#### I.1.2) Automatiquement :

Cette option est très utile lorsque le nombre d'objets de la base est très important. En exécutant dans le menu principale « ***Base / Création automatique*** », ou bien à partir du clavier en lancent la commande « ***Ctrl*** + *C* », elle lance une boite de dialogue de sauvegarde de fichier pour saisir le nom de la base (comme pour une base normale) . Mais, au lieu de faire insérer les objets un par un, il suffit juste de choisir un répertoire qui contient des sous-répertoires représentent les objets, et les images sont contenues dans ces derniers .

### I.1.3) Afficher les informations :

Après la création d'une base, l'option « ***Base / afficher information*** » permet d'afficher les informations (nombre d'objet, d'images, de descripteurs, de nœuds, de feuille , et le taux de remplissage de la base ) concernant la base ouverte .

Le bouton « ***Enregistrer les éléments*** » permet d'avoir le contenu de tous les nœuds et les feuilles de la base , sous la forme d'un fichier texte.

### I.2) La recherche d'une image :

Après avoir ouvrir une base déjà existante soit par le baisais de l'option « ***Base / Ouvert*** », ou bien soit à partir de la commande du clavier « ***Ctrl + O*** ». Ou bien après avoir créer une nouvelle base, la recherche d'une image requête est lancée par « Recherche / Rechercher », ou bien par «***Ctrl + R*** », une boite d'ouverture de fichier apparaît, pour sélectionner l'image requête.

Le résultat de la recherche est sous forme d'un tableau, contenant les objets classés suivant leur degré de ressemblance avec l'image requête.
Donc ce tableau retourne les objets classés suivant l'ordre décroissant du nombre de vote maximum obtenu. Avec, chaque ligne contient le nom d'objet , le nombre maximum de vote , le nombre total de vote , et le numéro de l'image qui a obtenu le maximum de vote .

Le bouton « ***Voir Détails*** » permet d'avoir les résultats de vote pour chaque image, ces images sont classées selon leur ordre décroissant de leur résultat de vote . Dans chaque ligne , on a le numéro de l'image , ainsi que son numéro par rapport à l'objet qu'elle représente et son nombre de vote .

# II) Exemples :

## II.1) Exemple.1 :

Exemple d'une base contient 4 objets : boulon, camion, disquette et table.

