Laboratoire d'Informatique
En Images et Systèmes d'information
UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon1/ Université Lumière

Rapport d'état d'avancement

Recherche par similarité dans les grandes bases de données multimédias : application à la recherche par le contenu dans les bases d'images

Encadré par : Khalid IDRISSI

Imane DAOUDI

Le 31- 03- 2007

Bibliographie

1 Introduction.

Jusqu'à présent, la recherche et l'indexation d'images constituent un problème majeur dans de larges bases de données multimédias notamment dans les grandes bases d'images. L'approche actuelle d'indexation consiste à décrire les images par leurs contenus à l'aide de descripteurs constitués de paramètres de bas niveau relatifs à la couleur, la forme et la texture. Ainsi, à chaque image correspond un ou des vecteurs caractéristiques formant les index de cette image. La recherche de similarité n'est donc, pas mesurée sur les images directement mais plutôt sur la base des vecteurs caractéristiques. Etant donnée une requête, un système de recherche d'images doit être capable, sur la base des index dont il dispose, de retrouver les images les plus similaires à une images requête en terme d'une distance donnée. Le problème de recherche d'images se ramène donc à un problème de recherche des k plus proches voisins dans un espace de vecteurs multidimensionnels. Ainsi, les systèmes de recherche d'images par le contenu se décomposent généralement des étapes suivantes :

1. Une étape d'indexation : celle-ci consiste à extraire des signatures compactes du contenu visuel de l'image.
2. Une étape de structuration de l'espace de description : qui consiste à mettre en place une structure d'index multidimensionnels permettent une recherche efficace des dizaines voire des centaines de milliers d'images.
3. Une recherche de similarité : dans la plupart des méthodes, une distance est associée à chaque descripteur et une recherche des k plus proches voisins est effectuée. La figure 1 résume les différentes étapes citées précédemment.

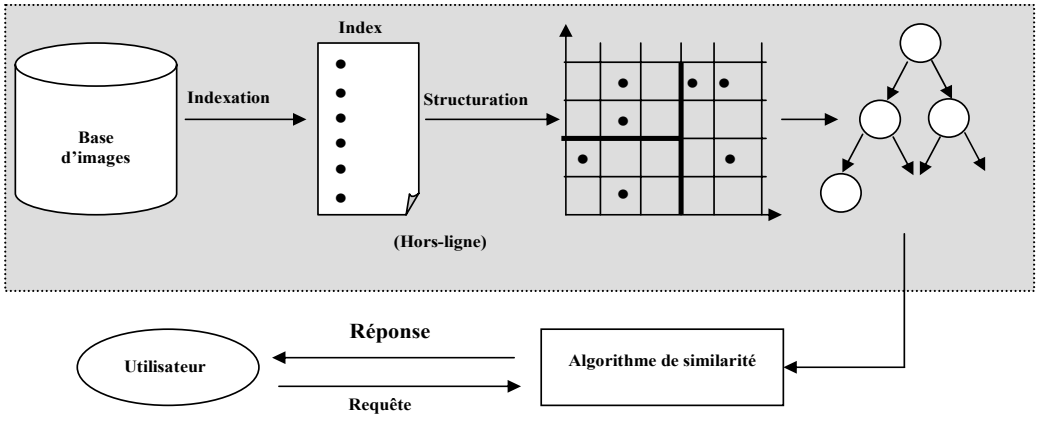


Fig. 1- Principe général de l'indexation

Construire un tel système nécessite donc des compétences que l'on trouve à la fois dans la communauté traitement d'images et celle des bases de données.

Du domaine du traitement d'images sont tirées les techniques d'analyse du signal qui permettent d'extraire des descripteurs des images. Ces descripteurs doivent permettre de reconnaître le contenu de l'image même en présence de certaines variations : illuminations différentes, recadrages, points de vues différents, déplacements d'objets dans l'image, etc.

Les compétences en bases de données sont nécessaires dès que le volume de la base d'images devient trop important pour tenir en mémoire centrale. Il devient dans ce cas obligatoire de stocker les descripteurs sur disque. Les techniques utilisées en BD sont alors employées pour organiser et structurer spécifiquement ces descripteurs, ils permettent d'accroître la vitesse des recherches et minimiser les coûts liés aux entrées-sorties.

Dans la littérature des bases de données, il existe un grand nombre de structures d'index multidimensionnelles qui permettent de structurer et d'organiser les données en mémoire pour une recherche efficace des plus proches voisins. Nous verrons dans l'état de l'art sur l'indexation qu'il existe deux grandes familles de méthodes. La première consiste à regrouper en paquets les vecteurs de la base, puis de les englober dans des cellules ayant une forme simple à manipuler, ces méthodes sont très limitées dans le cas d'espaces de grandes dimensions et leurs performances se dégradent très sévèrement lorsque la dimension devient considérable $d>10$ [1]. La seconde famille repose sur le principe de l'approximation des vecteurs elle est considérée efficace pour gérer les descripteurs de grandes dimension mais elle nécessite des améliorations.

Issus d'une communauté de base de données, nous nous engagerons très naturellement dans ce sens. Notre travail concernera donc l'aspect gestion et organisation de données plutôt que l'aspect description caractérisation des images. Il s'agira alors d'une part, d'améliorer la performance des méthodes d'indexation basées sur l'approche approximation ou filtrage en tenant compte des différentes descriptions des images, et d'apporter des réponses au problème de passage à l'échelle, et d'autre part, d'intégrer les résultats de la recherche sur l'accès aux grandes bases d'images au moteur de recherche

existant. L'idée est d'explorer ce que l'on peut faire en bases de données dans le cadre d'une approche conjointe indexation/recherche, il ne s'agit plus de considérer les deux processus comme indépendant mais plutôt les traiter en parallèles en tenant compte dans chacun des deux, des contraintes imposées par l'autre.

Ce rapport sera organisé de la manière suivante : le paragraphe 2 propose des définitions de base sur la similarité et la recherche d'images par le contenu, Nous passons en revue, dans le paragraphe 3, les principales techniques d'indexations multidimensionnelles, en précisant à chaque fois les limites et la particularité de chaque méthode. Enfin le paragraphe 4 sera consacré à l'étude et l'analyse du système existant, puis le dernier paragraphe présentera quelques pistes de solutions pour la recherche d'image à l'aide de ce système intégrant des méthodes d'indexation des BI (Base d'images).

2 Définitions de base

2.1 Recherche par similarité :

Il s'agit de retrouver les vecteurs les plus similaires à un vecteur requête au sens d'une mesure de similarité donnée. La recherche par similarité peut être exécutée soit par une recherche des k plus proches voisins (k-ppv), soit par une recherche à ε près appelée également *recherche par intervalle*

Recherche à ε prés

Il s'agit de rechercher l'ensemble des vecteurs qui se trouvent à une distance $<\varepsilon$ du vecteur requête au sens de la mesure de similarité associée aux vecteurs. Cet ensemble est défini par : Q (q, ε)= $\{ v \in BD / sim(q,v) < \varepsilon \}$ où BD est la base de vecteurs, v est un vecteur de la base, q est le vecteur requête et sim est la mesure de similarité associée aux vecteurs.

