République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Saad Dahlab, Blida
USDB.

Faculté des sciences.
Département informatique.

Mémoire pour l'obtention
d'un diplôme d'ingénieur d'état en informatique.
Option : Système d'information

Sujet :

Conception d'un système pour la recherche d'information en imagerie

Présenté par : *MADADI Miloud*
CHIBANE Mohamed

Promoteur : *M^elle N.Benblidia*
Encadreur : *M^elle F.Reguieg*

Promotion : 2007/2008

Remerciements

Nous tenons tout d'abord a remercier nos promotrices M^{elle} N.Benblidia, M^{elle} FZ.Reguig pour nous avoir encadré tout au long de ce travail, et aussi pour leurs encouragements.

Nos remerciements vont également à tout ceux qui nous ont aidés de manière directe ou indirecte pour réaliser ce modeste travail.

Dédicaces

Je dédie ce modeste travail à mes parents.

A mes frères (Mohamed, Redhouane, Kamel, Toufik, Nacer, Sofiane).

A ma sœur.

A mes amis……

Miloud…

Dédicace

Je dédie ce modeste travail :

A ma chère mère pour son soutient,

A mon père pour son appui,

A mon unique frère Sid Ahmed,

A ma Grande Mère

Chibane...

Table de Figure

Liste des tableaux

Résumé

Nous présentons dans ce mémoire un ensemble qui concerne : Une étude générale sur les images, et les techniques d'indexation des images couleurs, puis la recherche des similaires par le contenu dans les bases des images. Cet ensemble de techniques concourt au développement d'un système de reconnaissance automatique des images couleur.

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

Abstract

We present in this thesis a combination of the following: A general survey on the images and indexing techniques of color images, then search for similar by the content in databases of images. This set of techniques contributes to the development of a system of automatic recognition of color images.

As a first step, we present a literature review with the key techniques for indexing images adapted specifically for color images, and then we proposed a detailed algorithm for either of these approaches, so on we experimentally verify their validity .

Introduction

Introduction Générale

Jusqu'à présent, l'exploitation d'une collection d'images faites appel à des techniques de descriptions manuelle et textuelles inscrit à la main d'un ensemble de mots clés décrivant le contenu des images. Il est alors possible d'interroger la collection d'images à partir d'un ensemble de mots décrivant l'image recherchée. Ceci permet de retrouver, par exemple, les images relatives à un thème particulier.

Cette manière de description porte beaucoup de sémantique sur la description textuelle des images par un humain, cependant plusieurs interprétations peuvent être données au contenu d'une même image. En effet deux personnes peuvent ainsi donner deux descriptions différentes d'une même image. De plus, ce mode de description étant basé uniquement sur le texte, il est principalement limité au contenu exprimé à l'aide de mots. Il est également difficile à mettre en œuvre lorsqu'on doit décrire un très grand nombre d'images.

Avec l'arriver d'autres techniques de description, le traitement devient automatique. Réduisant l'intervention humaine, elles peuvent servir à décrire de très larges collections d'images. Ces techniques sont basées sur l'extraction de paramètres des images (couleur, texture, forme...).

Décrire automatiquement le contenu visuel des images permet d'arriver à un nouveau mode d'interrogation : l'interrogation par l'exemple. Dans ce cas, le principe est de retrouver les images qui ressemblent fortement à l'image donnée en exemple, c'est-à-dire celles qui ont une forte similarité visuelle avec l'exemple. La description visuelle des images est faite par des descripteurs.

On introduit ensuite la notion de « gestion de bases de données » qui étudie le stockage et l'interrogation des données numériques. Une base de données informatique est donc un ensemble d'informations numériques stockées selon un modèle dans le but de les conservées, de les enrichir et de les interroger avec la

garantie de l'intégrité de ces données. Ces informations peuvent être de n'importe quel type : texte, image, son ou vidéo. Dans le cadre de notre travail, notre base contient uniquement que des images.

Notre système, on a appliqué le procédé de la recherche d'images par l'exemple, On exploitant une base contenant des images déjà définie. Tout d'abord, on va extraire de l'image requête un descripteur visuel, et pour chaque image de la base, on extrait des descripteurs respectifs. En suite, on a comparé chaque image de la base avec l'image requête par le calcul de distances entre descripteurs. On filtre ensuite les distances trouvées par l'application d'un seuil fixé selon la base d'images et l'application ; à fin d'obtenir une ou plusieurs images qui soient les plus similaires à notre image exemple.

Ce mémoire est organisé en quatre chapitres :

- ✓ Dans le premier chapitre, nous présentons quelque notion sur l'image, ainsi que les méthodes d'analyse d'image.
- ✓ Dans le deuxième chapitre, nous exposons les différentes structures et techniques d'indexations. on a opté pour l'arbre quaternaire en utilisant la technique de hachage.
- ✓ Dans le troisième chapitre, nous décrivons la démarche à suivre dans la recherche d'image, ainsi que la structure de la base d'images à implémenter.
- ✓ Dans le quatrième chapitre, nous présentons l'implémentation de notre système, plusieurs tests seront effectués pour valider notre méthode.

Généralités

I- Introduction

Le stockage et la transmission des images numériques semblent être un problème en voie d'être résolu, celui de leur interprétation automatique demeure un problème difficile. Or, l'interprétation automatique des images ouvrirait la porte à de nombreuses applications pour lesquelles la machine viendrait au secours de l'homme, incapable de traiter de telles masses d'images. Nous pouvons par exemple imaginer des systèmes automatiques capables d'analyser simultanément plusieurs milliers de caméras de surveillance, ou encore un assistant qui viendrait aider l'utilisateur à rechercher des images dans une collection de photos trop importante pour être parcourue entièrement. L'approche statistique a fait l'objet de plusieurs recherches et a donné la naissance à un nombre considérable de méthodes. Ces méthodes sont très utilisées dans l'analyse des images texturées du fait qu'elles évaluent les attributs de texture directement à partir de l'image considérée.

Chaque méthode contient ses propres attributs qui permettent de caractériser la texture d'une image, certains de ces attributs de texture sont calculés soit pour toute l'image qui est donc de types globaux, soit pour chaque pixel de l'image (locaux).

Ces paramètres sont exploités et regroupés dans un vecteur appelé descripteur d'image qui donne l'index de l'image, qui facilite la recherche rapide d'image à partir de la base de données qu'il occupe.

La figure suivante représente les détails de notre système de recherche :

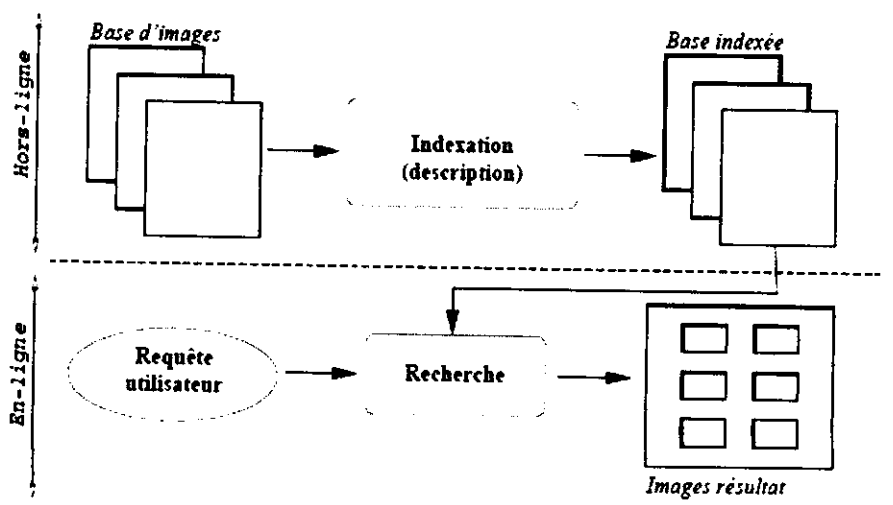


Figure I.1 : Schéma explicatif du système de recherche

II- Notion sur l'image

II-1- Définition

On peut trouver de nombreuses définitions du mot image.

Une image est un support d'informations présentant les éléments d'une scène qui a été captée soit par un appareil photographique, soit par un satellite…etc. [TIT 05].

Mathématiquement, l'image est représentée par une matrice de données numériques, on peut concevoir ces images en 2 dimensions comme un tableau de valeurs auxquelles on fait correspondre une position sur un plan (x, y) et une couleur pour visualiser l'image sur l'écran d'un ordinateur [TIT 05].

Chaque image est composée par des unités élémentaires appelées pixel. Elle est définie par le nombre de pixels qui la compose en largeur et en hauteur ainsi que l'étendue des teintes de gris ou des couleurs que peut prendre chaque pixel. [PUB 03].

II-2- Le pixel (Picture élément)

Il constitue le plus petit élément de l'image. Le pixel est lui-même codifiable en tonalite, en clarté et en saturation [TIT 05].

Un pixel est une entité calculable qui peut recevoir une structure et une quantification. Toutes les données correspondant aux informations contenues dans l'image sont structurées d'une certaine façon afin de permettre leur stockage. Il existe un grand nombre de formats d'images : tous ces formats ne correspondent ni plus ni moins qu'à une structuration particulière des données concernant l'image.

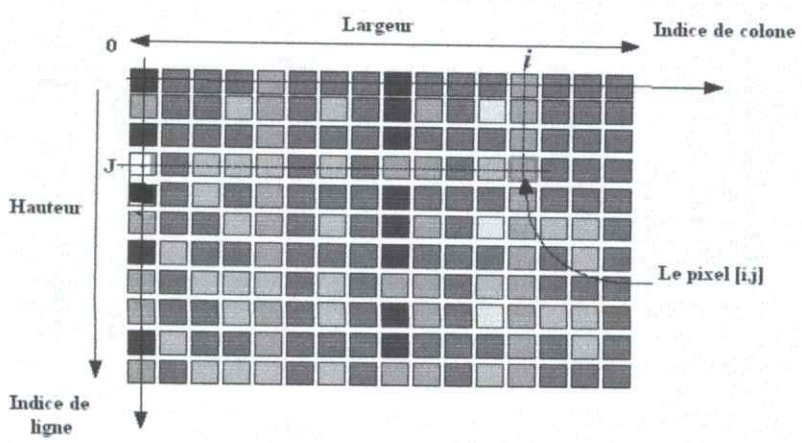


Figure I.2 Détail d'une Image

II-3- Résolution d'image

La résolution d'une image est définie par le nombre de pixels par unité de longueur de la structure à numériser (classiquement en dpi (dots per inches) ou ppp (points par pouce)). Ce paramètre dépend principalement des caractéristiques du matériel utilisé lors du processus de numérisation. Plus le nombre de pixels est élevé par unité de longueur de la structure à numériser, plus la quantité d'information qui décrit cette structure est importante et plus la résolution est élevée. La résolution d'une image numérique définit le degré de détails qui va être représenté sur cette image.

Exemples

- ✓ La **résolution** pour l'écran est 72 DPI.
- ✓ La **résolution** pour l'impression papier entre 150 et 600 DPI (ex : impr. laser standard = 300 DPI).
- ✓ La **résolution** pour les machines professionnelles 1200 DPI.

II-4 Types d'images

a- Les images binaires

Ce sont les images les plus simples ou le pixel peut prendre uniquement deux valeurs qui correspond généralement au noir et blanc.

b- Les images en niveaux de gris

En général, les images en niveaux de gris renferment 256 teintes de gris, par convention la valeur zéro représente le noir (intensité lumineuse nulle) et la valeur 255 le blanc (intensité lumineuse maximale) ; ces images sont codées sur un octet.

c- Les images couleurs

La couleur est une composante naturelle de l'image. De plus, la couleur est une propriété du rayonnement lumineux que l'on identifie par les longueurs d'onde des diverses radiations qui composent le flux lumineux, ainsi que par les énergies relatives de chacune d'elles. En infographie, la couleur est obtenue par un procédé simplificateur, mais opérationnel, dont le principe consiste à construire,

comme en télévision ou en vidéo, une couleur quelconque à partir de trois couleurs de base (le rouge, le vert, le bleu).

II-5- Les Modèles de représentation de couleurs

1- Espace de couleur RVB (RGB)

Acronyme de Red Green Blue. Ce mode de composition des couleurs est basé sur le principe des couleurs additives : le rouge, le vert et le bleu sont les trois primaires utilisées dans la constitution de couleurs à partir de sources lumineuses. Une image RVB est composée de trois couches, codées chacune sur 8 bits, (soit 256 niveaux de couleur par couche, ce qui donne 16 millions de couleurs). Le mode de représentation RVB est surtout utilisé pour la reproduction de couleurs sur écran [ADE 05].

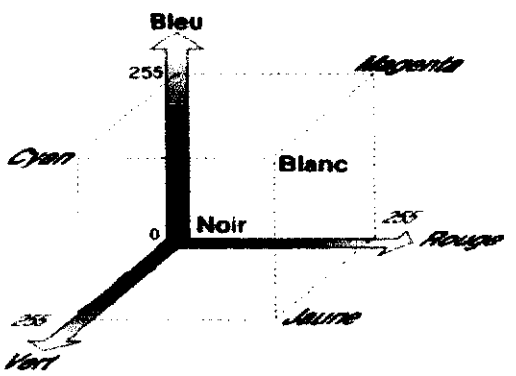


Figure I.3 : Espace de couleur RVB

2- Espace de couleur CMJN (CMYK)

Acronyme de Cyan Magenta Yellow black. Le cyan, le magenta, le jaune et le noir sont les quatre couleurs d'encre utilisées dans le procédé d'impression quadrichromique et dans tout procédé de reproduction à base de pigments ou de colorants. Sur chaque couche, le pixel est défini par un pourcentage de la couleur correspondante.

Le mode CMJN est la norme pour l'imprimerie [ADE 05].

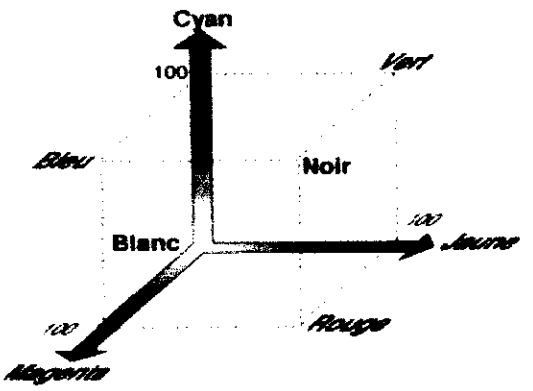


Figure I.4 : Espace de couleur CMJN

3- Espace de couleur Lab

Les couleurs sont définies par trois valeurs :

L : Luminosité (luminance) codée en pourcentages (%), a et b correspondent à l'information colorée (chrominance) où la couleur est définie à partir d'un mélange de vert à magenta (a) et un mélange de bleu à jaune (b). Les valeurs sont comprises entre -120 et +120 pour a et b [ADE 05].

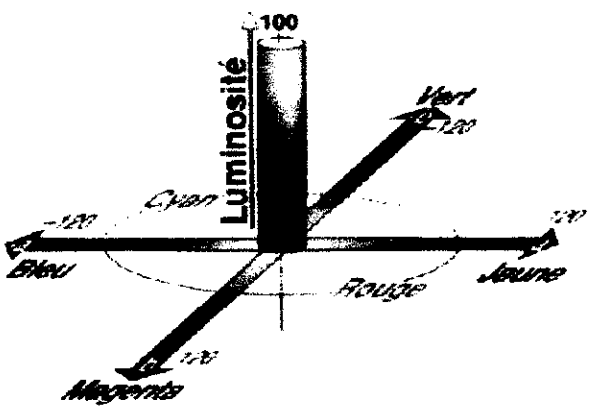


Figure I.5 : Espace de couleur Lab

4- Espace de couleur TSI (HLS)

Il spécifie chaque couleur en termes de Teinte (Hue), Saturation, et Intensité (Luminance) [ADE 05], où :

- ✓ **la teinte :** C'est la longueur d'onde de la lumière réfléchie, ou transmise par un objet. Elle correspond à son emplacement sur la roue chromatique, dans un angle compris entre 0° et 360°. Le spectre circulaire part du rouge, passe par le vert et le bleu pour revenir au rouge.

- ✓ **la saturation :** Définit l'intensité de la couleur des couleurs grisées - 0 %; aux couleurs vives - 100 %.

- ✓ **l'intensité :** Indique la variation d'intensité lumineuse d'une couleur, entre 0 % - noir et 100 % - blanc.

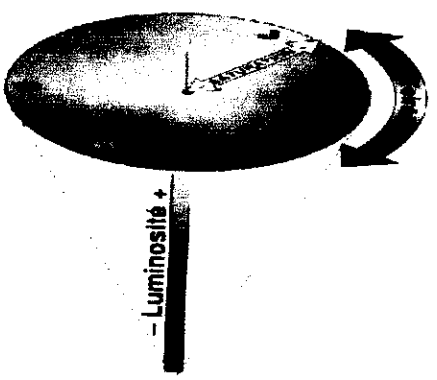


Figure I.6 : Espace de couleur TSI

II-6- Formats d'images

Un format d'image comprend en général un en-tête qui contient des informations sur l'image suivie des données de l'image [PUB03]. La structuration des données propres est différente pour chaque format d'image. Il s'agit de regrouper les informations contenues dans plusieurs pixels, au lieu de coder pixel à pixel les données des images : une zone homogène en couleurs pourra être codée par les coordonnées de ses pixels et par une valeur unique de couleur. Le codage de l'information couleur des fichiers de données image, peut s'effectuer en codant chaque pixel par le triplet des valeurs des composantes Rouge, Vert, Bleu dans le

cas des images couleurs, ou pour une seule valeur pour les images en niveaux de gris.

Nous distinguons plusieurs types de formats :

- **Le Format JPEG (Joint Photo Expert Group)**

Ce format permet une représentation en vraies couleurs (16 millions). C'est la représentation de prédilection des images "naturelles" avec des grands dégradés de couleurs [PUB 03].

- **Le Format GIF**

Le format GIF est essentiellement destiné à l'affichage des images sur le web et s'applique aux images bitmap, en niveau de gris ou en couleurs indexées. Il s'agit d'un format compressé conçu pour minimiser le temps de transfert en ligne [PUB 03].

- **Le Format TIFF (Tag Image File Format) :**

Le format TIFF (Tag Image File Format) Format de sauvegarde d'images bitmap avec possibilité de compression LZW (Algorithme de compression qui ne détériore pas les données compressées, par opposition au format JPEG). Ils peuvent coder les couleurs en RVB, CMJN ou les indexer sur une palette de couleurs. TIFF est souvent utilisé par les logiciels d'images pour MAC [PUB 03].

- **Tableau récapitulatif**

L'utilisation d'un format d'image donné est conditionnée par l'utilisation que l'on souhaite faire de l'image.

Par exemple, mettre une image sur une page web conduira au choix des formats gif, png ou jpeg les moins encombrants; le traitement et l'analyse d'une image conduiront par contre au choix des formats tif ou bmp qui respectent le plus les données pixels à pixels.

Format	Bitmap vectoriel	Compression des données	Nombre de couleurs supportées	Affichage progressif	Animations	Transparence
JPEG	Bitmap	Oui, réglable (avec perte)	16 millions	Oui	Non	Non
GIF	Bitmap	Oui (sans perte)	256 (palette)	Oui	Oui	Oui
PNG	Bitmap	Oui (sans perte)	De 256 (palette) jusqu'à 16 millions	Oui	Non	Oui (couche alpha)
TIFF	Bitmap	Oui (sans perte) ou non (au choix)	16 millions	Non	Non	Non

Tableau I.1 : Comparaison des Formats d'images

III- Notions sur la texture

La texture joue un rôle très important dans l'identification et l'extraction des informations thématiques contenues dans l'image. Ces informations sont extraites à partir de plusieurs méthodes comme par exemple la matrice des cooccurrences.

- **Définition**

La texture peut être définie comme un ensemble de primitives arrangées selon des règles particulières de placement (Gross, 1983 ; Wu, 1992). Une primitive est un ensemble connexe plus ou moins important de pixels de niveaux de gris à peu près semblables ; il s'agit en fait d'un motif de base. Il faut cependant préciser l'échelle à laquelle se fait l'observation.

Pour R. M. Haralick et Gagallowicz la texture est un phénomène à deux niveaux. Le premier niveau concerne la description des primitives dont est composée l'image et qui sont ses propriétés caractéristiques; le deuxième implique la dépendance spatiale entre ces primitives. [BRI 05]

Pour Serra, la texture est l'ensemble des propriétés spatiales, périodiques ou non, d'un phénomène se déployant dans le plan de l'image. L'étendue de ce phénomène et la géométrie de ses limites ne participent pas à la texture : seul importe le contenu spatial, exprimé en termes statistiques ou morphologiques, et non pas son contenu [BRI 05].