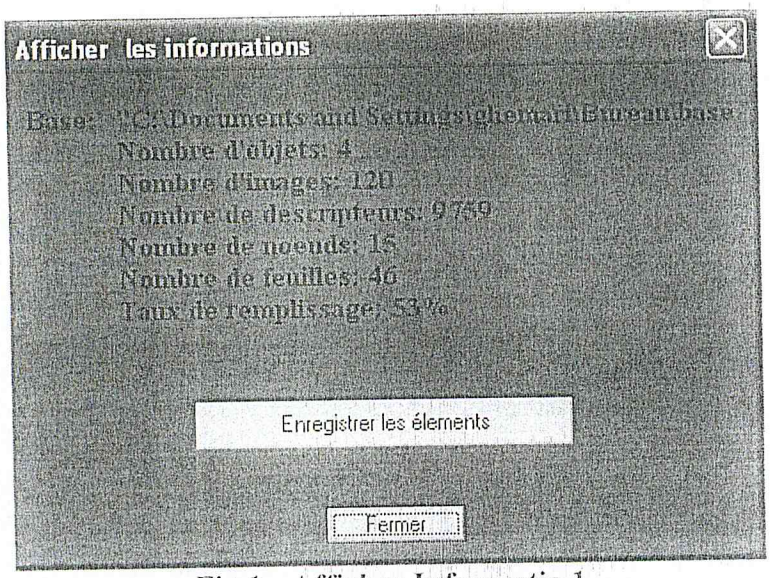



**Fig.1 : Afficher Informatio.1**

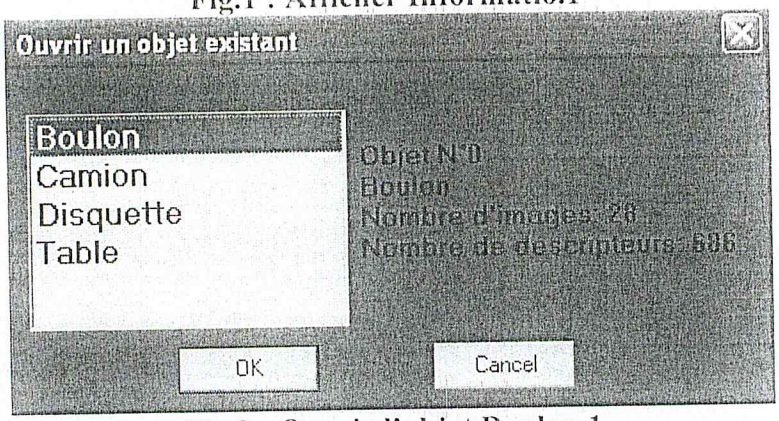



**Fig.2 : Ouvrir l'objet Boulon.1**

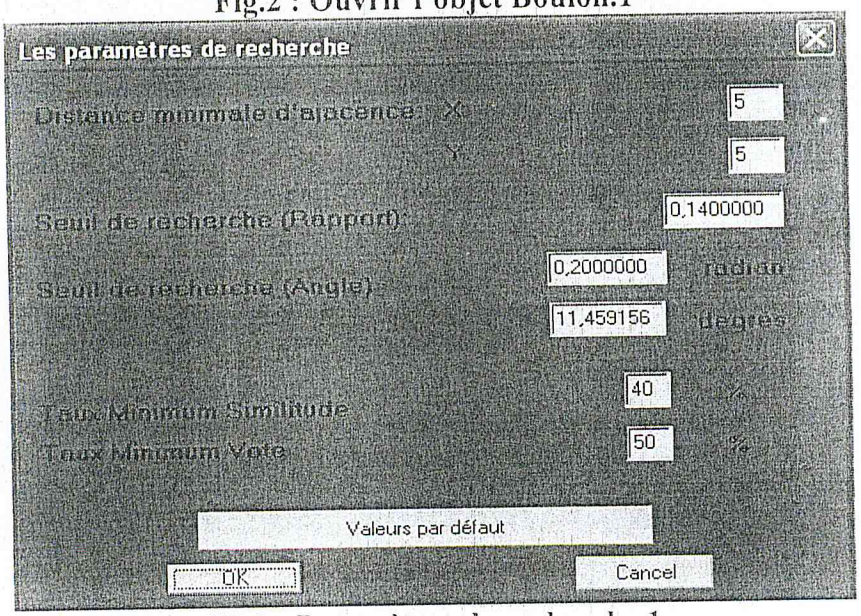



**Fig3 : Paramètres de recherche.1**

Voici les résultats de la recherche d'une image requête de l'objet disquette appartient à la base, avec des paramètres de recherche par défaut :