Recherche des k plus proches voisins :

Il s'agit de chercher les k vecteurs les plus proches du vecteur requête. Cet ensemble est défini par : $(\forall\, v_1 \in kppv)(\forall\, v_2 \in BD / kppv) \quad sim(q, v_1) \leq sim(q, v_2)$.

Chacune de ces techniques de recherche possède des avantages et des inconvénients. La recherche des k plus proches voisins garantit systématiquement k vecteurs dans l'ensemble résultat. Cependant, certains de ces vecteurs peuvent être très éloignés du vecteur requête et ne peuvent être considérés comme similaires. La recherche à ε près permet à un utilisateur expérimenté qui maîtrise la distribution de ces vecteurs d'éviter ce problème en choisissant une valeur appropriée de ε. Mais dans le cas général, le choix de ε reste problématique : une trop petite valeur impose une recherche très restrictive et peut donner lieu à des ensembles résultats vides, une trop grande valeur peut engendrer à l'inverse, des ensembles résultats de très grande taille.

En pratique, c'est la recherche des k plus proches voisins qui est le plus souvent privilégiée car elle ne nécessite aucune connaissance sur la distribution et sur l'ordre de grandeur des vecteurs.

Par ailleurs, une étude a posteriori de la distribution des distances entre le vecteur requête et ses k plus proches voisins permet d'éliminer les vecteurs les plus éloignés et de remédier ainsi l'inconvénient majeur de cette technique de recherche. Par conséquent, nous considérons dans la suite de ce travail uniquement la recherche des k plus proches voisins.

2.2 Base de données :

Nous considérons dans notre recherche de similarité, que les images sont représentées par des points dans un espace vectoriels avec une dimension fixe est finie d. Ainsi, la base de données est un ensemble de points dans un espace de donnée DS de dimension d, l'espace DS est un sous ensemble de R^d. Dans la plupart des cas on considère que DS est un hyper cube unité de dimension d DS=$[0..1]^d$. la base de données DB est définie comme étant un ensemble de n points dans un espace de dimension d tel que : $DB = \{P_0, \ldots, P_{n-1}\} / P_i \in DS, i=0..n-1, DS \subseteq R^d$

2.3 Espace vectorielle métrique

La similarité est basée sur la notion de distance entre deux points P et Q dans un espace de données. Il existe différentes métriques pour définir la distance entre deux points. La métrique la plus utilisée est la distance euclidienne elle est définie comme suit :

$$d_{euclid}(P,Q) = \sqrt[2]{\sum_{i=0}^{d-1}(Q_i - P)^2{}_i}$$

Il existe aussi d'autres distances peu utilisées comme la distances de Manhattan L_1 et la distance max L_∞ :

$$d_{Manhat\tan}(P,Q) = \sum_{i=0}^{d-1}\left|Q_i - P_i\right|$$

$$d_{Max}(P,Q) = \max\left\{\left|Q_i - P_i\right|\right\}$$

Si des poids supplémentaires sont rajoutés à la dimension dans ce cas on définie la distance euclidienne pondérée et la distance max pondérée :

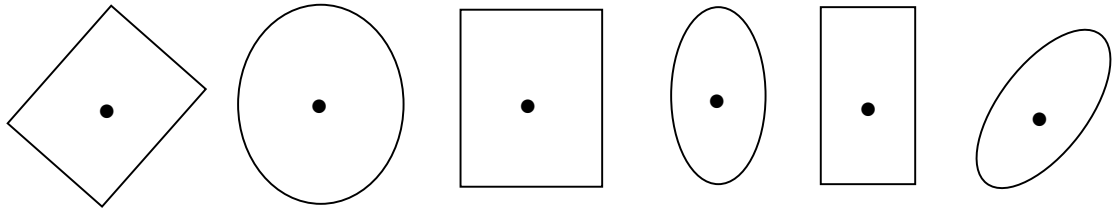
$$d_{W.euclid}(P,Q) = \sqrt[2]{\sum_{i=0}^{d-1} W_i (Q_i - P)^2}_i$$

$$d_{W.Max}(P,Q) = \max\left\{W_i . \left|Q_i - P_i\right|\right\}$$

Pour une recherche de similarité adapté il existe une distance dite distance ellipsoïde définie comme suit :

$$d_{ellipsoid}(P,Q) = (P-Q)^T . W . (P-Q)$$

La figure 2 représente les différentes distances définies auparavant :



Manhattan | Euclidienne | Maximum | Euclidienne pondérée | Maximum pondéré | Ellipsoïde

Fig.2- métriques de l'espace de données

3 Etat de l'art des méthodes d'indexation multidimensionnel

Les techniques d'indexations multidimensionnels sont basées sur la structuration et le groupement de données en cluster, elles reposent sur le même principe que les index traditionnels tel que les B-Tree [2], elles sont nécessaires dès que la base d'images devient conséquente et qu'il est obligatoire de stocker les descripteurs sur disque. Elles structurent les données en index pour accélérer les recherches de similarités en limitant autant que possible les entrées/sorties. La plupart des techniques d'indexation multidimensionnelles englobent les données en des cellules ayant une forme simple à manipuler (rectangles / sphères), ce qui permet lors d'une recherche de sélectionner les cellules pertinentes et de n'accéder qu'aux cellules sélectionnées, cela permet d'une part de réduire le temps d'entrée /sortie et le nombre de calculs de distance à effectuer, et par conséquent, le temps globale de la recherche. Elles peuvent être classifier en deux

grandes catégories : les méthodes d'indexation conventionnels et les méthodes basées sur l'approximation ou le filtrage.

3.1 Les techniques d'indexation conventionnelles

Les techniques d'indexation conventionnelles se classifient en deux grandes familles : La première famille est basée sur le partitionnement de données, elles regroupent les données en fonction de leur proximité dans l'espace. Tandis que la deuxième famille est basées sur le partitionnement le l'espace de données, elles découpent a priori l'espace multidimensionnel et, ensuite, stockent les données selon ce découpage. Toutes ces approches peuplent l'espace avec les descripteurs ou avec des approximations géométrique de ces descripteurs.

3.1.1. Les techniques basées sur le partitionnement de données

3.1.1.1 R- Tree, R*-Tree, R^+-Tree

Le R-Tree [3] est l'extension directe du B-Tree [2] aux espaces multidimensionnels. C'est un arbre équilibré dans lequel les vecteurs regroupés selon leur proximité relative, sont stockés dans les feuilles, alors que les nœuds interne sont constitués par une hiérarchie d'hyper rectangles englobants, les nœuds du niveau supérieur englobent les données du niveau inférieur. Un hyper rectangle est appelé rectangle englobant minimal (REM) et correspond au plus petit hyper rectangle pouvant englober l'ensemble des données du niveau inférieur et mémorise les caractéristiques de l'hyper rectangle englobant tous les objets (REMs ou vecteurs) stockés dans le sous arbre pointé. On peut dire que les MBRs d'un niveau inférieur appartiennent au REM du niveau supérieur, et par extension, un vecteur est dit appartenir au REM qui l'englobe. La taille des nœuds est limitée et fixée a priori. Elle correspond à la taille d'une page du disque. La figure 3 montre un exemple du R-Tree. Pour simplifier, seuls quelques points (rond noirs sur la figure) sont représentés. Les REMs A, B, C et D sont englobés dans le REM_1. Par construction, les REMs peuvent se chevaucher.