Pour Henri Maître une texture est un champ de l'image qui apparaît comme un domaine cohérent et homogène, c'est-à-dire formant un tout pour l'observateur. C'est cette propriété de cohérence de la texture placée dans son contexte d'être perçue comme un tout homogène par l'oeil humain qui sera recherchée le plus souvent par le traiteur des images, dans le but d'isoler les textures, soit pour segmenter l'image, soit pour reconnaître des régions [BRI 05].

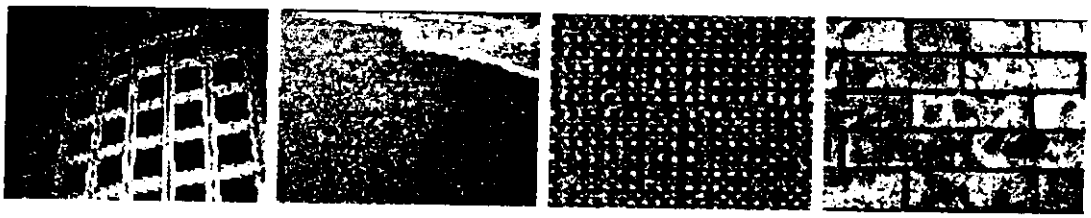


Figure I.7 : Exemple de deux textures différentes

Pour étudie la texture nous distinguons plusieurs méthodes :

III-1- Les méthodes statistiques

III 1-1- Méthodes du premier ordre

L'analyse par les méthodes de premier ordre se fait au niveau des pixels individuels d'une région de l'image (ROI). Les paramètres sont calculés à partir de l'histogramme des intensités. Celui-ci décrit, au niveau de la ROI choisie, la fréquence d'apparition de chaque niveau sur l'échelle des couleurs.

La moyenne, la variance, le skewness et le kurtosis sont les paramètres le plus souvent utilisés pour caractériser une texture.

1- La moyenne

Elle donne la valeur moyenne ou intensité moyenne des niveaux de couleur appartenant à tous les pixels de la ROI. , Ce paramètre représente l'emplacement de l'histogramme sur l'échelle des niveaux de couleur.

$$MOY = \frac{1}{N}\sum_{i,j} g(i,j)$$

Où g (i, j) représente la valeur du niveau de couleur du pixel (i, j).
N : est un facteur de normalisation qui correspond au nombre total de pixels.

2- La variance

Elle correspond au moment d'ordre 2 ; elle mesure la répartition des niveaux de couleur autour de la valeur moyenne.

$$VAR = \frac{1}{N}\sum_{i,j}\left(g(i,j) - MOY\right)^2$$

3-Le skewness

Il correspond au moment d'ordre 3 centré autour de la moyenne. Ce paramètre mesure la déviation de la distribution des niveaux de couleur par rapport à une distribution symétrique.
Pour une déviation par les valeurs élevées, le skewness est positif ; alors que pour une déviation vers les basses valeurs, il est négatif.

$$SKEW = \frac{1}{N}\sum_{i,j}\left(g(i,j) - MOY\right)^3$$

4- Le kurtosis

Il correspond au moment d'ordre 4 centré autour de la moyenne. Il caractérise la forme du sommet de l'histogramme : plus le kurtosis est faible et plus le sommet de l'histogramme est arrondi.

$$KURT = \frac{1}{N}\sum_{i,j}\left(g(i,j) - MOY\right)^4$$

5- Le rapport signal sur bruit (SNR)

En imagerie le SNR peur être défini de la façon suivante :

$$SNR = \frac{MOY}{\sqrt{VAR}}$$

Il correspond au rapport de la moyenne sur l'écart type. Ce paramètre permet de rendre compte de l'hétérogénéité d'une texture. Une texture hétérogène possèdera un signal sur bruit faible car la variance sera élevée.

6- Les percentiles pi

Ils sont déterminés par extrapolation sur l'histogramme cumulé. Celui-ci correspond à la fréquence cumulée d'apparition d'un niveau de couleur.
Pi correspond à la proportion (en %) de pixels ayant une valeur de niveaux de couleur inférieure à la valeur correspondante à p_i.

III-1-2- Les méthodes du deuxième ordre (matrice de cooccurrence)

En 1973, Haralick a proposé une méthode en se basant sur les matrices de co-occurrences de niveaux de gris.
Cette méthode est largement utilisée dans le monde du traitement d'image elle présente une grande simplicité de mise en oeuvre et donne de bons résultats sur la plupart des types d'images. C'est ce qui justifie notre choix. Il s'agit à partir d'une image représentée en niveaux de gris d'obtenir la matrice des moyennes d'espace du second ordre, appelée matrice de cooccurrence. Cette matrice contient une

masse très importante d'informations difficilement manipulable. C'est pour cela qu'elle n'est pas utilisée directement mais à travers des mesures dites indices de texture. Haralick a proposé quatorze indices. Bien que corrélés entre eux, ils réduisent l'information contenue dans la matrice de cooccurrence et permettent une meilleure discrimination entre les différents types de textures.

- **Définition**

La matrice de cooccurrence MC est une matrice carrée N*N (N=8, 16,32), ou N est le niveau de gris, chaque élément MC (i,j) représente le nombre de paires de pixels sépares par une distance d pour une direction θ.

On prend généralement comme valeurs 0°, 45°, 90°, 135° pour l'angle θ et 1 pour la valeur de d.

- **Exemple**

Une région de 15 pixels quantifiés sur 8 niveaux de gris.

1	2	1	3	4
2	3	1	2	4
3	3	2	1	1

La Matrice de cooccurrence MC, avec d = 1 et θ = 0.

b a	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	0	0	0	0	0
2	0	1	0	2	0	0	0	0
3	0	0	1	1	0	0	0	0
4	0	1	0	0	1	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0

Les paramètres extraits de la matrice MC (paramètres de Haralick) sont

1- La moyenne

$$MOY = \sum_i \sum_j p(i,j)$$

Où p (i, j) correspond aux éléments de la matrice de cooccurrence ; c'est à dire à la Probabilité de passer d'un pixel de niveau de couleur i à un pixel de niveau de couleur j.

2- La variance

$$VAR = \sum_i \sum_j (i - MOY)^2 p(i,j)$$

Elle caractérise la distribution des niveaux de couleur autour de la valeur moyenne M calculée précédemment.

3- L'énergie

L'un des indices le plus utilisé pour caractériser la texture dans le plan Spatio-fréquentiel est la mesure d'énergie. Du fait que les images transformées ont des fréquences, des orientations et des échelles différentes, l'indice d'énergie est une mesure locale de la distribution des coefficients d'ondelettes en fonction de la fréquence, l'orientation et l'échelle. Elle a été utilisée avec succès pour des problèmes de classification des textures. L'expression de l'énergie exprimée sous forme normalisée est donnée par :

$$E = \sum_{i,j} (p(i,j))^2$$

4- Le contraste (=inertie)

Il mesure les variations locales des niveaux de couleur. Si elles sont importantes (c'est à dire s'il existe des régions homogènes), alors le contraste sera élevé.
Ce paramètre permet aussi de caractériser la dispersion des valeurs de la matrice de cooccurrence par rapport à sa diagonale principale.

$$CONT = \sum_i \sum_j (i-j)^2 p(i,j)$$

Une texture apparaît plus nette lorsque le contraste est plus élevé.

5- La corrélation

$$COR = \sum_i \sum_j (i-\mu_x)(j-\mu_y) p(i,j)$$

Où : μ_x et μ_y représentent les moyennes respectivement des lignes et des colonnes de la matrice de cooccurrence.

6- La corrélation normalisée

Elle permet de déterminer si certaines colonnes de la matrice sont égales. Plus les valeurs sont uniformément distribuées dans la matrice de cooccurrence et plus la corrélation est importante.

$$COR_N = \sum_{i,j} \frac{ijp(i,j) - \mu_x \mu_y}{\sigma_x \sigma_y}$$

Où σ_x et σ_y représentent les écarts types respectivement des lignes et des colonnes de la matrice de cooccurrence.

7- L'entropie

Elle mesure la complexité de l'image. Lorsque les valeurs de la matrice de cooccurrence sont presque toutes égales, l'entropie est élevée.

$$ENT = -\sum_{i}\sum_{j} p(i,j)\log p(i,j)$$

Elle permet de caractériser le degré de granulation de l'image. Plus l'entropie est élevée et plus la granulation est grossière.

8- L'homogénéité ou moment différentiel inverse

$$HOM = \sum_{i,j} \frac{1}{1+(i-j)^2} p(i,j)$$

Ce paramètre à un comportement inverse du contraste. Plus la texture possède des régions homogènes et plus le paramètre est élevé.
On peut aussi définir les paramètres suivants qui sont cependant moins utilisés :

9- Le moment diagonal

$$DIAG = \sum_{i}\sum_{j} \left(\frac{1}{2}|i-j|p(i,j)\right)^{1/2}$$

10- Le cluster shade

$$CSHADE = \sum_{i}\sum_{j} (i+j-2MOY)^3 p(i,j)$$

11- Le cluster prominence

$$CPROM = \sum_i \sum_j (i + j - 2MOY)^4 p(i,j)$$

III-2- Les méthodes d'ordre supérieur

Elles étudient les interactions entre plusieurs pixels. Le voisinage est de type mono ou bidimensionnel. La méthode des longueurs de plages de niveaux de couleur est la plus souvent utilisée (Conners, 1980), (Chu, 1990), (Galloway, 1975).
Elle consiste à compter le nombre de plages d'une certaine longueur j, de niveau de couleur i dans une direction θ donnée (on prendra généralement comme valeurs 0°, 45°, 90°, 135°).
A chaque direction θ correspondra donc une matrice R (θ) définie par :

$$R(\theta) = [r(i,j)/\theta]$$

Avec r (i, j): nombre de plages de pixels de niveau de couleur i, de longueur j
θ: direction de la plage de niveau de couleur .
Une plage de niveaux de couleur correspond à l'ensemble des pixels d'une image ayant la même valeur de niveau de couleur.
La longueur de la plage correspond au nombre de pixels appartenant à la plage ; ainsi, on peut dire qu'une texture fine possède peu de pixels dans une plage.
Les principaux paramètres issus de la matrice des longueurs de plages sont les suivants :

1- Le poids des plages courtes

$$SRE = \frac{1}{nr} \sum_{i=1}^{M} \sum_{j=1}^{N} \frac{r(i,j)}{j^2}$$

Où

$nr = \sum_{i=1}^{M} \sum_{j=1}^{N} r(i,j)$: représente le nombre total de plages de l'image.

M : Correspond au nombre de niveaux de couleurs dans l'image et N à la longueur de la plage maximale.

2- Le poids des plages longues

$$LRE = \frac{1}{nr}\sum_{i=1}^{m}\sum_{j=1}^{N} j^2 r(i,j)$$

3- La distribution des niveaux de gris

$$GDIST = \frac{1}{nr}\sum_{i=1}^{M}\left[\sum_{j=1}^{N} r(i,j)\right]^2$$

Ce paramètre mesure l'uniformité de la distribution des plages

4- Le pourcentage de plages

$$RPC = \frac{nr}{np}$$

$$np = \sum_{i=1}^{M}\sum_{j=1}^{N} jr(i,j)$$

: correspond au nombre de pixels de l'image.

Plus la texture est homogène et plus ce paramètre est élevé.

5- La distribution des longueurs de plages

$$RLDIST = \frac{1}{nr}\sum_{j=1}^{N}\left[\sum_{i=1}^{M} r(i,j)\right]^2$$

IV- Conclusion

La texture joue un rôle très important dans l'identification et l'extraction des informations thématiques contenues dans l'image numérique.

Dans ce chapitre on a étudié quelques notions de base concernant les images numériques, puis les techniques nécessaires pour faire l'extraction des paramètres de textures, ces méthodes sont les méthodes statistiques et les méthodes d'ordre supérieurs...etc. Chaque type de méthode définit un ensemble des paramètres de textures.

Les paramètres extraits, a sont but de définit la nature de chaque image analysée. Pour exploité à partir des méthodes d'indexations dans le domaine de recherche d'image (exemple : la recherche de similarité...).

Structures et techniques d'indexation

I- Introduction

Pour faire une reconnaissance, une stratégie consiste à représenter chaque image par des vecteurs de descripteurs.

Pour faire une recherche pertinente par similarité dans des bases d'images, on doit choisir une structure d'indexation qui soit adaptée à nos besoins matériels et, à revoir au temps de réponse.

Il existe plusieurs techniques d'indexation de données, chacune utilise des concepts différents pour indexer les données [MAT 99].

II- Les différentes structures d'indexation

Elles sont classées en deux groupes

- ✓ La première regroupe les données en fonction de leur proximité.
- ✓ La deuxième découpe à priori l'espace multidimensionnel ensuite stocke les données selon ce découpage [AMS00].

II-1- B-Arbre

Cette structure a été fournie en 1970 par R.Bayer [BAY 71], elles sont surtout utilisées 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 réorientations courantes des données.

- **Principe**

Un B-Arbre est un arbre équilibré puisque 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 peut 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 m/2 descendant.
- ✓ Chaque nœud contient k clés, avec m-1≤ k ≤ 2m-1. sauf la racine pour laquelle 1 ≤ k ≤ 2m-1.
- ✓ Pour tout nœud non feuille, le i^{eme} fils contient des valeurs comprises entre la i^{eme} et la $(i+1)^{eme}$.

Comme l'arbre est équilibré alors toutes les feuilles apparaissent au même niveau.

- **Recherche**

Pour faire la recherche d'un élément quelconque à une clé donnée, tout d'abord on compare sa clé avec la racine et on détermine le fils correspondant.
S'il est un nœud, alors on continue sur le fils correspondant de la même façon.
Sinon si c'est une feuille un accès séquentiel aux éléments de cette feuille va déterminer si cet élément existe ou non.

- **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 ou l'élément à insérer aurait dominé se trouver s'il était dans la structure alors on essayera de l'insérer dans la feuille en question.

II-2- Quad –Tree

L'arbre quaternaire est une structure à précisions infinie ou chaque nœud peut avoir au maximum quatre fils, pour le stockage des données bidimensionnelles [LAM 98].

- **Principe**

L'arbre quaternaire (ou quadtree) est une structure de données non équilibrée construite par division récursive de l'espace en quatre quadrants disjoints, chaque nœud de l'arbre possède au maximum 4 fils.

L'arbre quaternaire est très utilisé dans le domaine des images, aussi bien pour le stockage, la compression, l'extraction de l'information sur le contenu des images, que pour la recherche d'images par le contenu.

Dans le domaine de la recherche d'images par le contenu, chaque image de la base de données peut être représentée par un arbre quaternaire équilibré complet dont chaque noeud stocke un descripteur visuel du quadrant correspondant dans l'image. Ce descripteur peut être un histogramme de couleurs, une forme ou un histogramme combinant à la fois la couleur et la texture. Ce type de structure est appelé un descripteur visuel Multi-Niveau.

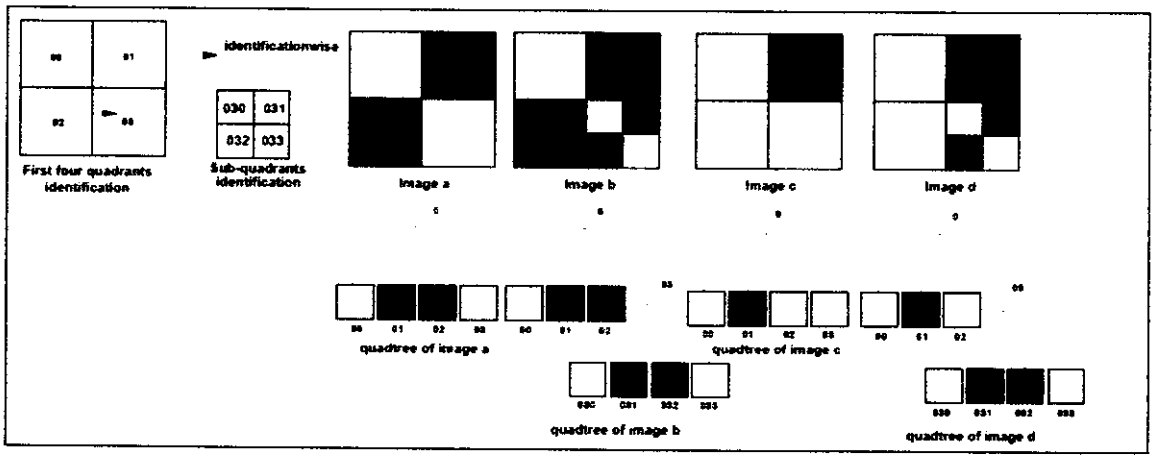
Un descripteur Multi-Niveau d'image peut être représenté par un arbre quaternaire stockant les descripteurs de tous les quadrants de l'image. Chaque noeud de l'arbre quaternaire peut stocker n'importe quel type de descripteur, ce dernier pouvant être compact pour éviter les problèmes liés aux données de grande dimension.

En outre, les descripteurs stockés dans les noeuds peuvent être différents en fonction du niveau des noeuds dans l'arbre quaternaire. Par exemple, la racine de l'arbre peut stocker un descripteur de couleur et ses quatre fils, représentant les quatre premiers quadrants de l'image, peuvent stocker des descripteurs de textures.

Un descripteur multi niveau permet également de faire de la recherche d'images par similarité de régions. Une région r est, dans ce cas, composée de plusieurs quadrants d'image choisis par l'utilisateur dans une grille 4×4 ou 16×16. L'utilisateur spécifie ensuite un vecteur de caractéristiques, puis un seuil pour chaque quadrant. Il peut également choisir les quadrants d'une image existant dans la base.

Le système détecte automatiquement le rectangle minimum englobant les quadrants sélectionnés par l'utilisateur. La requête qr ainsi déduite peut être transformée en plusieurs sous requêtes, chaque sous requête correspond par exemple à une transformation géométrique (rotation ou translation) du rectangle minimum englobant.

La figure ci-dessus présente une image classique récursivement découpée en quadrants pour être représentée par un descripteur Multi-Niveau. Les nœuds de l'arbre quaternaire et les quadrants correspondants dans l'image sont identifiés en suivant un ordre en Z, correspondant aux directions Nord-Ouest, Nord-Est, Sud-Ouest et Sud-Est. La racine de l'arbre est l'image entière est identifiée par 0. Les entiers 0 à 3, précédés de 0, identifient les quatre premiers quadrants de l'image et les quatre noeuds d'arbre quaternaire associés. De manière récursive, les noeuds (resp. quadrants) fils d'un noeud (resp. sous quadrants d'un quadrant), identifié par k, sont identifiés par kx avec x un entier prenant sa valeur dans {0 ; 1 ; 2 ; 3}.



Figurell.1 : Exemple d'un quad-tree

L'avantage de cette structure s'il s'agit d'une recherche exacte, est que les points stockés dans les nœuds peuvent appartenir à l'ensemble des données, ce qui diminue le temps de réponse d'une requête ; mais s'il s'agit d'une recherche de vecteur similaire, cet avantage ne compte pas et si les données sont stockées immédiatement de l'index, ceci donne une plus grande souplesse pour les opérations d'insertion et de suppression.

- **Recherche**

1- Recherche d'un point précis

Pour chercher un point précis dans un arbre quaternaire, on va comparer ses coordonnées avec les coordonnées de la racine, après avoir déterminé le fils correspondant au quadratique qui contient la requête, ensuite on compare de nouveau les coordonnées requête avec celles du nœud de fils, et ainsi de suite jusqu'au positionnement de la requête. Si on rencontre un pointeur vide, le point requête n'appartient pas à la base.
Si on dispose d'un ensemble de données contenu dans 4^n feuilles, on a besoin que de (n+2) accès pour la requête.

2- Recherche de points similaires

La recherche des points similaires à un point donnée, suit les deux critères suivants :

- ✓ Si ce nœud pointe sur une feuille alors on détermine les points contenus dans la zone requête [MAT 99].
- ✓ Sinon, pour chacun de ses fils dont le rectangle correspondant se chevauche avec la zone requête, et on lance récursivement cette procédure de recherche.