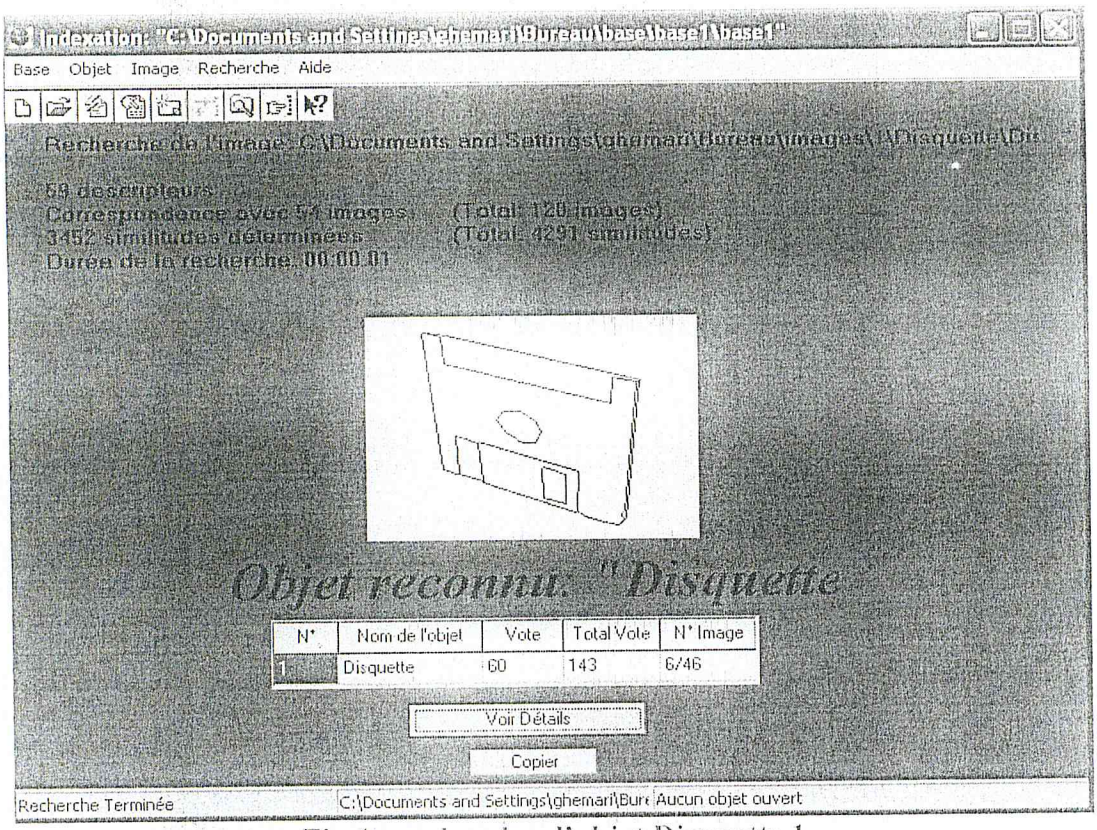



Fig.4 : rechercher l'objet Disquette.1

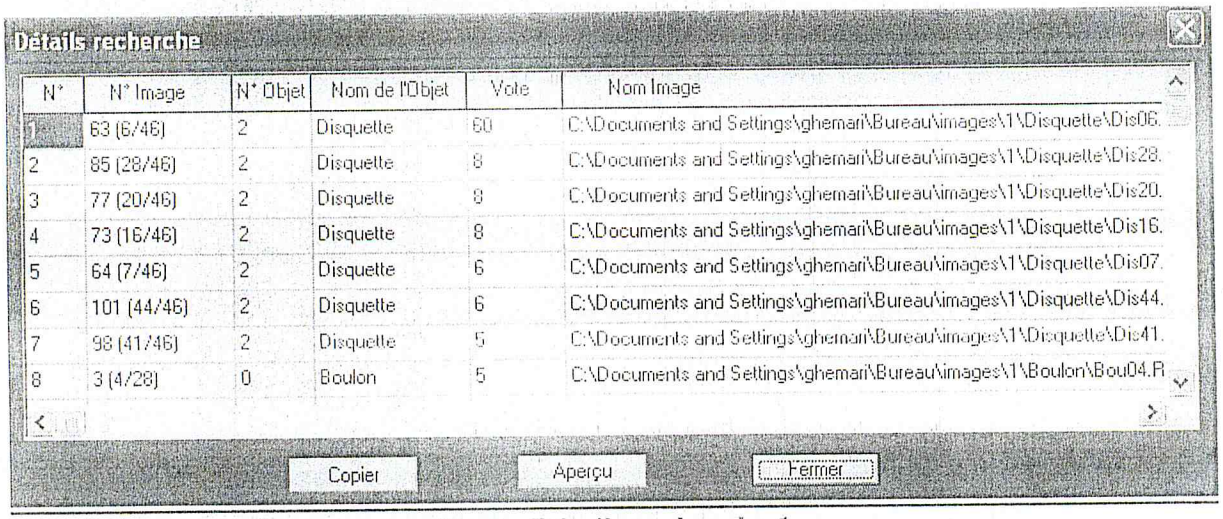



Fig.5 : Détails recherche.1

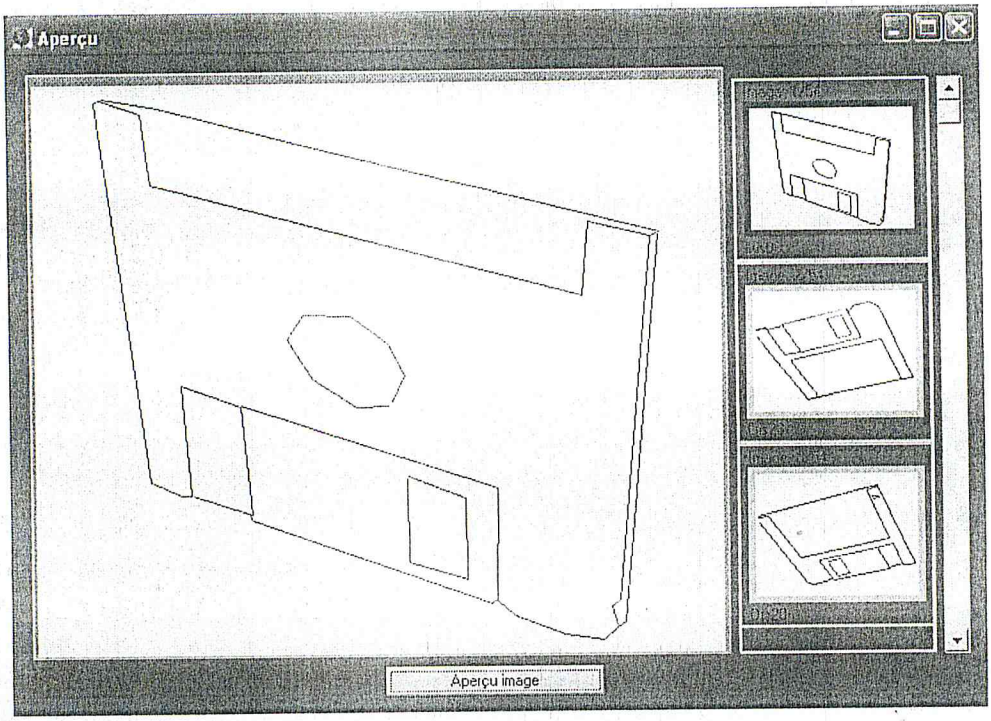



Fig.6 : Apercu.1

## II.2) Exemple.2:

Un autre exemple d'une base qui porte 10 objets ( 279 images ) qui sont : fourchette , guidon , hache , missile , niche , phone , règle T , sabre , scie et sou marin , la recherche