Selon l'algorithme présenté par [3], la construction de l'arbre s'effectue de manière dynamique par insertions successives, ces insertions peuvent causer des fractionnements de feuilles, de nœuds, et éventuellement augmenter la hauteur de l'arbre.

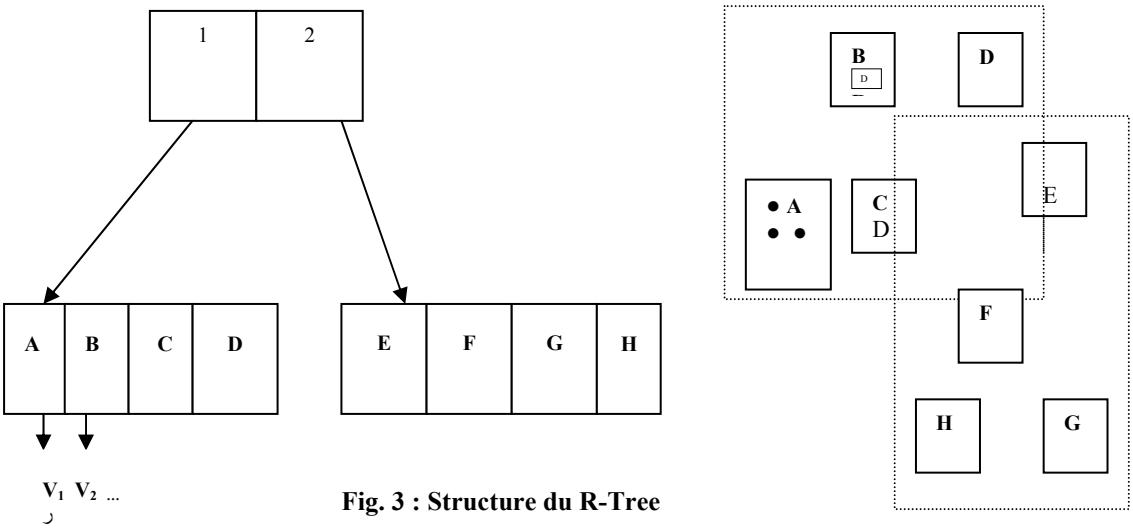


Fig. 3 : Structure du R-Tree

Pour gérer les dépassements de capacité des nœuds, les REMs du nœud saturé sont partitionnés en deux sous ensembles de même cardinalité, de façon à minimiser le volume de l'espace couvert par les deux formes englobantes des deux nouvelles sous ensembles. Le fait d'avoir deux sous ensembles de même cardinalité permet de préserver l'équilibre de l'arbre. La figure 4 illustre deux partitionnements possibles des REMs A, B, C et D. Le partitionnement (b) génère deux REMs qui occupent moins de surface que ceux du partitionnement (a). Minimiser les volumes des REMs permet d'éviter de gérer les zones d'espaces vides entre les REMs, ce qui permet d'accroître l'efficacité des règles de sélection des algorithmes de recherche.

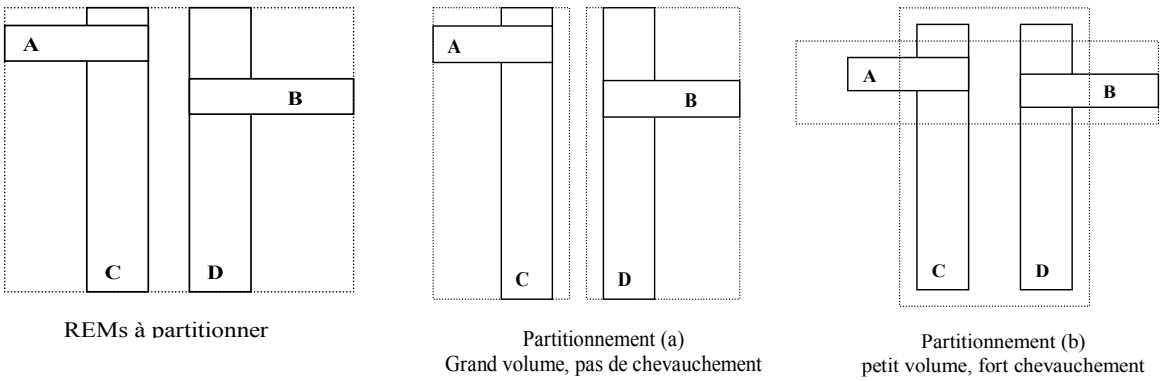


Fig. 4 : Deux exemples de partitionnement d'un ensemble de quatre REMs dans un espace de dimension 2

couvert par les REMs afin d'éviter de gérer les régions vides, et d'autre part de minimiser le taux de chevauchement entre les entrées du même noeuds. Pour éviter le chevauchement, Sellis propose d'autoriser à couper en 2 (voir rectangle D de la figure 5) et à mettre de chaque côté un objet sur la frontière du fractionnement à opérer. Un objet ainsi coupé est référencé par chacun de ses morceaux, ce qui entraîne la duplication de l'identifiant de l'objet autant de fois qu'il y a de fractions le constituant. Aucun chevauchement n'existe alors entre les formes englobantes. Un inconvénient de cette technique est le coût relativement élevé des opérations d'insertion puisque le fractionnant d'un hyper rectangle se répercute sur tous les niveaux inférieurs où doivent être dupliquées certaines informations