- **Insertion**

L'insertion dans un arbre quaternaire est difficile, parce qu'on peut avoir des cas spéciaux. Elle consiste à chercher le nœud dont le rectangle correspondant est le plus petit rectangle contenant ce point.
On distingue deux cas possibles

- ✓ Si ce point pointe sur une feuille déjà existante, alors le point est inséré, sinon s'elle est pleine, on partage le rectangle en quatre nouveaux quadratiques ; c'est

l'opération d'éclatement. Cette feuille va alors être remplacée par un nœud pointant sur quatre nouvelles feuilles et ses vecteurs seront insères dans ces feuilles.

- ✓ Si ce nœud ne pointe pas sur une feuille correspondant à ce quadratique, alors une nouvelle feuille est affectée et ce point est inséré.

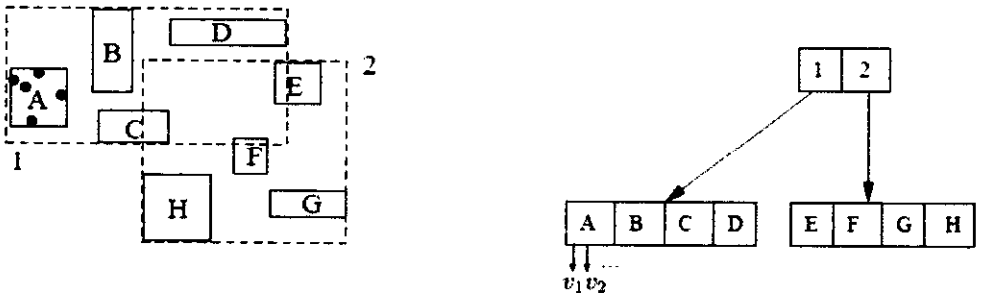
II-3- R-Tree

Le R-Tree est la première structure d'indexation multidimensionnelle basée sur la structure B-Arbre ; cette structure a été proposée par A.GUTTMAN.
Le R-Tree permet d'interroger les données qui peuvent être représentées sous forme de points ou de régions dans un espace de grande dimension [Ghem 03].

Principe

Cette méthode est basée sur l'approximation des objets complexes multidimensionnels par des RME_S (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 hyper rectangle à n dimensions.
Dans le R-Tree, les objets ne sont contenus que dans les feuilles, 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 correspond à ce RME, qui est le 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 ; la racine contient entre 2 et M fils.

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



Figurell.2 : Exemple d'un R-Tree

- **Recherche**

Cette structure utilise la même procédure que B-Arbre ; la seule différence entre les deux structures est que R-Tree compare les coordonnées des rectangles pour voir s'il existe une intersection ou non. La recherche est meilleure si le chevauchement entre les rectangles est minimum (Recherche optimale).

- **Insertion**

L'algorithme d'insertion doit déterminer la feuille hôte, car l'insertion d'un objet dans un R-Tree se fait au niveau des feuilles, en utilisant l'algorithme de recherche précédent ; plusieurs cas peuvent se présenter :
Si l'objet à insérer appartient à une seule feuille, alors en l'insère dans cette feuille.
Si cet objet appartient à plusieurs feuilles alors en 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 la plus petite augmentation de la surface.

II-3-1- Les variantes de R-Tree

Le R-Tree a été repris dans de nombreuses variantes telles que le Packed-Tree, R^+-Tree, le R^*-Tree et le X-Tree.

a- Le Packed-Tree

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-Tree.
Cette structure est utilisée dans les bases de données statiques.

b- Le R^+-Tree

Le R^+-Tree est développée par N.Roussopoulos et, et T.sellis, C.Faloutsos (Université de Maryland). Il consiste à éviter le chevauchement des régions en divisant si nécessaire les régions inférieures.
Il réduit efficacement le nombre de chemin lors de la recherche.

c- Le R^*-Tree

Le R^*-Tree est proposé par Beckmann, N.Kriegel, Schneider.R (Université de Maryland). Il est apparu après de nombreuses études et expériences.
Il utilise différentes stratégies dans les algorithmes d'insertion et de divisions.

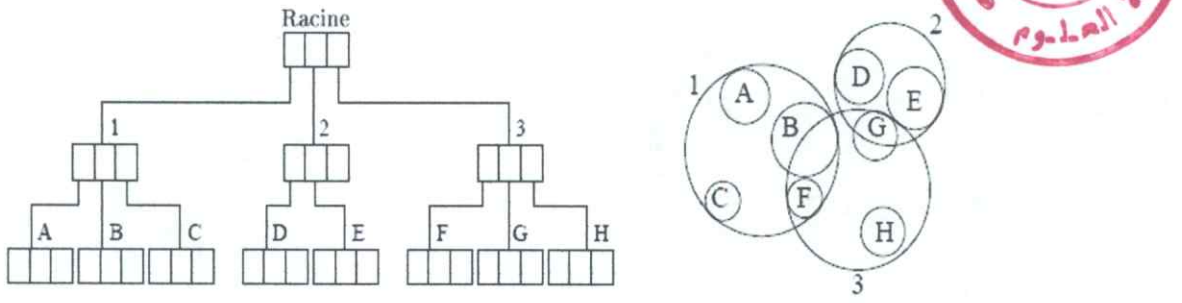
d- Le X-Tree

Le X-Tree(extented Tree), est une structure d'indexation qui supporte la recherche de similarité dans un espace de données multidimensionnel similaires au R-Tree, mais équipé d'une nouvelle stratégie de partitionnement de noeuds rejetant le fractionnement si celui-ci génère un taux de chevauchement trop élevé [GUT 84].
Le chevauchement entre les zones contenant les données RMEs est un grand problème qui apparaît dans les bases de données structurées par le R-Tree, ce qui indique une détérioration des performances avec l'augmentation de la dimension.
Le X-Tree comporte des super-nœuds qui sont des nœuds de taille variable.
Trois étapes ponctuent l'insertion dans un noeud saturé. Un algorithme de fractionnement similaire à celui du R-Tree utilisant les propriétés géométriques des formes englobantes est d'abord employé. Si le fractionnement proposé entraîne un chevauchement inférieur à un seuil fixé, le fractionnement a lieu et l'insertion se termine. Sinon, un autre fractionnement est calculé, mais dans ce cas, l'historique des fractionnements qui ont touché les formes englobantes du noeud est pris en compte : le but est de choisir la dimension selon laquelle la majorité des formes englobantes a été fractionnée, car cela minimise le taux de chevauchement.

e- Le SS-Tree

Le SS-Tree est une structure d'indexation qui utilise les mêmes principes que le R*-Tree. Il utilise toutefois des hypersphères englobantes au lieu des hyper rectangles. Le centre de la sphère est le centre de gravité des vecteurs englobés. La figure II.3 illustre un exemple de SS-Tree ainsi que la représentation géométrique de ses entrées. Lors d'une insertion, un point est assigné à l'hypersphère dont le centre est le plus proche. Quand une page (noeud ou feuille) est saturée, le fractionnement s'effectue selon la dimension suivant laquelle la dispersion des données est la plus

grande. Ceci permet de réduire le chevauchement entre hypersphères et d'améliorer les performances des requêtes du type plus proches voisins. Par ailleurs, une hypersphère est représentée par un centre (point multidimensionnel) et un rayon (valeur réelle), alors que pour représenter un hyper-rectangle deux points multidimensionnels sont nécessaires. Par conséquent, l'utilisation d'hypersphères permet d'accroître la capacité des nœuds et de réduire la taille de l'arbre [BER 04].



Figurell.3 : Exemple d'un SS-Tree

De plus, le SS-Tree emploie aussi la réinsertion des entrées d'un noeud saturé avant de le fractionner. Cette stratégie est toutefois modifiée: la réinsertion d'un sous-ensemble des entrées d'un noeud saturé se répète jusqu'à ce que le fractionnement soit évité, ou bien jusqu'à ce que toutes les entrées réinsérées se retrouvent dans le même noeud. Dans ce dernier cas seulement, le noeud est fractionné.

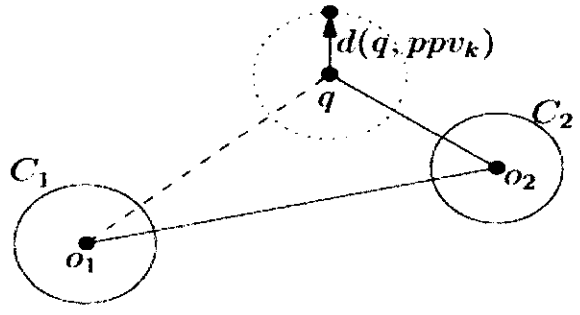
II-4- Le M-Tree

Le M-Tree est une structure d'indexation qui permet la recherche des similarités dans des espaces de grande dimension dotées d'une métrique [BER 04].

- **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 contraintes d'ordre, Seules, les informations liées au calcul des distances sont stockées dans les nœuds.
Cette structure est basée sur le regroupement des vecteurs dans des sous ensembles qui s'appellent des 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 de l'arbre est doté d'un pointeur vers son père sauf la racine, et des pointeurs vers des racines des sous arbres.



Figurell.4 : Exemple d'un M-Tree

- **Recherche**

La recherche dans un M-Tree commence par le nœud racine et consulte toutes les branches qui ne peuvent pas être exclues.
L'algorithme de recherche détermine pour chaque noeud parmi ses fils ceux dont l'hypersphère associée se croise avec la zone de recherche et pour chacun de ces fils la procédure est relancée récursivement jusqu'à atteindre les feuilles.

- **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 inséré.
- ✓ Si aucun nœud ne peut contenir le nouvel objet, on prend celui dont l'hypersphère correspondante va présenter après l'insertion la plus petite augmentation dans son rayon.

Conclusion

Sur les structures qu'on a présentées ; nous allons choisir la structure d'arbre quaternaire. Nous allons dans la partie qui suit présenter les principales techniques que l'on a rencontrées dans la littérature.

III- Techniques d'indexations

Plusieurs techniques d'indexations sont utilisées en base de données pour la reconnaissance d'objets.
Elles structurent les données en index pour accélérer les recherches de similarité entre l'image inconnue et les modèles de la base [BER 04].
Dans ce chapitre, nous présentons les techniques les plus importantes d'indexation qui sont basées sur les descripteurs de paramètres extraites à partir de l'image.

III-1- Indexation géométrique étendue

Cette technique a été proposée par Gros Iamiroy [LAM 96] [LAM 98], son principe est basé sur les données géométriques pour caractériser les images, elle utilise seulement deux quasi-invariants qui sont l'angle (θ) formé par deux segments adjacents, et leur rapport de longueur (ρ).
Le principe de l'algorithme de reconnaissance par indexation géométrique est de comparer l'image requête à chacun des modèles de la base et d'offrir un classement de ces modèles. Comme le coût d'exécution ne permet pas de parcourir tous les modèles et de les comparés l'un a un avec l'image requête, il faut alors organiser les quasi-invariants (ρ,θ) pour réduire la reconnaissance à une mise en correspondance entre deux images.
Pour chaque image, on calcule toutes les configurations de deux segments convergents. On calcule ensuite les quasi-invariants associés, formés par l'angle et le rapport de longueur des deux segments intervenants. On peut considérer que les couples de configurations qui ont des quasi-invariants associés proches forment des appariements possibles.

On établit par la suite une table d'indexation bidimensionnelle qui a comme clés les quasi-invariants précédemment calculée. Chaque cellule de la table pointe vers une liste de couples (vecteur d'invariants (ρ,θ), modèles correspondants).

Pour la reconnaissance, l'image requête est segmentée et ces descripteurs sont comparés avec les descripteurs stockés dans la table.

III-2- Techniques d'indexation évoluées

Elles contiennent de nouvelles techniques pour la recherche des plus proches voisins. Contrairement aux index multidimensionnels traditionnels, ces techniques ont été proposées en prenant en compte explicitement les problèmes liés aux grandes dimensions. Ainsi, toutes ces techniques proposent des idées originales leur permettant de mieux résister face à la malédiction de la dimension. Nous présenterons le Pyramid-Tree qui se base sur le principe du partitionnement de l'espace, mais qui utilise une stratégie de partitionnement permettant de créer un nombre de cellules dont le nombre croît linéairement et non pas exponentiellement lorsque la dimension augmente. Nous présenterons également le VA-File qui est une méthode basée sur la compression des données, mais qui, au final, peut être considérée comme une amélioration de la recherche séquentielle [BER 04].

a- Le VA-File

Cette technique très efficace pour les grandes dimensions, a été proposée par Weber, Schek et Blott [WEB 98], pour améliorer les performances de recherche séquentielle.

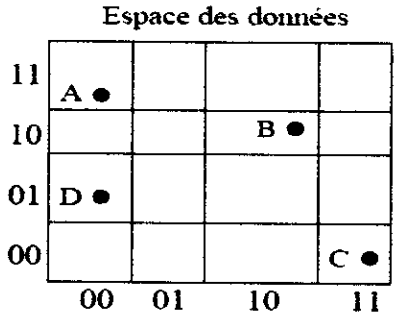
Le principe de base du VA-File est basé sur la gestion de deux ensembles de données : un fichier qui contient tous les vecteurs de la base, et un autre qui contient des approximations géométriques de ces vecteurs. La taille du fichier d'approximation est plus petite que celle du fichier de données [WEB 98].

Lors d'une interrogation, un premier parcours séquentiel du fichier d'approximation permet de sélectionner les vecteurs qui ont le plus de chances d'appartenir à l'ensemble résultant. Après, l'accès au fichier de données est effectué sur la base des résultats issus de la première phase. La recherche séquentielle s'effectue sur le fichier d'approximation, celle-ci est donc très rapide et n'entraîne ensuite l'accès à la base que pour un sous-ensemble réduit de vecteurs.

Ce procédé diminue donc le nombre d'opérations d'E/S et le coût de calcul CPU par rapport à la recherche séquentielle qu'elle analyse la totalité de la base.

Pour calculer les approximations géométriques, chaque dimension d est partitionnée en 2^b intervalles où chaque intervalle est codé sur b bits.

Les intervalles sont calculés de façon à contenir le même nombre de vecteurs. De cette façon, à chaque cellule, ainsi qu'aux vecteurs qu'elle contient, est attribué un code binaire de longueur $\sum^d = 1^b$. La figure ci-dessus illustre un exemple de codage de vecteurs définis dans un espace de dimension deux.

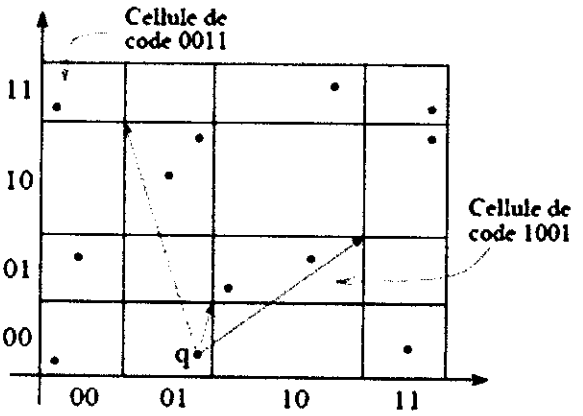


Approximations

A	0011
B	1010
C	1100
D	0001

Figurell.5 : Création du fichier d'approximation

Le fichier d'approximation VA-File est composé des approximations de paires (identifiant de descripteur, numéro de case).



Figurell.6 : Principe de recherche dans le VA-File

Lors de la recherche, le descripteur requête subit le même processus d'approximation. L'algorithme détermine les cases (hypercubes) qui s'ont 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 opère comme un filtre qui élimine de la recherche les hypercubes n'ayant aucune chance de faire partie de la réponse, en limitant le nombre de comparaisons à faire par rapport à la recherche séquentielle.

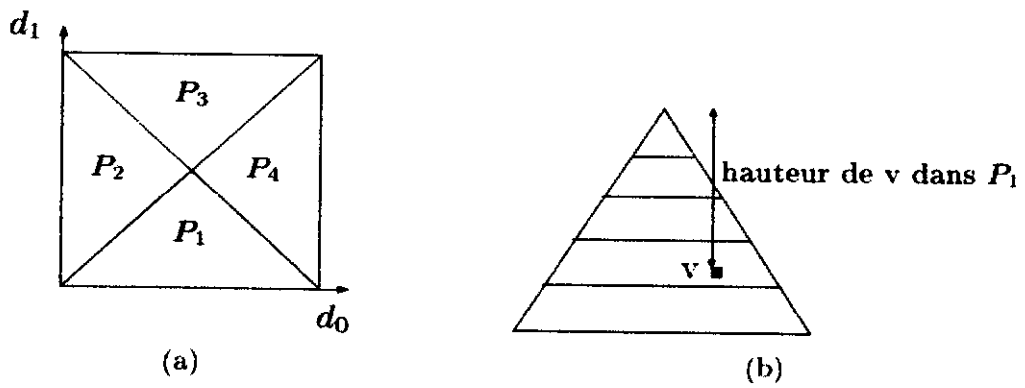
On peut dire que l'objectif principal de VA-File est :

Le filtrage des données par approximation en mémoire centrale, La recherche séquentielle et exhaustive dans l'espace de recherche approché.

b- Le Pyramid-Tree

Le Pyramid-Tree a été proposée par Berchtold, Bohm et Kriegel [BER 98], 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 $[0,1]^d$. Cet espace est ensuite divisé en deux étapes :
Au début, il est découpé en 2d pyramides. Chaque pyramide est numérotée 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.
Ce double découpage permet à tout point x de l'espace d'être exprimé comme une paire (i,h_x).
Ou : i est le numéro de la pyramide à laquelle appartient le point, et h_x la hauteur de ce point dans cette pyramide.
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 réelles 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 disjoints.



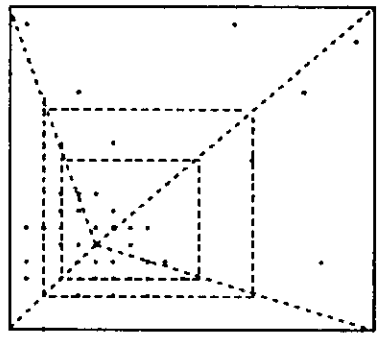
Figurell.7 : Principe du découpage du Pyramid-Tree

Ayant la transformation déterminant la valeur pyramidale d'un point, l'étape d'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 correspondants puisque la transformation vers la valeur pyramidale n'est pas injective. Le problème de stockage de vecteurs d-dimensionnels se réduit donc 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 :

$$(x_{1min}, x_{1max}), (x_{2min}, x_{2max}), \ldots, (x_{dmin}, x_{dmax}).$$



Figurell.8 : Modélisation de la requête (Espace 3D)

Les vecteurs contenant 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. 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-3- Le Hachage

C'est une technique de stockage et de recherche de l'information dans une table de hachage en utilisant des indexes entiers issus d'une fonction de hachage *h* à partir des clés définies par les invariants [CAR 97].
Le hachage consiste à calculer l'adresse entière, de l'emplacement d'un élément dans la table de hachage à partir de sa clé, définit par une fonction de hachage *h*.

- **Principe**

Son principe consiste à répartir les éléments dans une Table T, l'adresse de la Table ou chercher l'élément est calculé à partir de la clé de l'élément. Les méthodes de hachages consistent à rechercher une clé parmi n clés, en un nombre constant d'opérations.
Si m est le nombre de places de T, les places de T sont numérotées de 0 à m-1.
On associe à chaque clé une adresse dans la table T calculée à partir de la fonction de hachage *h*.

$$h(\text{clé})=\text{adresse}(T).$$

Le hachage est un concurrent de l'arbre B^+, il est meilleur pour la recherche par clé, et n'occupe aucune place. Malheureusement, on peut avoir deux clés de même adresse dans la table (Collision).

a- Différents types de hachage

Il existe deux types de hachage [CAR 97] :

1- Hachage statique

La table de hachage de taille fixe, dans laquelle les éléments sont placés dans des emplacements dont l'adresse est calculée à l'aide d'une fonction de hachage fixe appliquée à la clé.

- **Avantages du Hachage statique**

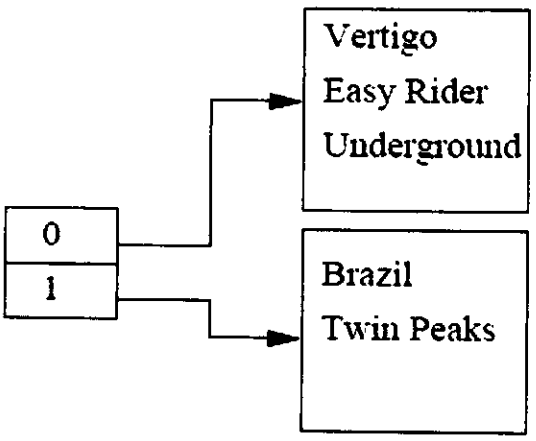
✓ Accès sélectif sur la clé de placement est très performant (sans débordement).
✓ Bien adapté pour des relations de taille moyenne à faible croissance.
✓ La solution d'organisation du Hachage statique ce fait par Hachage dynamique.