est faite avec comme paramètres de recherche les paramètres par défaut .

La requête est prise de deux façons différentes :

### II.2.1) La requête appartient à la base

L'image requête de l'objet phone appartient à la base.

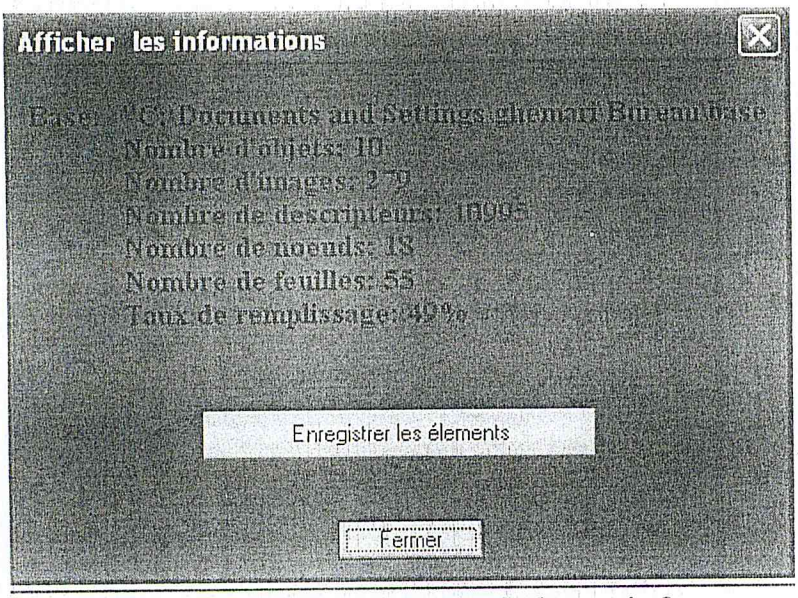



Fig.7 : Afficher Informatio.2

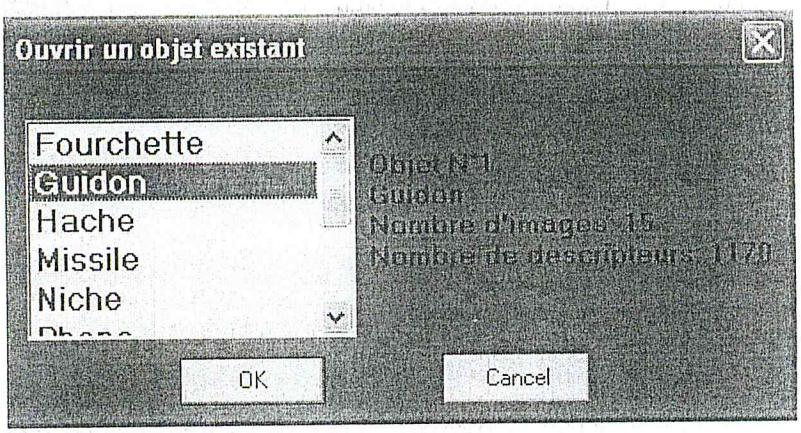



Fig.8 : Ouvrir l'objet Guidon.2

Voici les résultats de la recherche :