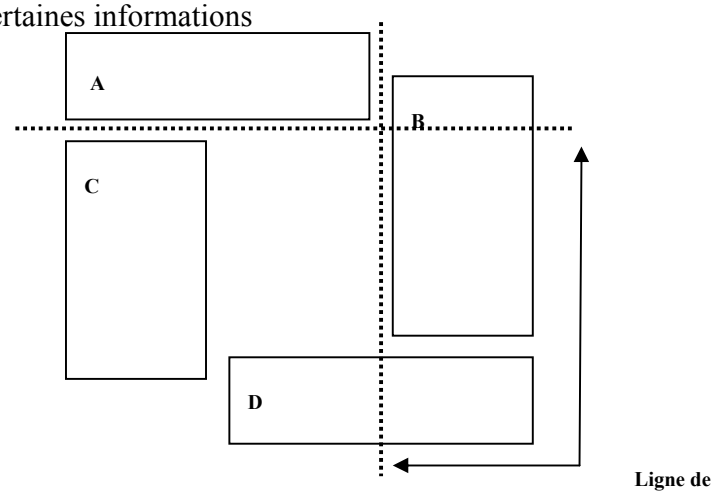


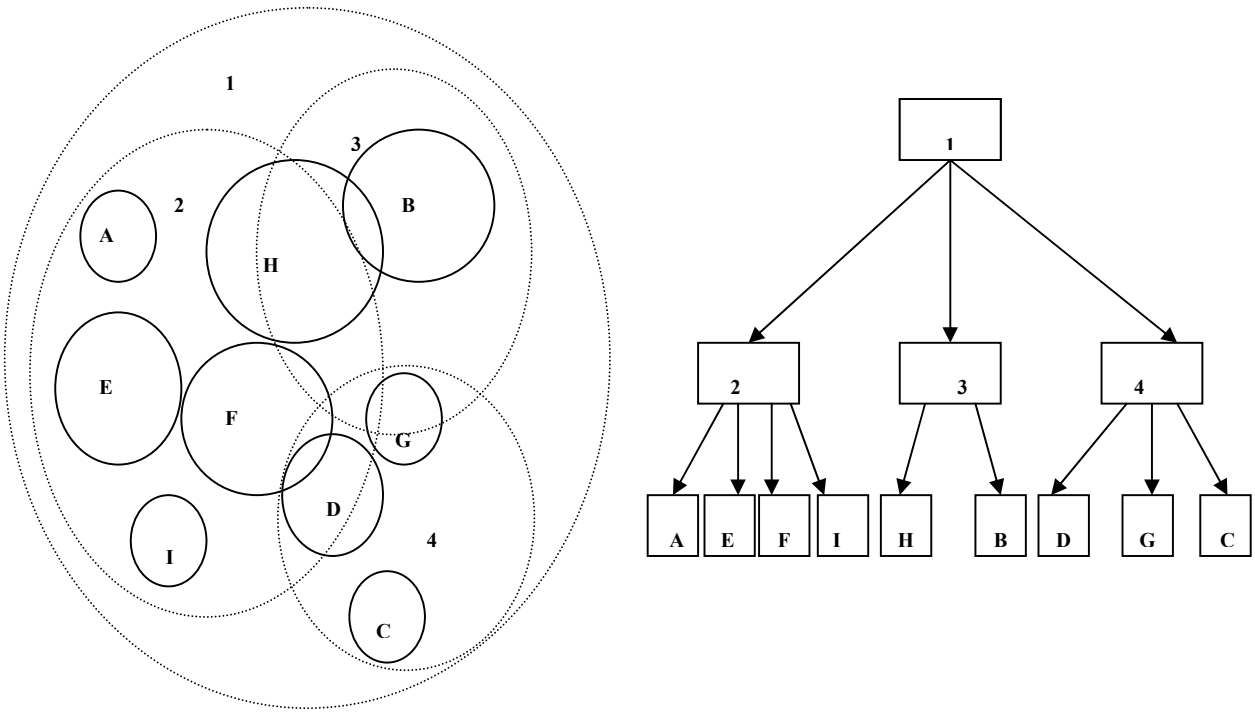
Fig. 5 Partitionnement d'un ensemble de REMs dans un espace de dimension 2

R*-Tree [5] cette technique améliore considérablement les performances de R-Tree, elle permet de limiter la sensibilité des R-Tree à l'ordre d'insertion des vecteurs. Beckmann propose alors de supprimer puis de réinsérer dans le même niveau de l'arbre un sous ensemble des entrées d'une page saturée avant de la fractionner. Cette réinsertion, permet dans la plupart des cas d'éviter le fractionnement. De plus elle permet de réorganiser l'arbre continuellement et diminuer les effets d'indéterminisme dus aux changements dans l'ordre d'insertion des données.

3.1.1.2 SS-Tree

SS-Tree [6] (similarity search Tree) est une structure d'index qui englobe les données dans des hyper rectangles. Le centre de la sphère est le centre de gravité des vecteurs

englobés (voir fig 6). La structure d'index est générée par insertions successives des données. Lors d'une insertion, un point est assigné à l'hyper sphère dont le centre est le plus proche. Chaque nœud est caractérisé par une certaine capacité définie par l'utilisateur. Lorsqu'un nœud est saturé, un partitionnement de ce nœud est réalisé selon la dimension suivant laquelle la dispersion des données est la plus grande. Ainsi, une hyper sphère est représenté 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'hyper sphère permet d'accroître la capacité des nœuds (fanout) et réduire la taille de l'arbre. Comme dans le R^*-Tree, le SS-Tree la réinsertion des données d'un nœud saturé avant de le fractionner, sauf que dans le cas des SS-Tree, la réinsertion d'un sous ensemble des entrées d'un nœuds 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 nœud. Dans ce dernier cas seulement le nœud est fractionné.



a. représentation des vecteurs dans un espace de dimension 2

b. représentation des vecteurs dans la mémoire

Fig.6 la structure SS-Tree

3.1.1.3 SR-Tree

SR-Tree [7] (Sphere/rectengle-Tree) est une structure d'index qui utilise à la fois des formes géométriques sphériques et rectangulaires pour définir les régions. Une région est définie par l'intersection d'un rectangle et d'une sphère. Les études menées en [7], montrent que les hyper rectangles divisent un espace de grande dimension en régions de petits volumes et de grandes diagonales. Tandis que les hyper sphères possèdent de petits diamètres et des volumes comparativement plus importants que ceux des hyper rectangles. En général, des régions avec de petits volumes diminuent le taux de chevauchement, et des régions de petites diagonales sont des régions compactes qui améliorent les performances des algorithmes de recherche. Ainsi le SR-Tree est une structure d'index qui combinent les avantages des deux formes géométriques, par contre la forme résultante des régions est complexe, et la taille qu'occupe chacune est grande ce qui diminue par conséquent la taille des nœuds (fanout) du SR-Tree. De plus, la complexité des formes englobantes accroît le coût des opérations d'insertion, de mise à jour et de recherche.

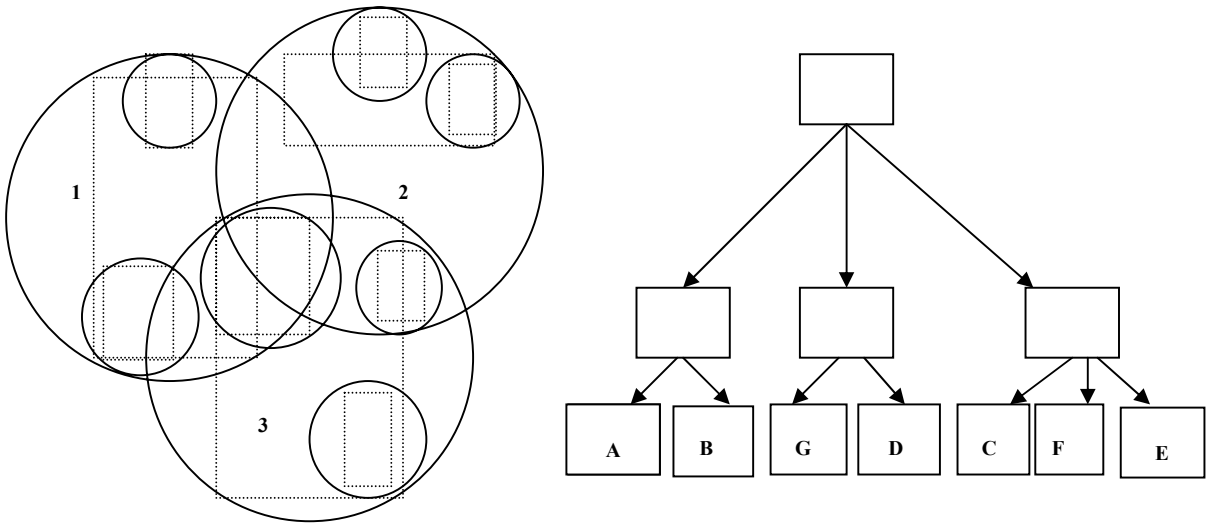


Fig. 7 la structure SR-Tree

3.1.2. Les techniques basées sur le partitionnement de l'espace de données

3.1.2.1 K-D-B-Tree

K-D-B-Tree [8] est une structure d'index multidimensionnel qui se base sur le partitionnement de l'espace en des régions disjointes organisées dans une arborescente équilibré. Le nœud racine de l'arbre correspond à la région qui contient les données de l'espace de dimension d. Les nœuds internes de l'arbre appelés pages région, contiennent les pairs (region, pageID) tel que « region » est la région délimitant l'ensemble de données et pageID est un pointeur sur la page fille. Les feuilles de l'arbre appelées « pages point » sont composées des paires (point, localisation). Le nombre de points de chaque feuille est limité par une certaine capacité appelée taille de la feuille.