2- Hachage virtuel

Son intérêt est d'agrandir l'espace d'adressage avec les insertions des éléments. Il se compose en 2 types :

- **Hachage dynamique**

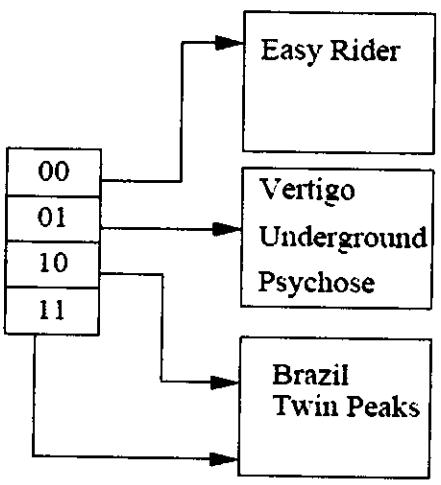
Ce type réorganise la table de hachage en fonction des insertions et des suppressions ; le nombre d'entrées dans les répertoires est une puissance de 2.
La fonction de hachage *h* donne toujours un entier sur 4 octets (32 bits).
On utilise les n premiers bits des résultats de h avec n<32.
La construction de la table de hachage utilise au départ seulement les premiers bits de la fonction *h*, donc on a 2 valeurs possibles (0 et 1), deux entrées possibles alors on a deux blocs nécessaires pour l'affectation d'un enregistrement.
Généralement, l'affectation d'un enregistrement dépend de l'ancien enregistrement dans la table et la fonction d'affectation (fonction de hachage).



titre	*h*(titre)
Vertigo	01110010
Brazil	10100101
Twin Peaks	11001011
Underground	01001001
Easy Rider	00100110
Psychose	01110011
Greystoke	10111001
Shining	11010011

Figurell.9 : Exemple d' Hachage dynamique

Cette représentation est limitée, car si on a 3 titres par blocs, l'insertion du titre Psychose (valeur : 01110011) entraîne le débordement du premier bloc, qui nécessite alors de doubler la taille du répertoire (deux bits) pour allouer un nouveau bloc de l'entrée 01 du psychose.



Figurell.10 : Exemple d'un table de Hachage

- **Hachage extensible**

Le hachage extensible utilise une fonction de hachage fixe *h*, qui appliquée à une clé *K* renvoie une pseudo-clé *K0* = *h*(*K*).

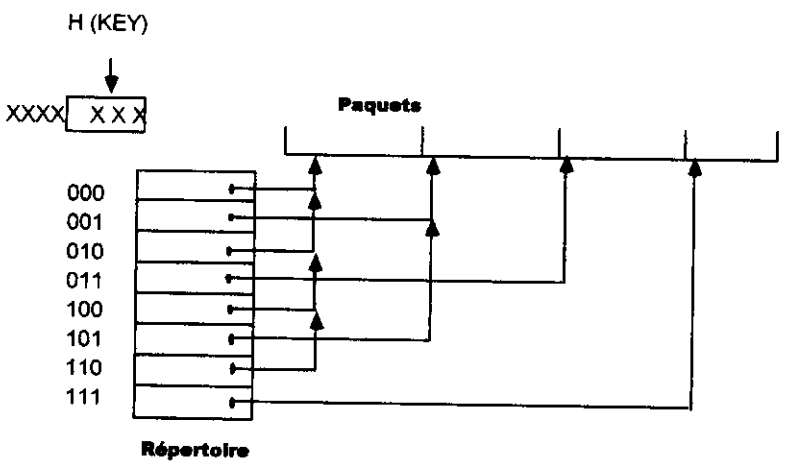
Les pseudo-clés sont les indices d'un tableau ; le répertoire central, contient des pointeurs sur la page contenant les données. L'entête du répertoire contient sa profondeur *p*, qui détermine le nombre de pseudo-clés 2^p.

Une même page peut avoir plusieurs entrées dans le répertoire, plusieurs pseudo-clés. La profondeur *p0* d'une page définit le nombre de pseudo-clés qui lui sont associées.

- **Avantages**

✓ Accès sélectif sur clé de placement toujours performent.

✓ Adapté à des relations à forte dynamique.

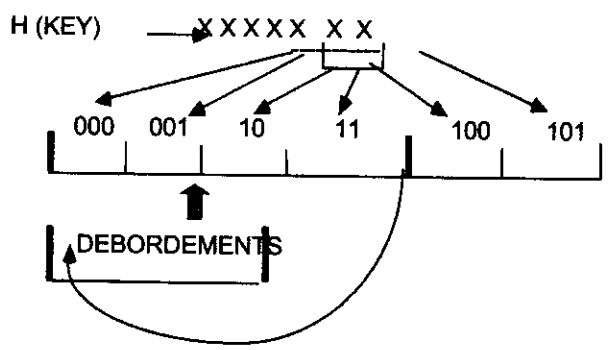


Figurell.11 : Exemple sur le Hachage extensible

- **Hachage linéaire**

Ce type de hachage a été introduit par LITWIN en 1980. Dans ce type de hachage, on fait l'extension ou la réduction dynamique du nombre de blocs d'accès occupés par un fichier sans utiliser de structure d'accès additionnelle (répertoire), comme c'était le cas pour les deux techniques précédentes.

On suppose que le fichier initial occupe M blocs numérotés de 0 à M-1, et que la fonction de hachage utilisée est la fonction modulo (i.e. h (K) = K MOD M).



Figurell.12 : Exemple sur le hachage linéaire

IV- Conclusion

Dans ce chapitre, nous avons présenté plusieurs structures d'indexation, qui vont servir comme des outils de stockage de base pour les techniques d'indexation.
Nous avons par la suite présenté les différentes techniques d'indexation pour étudier la similarité entre les images, et on a choisi comme technique l'hachage, car :

- ✓ Elle suit une technique d'affectation bien définie (la fonction d'hachage).
- ✓ Accès sélectif sur clé de placement toujours performent.
- ✓ Facilite la recherche d'un élément dans la table de hachage.

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.

Architecture et mise en œuvre d'un système d'indexation

Dans la plupart des systèmes de recherche d'image, l'indexation est un procédé de choix puisqu'elle offre un parcourt rapide dans des bases d'images qui s'agrandissent considérablement au fil des utilisations. Le choix de l'arbre quaternaire joue un rôle important dans la clarté de la représentation et le volume des informations parcourt, et aussi le choix de la technique de hachage qui améliore la rapidité du parcourt des images.

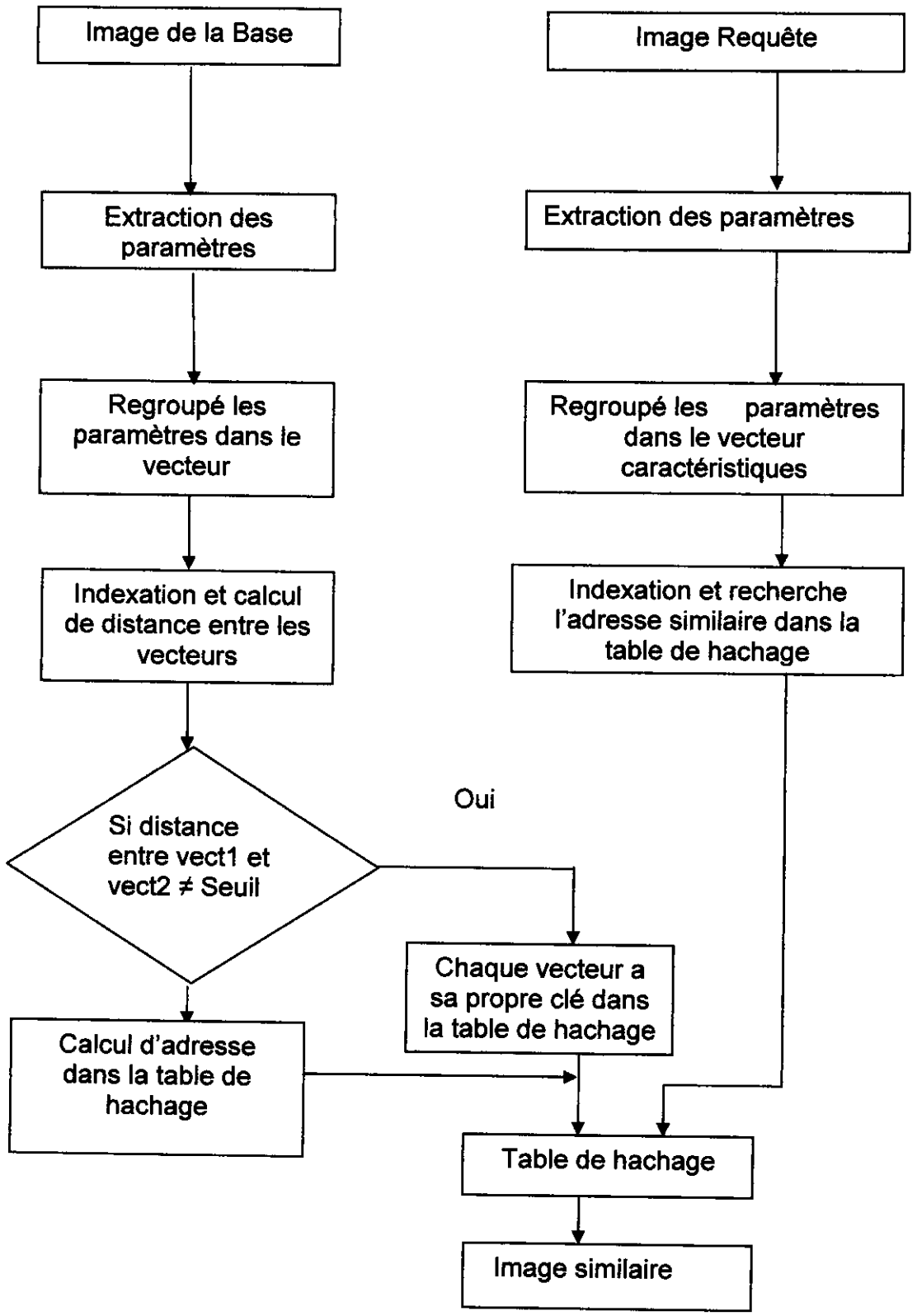
L'introduction du terme base d'images nous oblige à choisir un système de modélisation qui soit le plus standardisé de nos jours. UML offre la possibilité de représenter notre système de différents points de vus possible avec plusieurs niveaux de détails.

I- Architecture du système d'indexation

Dans ce qui suit, nous allons détailler l'architecture de notre système d'indexation, comment il structuré et comment le mettre en œuvre dans notre application.

Ensuite, nous allons essayer de modéliser notre base suivant le langage UML.

Le schéma suivant représente notre système :



FigureIII.1 : Schéma synoptique du système d'indexation

Les principales fonctions de notre système de recherche sont les suivantes :

1- L'analyse et la représentation du contenu de l'image

- ✓ Les données source mises sous forme de vecteur caractéristique. L'information obtenue dans cette étape est une sorte de résumé des images de la base texture. La transformation est généralement gourmande en temps de calcul parce que les images sont traitées d'une manière séquentielle dans la base d'images. Mais elle est exécutée en temps mort (off line).
- ✓ L'analyse et la représentation du contenu de l'image suivent des méthodes bien définies, plusieurs méthodes en été élaboré pour définir ces caractéristiques :

a- Les méthodes statistiques : Qui divise regroupe deux grandes méthodes d'analyse

- ✓ Méthodes du premier ordre (Analyse par des histogrammes.).
- ✓ Les méthodes du deuxième ordre (matrice de cooccurrence).

b- Les méthodes d'ordre supérieur :(ces méthodes n'est pas intéressons pour notre étude).

2- Extraction des paramètres

Pour méthode d'extraction, nous avons déjà choisi ***la matrice de cooccurrence*** (qui est déjà expliqué dans le chapitre I), Les matrices de cooccurrences contiennent les moyennes d'espace du second ordre. Quatorze indices peuvent être calculés à partir de ces matrices (définis par Haralick) qui correspondent à des caractères descriptifs des textures. Nous avons prendre trois indices qui donnent le meilleur résultat de recherche (réduction de paramètres) car, si on prend les quatorze paramètres notre descripteur va être de dimension quatorze est notre étude va très difficiles, en plus de ça l'augmentation de dimension de vecteur descripteur influe sur les techniques d'indexation et le rendre inutilisable et la performance sera réduite.

Pour que les techniques d'indexation fonctionnent bien, on a donc besoin de diminuer la dimension de vecteur avec l'élimination des paramètres moins importants.

L'algorithme de la matrice de cooccurrence, est donné par:

Algorithme Matrice de cooccurrence

Variables d'entrée :

W : Masque centré autour d'un pixel de l'image (**W** est
Une matrice de taille Tx par Ty) ;

L : Nombre maximal de niveaux de gris de l'image ;

Variable de sortie :

MCOOC : Matrice de cooccurrence ;

Variables locales :

i, j, x, y : **Entiers** ;

Début

Pour j variant de 0 à L-1 **Faire**

Pour i variant de 0 à L-1 **Faire**

MCOOC [i, j] ← 0 ;

Pour y variant de 0 à Ty-1 **Faire**

Pour x variant de 0 à Tx-2 **Faire**

Début

I ← **W**[x, y] ;

J ← **W**[x+1, y] ;

MCOOC [I, J] ← **MCOOC** [I, J] + 1;

Fin

MCOOCs ← Symétrique de **MCOOC** ;

Pour j variant de 0 à L-1 **Faire**

Pour i variant de 0 à L-1 **Faire**

MCOOC [i, j] ← ((1/2)*(**MCOOC** [i, j] +**MCOOCs** [i, j])/ (2*Ty*(Tx-1)));

Fin.

Figure III.2 : Construction de la matrice de cooccurrence

Avec la matrice de cooccurrence, Haralick introduit 14 paramètres.
Notre choix comme paramètres est :

✓ **La moyenne :**

$$MOY = \sum_i \sum_j p(i,j)$$

✓ **Le contraste :**

$$CONT = \sum_i \sum_j (i-j)^2 p(i,j)$$

✓ **L'énergie :**

$$E = \sum_{i,j} (p(i,j))^2$$

Ou : p (i, j) est la Probabilité de passer d'un pixel de niveau de couleur i à un pixel de niveau de couleur j.

Bien que ces indices sont corrélés, la réduction de l'information contenue dans la matrice de cooccurrence et permettent une meilleure discrimination des textures.

3- Regroupement des paramètres

Les paramètres obtenus sont regroupés dans un vecteur caractéristique qui sera l'identité de l'image dans la base. Le descripteur a la forme suivante :

Seuil	Moyenne	Contraste	Energie	Id-Image

FigureIII.3 : La forme d'un descripteur

Seuil : Pour mesurer la similarité.

Id-Image : Numéro de l'image.

Remarque : ces trois étapes sont appliquées pour les images de la base ainsi que pour l'image requête.

4- Indexation et le stockage des caractéristiques

Elle devra être dynamique pour répondre â la variation en taille des bases et rapide pour les nécessités de type temps réel. Ces contraintes requièrent une stratégie de recherche pour naviguer efficacement en bases de données. Les techniques modernes de gestion de base multimédia incorporent de tels concepts.
La mise en place du système nécessite une étape préalable d'indexation des images de la base, afin d'extraire les descripteurs qui serviront à classifier la base dans le cadre de la navigation et à évaluer la similarité entre images dans le cas de la recherche directe. Cette phase est divise en trois étapes :

Étapes1 : Insertion des images dans la base de données :
L'insertion des images ce fait à l'aide de l'algorithme suivant :

```
Algorithme Insertion_Image (Image)
 Début
        Pour tout Image Faire
       // Insertion d'image dans la base
            Insérer_Image (Image) ;

        Pour chaque Image Faire
            Insérer_dans_la_base (NImage, Nom Image) ;
            Découper_Image (Image) ; // découpage de l'image en quadrants
            Calculer_Paramètre (Matricecooc) ;
            Calculer_Descripteur (Paramètre) ;
            Insérer_Arbre (Descripteur) ; // insertion descripteur dans l'arbre
quaternaire

 Fin.
```

Figure III.4 : Insertion des images dans la base

Étapes 2 : Insertion des descripteurs dans la base

Les paramètres extraits dans la 1ére phase sont regroupés dans un vecteurs « vecteur caractéristique », donc notre vecteur contient trois cases ou chaque case correspond aux trois paramètres moyenne, contraste, énergie. Par ailleurs, chaque image est découpée en quatre quadrants suivant la structure d'arbre quaternaire, chaque nœud a quatre fils. « Chaque nœud est alloué par un vecteur descripteur correspond au quatre quadrants de même niveau ». L'algorithme suivant donne une définition générale pour ce mécanisme :

```
Algorithme Insertion_ Arbre (Descripteur)
Début
    Nœud :=Racine ;
      Tant que ((Nœud<>NULL) et (Nœud <>Feuille)) Faire Avancer dans
l'arbre ;
            Si (Nœud= Feuille) alors  Insérer (Descripteur) ;
              Début
                  Si (Feuille non pleine) Alors Insérer (Descripteur)
                     Sinon
                        Eclatement (Feuille);
                        Insérer (Descripteur);
                     Fin ;
                Fin ;

            Sinon
                     Allouer_Espace (Nouvelle_Feuille);
                     Insérer (Descripteur) ;
             Fin ;
        Fait
Fin.
```

Figure III.5 : Insertion des descripteurs

Étape 3 :L'indexation des vecteurs caractéristiques

Pour améliorer le temps de recherche d'images similaires dans la base d'images, il existe plusieurs méthodes d'indexation. Nous avons choisi comme méthode le Hachage (Chapitre II). La fonction de hachage doit associer aux vecteurs

caractéristiques une valeur entière (leur adresse dans la table de hachage), nous allons définir les notions suivantes :

a- Table de hachage

Les tables de hachages sont des structures de données composées de tableaux statiques et possédant des algorithmes de recherche très performants (en O (1)).En effet, la position de l'élément dans le tableau est en fonction de l'élément lui-même.

Clé : Une clé est une partie d'un élément qui permet de désigner le contenu de cet élément de manière non ambiguë.

b- Fonction de hachage

- Définition

Soit ***E*** l'ensemble des clés possibles, et ***F*** l'ensemble des indices du tableau. Une fonction de hachage **H** est une fonction qui associe à toute clé ***K*** un indice dans le tableau :

H : $E \rightarrow F$

H (***K***) = ***i***

En pratique, Il est difficile de trouver une bonne fonction de hachage, car : Une fonction de hachage doit être injective, sinon on a H (K1) = H (K2) = i et 2 éléments sont stockés au même indice. Quand elle existe, elle est alors parfois complexe et le calcul H (K) peut être coûteux.

c- La Collision

- Définition

Le cas ou plusieurs clés donnent le même indice dans la table de hachage résulte en une collision.

Pour régler ce problème, on a deux méthodes possibles :

a- Le chaînage externe

On déclare un tableau de pointeurs au lieu du tableau d'éléments tab[i] qui contiendra la liste des éléments dont les clés K_i ont la même image par H.

➢ **Avantages de cette méthode**

✓ Un seul tableau de pointeurs.

➢ **Inconvénients de cette méthode**

✓ liste chaînée.

✓ la recherche d'un élément n'est plus immédiate.

b- Le tableau de collisions

On augmente la taille du tableau de hachage et on ajoute un tableau supplémentaire en parallèle du tableau de hachage pour gérer les collisions.
Si 2 éléments E1 et E2 sont en collisions (H (K1)=H (K2)=i), alors on a :
Tab[i] = E1 et col [i] = i', i' est l'indice tel que tab [i'] = E2, Sinon tab[i] = E1
et col[i] = -1.

✓ Notre fonction de Hachage est définie comme suit :

On fait un regroupement d'images similaires, ou chaque groupe de la base est pointé sur une adresse de la table, dans notre table de hachage il n'existe pas de collusion, car on a utilisé le chaînage arrière pour attribuer les adresses à chaque groupe d'image.

On a commencé par mesuré la distance entre les images par rapport au seuil et chercher les images similaires, puis affecter l'adresse première pour le premier groupe, puis les autres adresses sont calculer à partir de l'adresse origine on ajoutant un.