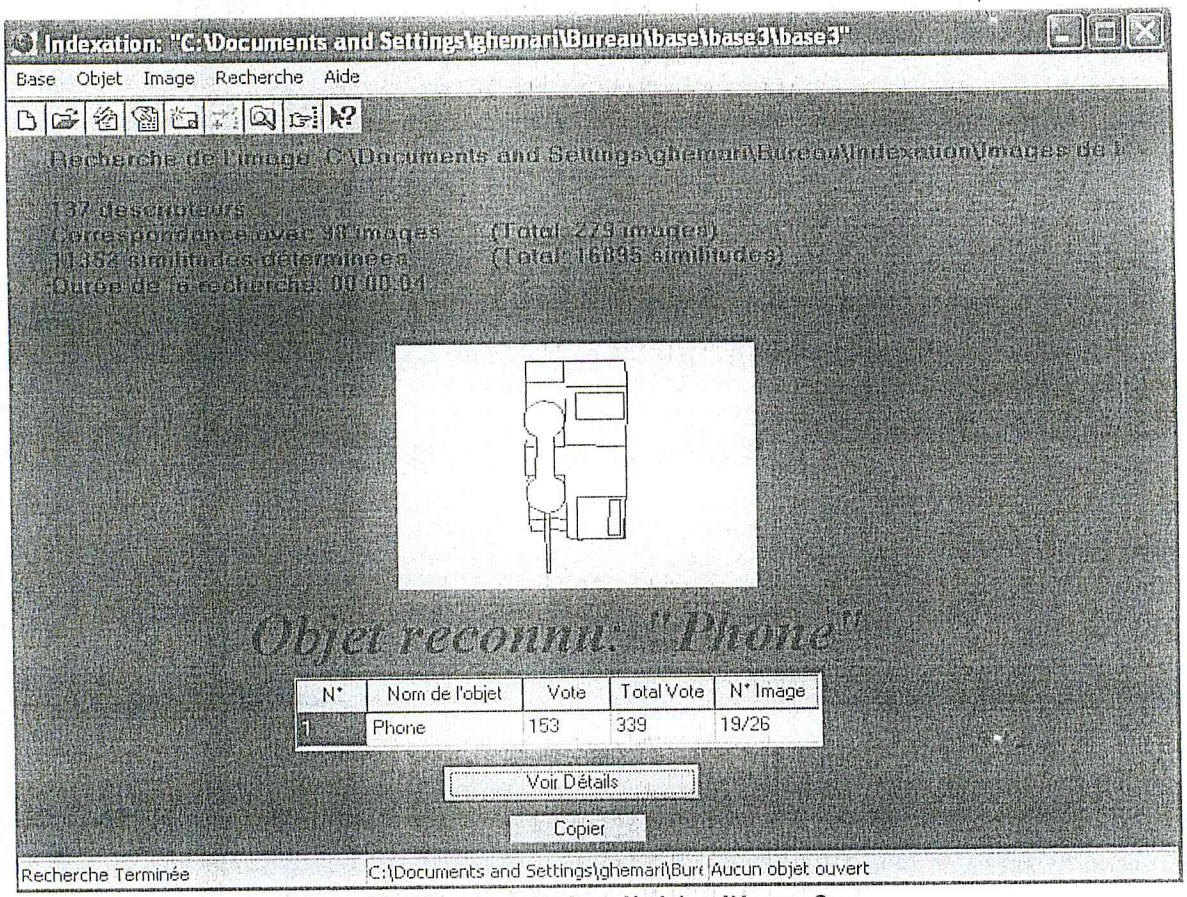



Fig.9 : rechercher l'objet Phone.2

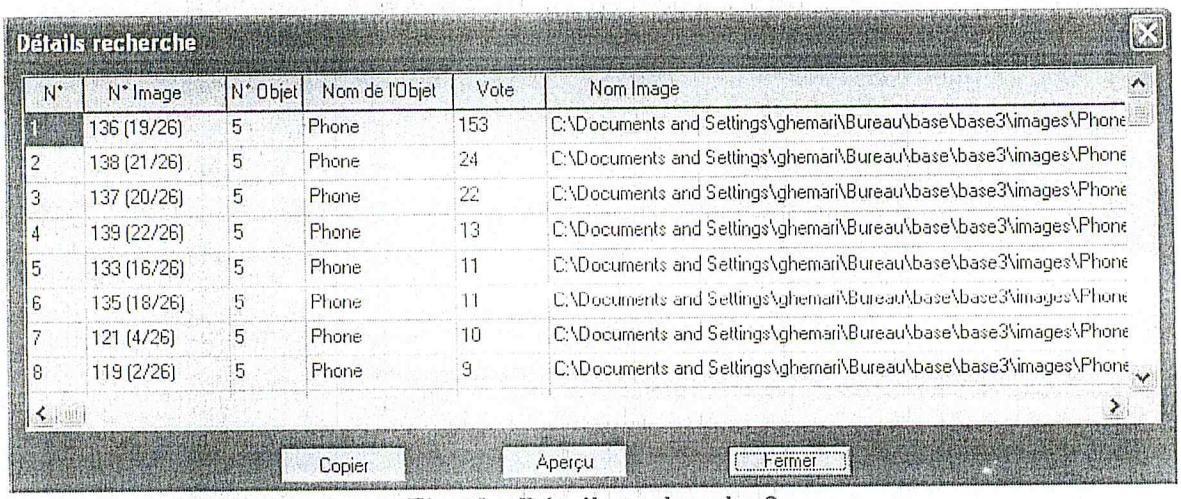



Fig.10 : Détails recherche.2

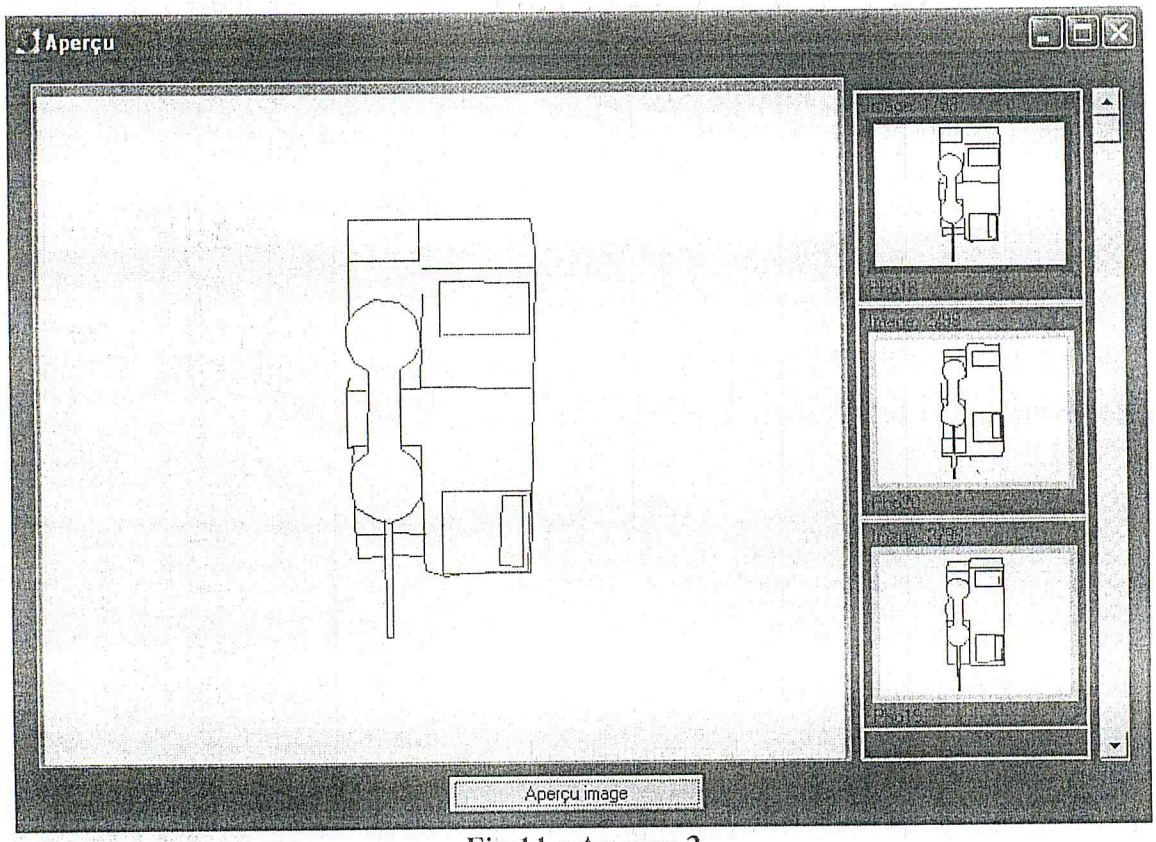



**Fig.11 : Apercu.2**

## II.2.2) La requête n'appartient pas à la base :

L'image requête de l'objet phone qui n'appartient pas à la base.