La figure 8 représente la structure de K-D-B-Tree dans un espace de dimension 2

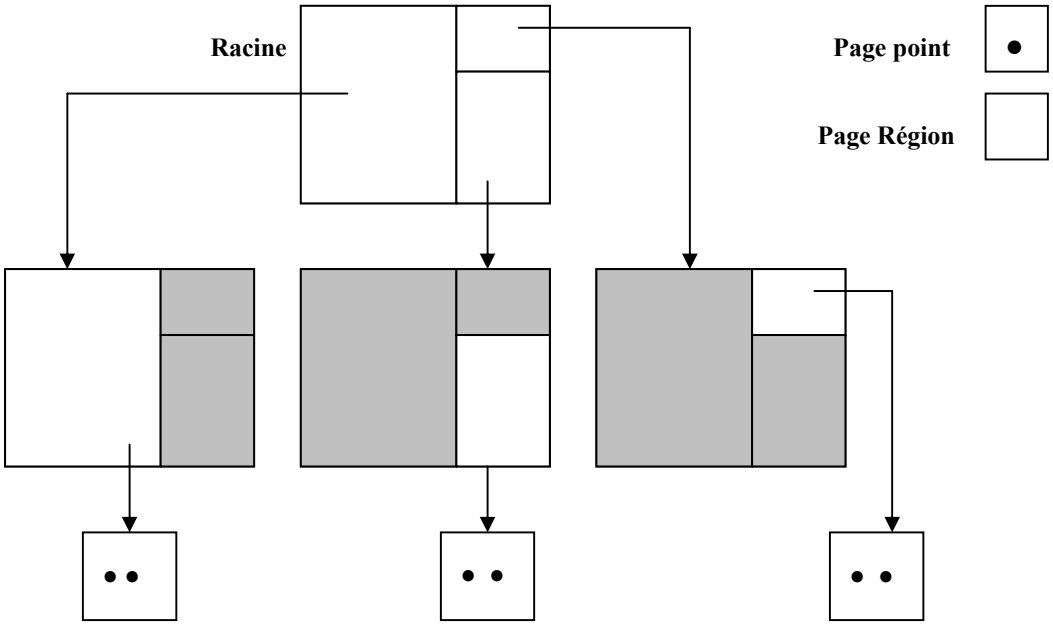


Fig 8. Structure d'un 2-D-B-Tree

Les nœuds de l'arbre sont caractérisés par une certaine capacité, lors d'un dépassement de capacité un partitionnement aura lieu pour préserver l'équilibre de l'arbre. On peut distinguer entre deux types de subdivisions : la subdivision des pages point et celle des pages régions.

i. La subdivision des pages point :

Elle aura lieu lors de l'insertion d'une nouvelle donnée dans une page point saturée c.à.d. le nombre de vecteurs de la page point est supérieure à la taille de la feuille. Dans ce cas, la page et décomposée en deux sous pages suivant un hyperplan de subdivision x_i perpendiculaire à un axe de coordonnée. La page point initiale est remplacée par les deux nouvelles pages point, et une nouvelle entrée est insérée dans la page région mère pour la $2^{ème}$ page point. La propagation de la subdivision des pages points est dite **upward.**
Un exemple d'une décomposition d'une page point est illustré dans la figure 9.

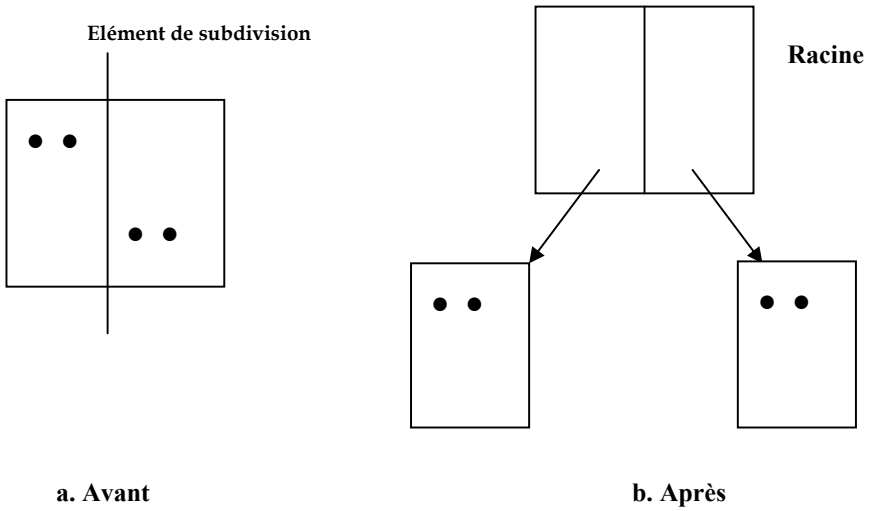
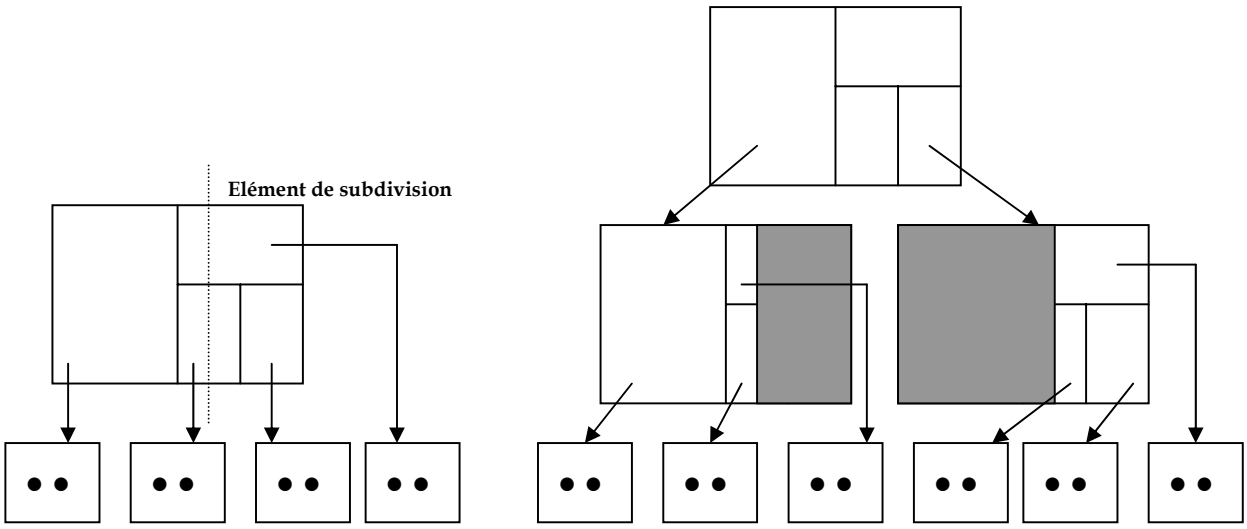