Adresse courant=ancien Adresse+1.

c- Mesure de similarité

Pour mesurer la similarité entre deux Images x et y (ou bien entre un image x et une requête y) représentés par des vecteurs multidimensionnels $x = (x_1, x_{2,\dots}, x_n)$ et $y = (y_1, y_{2,\dots}, yn)$, on a coutume de prendre l'inverse ou l'opposé d'une distance comme les distances L_P.

$$L_p(x,y) = \left(\sum_{j=1}^{n}(x_j - y_j)^p\right)^{\frac{1}{p}} \qquad p \in [1, \infty[$$

Pour $p=2$, elle correspond à la distance euclidienne qui est la plus utilisée.
La mesure de similarité se fait à l'aide de deux types de requêtes :
Soit : B : Base d'image, i : Image de la base, q : Image requête, S : Seuil,
Q : Résultat.

- ✓ **Requêtes d'intervalle** : Q= {i/ L_P (q, i) ≤S, i appartient a B}.
- ✓ **Requêtes de voisinage** : Quelque soit : i appartient à Q, Quelque soit : j n'appartient pas à Q : L_P (q, i) ≤ L_P (q, j).

d- Déduction du seuil

Notre base d'images contient 30 images. Le seuil est obtenu en fonction de notre base, et notre application.

- ✓ On compare la distance L_P (x, y) avec le seuil calculé à partir de L'image de base :

Si la distance L_P (x, y) du niveau zéro approximativement égal au seuil, alors les deux images sont similaires, et **H** (descripteur1)=**H** (descripteur2). Les deux descripteurs pointent sur la même adresse dans la table.

Sinon on passe au niveau un, ou l'image est divisée en quatre quadrants, chaque quadrant est associé à un descripteur. Si la distance L_P (x, y) de niveau un est approximativement égal au seuil, on dit que les deux images sont similaires et les deux descripteurs associés aux deux images, pointent sur les mêmes adresses de la table de hachage. On va répéter cette instruction jusqu'au niveau deux qui est le

dernier niveau. Donc, l'idée de base consiste à transformer les vecteurs en utilisant une fonction de hachage. Chaque transformation vise à maximiser la probabilité de collision entre les vecteurs proches dans l'espace. Le but de chaque transformation est ainsi d'attribuer aux vecteurs similaires (ou proches) une même valeur. Cette valeur est insérée dans une table appelée « table de hachage » pour effectuer des recherches de plus proches voisins approximatives puisque seuls les vecteurs de la base ayant au moins une valeur transformée commune avec le vecteur requête sont considérés lors d'une recherche.

```
Algorithme Insertion_Table (Arbre [], Nombre Max, Distance, H, Seuil)
Début
Booléen Même Niveau ;
Niveau []= {0, 1, 2}
1 : Pour i=1 à (Nombre Max-1) Faire
      Pour j=i+1 à Nombre Max Faire
            Pour chaque Niveau Faire
                  Si Même Niveau
                     Calculer_Distance (Arbre[i], Arbre [j]) ;
                  Sinon Même Niveau =False ; Aller a 1 ;
2 : Pour i=1 à Nombre Max Faire
      Pour j=i+1 à Nombre Max Faire
            Pour chaque Niveau Faire
                  Si Distance (Arbre[i], Arbre [i+1] <=Seuil)
                     A= {Arbre[i], Arbre [i+1]},
                     H (A)= Zéro; // La première adresse de la table
                  Si non
                     A= {Arbre[i]}; B= {Arbre [i+1]};
                  H (A)= la première adresse de la table ; // chaînage arrière
                  H (B)= la première adresse de la table+1 ;
      Aller a 2 ;
Fin.
```

Figure III.6 : Construction de la table de Hachage

Algorithme de recherche dans la table de hachage

```
Algorithme Recherche_Table (Desc_ImageReq, Table [])
Début
        Pour chaque Arbre_ImageReq Faire
              Pour i=1 à Nombre Max Faire
                Si Calculer_Distance (Arbre_ImageReq, Arbre [i] <=Seuil) ;
              Afficher toutes les images de groupe (Liste) qui appartient l'image i
plus que l'image i ;
              Afficher_Sim (Groupe (i)) ;
                Si non
              // Pas d'images similaire
Fin.
```

Figure III.7 : Recherche de similaire

5- L'analyse et représentation de la requête utilisateur sous une forme compatible avec celle de la base. Cette opération est analogue â celle de la première étape, mais appliquée seulement à l'image requête.

6- La mesure de similarité entre les images requête et source, mesure généralement basée sur la distance euclidienne.

7- L'interface utilisateur : C'est la vitrine du système permettant la présentation des résultats ainsi que l'interaction avec l'humain. Elle doit être intuitive et simple pour offrir à l'usager le confort nécessaire à une utilisation souple et efficace.

Notre système de recherche est schématisé comme suit :

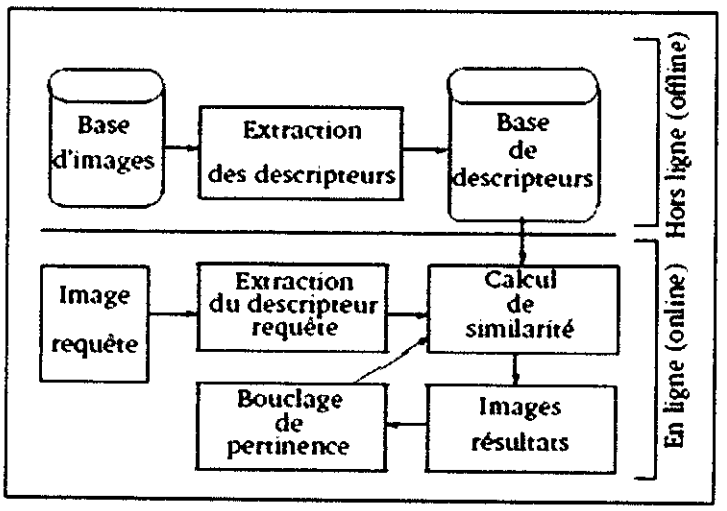


Figure III.8 : Structure de la base de recherche

On distingue deux phases indépendantes :

1- Une phase **hors ligne** (*Off Line*) : dans laquelle sont réalisés l'extraction et le stockage des descripteurs des images de la base. Durant cette phase, aucun utilisateur n'est connecté à la base d'images. Cette phase peut donc prendre le temps nécessaire à l'évaluation des descripteurs.

2- Une phase **en ligne** (*On Line*) : où l'utilisateur interroge la base à l'aide de son image exemple. Durant cette seconde phase, le temps de réponse du système est crucial, il faut le réduire au maximum.

Cette architecture générale présente cependant deux inconvénients majeurs :

- ✓ L'utilisateur ne dispose pas toujours d'une image exemple pour formuler sa requête.
- ✓ Lorsque le nombre d'images de la base augmente, le temps de recherche devient rapidement démesuré.

Nous utilisons l'UML comme langage efficace pour faire la conception de notre système de recherche par similarité.

II – UML un outil de modélisation

1- Définition

UML (Unified Modeling Language) est un outil de modélisation orienté objet, unifié. Il résulte de l'effort de standardisation de plusieurs méthodes de conceptions orienté objets (la méthode OMT de Rumbahgh, la méthode BOOCH' 93 DE Booch, et la méthode OOSE de Jacobson), dans le but de garantir la compatibilité entre les applications programmées fonctionnant sur des réseaux hétérogènes. [Mul 97]

En 1994, G Booch et J.Rumbaugh ont décidé d'unifier leurs méthodes au sein de la société Rational Software. Un an après I. Jacobson a rejoint Rational Software pour travailler sur l'unification. Les travaux sur UML ont continué avec sont adoption par de grands acteurs industriels dont HP, Microsoft, Oracle, et Unisys. Le travail a abouti en janvier 1997 à UML 1.0. [Ket 01].

2-Historique

- **Le schéma suivant illustre l'historique d'UML**

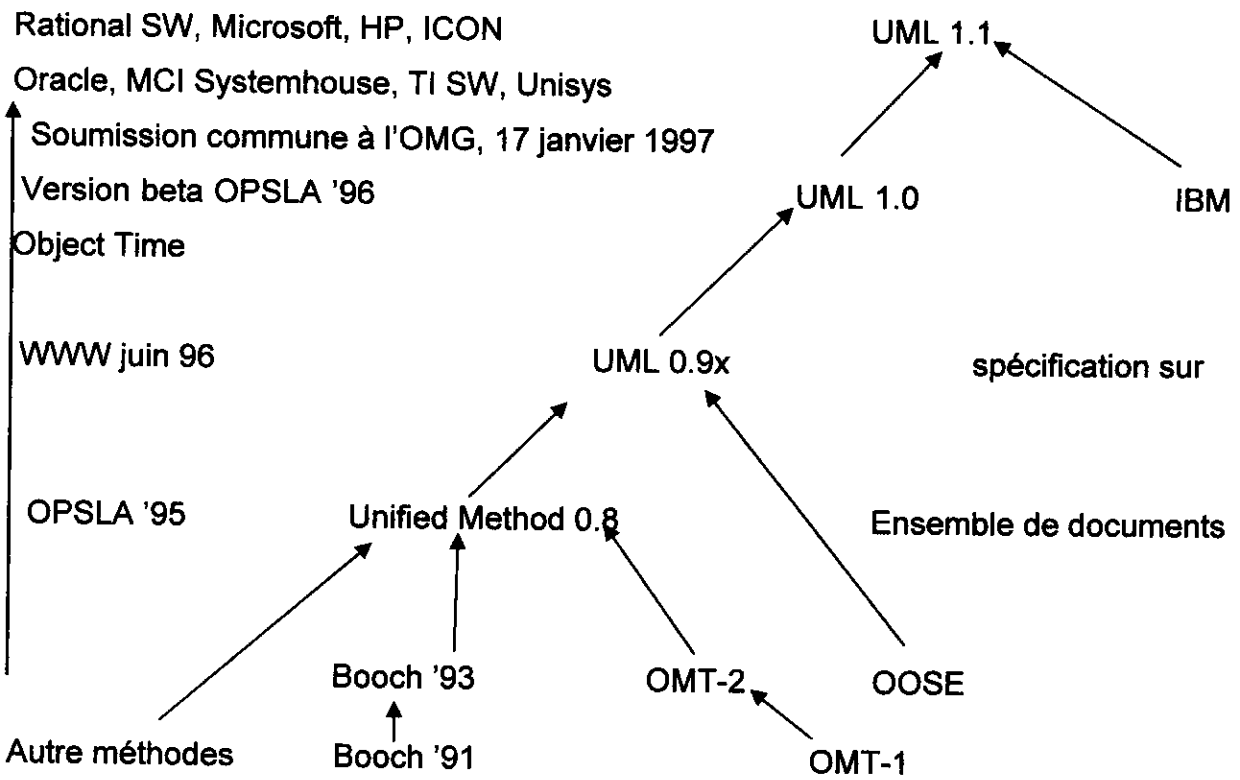


Figure III.9 : Historique d'UML [De Merise à UML97]

3- Notion UML

UML définit neuf diagrammes pour représenter les différents points de vue de modélisation. L'ordre de présentation de ces différents diagrammes ne reflète pas un ordre de mise en œuvre dans un projet réel, mais simplement une démarche pédagogique qui essaie de minimiser les prérequis et les références croisées. Ces neuf diagrammes sont : diagramme d'objet, diagramme de déploiement, diagramme de collaboration, diagramme de composition, diagramme de séquences, diagramme de cas d'utilisation, diagramme d'états transitions, diagramme de classes,

Diagramme d'activités. Les diagrammes précédemment situés sont classes dans deux grands modèles :

- ✓ **Modèle statique :** comporte les diagrammes suivants :

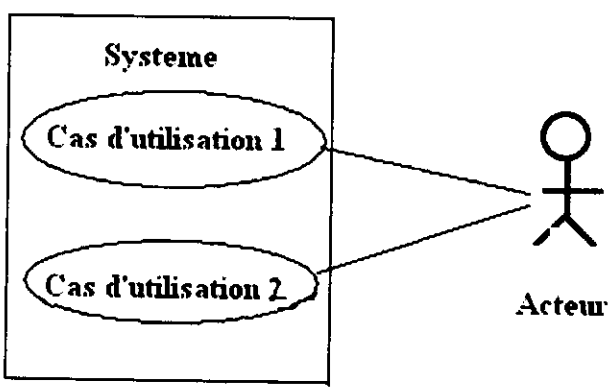
Diagramme de classes, diagramme de déploiement, diagramme de composition, diagramme de cas d'utilisation, diagramme d'objet.

- ✓ **Modèle dynamique :** comporte les diagrammes suivants : diagramme de collaboration, diagramme de composition, diagramme de séquences, diagramme d'activités.

a- Diagramme des cas d'utilisations

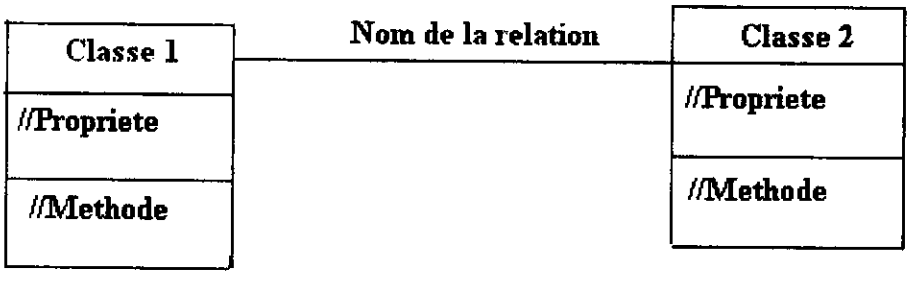
Un cas d'utilisation décrit le comportement d'un système de point de vue d'un utilisateur. Il permet de définir les limites du système et les relations entre le système et l'environnement. Un diagramme des cas d'utilisation comprend les cas d'utilisations qui sont les activités qui symbolisent les étapes d'un processus, et les rôles qui accomplissent ces activités.

Les acteurs sont représentés sous la forme de petits personnages qui déclenchent des cas d'utilisations ; ces derniers sont représentés par des ellipses.



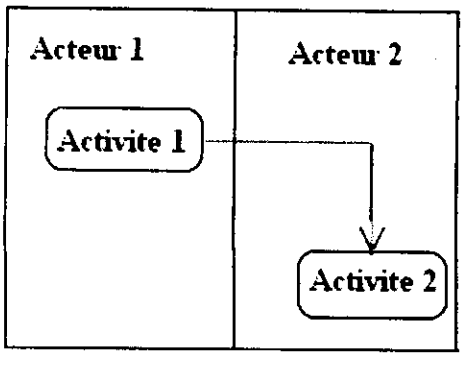
b- Diagrammes de classes

Un diagramme de classes exprime les liens d'un objet donné, qui transite par les activités et chemins, et qui subit des transformations, avec d'autres objets.
Une classe est représentée par des rectangles compartimentés. Le premier compartiment contient le nom de la classe. Les deux autres compartiments contiennent respectivement les attributs et les opérations de la classe.
Les relations entre les classes sont les associations sont binaires, c'est-à-dire qu'elles connectent deux classes. Les associations se représentants en traçant une ligne entre les classes associées.



c- Diagramme d'activités

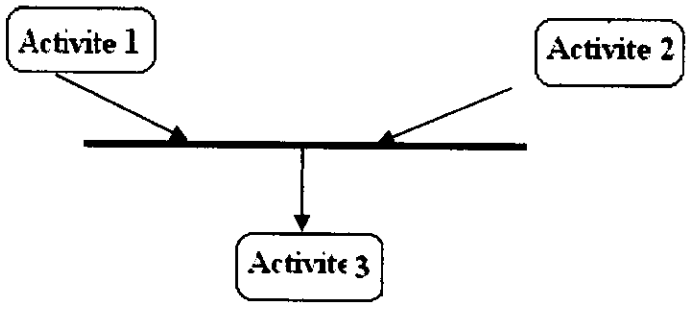
Un diagramme d'activités représente la répartition des rôles entre les acteurs, ainsi que la route (ou itinéraire) qui représente les transitions entre les activités. Une activité représentée par un rectangle arrondi. Les activités sont reliées par des transitions automatiques, représentées par des flèches. Lorsqu'une activité se termine, la transition est déclenchée et l'activité suivante démarre.



Les transitions entre activités peuvent être gardées par des conditions booléennes, mutuellement exclusives. Une condition est matérialisée par un losange d'où sortent plusieurs transitions.

Les diagrammes d'activités représentent les synchronisations entre activités, au moyen de barre de synchronisation. Une barre de synchronisation permet d'ouvrir et de fermer des branches parallèles au sein d'un flot d'exécution d'un cas d'utilisation. Les transitions au départ d'une barre de synchronisation sont déclanches simultanément. Inversement, une barre de synchronisation ne peut être franchie que lorsque toutes les transitions en entrée sur la barre ont été déclenchées.

Les deux cas illustrés dans le schéma suivant :

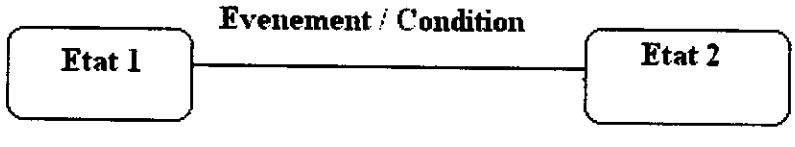


d- Diagramme d'états transitions

Un diagramme d'états transitions fournit une vision globale du cycle de vie de l'objet.

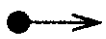
Un état est une condition ou une situation, dans la vie d'un objet, durant laquelle il satisfait une condition, il effectue une activité ou il attend un évènement donné.

Une transition est une relation entre un état source et un état cible.



Etat initial

La représentation de l'état initial est :



Etat final

La représentation de l'état initial est :

4- Avantage d'UML

UML offre plusieurs avantages dont :

- améliorer la compréhension d'un système complexe.
- faire ressortir les aspects importants du système.
- améliorer la fiabilité de l'architecture du système.
- faciliter la communication entre membres du projet et clients.
- utiliser facilement par l'homme et la machine.

5- Conception et réalisation : Nous utilisons quatres types de diagrammes

1- Le diagramme de séquence

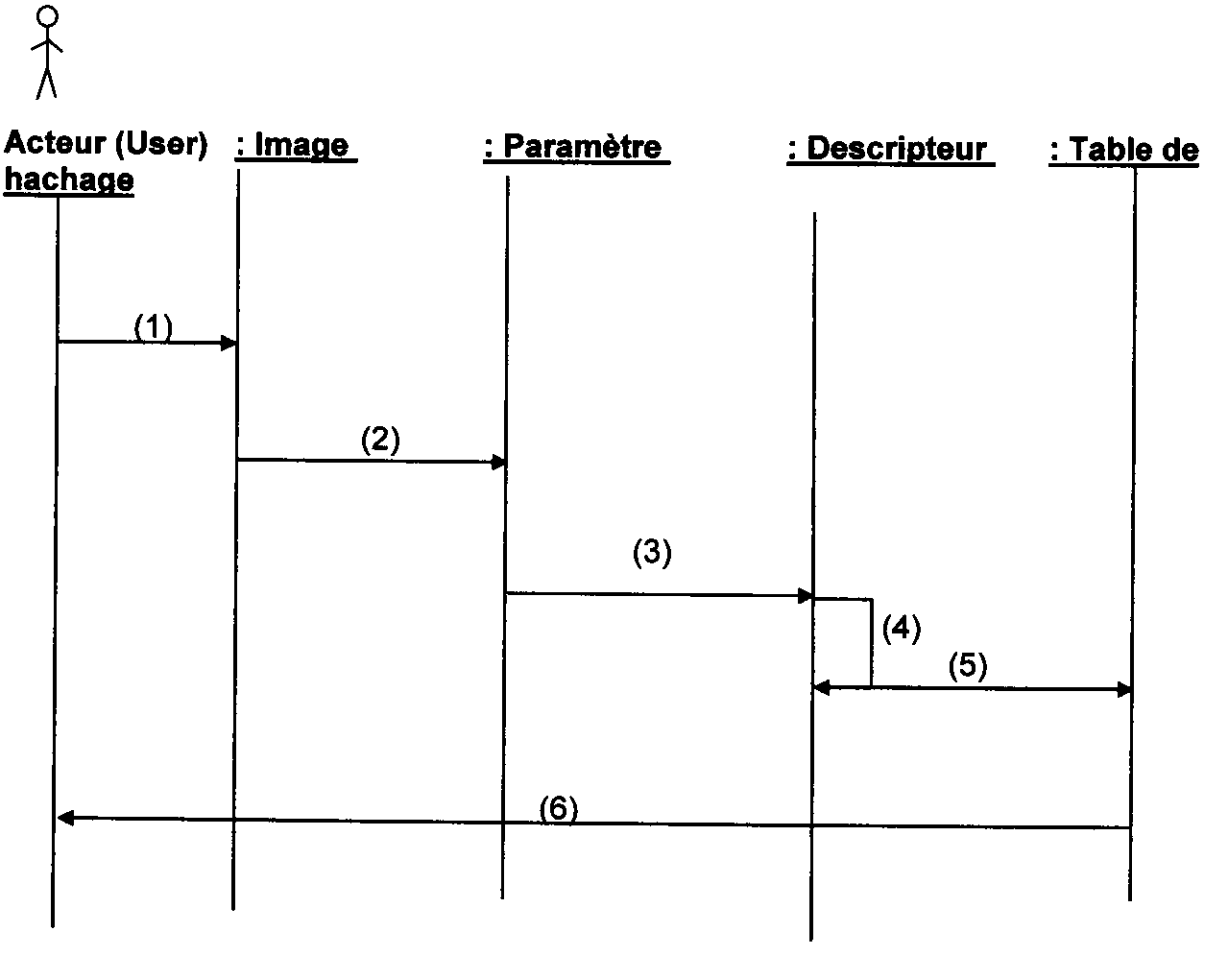


Figure III.10 : Diagramme de séquence

✓ **Le scénario associé**

1 : L'utilisateur lance l'image requête pour rechercher les images similaires.