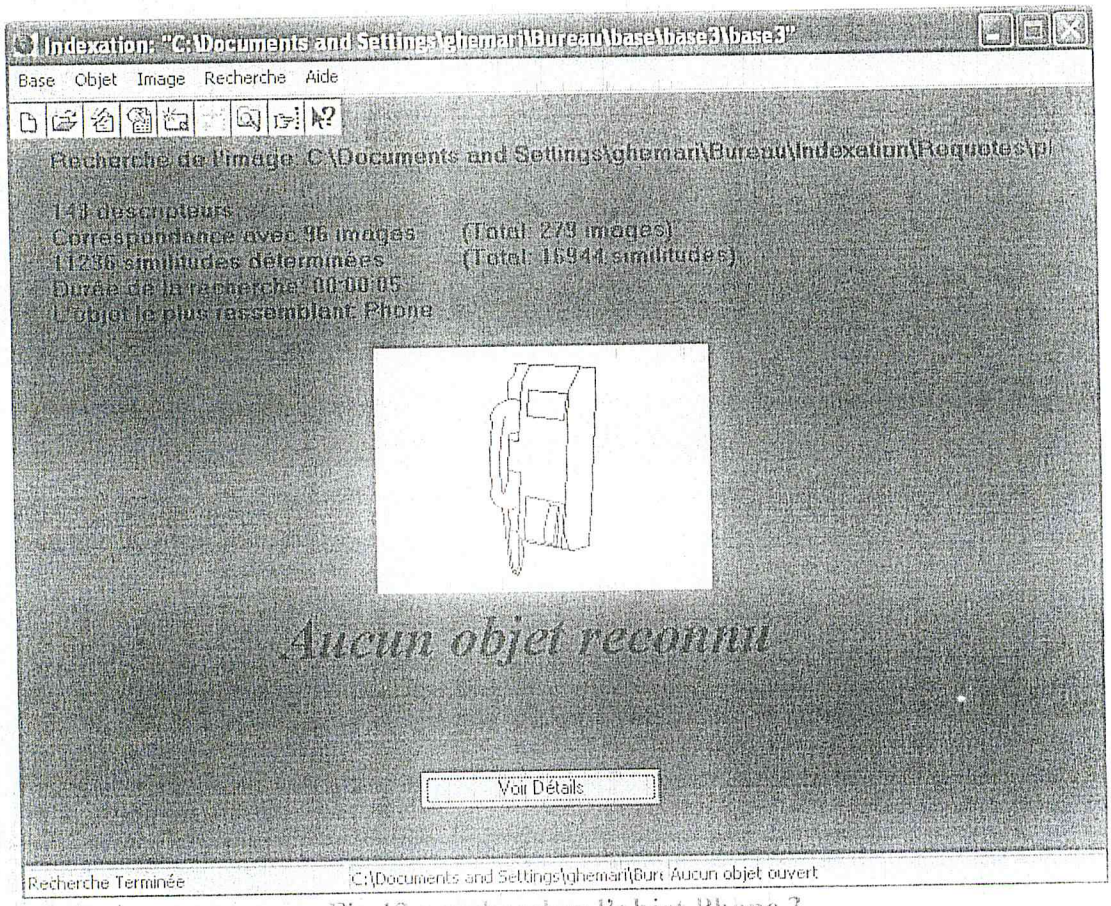



Fig.12 : rechercher l'objet Phone.3

Voici les résultats de la recherche :