Fig.9. Exemple de décomposition d'une page point

ii. La subdivision des pages région :

Une page région et subdivisée, si le nombre d'entrées dans cette page dépasse la capacité de la région. Dans ce cas, le nombre d'entrées de cette région est partitionné en deux nouvelles pages régions disjointes. En effet, toutes les (region, pageID) de la page région ayant une intersection non nulle avec l'hyperplan de subdivision sont subdivisées ce qui entraîne par la suite, la subdivision des pages points. Donc la propagation de la subdivision des pages région est dite **downward**. Un exemple de subdivision des pages région est illustré dans la figure 10.



a. Avant **b. Après**

Fig. 10. Décomposition d'une page région

Notons que la subdivision des pages région est nuisible au stockage de données. En effet, la partitionnement d'un nœud interne (page région) saturé entraîne des restructurations importante de la totalité de l'arbre pour préserver les propriétés de la structure K-D-B-Tree, ces restructurations entraînent souvent la création des feuilles vides ou presque vide, ce qui ne peut garantir un taux minimal de l'utilisation de l'espace alloué.

3.1.2.2 LSD-Tree & LSDh-Tree

LSD-Tree [9] est une structure d'index organisée à deux niveaux, le premier niveau réside en mémoire centrale, alors que le deuxième niveau est stocké sur disque, c'est un arbre binaire dans lequel chaque nœud représente un partitionnement de l'espace en deux à l'aide d'un hyperplan. Les nœuds de l'arbre sont représentés par le numéro de la dimension selon laquelle le partitionnement est effectué, et la position sur l'axe associé à cette dimension. Les noeuds du bas niveau contiennent les pages de données. Ces pages sont de taille limitée et fixée a priori. La construction de l'arbre se fait par insertion successive des vecteurs. Lors d'un dépassement de la capacité des régions, un partitionnement du nœud est effectué, le choix de la position et la dimension du partitionnement se fait aléatoirement. La structure d'index ainsi crée est un arbre binaire non équilibré. L'exemple de la figure 11 montre un exemple de LSD-Tree

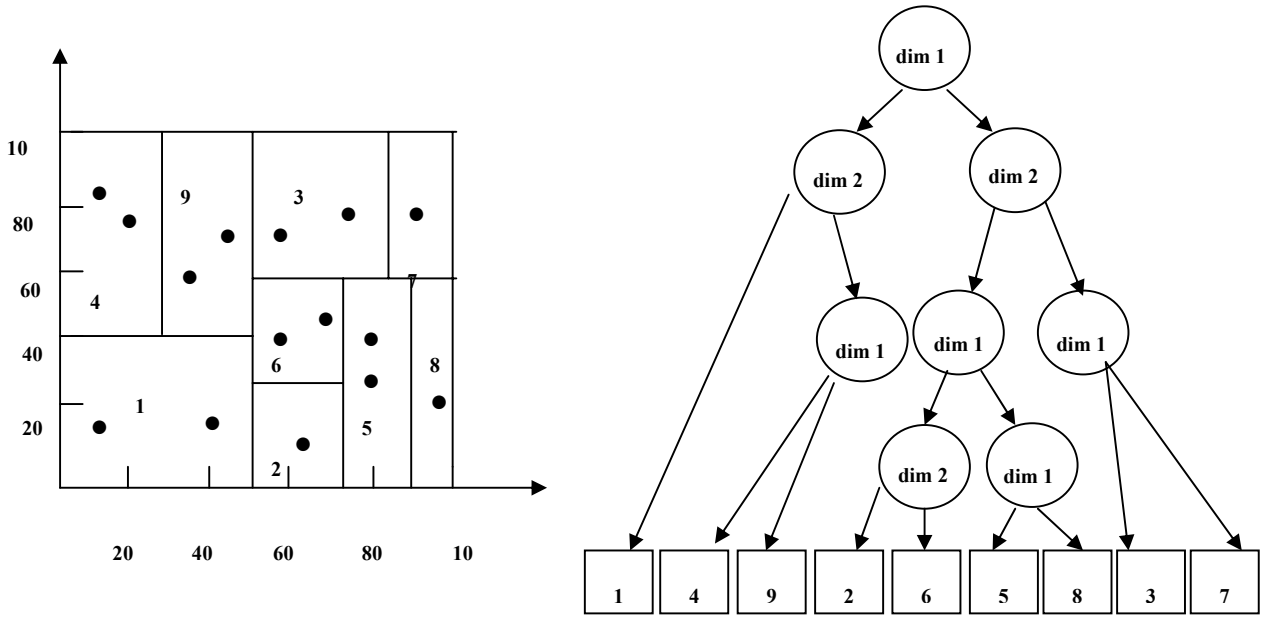


Fig.11 Exemple d'un LSD-Tree associé au partitionnement d'un carré dans un espace de dimension 2

La structure non équilibré de LSD-Tree ne permet d'avoir ni une estimation du temps de réponse ni une estimation de la taille du LSD-Tree, sa complexité n'est donc pas calculable. D'un autre côté, la représentation d'une région en utilisant uniquement la dimension et la position de partitionnement de la région mère ne permet pas d'avoir une localisation précise es vecteurs au sein de la région. Cela réduit les performances des algorithmes de recherche et entraîne une gestion inutile de grandes zones d'espace vide. De plus, la détérioration des performances dues à cet inconvénient s'amplifie lorsque la dimension des données croit.

Pour remédier à ce problème, le LSD-Tree a été étendu aux espaces de grande dimension par l'introduction du LSD^h-Tree [10]. Pour diminuer la taille de la structure d'index et éviter de gérer les régions vides de l'espace, le LSD^h-Tree utilise la notion de régions exactes codées. Il s'agit de retrouver au sein de chaque région la sous région où l'on y trouve effectivement les vecteurs, et de mémoriser cette sous région plutôt que la région elle-même. LSD^h-Tree englobe les vecteurs appartenant aux régions gérées par le LSD-Tree dans les hyper rectangles puis les codes à l'aide de chaînes binaires de petites tailles. Les régions codées sont ainsi des hyper rectangles englobants mais non minimaux.