2 : Opération d'extraction des paramètres de textures (trois paramètres).

3 : Opération de regroupement des paramètres dans le vecteur caractéristique (descripteur).

4 : Calcul la distance entre les descripteurs, puis comparer avec le seuil.

5 : Rechercher l'adresse similaire dans la table de hachage.

6 : Le résultat est l'affichage des images similaires.

2- Diagramme de classes

Dans ce diagramme, on a :

- ✓ une image peut être de la base d'image ou elle peut être une image requête.
- ✓ une image possède son propre descripteur.
- ✓ un descripteur peut être commun à plusieurs images, si elles sont identiques.
- ✓ un descripteur regroupe trois paramètres et possède trois niveaux.

Les identificateurs sont :

NImage : Numéro de l'image.
NomIm : Nom de l'image.
Height : L'hauteur de l'image.
Width : Largeur de l'image.
FrmImage : Format d'image.
TypImage : Type de l'image.
Num descript : Le numéro du descripteur.
Nom Param : Le nom du paramètre.
NNiveau : Le numéro du niveau.
Num Table : Numéro de la table de hachage.
Nom type : Le nom de type du paramètre.

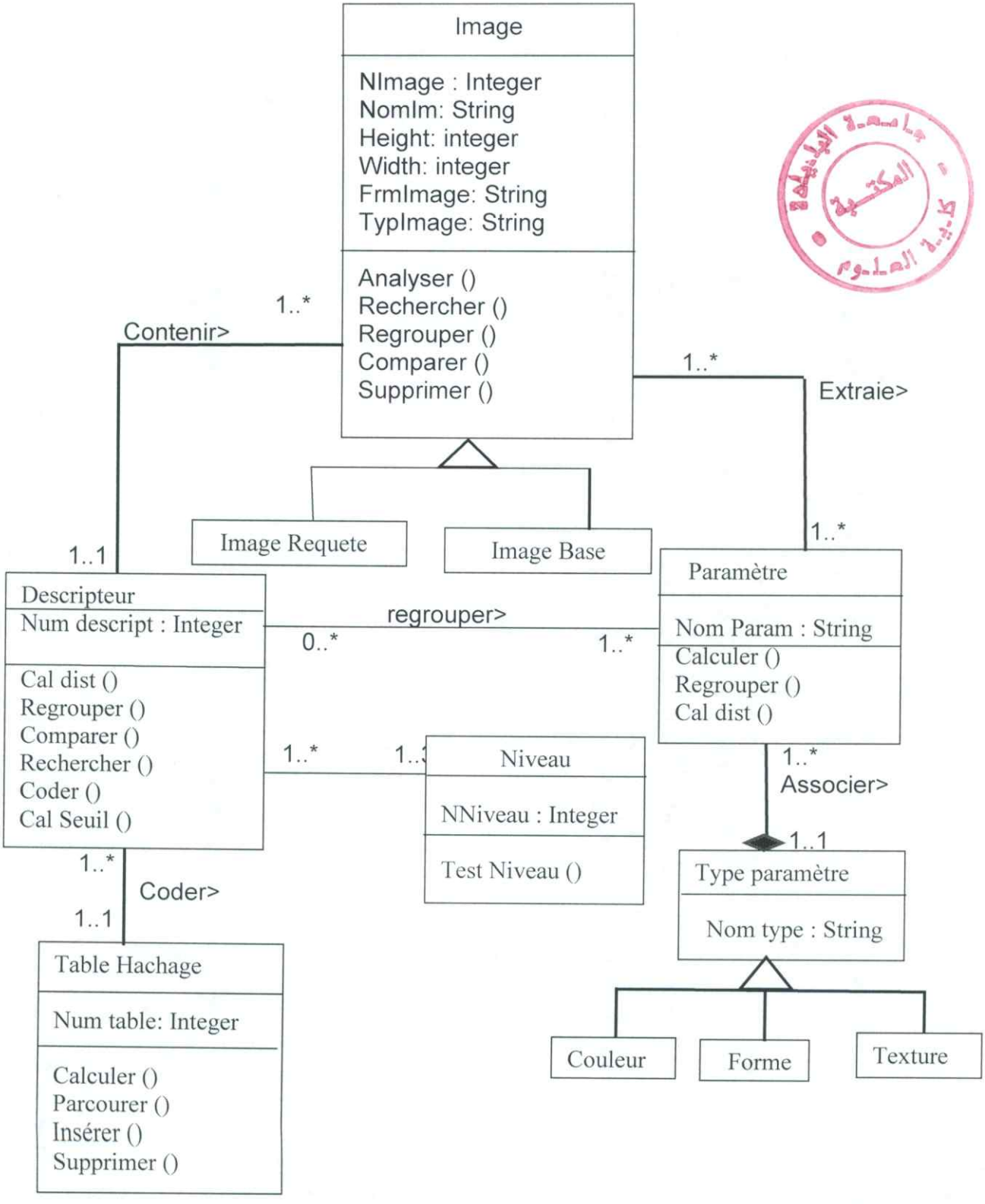


Figure III.11 : Diagramme de classes

3- Diagramme de cas d'utilisation

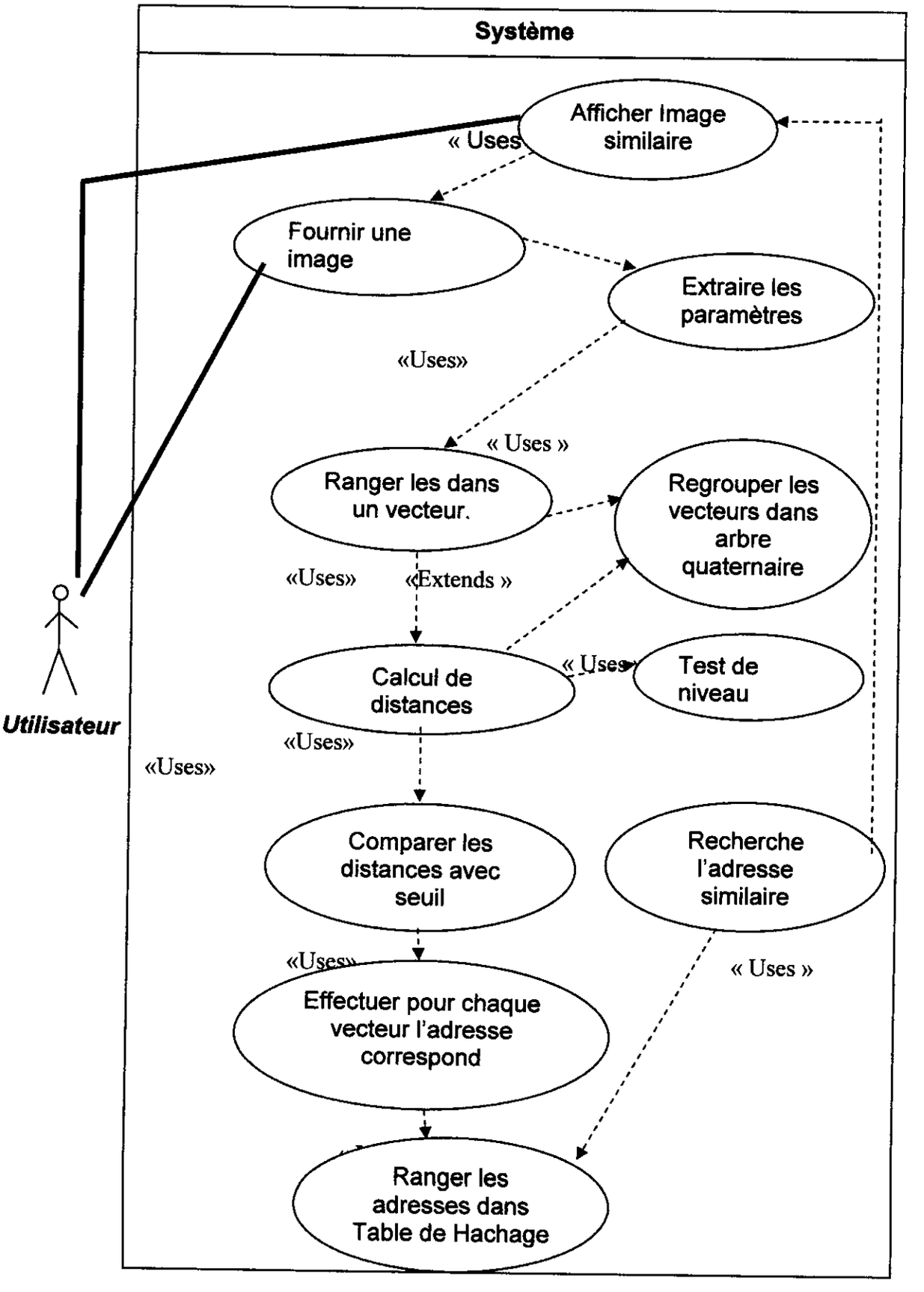
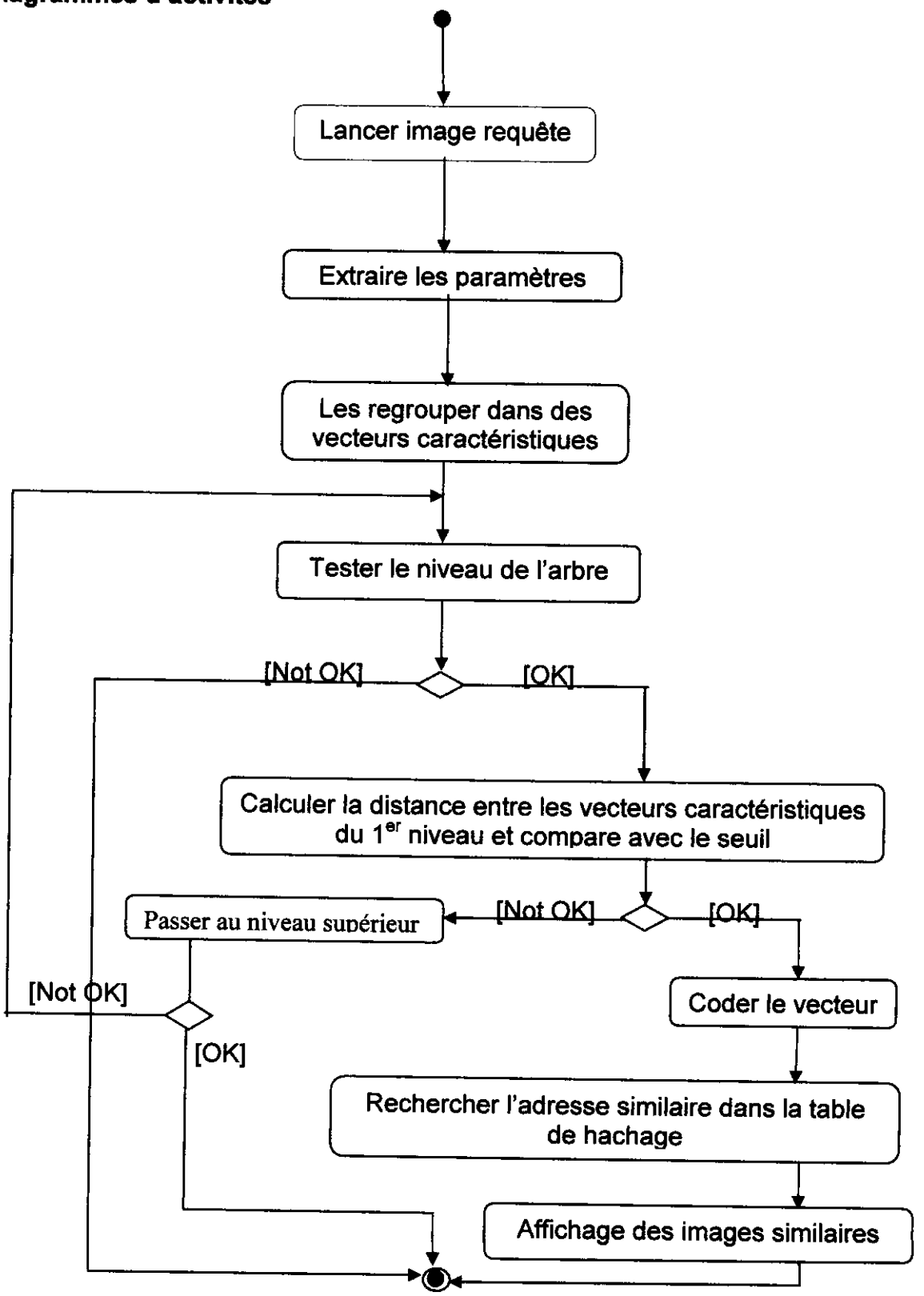


Figure III.12 : Diagramme de cas d'utilisation

4-Diagrammes d'activités



FigureIII.13 : Diagramme d'activité

III- Conclusion

Dans ce chapitre, on a présenté notre démarche de projet en détail avec toutes les étapes nécessaires de l'extraction, le regroupement puis l'indexation des images jusqu'à l'affichage de l'image similaire et les différents algorithmes nécessaires pour l'implémentation de notre application.

Avant de réaliser les étapes précédentes, on a besoin d'une base d'images pour le stockage des images, chaque base de données possède sa propre conception.

La conception de notre projet basé sur le langage **UML**, qui offre à l'utilisateur de faire leur conception correcte et définit les modules nécessaires pour le développement du projet. On a défini quatre types de diagrammes : diagramme de classe, diagramme de séquences, diagramme d'états, diagramme de cas d'utilisation.

Dans ce qui suit, nous allons présenter notre application avec quelques exemples bien expliqués

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

I- 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 256 Méga octets, et d'un disque dur de 80 Giga octets.

I-1 Choix du langage de programmation

Nous avons réalisé notre projet en utilisant le compilateur C++Builder version 6 sous environnement Windows XP.

Notre choix est justifié par plusieurs raisons :

- ✓ La puissance du langage C++ pour les applications scientifiques.
- ✓ La facilite de réalisé une interface graphique interactive, contenant des menus, des boites de dialogues ;...etc. Ce qui permet d'utiliser notre projet d'une manière très simple.
- ✓ Les possibilités de la programmation orientée objet qui donnent l'accès à l'environnement Windows grâce à la librairie OWL (Object Windows Library). Cette bibliothèque de classes facilite grandement la programmation Windows.

Pour implémenter notre base de données nous avons utilisé le langage SQL pour insérer, lire et stocker les images.

II- Présentation du Logiciel

Le logiciel réalisé est représenté par la fenêtre principale donnée par la figure (IV.1) :

Figure IV.1 : Fenêtre principale

ENTREE : permet d'accéder à l'interface principale de l'indexeur (Figure IV.2).

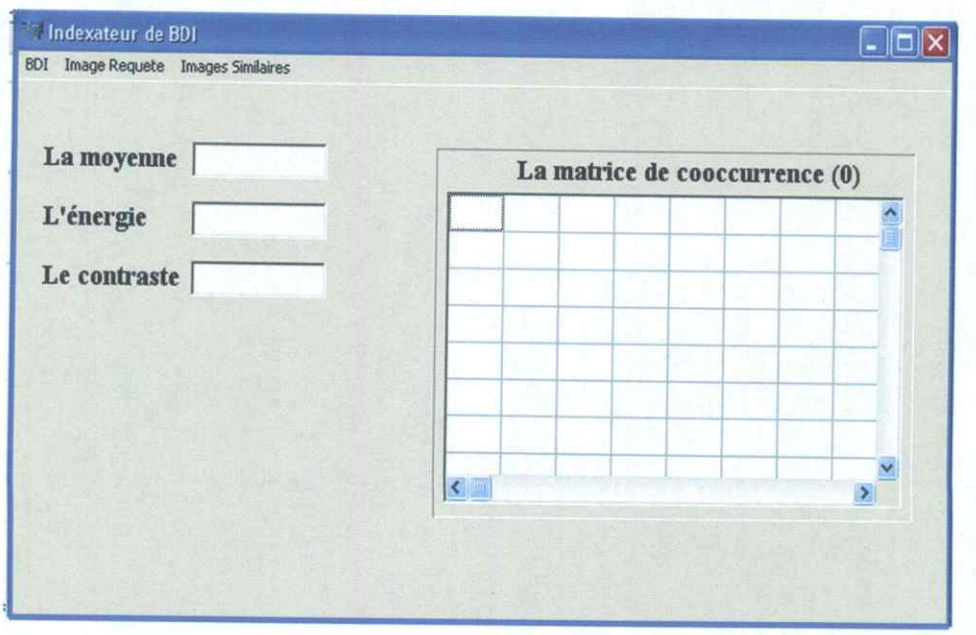


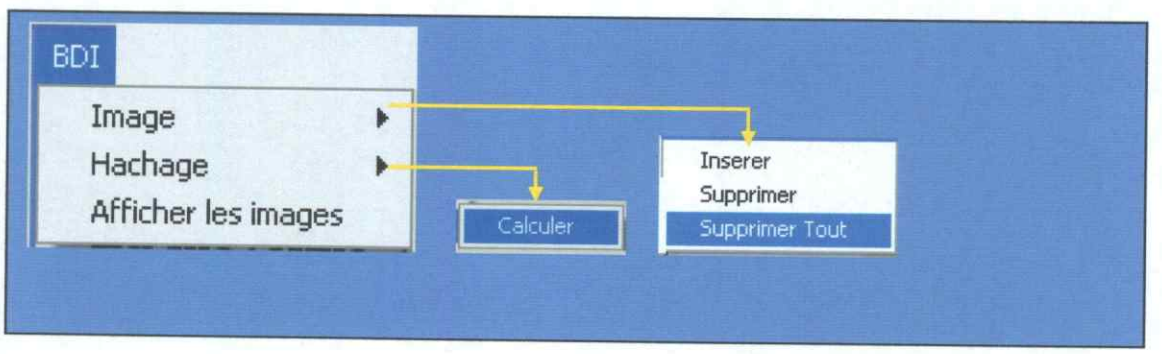
Figure IV.2 : Interface de l'indexeur

L'interface de l'indexeur est caractérisée par les menus suivants :

II-1- Menus

1-BDI

Ce menu concerne la gestion de la base de données. Il contient les commandes suivantes :



- **Image :** Cette commande contient deux sous menus

  - ✓ **Insérer :** permet d'insérer des images dans la base de données, à partir de la boite de dialogue suivante :

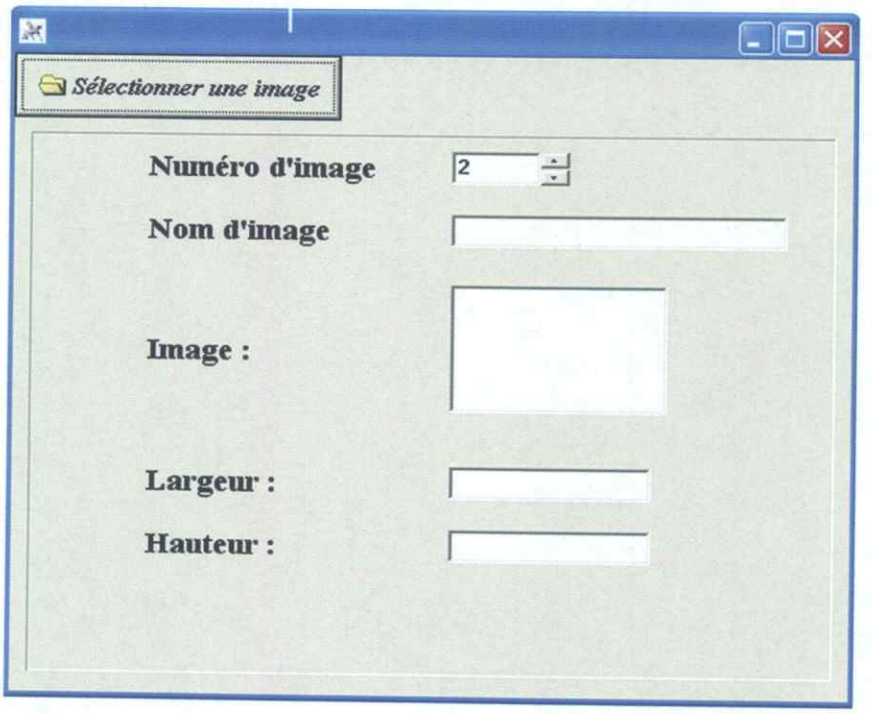


Figure IV.3 Sélectionner l'image à insérer dans la base

✓ **Supprimer :** Permet de supprimer les images de la base de données. La suppression se fait à partir de l'identifiant de l'image.