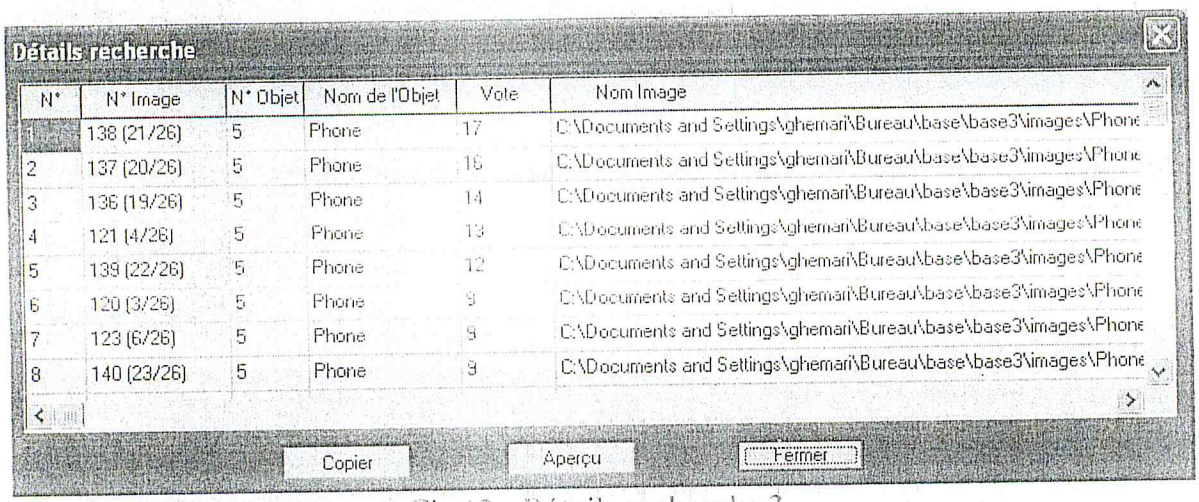



Fig.13 : Détails recherche.3

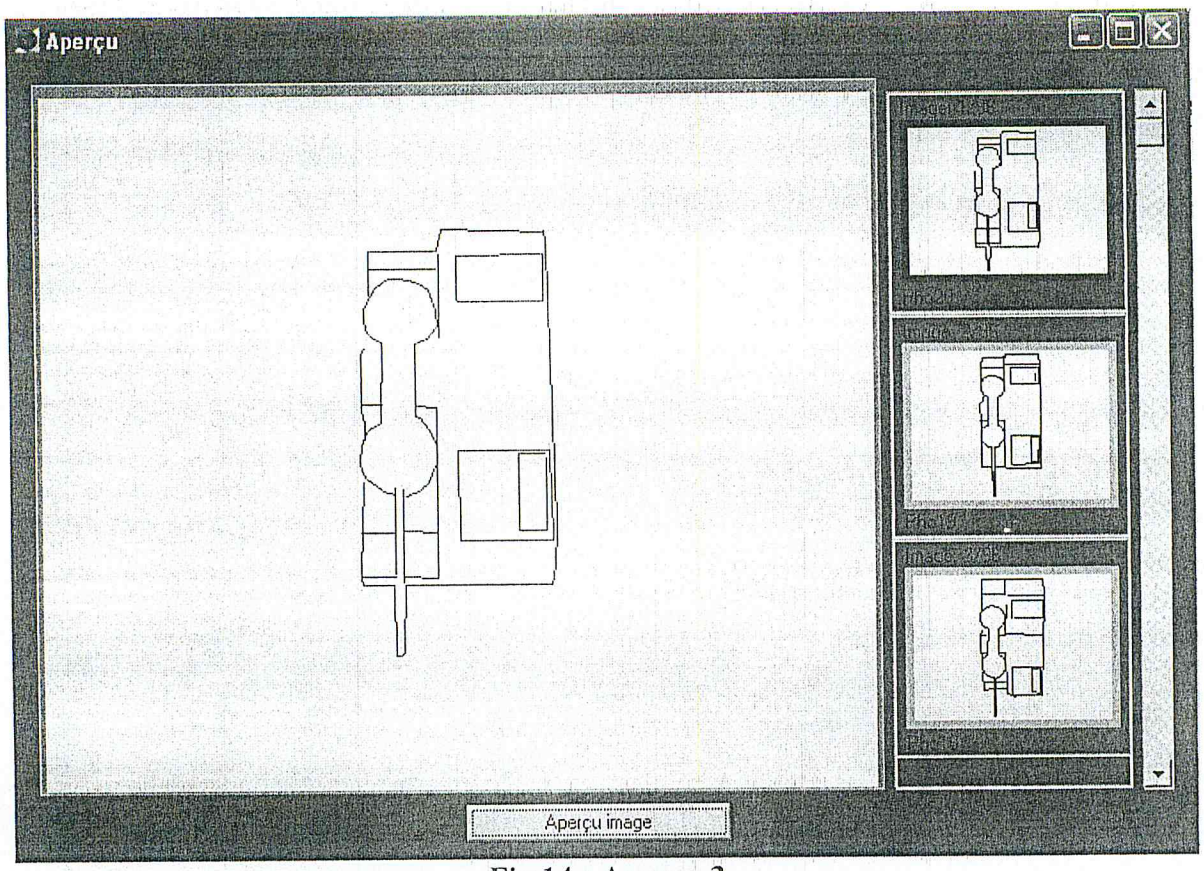
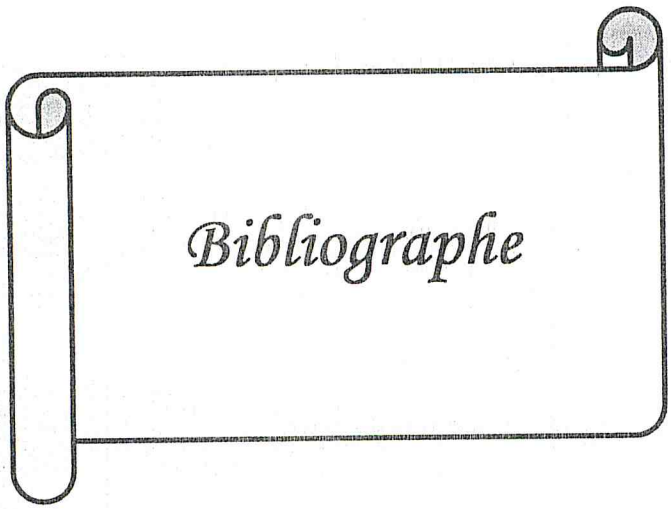



**Fig.14 : Apercu.3**


Bibliographe

A.Abi ayed
[AIT95] Calibrage statistique et dynamique des camera –application à la manipulation d'oobjet polyédrique par un robot sous le contrôle d'une tête de vision stéréoscopique.
PhD thesis, Enseeiht (INPT), 1999

[AMS00] L. AMSALEG, P. GROS, R. MEZHOUD.
Mise en base d'images indexées par des descripteurs locaux: Problèmes et perspectives.
Rapport de recherche n°3903. INRIA. Mars 2000

[AOU93] S. AOUAT et K. HARBILI.
Utilisation des invariants projectifs dans l'appariement d'images stéréoscopiques.
Projet de fin d'étude.
Institut d'informatique, USTHB, Juin 1993

[AST94] K. ASTRÖM.
Affine and projective normalization of planar curves and regions.
Proceedings of the 3rd European Conference on Computer Vision, Stockholm, Sweden. Mai 1994.

[AUD00] P. AUDENINO.
Aspect de stéréovision humaine et artificielle: Application Algorithme mise en correspondance de points.
DEA Signal Image Parole Transmission
Ecole Polytechnique de Montréal et Laboratoire LIS. 2000

[BAL91] D.H. BALLARD.
Animate vision, in Artificial Intelligence.
1991.

[BAY71] R. BAYER.
Binary B-trees for virtual memory.
Proceedings of the ACM 1971 SIGFIDET November 1971

[BER96] S. BERCHTOLD, D.A. KEIM, H.P. KRIEGEL.
The X-tree: An index structure for high-dimensional data.
Proc. 22nd International conference on Very Large Data-Bases, Bombay, India, 1996.

[BER98] S. BERCHTOLD, C. BOHM, H.P. KRIEGEL.
The Pyramid-technique: Towards Breaking the Curse of Dimensionality.
Proc. Int. Conf. on Management of Data, ACM SIGMOD, Seattle, Washington, 1998.

[BIN93] T.O. BINFORD, T.S. LEVITT.
Quasi-invariants: Theory and exploitation.
Proceedings of DARPA Image understanding Workshop. 1993

[BRE93] T.M. BREUEL.
Higher-order statistics in object recognition.
International Conference on Computer Vision and Pattern Recognition. 1993.

[CAN86] J.F. CANNY.
Computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence.
Juin 1986.

[CAR97] C. CARREZ.

Structure de données en Java, C++ et Ada 95, Pratique et outils de contrôle.
CNAM, Paris InterEditions, 1997.

[CHA93] H.Chabbi
Construction de facettes 3D par stéréovision intégrant des principes de géométrie projective.
Thèse de doctorat.
INPL. Février 1993.

[CIA97] P. CIACCIA, M. PATELLA, P. ZEZULA.
M-tree: An efficient access method for similarity search in metric spaces.
Proc VLDB, 1997.

[COC95] JP.Cocqeerez et P.Syivie
Analyse d'imags: filtrage et segmentation. 1995.

[COU94] B. COUAPEL.
Stéréovision par ordinateur : Géométrie et expérimentation
Thèse de doctorat.
Institut de Formation Supérieure en Informatique et Communication.
Université de Rennes 1. Mars 1994.

[GRI90] W.E.L. GRIMSON, D.P. HUTTENLOCHER.
On the sensitivity of the HOUGH transform for object recognition.
IEEE Transactions on Pattern Analysis and Machine Intelligence. Mars 1990.