3.1.3. Synthèse

Les techniques conventionnelles d'indexation multidimensionnelles sont efficaces

dans les espaces de petites dimensions. Plusieurs travaux ont montré que ces techniques effectuent des recherches de plus proche voisins d'une manière efficace dans les espaces de dimension <10 [1], leurs performances se dégradent cependant très rapidement lorsque la dimension de données croit, ils présentent une complexité exponentielle en fonction de la dimension de l'espace de données. Seule la recherche séquentielle garde un coût linéaire en fonction de la dimension, mais malheureusement, elle reste inapplicable dans les application réelles. Le tableau ci-dessus représente un récapitulatif des méthodes d'indexations citées précédemment.

Index	**Forme englobantes**	**complexité**	**équilibre**	**chevauchement**
R-Tree	hyper rectangle	calculable	oui	oui
SS-Tree	hyper sphère	calculable	oui	oui
SR-Tree	∩ hyper rectangle et hyper sphère	calculable	oui	oui
KDB-Tree	hyper rectangle	calculable	oui	non
LSD-Tree	hyper rectangles	non calculable	non	non

Index	**Points faibles**	**Points forts**
R-Tree	Forme englobante avec une grande diagonale => processus de recherche dégradé; taille importante des nœuds; fort taux de chevauchement.	Forme englobante permettant d'affiner les règles de filtrage
SS-Tree	Forme englobante avec un grand volume => le taux de chevauchement élevé.	Forme englobante avec une faible diagonale => arbre compacte.

SR-Tree	Formes englobantes complexes.	Forme englobantes adaptées aux grandes dimensions.
KDB-Tree	Faible taux d'utilisation de l'espace alloué	Pas de chevauchement
LSD-Tree	Recherche non précise des données	Bonne utilisation de l'espace alloué.

Tableau 1. Récapitulatif de quelques propriétés générales des méthodes d'indexation conventionnelles

3.2 Les techniques basées sur l'approche approximation ou filtrage

3.2.1 VA-File & VA$^+$-File

VA-File (Vector Approximation File) [1] est la première technique d'indexation efficace pour la recherche des k-ppv dans l'espace de grande dimension, c'est une amélioration d'une simple recherche séquentielle.

L'idée de base de cette technique repose sur la gestion de deux ensembles de données : un fichier qui contient tous les vecteurs de la base et un autre, de ″petite″ taille, contenant des approximations géométriques de ces vecteurs. Lors d'une interrogation, un premier parcours séquentiel du ficher d'approximation permet de sélectionner les vecteurs qui ont le plus de chances d'appartenir à l'ensemble des résultats. Ensuite, 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 ″petit″ fichier d'approximations, celle-ci est donc très rapide et n'entraîne ensuite l'accès à la base que pour un sous-ensemble réduit de vecteurs (ceux retenus). Ainsi, ce procédé diminue le nombre d'opérations d'E/S et le coût de calcul CPU par rapport à la recherche séquentielle qui, elle, analyse la totalité de la base.

Compression des vecteurs : Chaque dimension d_i est partitionnée en 2^{bi} intervalles chacun est codé sur b_i bits. A chaque cellule est attribué un code binaire de longueur $b=\sum_{i=1}^{d} b_i$ qui sont numéroté de 0 à 2^b-1. Les descripteurs de la base sont ainsi lus les uns après les autres, et l'approximation d'un descripteur est déterminée par le numéro de la

case qui le contient. Le fichier d'approximations est ainsi composé de paires (identifiant de descripteur, numéro de case). La figure 12 illustre un exemple de codage de vecteurs dans un espace de dimension 2.

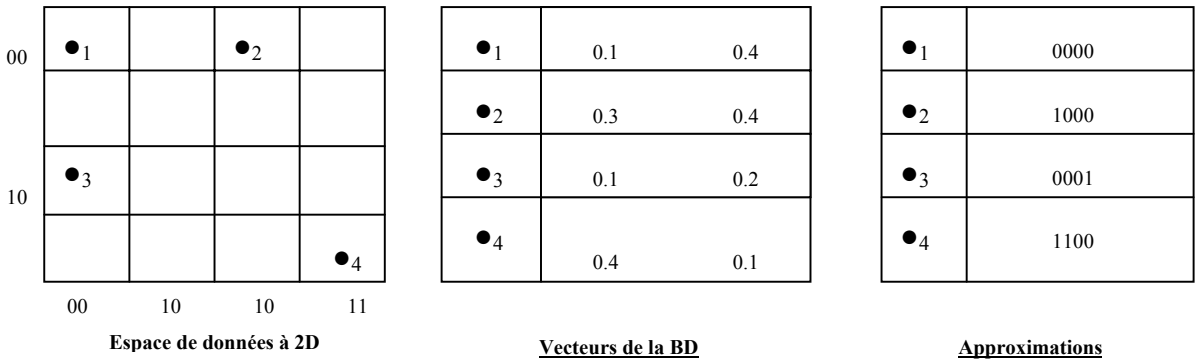


Fig. 12 : Construction du VA-File

Filtrage : Lors de la recherche du plus proche voisin, le fichier d'approximations est parcouru entièrement. Les bornes maximale et minimale sur la distance par rapport au vecteur requête peuvent être facilement déterminées en se basant sur les cellules rectangulaires représentées par l'approximation. Supposons que Ω est la plus petite borne maximale trouvée jusqu'à l'instant. Si une approximation est rencontrée telle que sa borne inférieure est supérieure à Ω, alors l'objet correspondant est éliminé puisqu'au moins un plus bon candidat existe. De la même manière, on peut définir une étape de filtrage lorsque k plus proches voisins doivent être retrouvés.

Accès aux vecteurs : Après l'étape de filtrage, un petit ensemble de points candidats reste. Ces candidats sont alors visités selon un ordre croissant de leur borne minimale par rapport au point requête q, et la distance exacte à q est calculée. Cependant, on ne parcoure pas tous les candidats. Plutôt, si une borne minimale rencontrée est supérieure à la plus poche distance jusqu'à cette étape, la recherche s'arrête.

Bien que cette méthode est efficace dans les grandes dimensions d>10 [1], elle reste limitée par plusieurs contraintes. En effet, la performance du VA-File dépend essentiellement de la taille du fichier d'approximation et du nombre de bits de codage utilisé dans la compression des vecteurs. Un grand nombre de bits de codage permet de créer une grille de subdivision fine (bonne précision sur la position des vecteurs) et génère en contre partie un fichier d'approximation assez grand.

D'un autre côté, VA-File suppose que les dimensions sont indépendantes, chaque

dimension est subdivisée indépendamment en des cellules hyper rectangulaires. Ceci ne peut être appliqué dans le cas des données non uniforme.

Pour remédier à ce problème VA-File a été étendu à VA^{+}-File [11], pour les données non uniformes. En effet, Hakan a remarqué que pour avoir une meilleure quantification, les dimensions de l'espace de données doivent être non corrélés entre eux. Il est donc important de les décorréler en appliquant une transformation unitaire sur l'ensemble de données. La transformation Karhumen Loeve permet de remplir efficacement cette tâche tout en réduisant la dimension de l'espace de données. Cette une technique très efficace pour les données non uniforme, elle donne des résultats très intéressants en grande dimension.