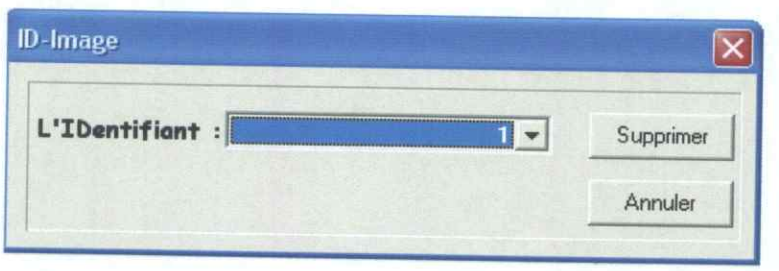


Figure IV.4 : Supprimer une image

✓ **Supprimer tout :** Permet de vider la base.

- **Hachage :** Cette commande contient un seul sous menu « Calculer ».

Il permet de créer la table de Hachage, et regrouper les images dans la table.

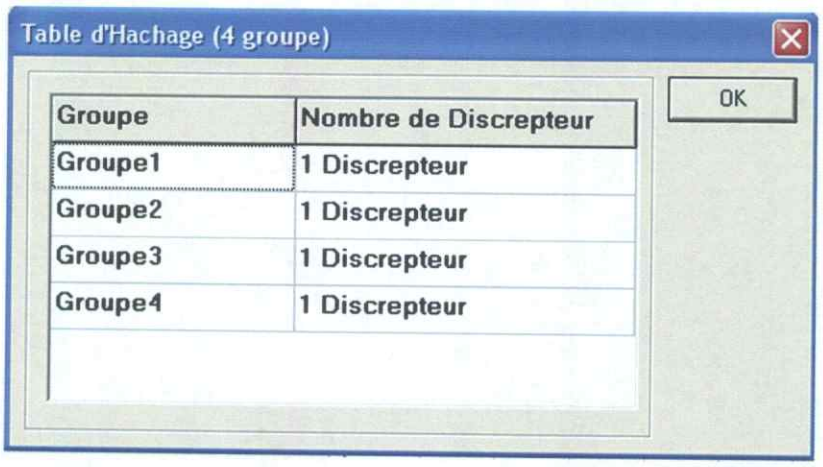


Figure IV.5 : Insertion des descripteurs dans la table

- **Afficher les images :** Il permet d'afficher toutes les images existant dans la base de données.

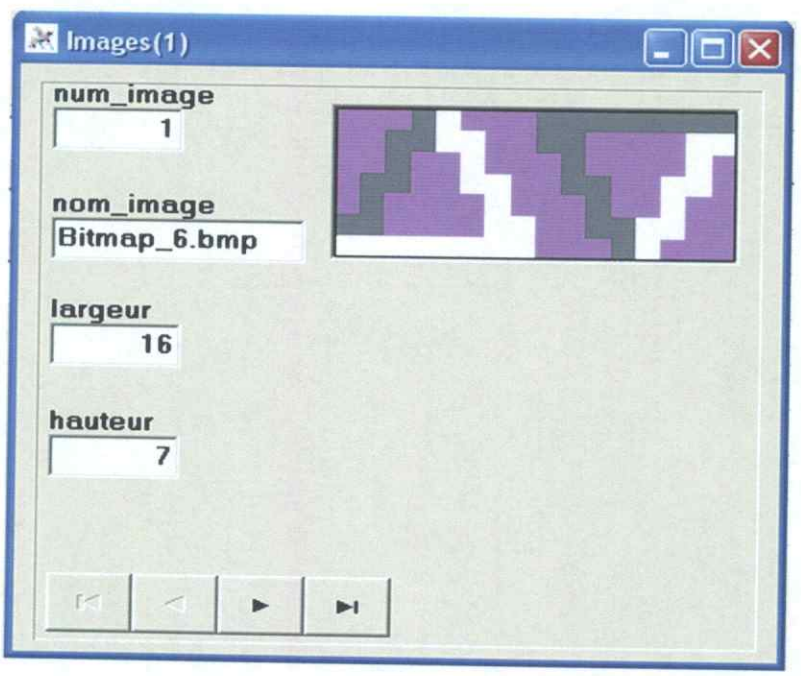
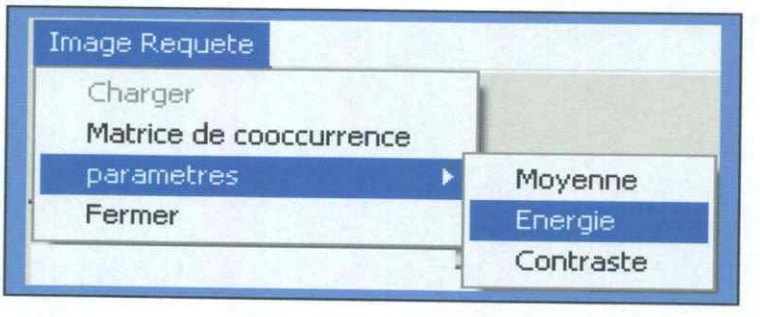


Figure IV.6 : Affichage des images de la base

2. Image Requête : Ce menu concerne l'analyse de l'image requête.



- ✓ **Charger :** Permet de charger une image quelconque.
- ✓ **Matrice de cooccurrence :** Calcule la matrice de cooccurrence de l'image à insérer.
- ✓ **Paramètres :** Permet de calculer les paramètres (Moyenne, Energie, Contraste) extraits de l'image requête à partir de la matrice de cooccurrence associée.
- ✓ **Fermer :** Permet de charger une autre image (Supprimer les paramètres de l'ancienne requête pour charger une autre image requête).
- ✓ **Image Similaire :** Permet de chercher l'image similaire de l'image requête dans la base.

- ✓ **Recherche tout :** Permet de chercher les images similaires d'une requête. Si elles existent il les affichent sinon il lance un message à l'utilisateur qu'il n'existe aucune image similaire.

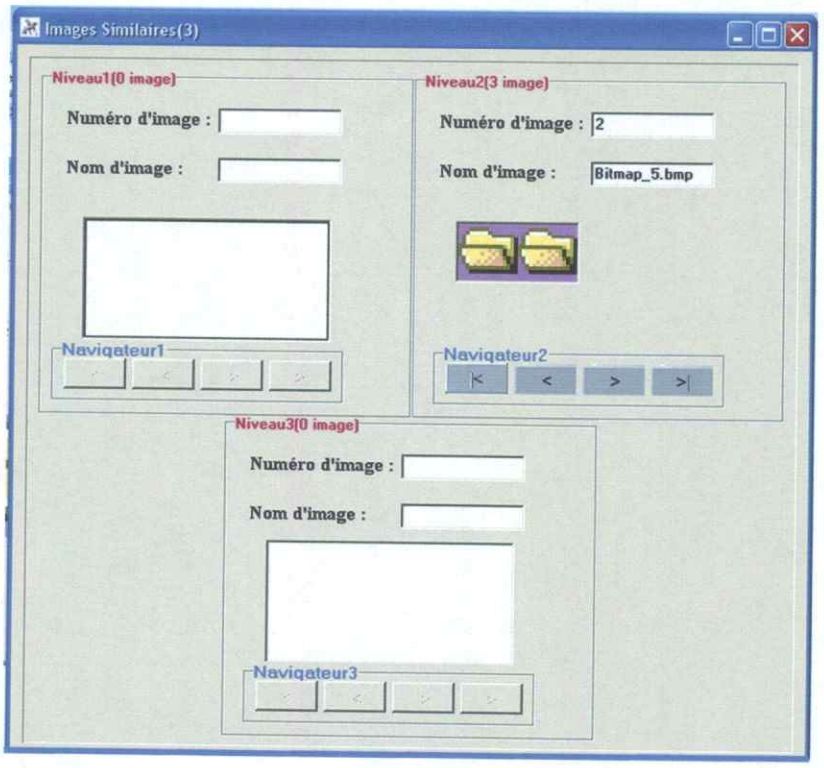


Figure IV.7 : Affichage des images similaire

- ✓ **Recherche de l'image la plus similaire :** Affiche l'image la plus proche dans la base en fonctions de la valeur de la distance.

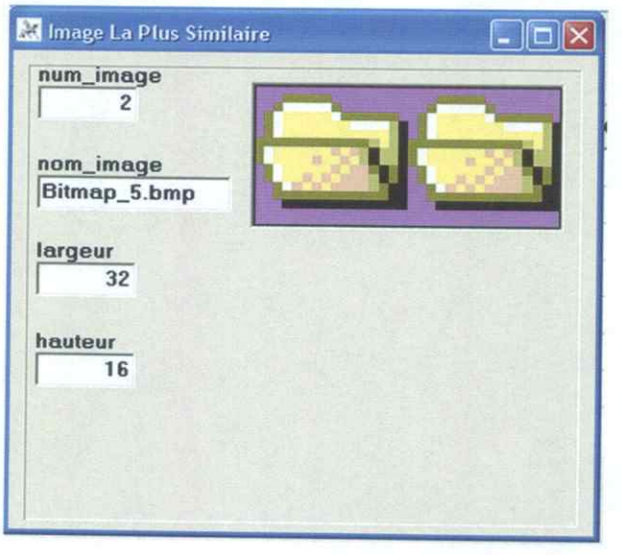


Figure IV.8 : Affichage de l'image la plus similaire

- **A PROPOS :** Ce bouton contient la boite de dialogue suivante :

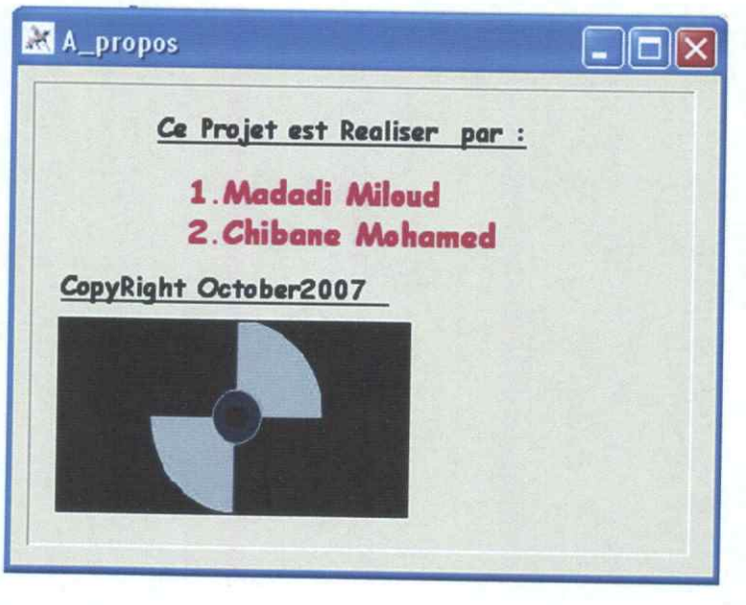


Figure IV.9 : A_PROPOS de notre logiciel

- **QUITTER :** Permet de fermer l'application, par une confirmation avec une boite de dialogue.

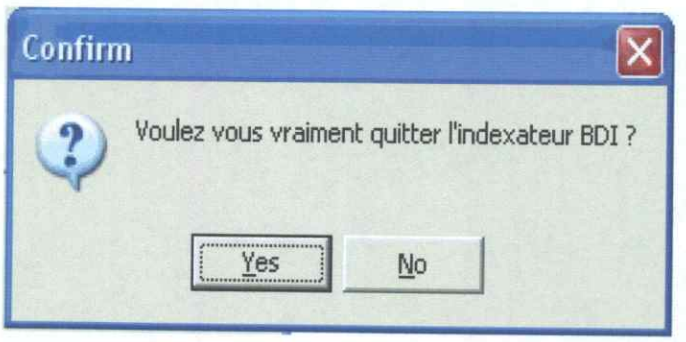


Figure IV.10 : Fermer l'application

III- Tests sur des exemples

III- 1- Exemple1

Dans cet exemple, on fait la recherche dans une base de 14 images, puis on calcule le temps de recherche.

a- insertion des images dans la base

Cette opération permet d'insérer une image ou plusieurs dans la base suivant les figures ci dessous.

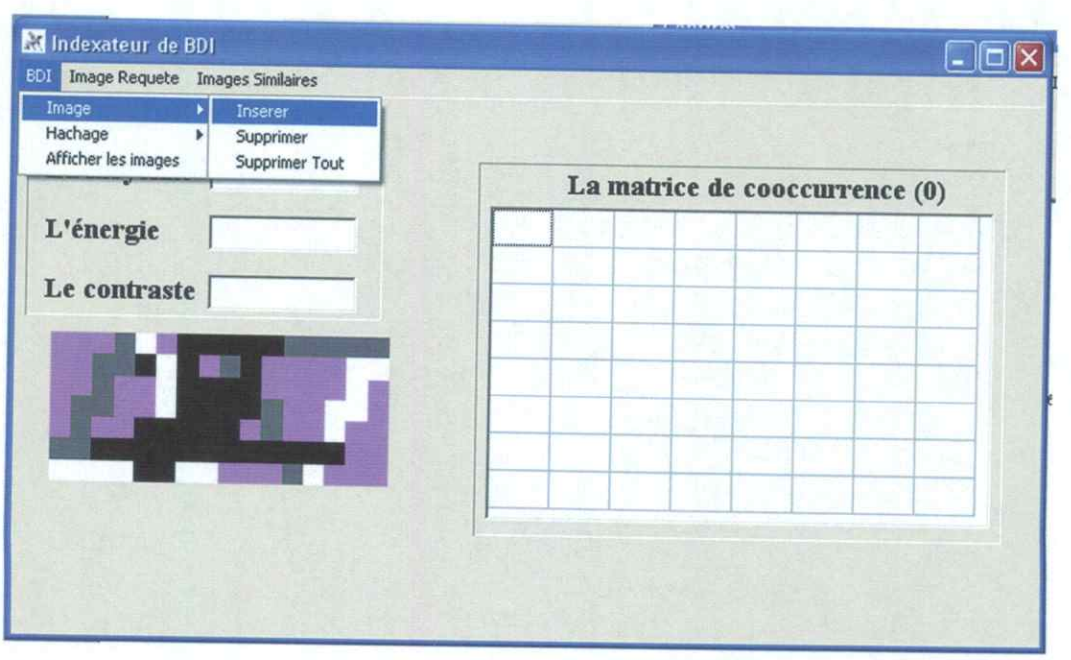


Figure IV.11 : Insertion des images

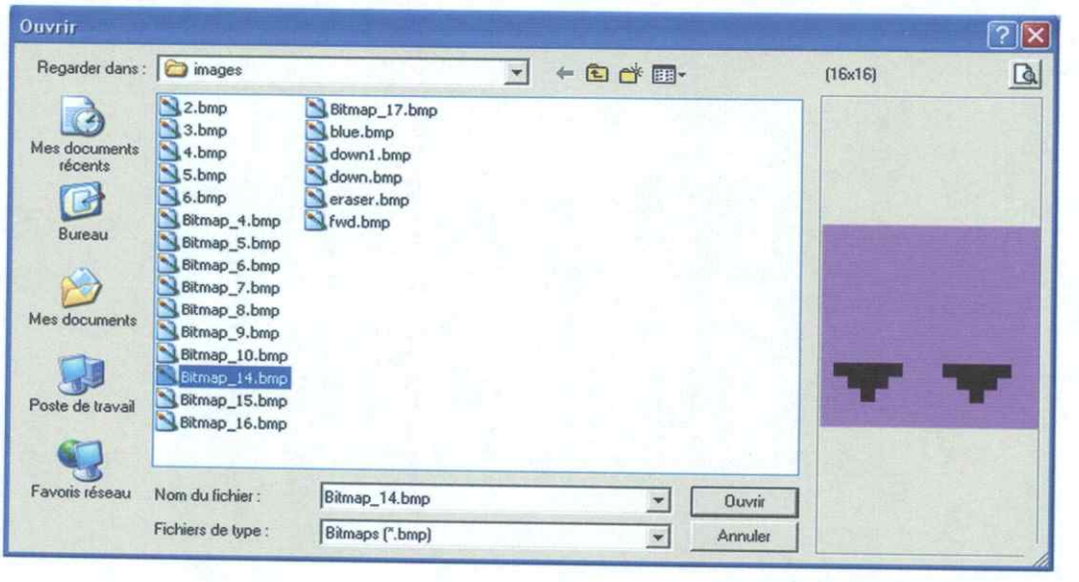


Figure IV.12 : Sélection de l'image insérer

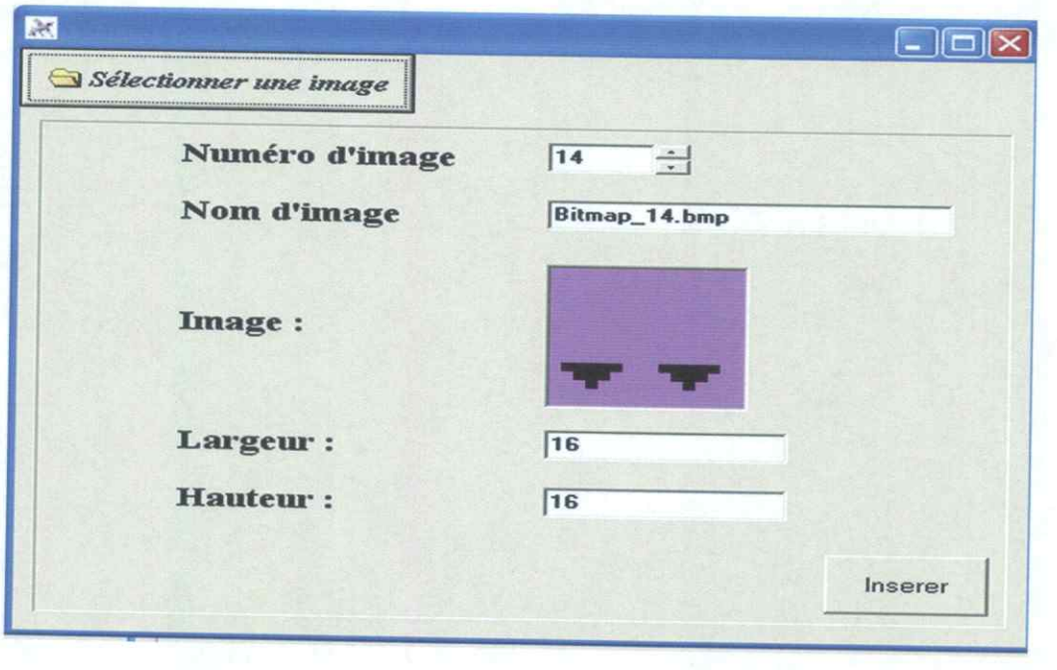


Figure IV.13 : Affichage de l'image a insérer

b- Calculer la table de hachage

Le calcul se fait selon l'état suivant :

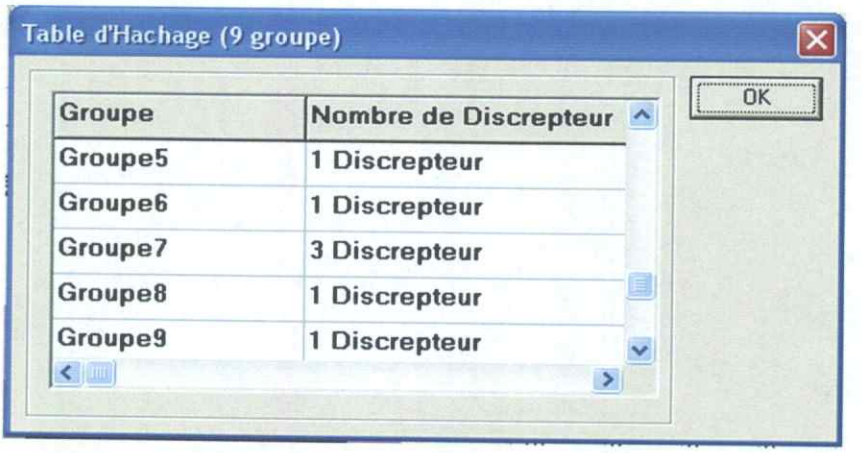


Figure IV.14 : Etat de la table

c- Sélectionner l'image requête

Cette opération permet de sélectionner l'image requête pour faire analysée et comparé avec les 14 images de la base (Figure IV.14, Figure IV.15).