[GRO93] P. GROS.
Outils géométriques pour la modélisation et la reconnaissance d'objets polyédrique.
Thèse de doctorat.
Institut National Polytechnique de Grenoble, France. Juillet 1993.

[GRO95A] P. Gros, O. Bournez, E. Boyer
Utilisation des quasi-invaraints géométriques pour l'appariement et la modélisation des images de segments de droites
Rapport de recherche n°2608. INRIA. Juillet 1995.

[GRO95B] P. GROS.
Matching and clustering : Two steps towards object modelling in computer vision.
The international Journal of Robotics Research. Décembre 1995.

[GRO97] P. GROS, G. MCLEAN, R.DELON, R. MOHR, C. SCHMID et G. MISTLER.
Utilisation de la couleur pour l'appariement et l'indexation d'images.
Rapport de recherche n°3269. Projet MOVI. INRIA Rhône-Alpes.
Septembre 1997

[GRO98] P. GROS.
De l'appariement à l'indexation d'images.
Thèse d'Habilitation à diriger des recherches.
Institut National Polytechnique de Grenoble, GRAVIR-IMAG. France.
Décembre 1998.

[GUT84] A. GUTTMAN.
R-trees: A dynamic index structure for spatial searching.
Proc. ACM SIGMOD conference, Boston, June 1984.

[HOR95] J. HORNEGGER et H. NIEMANN.
Statistical learning, localisation and identification of objects.
ICCV'95 Fifth International Conference on Computer Vision. 1995.

[KAB95] .Kabir et S.Derbane.
Etude et implémentation des méthodes déterministes se segmentation d'images par extraction de contours.
Thèse d'ingéniorat.
Uniersité de Blida, 1995.

[LAM96] B. LAMIROY, P. GROS.
Reconnaissance d'objets par indexation géométrique étendue
Rapport de recherche. Equipe MOVI. GRAVIR-IMAG-INRIA Rhône-Alpes. Mai 96.

[LAM88] Y. LAMDAN et H.J. WOLFSON.
Geometric hashing: A general and efficient model based recognition scheme.
ICCV'88 Second International Conference on Computer Vision. 1988.

[LAM98] B. LAMIROY.
Reconnaissance et modélisation d'objet 3D à l'aide d'invariants projectifs et affines.
Thèse de doctorat.
Institut National Polytechnique de Grenoble, GRAVIR-IMAG-INRIA Rhône-Alpes. Juillet 1998.

[LAR89] S.larabi
Utilisation et la géométrie en vision stéréoscopique binoculaire. Calcul des profondeurs à un facteur d'échelle
Press. 7ème congré AFCET, 1989.

[LAR91] S. LARABI.
Utilisation de la géométrie projective en vision par stéréoscopique binoculaire.
Thèse de doctorat.
Institut National Polytechnique de Toulouse. France. Avril 1991.

[MAR82] D. Marr
Vision computational investigation into human representation and processing of visual information.
Free man and company, San Francisco. 1982.

[MAT99] S. MATUSIAK.
Description invariante et locale des formes planes: Application à l'indexation d'une base d'images.
Thèse de doctorat. Université de Valenciennes et du Hainaut Cambresis. France. Septembre 1999.

[MOR93] L. MORIN.
Quelques contributions des invariants projectifs à la vision par ordinateur.
Thèse de doctorat.
Institut National Polytechnique de Grenoble, INFIA-IRIMAG. France. Janvier 1993.

[MUR95] H. MURASE et S.K. NAYAR.

Visual learning and recognition of 3D objects from appearance.
International journal of computer vision, 1995.

[ROU99] D. ROUSSEL.
Reconstruction de courbes et de surfaces 3d en stéréo-acquisition.
Thèse de doctorat.
Université Paris XI. Janvier 1999.

[SCH96] C. SCHMID.
Appariement d'images par invariants locaux de niveaux de gris :
Application à l'indexation d'une base d'objets.
Thèse de doctorat.
Institut National Polytechnique de Grenoble, GRAVIR-IMAG-INRIA Rhône-Alpes. Juillet 1996.

[SCH97] B. SCHIELE
Reconnaissance d'objets utilisant des histogrammes multidimensionnels de champs réceptifs
Thèse de doctorat, GRAVIR-IMAG-INRIA Rhône-Alpes, Juillet 1997.

[SKO88] T. SKORDAS.
Mise en correspondance et reconstruction stéréo utilisant une description structurelle des images.
Thèse de doctorat.
Institut National Polytechnique de Grenoble. France. 1988.

[SOS92] H. SOSSA.
Reconnaissance d'objets polyédriques dans une base de modèles.
Thèse de doctorat. Institut National Polytechnique de Grenoble. France. Décembre 1992.

[SWA91] M.J. SWAIN et D.H. BALLARD.
Color indexing. International Journal of Computer Vision. 1991.

[TOS89] G. TOSCANI, R. VAILLANT, R. DERRICH, O.D. FOUGERAS.
Stereo camera calibration using the environment.
The 6th Scandinavian conference on image analysis. Oulu, Finland 1989.

[WAN91] L. WANG et W.H. SIANG.
Camera calibration by vanishing lines for 3D computer vision.
IEEE. Transactions on pattern analysis and machine intelligence.
Vol. 113, April 1991.

[WEB98] R. WEBER, H.J. SCHEK, S. BLOTT.
A quantitative analysis and performance study for similarity-search methods in high dimensional spaces.
Proceedings of the 24th VLDB International Conference on Very Large Data Bases. New York. USA. August 1998.

[ZEZ96] P. ZEZULA, P. CIACCIA, F. RABITTI.
M-tree: A dynamic index similarity queries in multimedia database.