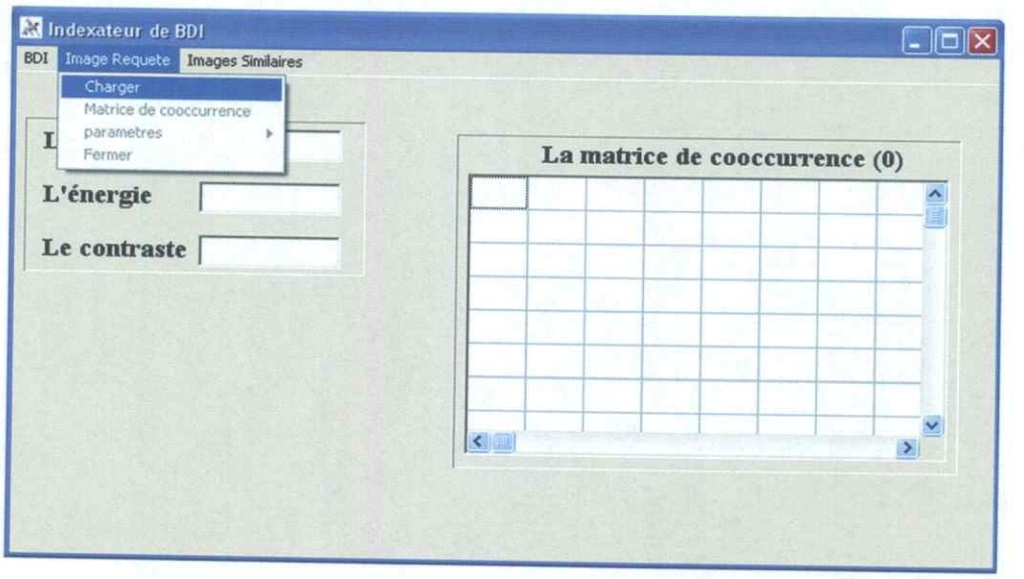


Figure IV.15 : Charger l'image a insérer

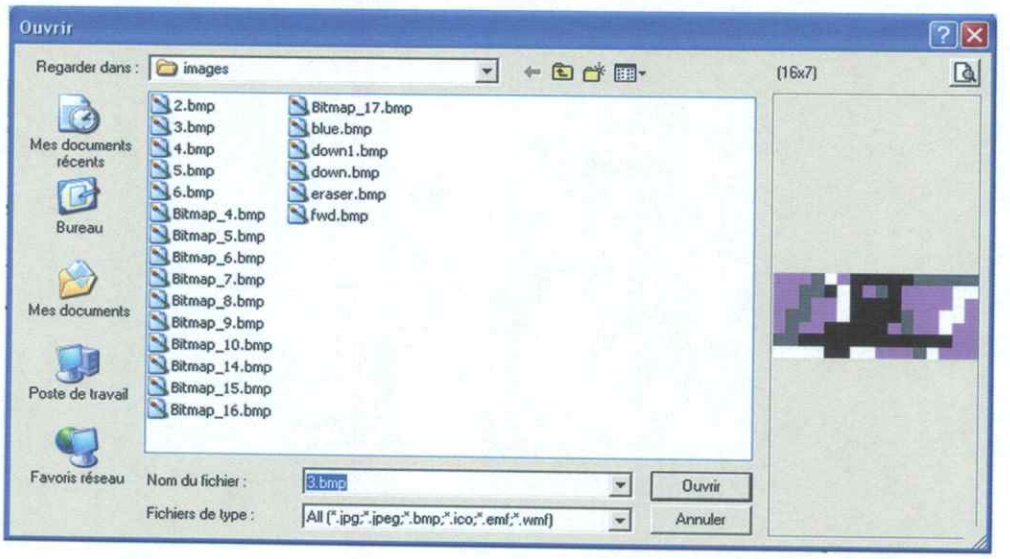


Figure IV.16 : Sélection de l'image Requête

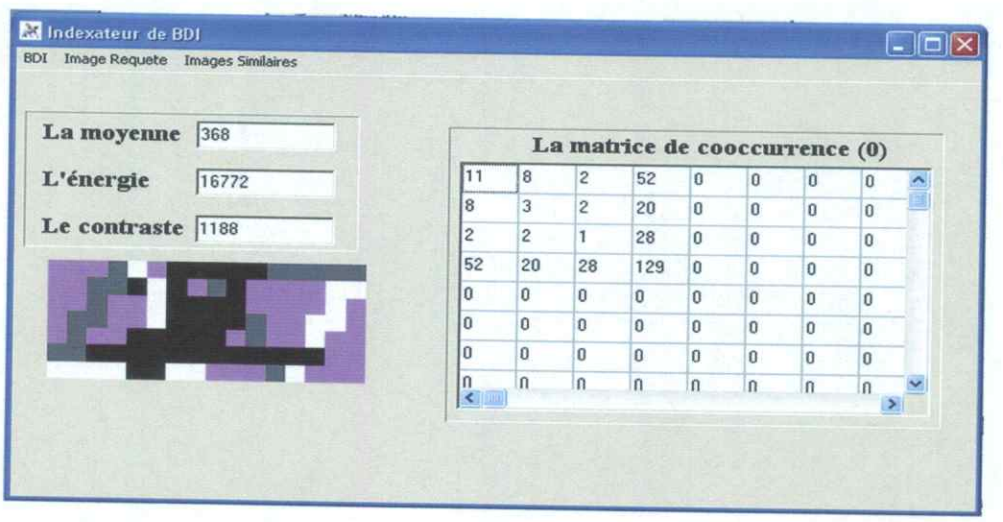


Figure IV.17 : Affichage de l'image Requête

✓ **Rechercher**

Cette opération permet de rechercher l'image similaire de l'image requête (Figure IV.18).

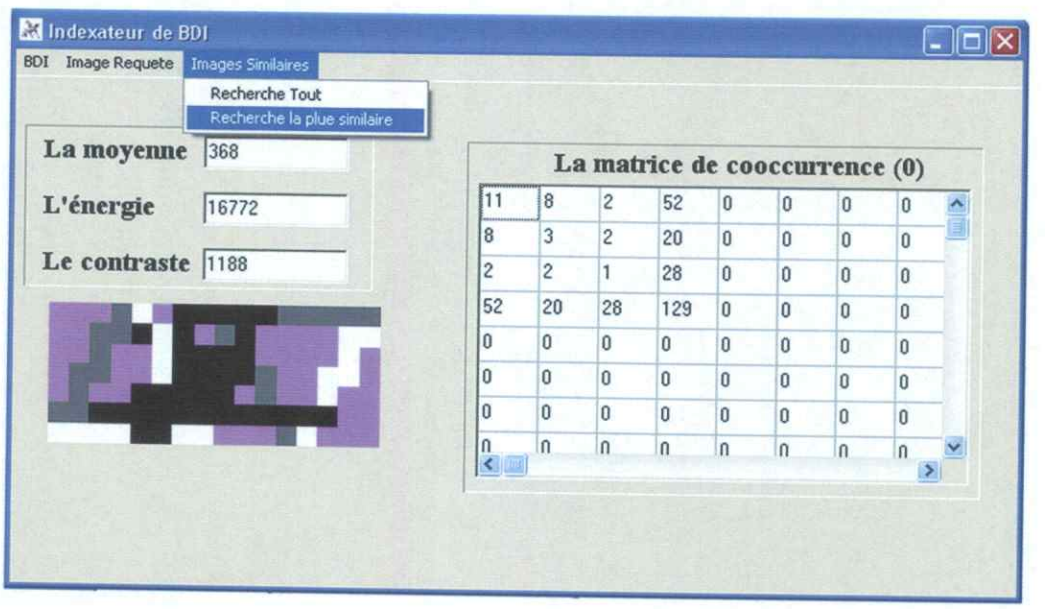


Figure IV.18 : Rechercher des images similaires

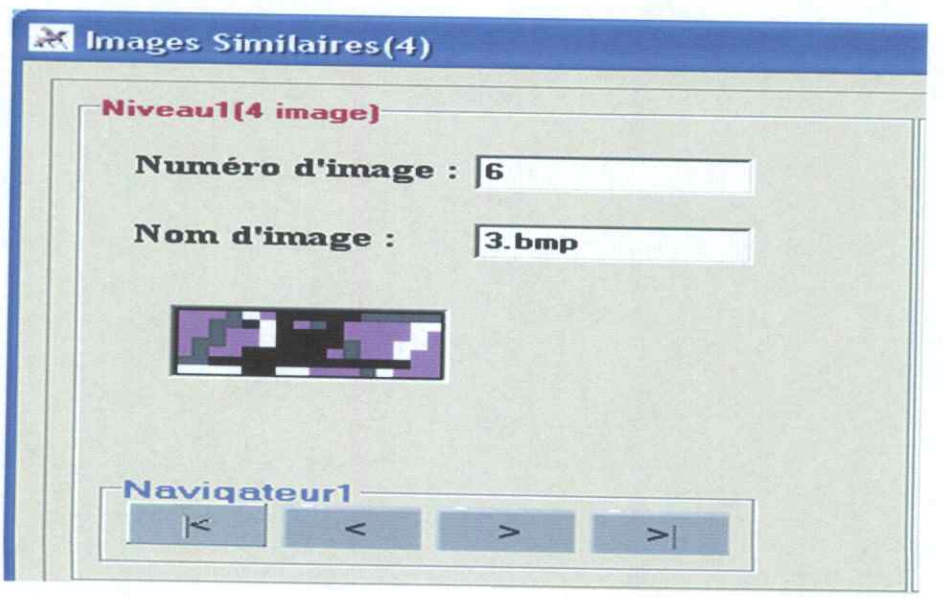


Figure IV.19 : Affichage de l'image similaire

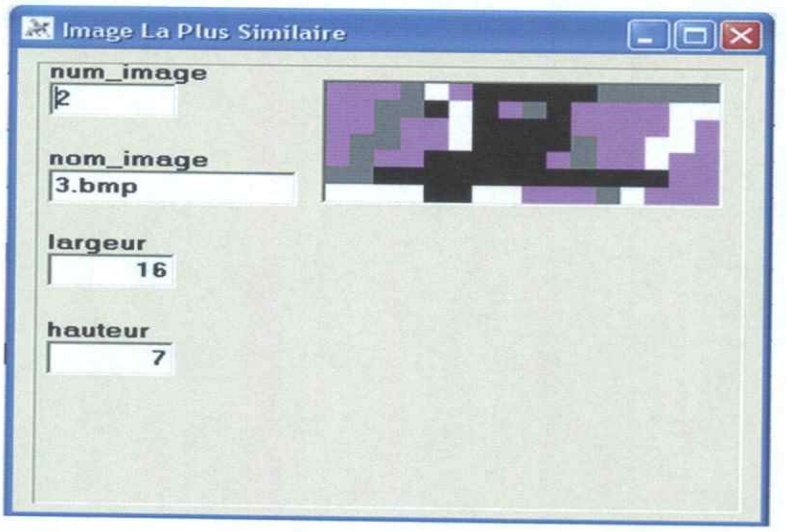


Figure IV.20 : Affichage de l'image la plus similaire

- **Comparaison et discussion**

Nous avons réalisé 7 tests dans le cadre de notre application. Les résultas de ces tests sont données dans le tableau suivant :

Nombre d'images insérées	14	30	45	60	75	90	100
Temps de calcul de la table	3.66	17.78	25.33	35.66	40.19	44.50	48.20
Temps de recherche	0.11	0.12	0.14	0.15	0.17	0.18	0.20

Tableau IV.I : Résultas des tests

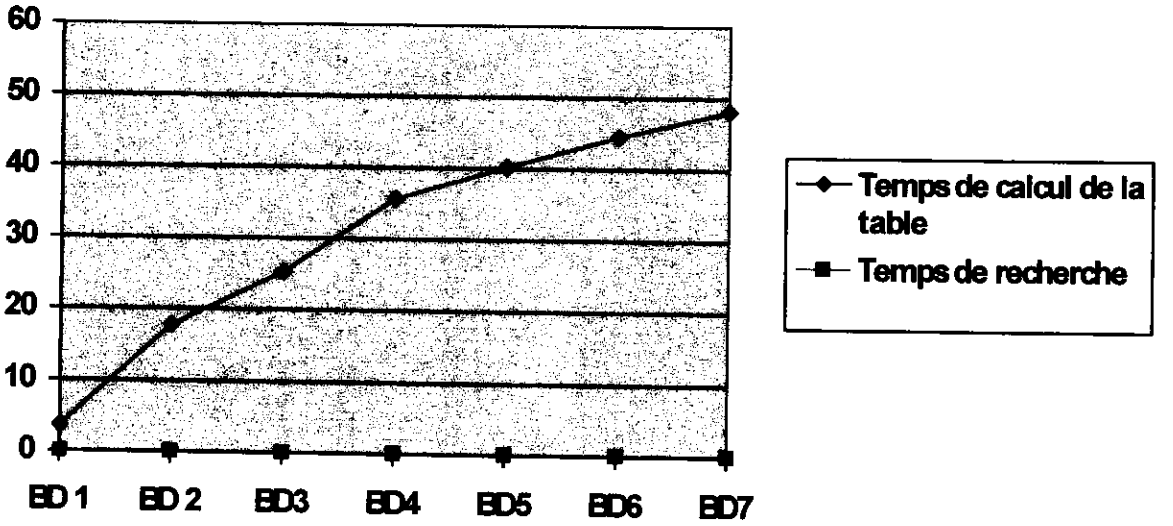


Figure IV.21 : Comparaison des tests

Avec : DB i : La base d'image i, qui prend la valeur du nombre d'images dans le tableau précédant .

On remarque que le temps de calcule de la table est très important, puisqu'il dépend de nombre d'images de la base.

On peut dire que les inconvénients de notre système sont le temps de calcul de table d'hachage et la taille de la base. Plus la taille de la base est augmente, plus le temps de calcul et de recherche augmente.

IV- Conclusion

Dans ce chapitre, nous avons présenté les détails de notre application, et la manière de fonctionnement. Puis les résultats des tests de l'algorithme de recherche. Dans ce chapitre, nous avons présenté les résultats de notre application réalisée sur plusieurs bases d'images.

Nous avons remarqué que le temps et la taille de la base sont les deux inconvénients de notre système de recherche.

Notre système de recherche engendre des bons résultats de recherche car il traite les images avec leur contenu, en plus de ça il donne des bons résultats de recherche (le temps de recherche est négligeable).

CONCLUSION

Conclusion générale

Dans ce mémoire, nous avons fait une étude bibliographique sur les principales méthodes d'indexation existantes destinées au domaine de la recherche d'images par le contenu.

Nous avons opté par la suite pour la méthode d'indexation le ***hachage.*** Cette méthode nécessite l'utilisation d'une fonction de hachage qui attribue à chaque image une adresse dans la table de hachage. Le calcul de cette adresse dépend des trois paramètres du texture extraits à partir de l'image a inséré ; ces derniers sont calculés par la méthode de la matrice de cooccurrence et regroupées dans un vecteur appelé descripteur local qui caractérise l'image en utilisant une structure d'indexation ***quad-tree.*** Cette dernière est bien adaptée aux index tridimensionnels, car elle permet un accès rapide aux données.

Nous avons ensuite, utilisé UML pour modéliser notre système, en indiquant les acteurs et les interactions entre eux ; Puis on a énuméré tous les traitements et les opérations appliquées.

L'exploitation de ce système d'indexation a nécessité l'utilisation d'une base d'images. L'image requête est ensuite en comparaison avec les images existant dans cette base.

Le traitement se fait en 2 étapes, off line et on-line. En off line, aucun utilisateur n'intervient ; tout se fait automatiquement (extraction des descripteurs ... base de descripteurs). En mode on-line, l'utilisateur introduit l'image requête, puis calcule automatiquement le descripteur requête. Ensuite, on calcule la similarité entre la base de descripteurs et le descripteur requête.

La validité de notre application a été faite en considérant des bases d'images différentes. Les résultats montrent que le temps de recherche est augmente en fonction de la taille des bases d'images.

Comme perspectives à notre travail, nous proposons d'améliorer les algorithmes de recherche. Nous pouvons citer en particulier :

- ✓ Changer le type des descripteurs (Exemple : des descripteurs qui contiennent les paramètres de textures, les paramètres de formes...etc.).
- ✓ Choisir d'autres paramètres de textures.
- ✓ L'amélioration des structures de données aux stockages, en utilisant d'autres structures (Exemple : **R-Tree, Le SS-Tree**...etc.)
- ✓ L'utilisation des objets plus complexes.

Références Bibliographiques

Bibliographie

Ouvrages

[CAR 97] C.Carrez, « Structure de données en Java et Ada 95, Pratique et outils de contrôle ».

[Ket 01] Nasser Kettani [et al.], « De Merise à UML », 4éme ed, Eyrollles, 2001.

[MUL 97] Muller Pierre-Alain, «Modélisation objet avec UML », Eyrollles, 1997.

[Mul 97] Muller Pierre-Alain, Gaertner Nathelie, «Modélisation objet avec UML », 2ème ed, Eyrollles, 2001.

Mémoires

[ADE 04] Adel Hafiane, « Caractérisation de textures et segmentation pour la recherche d'images par le contenu », Thèse de doctorat, Ecole doctorale sciences et technologies de l'information des télécommunication et des systèmes Paris, 2004.

[ADE 05] Adel Hafiane, « Caractérisation de textures et segmentation pour la recherche d'image par le contenu », mémoire de doctorat informatique, 2005.

[ALA 05] Alain Boucher, « Indexation et recherche d'image par le contenu », Mémoire de Master en informatique, France, 2005.

[AMS00] L.Amsaleg, P.Gros, R.Mezhoud, « Mise en base d'images indexées par des descripteurs locaux : Problèmes et perspectives », INRIA, 2000.

[BAY 71] R.Bayer, « Binary B-trees for virtual memory », Proceedings of the ACM 1971 SIGFIDET November 1971.

[BER 04] Berrani, Sid-Ahmed, «Recherche approximative de plus proches voisins avec contrôle probabiliste de la précision application à la recherche d'image par le contenu », Mémoire de Doctorat informatique, 2004.

[GHE 03] Fouzia Ghemari, Habiba Benadouane, « Indexation géométrique d'une base d'images », Mémoire d'ingénieur d'état en informatique, 2003.

[LAM 96] B.Lamioroy, « Reconnaissance d'objets par indexation géométrique étendue »,1996.

[LAM 98] B.Lamioroy, « Reconnaissance et modélisation d'objet 3D à l'aide d'invariants projectifs et affines », Thèse de doctorat, Institut National Polytechnique de Grenoble, 1998.

[TIT 05] Emmanuel TONYE, «Le traitement des images de télédétection par l'exemple» 2005.

[WEB 98] R.Weber, « A quantitative analysis and performance study for similarity-search methods in high dimensional spaces », U.S.A 1998.

Publications

[BRI 05] Brian Lam et Vic Ciesielski Blam, «Texture Feature Extraction Programs Using Genitic Programming», Université de RMIT Informatique, 2005.

[PUB 03] Jean-Michel Jolion, «Vision Artificielle» 1997.

[PUB 42] Genivière Jomier, Maude Manouvrier, Vincent Oria, « Indexation Multi-Niveau pour la recherche globale et partielle d'image par le contenu », Université de paris Dauphine 2004.