3.2.2 IQ-Tree

IQ-Tree [12] (Independant Quantization Tree) est une hiérarchie généralisée du VA-File. L'espace de donnée est subdivisé en des rectangles minimums englobant (MBRs) tel que les MBRs des nœuds feuilles sont approximés de la même manière que le VA-File. Le nombre de bits utilisés pour la quantification dépend des MBRs. En effet, les régions ayant un grand nombre de vecteurs, sont approximés par un nombre de bits assez élevé, et les régions peu denses, sont approximer par un nombre de bit est assez petit. C'est une méthode d'accès très performante puisqu'elle utilise des approximations relatives de la position des vecteurs dans l'espace de données, elle présente une sélectivité beaucoup plus meilleure que le VA-File par contre elle a une structure d'index assez complexe.

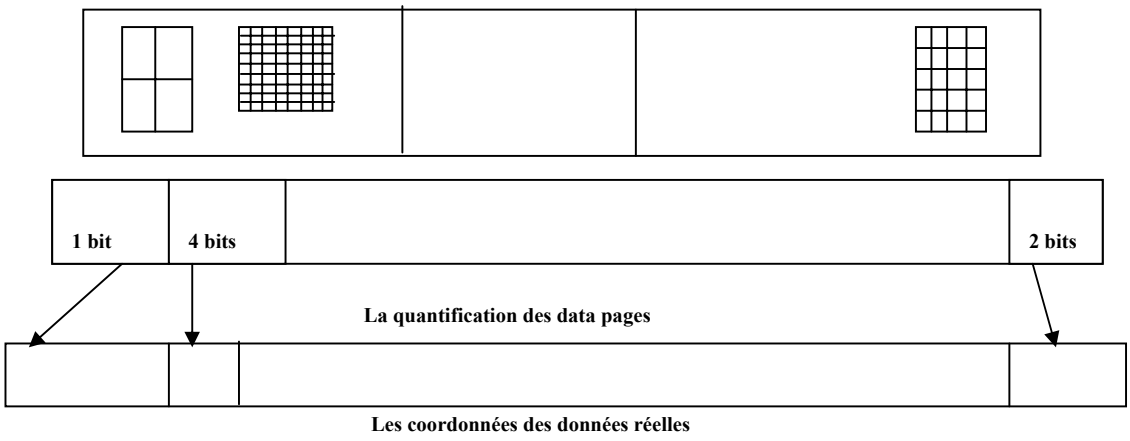


Fig. 13 Structure d'index IQ-Tree

3.2.3 A-Tree

A-Tree [13] (Approximation Tree) est une méthode d'indexation basée sur l'approximation similaire à R*-Tree. Contrairement à IQ-Tree, A-Tree utilise l'approximation de données dans tous les niveaux de l'arbre, elle introduit la notion de des rectangles englobants virtuels (VBRs) qui repose sur l'idée est d'approximer chaque MBR ou vecteur relativement par rapport à son MBR père. Cette approximation adaptative est beaucoup plus performante et permet d'avoir un arbre avec une grande capacité. Elle a montré de bonne performances par rapport à VA-File à la fois pour les données uniforme et non uniforme et pour une dimension d<64). Sa structure d'index est représentée dans la figure 14.

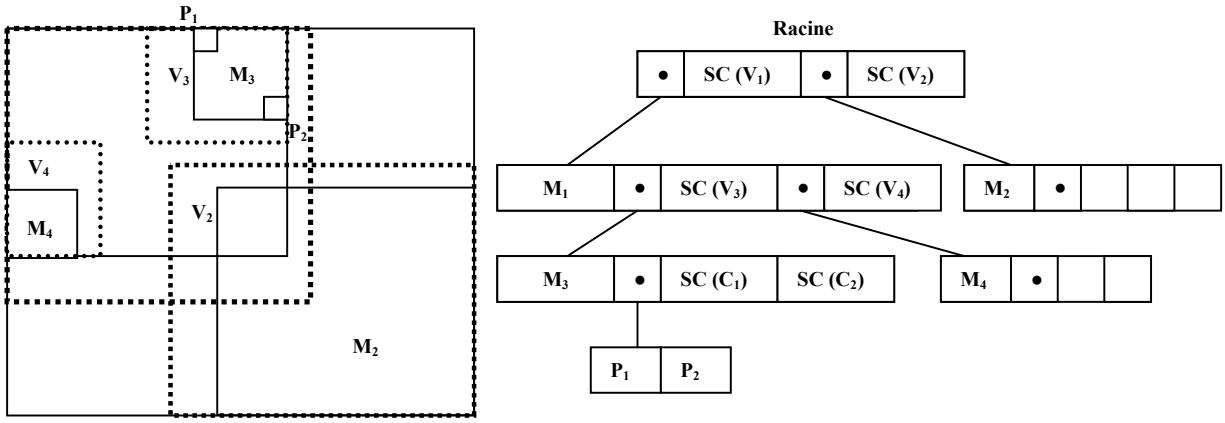


Fig.14 Structure d'index du A-Tree

3.2.4 RA-Blocks/ RA-Blocks adapté

RA-Blocks (Region Approximeted Blocks) [14], est une technique d'indexation basée sur l'approche filtrage. Elle suit la même démarche du VA-File dans la quantification de l'espace de grande dimension. En effet, l'espace de données est partitionné en des cellules hyper rectangles, chacune est représentée par une chaîne de bits. Ensuite, il est subdivisé en des régions disjointes en utilisant la méthode K-D-B-Tree. Ces régions sont codées par deux chaînes de bits correspondant aux deux hyper rectangles bas gauche et haut droit de chaque région, formant ainsi le fichier d'approximations à gérer. La figure 15 représente un exemple de codage des régions dans un espace de deux dimensions.

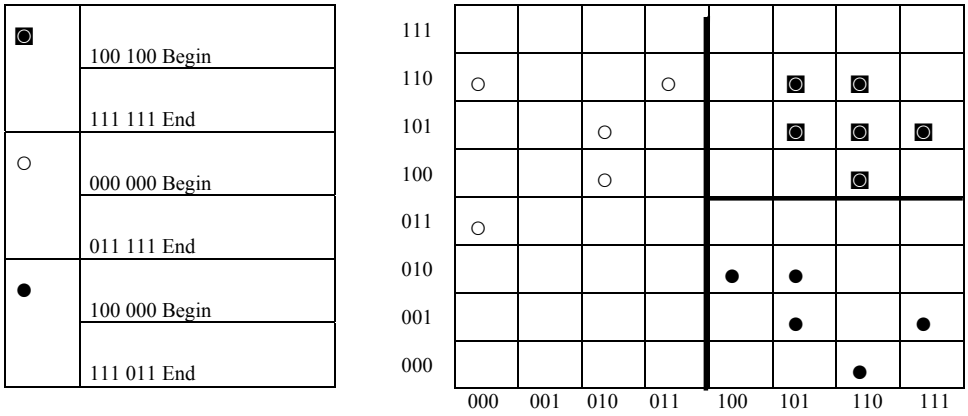


Fig. 13 : Exemple de codage des régions définies dans un espace de dimension 2.

Structure d'index : La structure d'index de RA-Blocks est une structure à deux niveaux : dans le premier niveau on trouve les RADATA (Region Approximeted Data) qui représentent les approximations codées en bits de chaque région, et dans le deuxième niveau, il y a les DATA-Pages qui contiennent les données réelles.

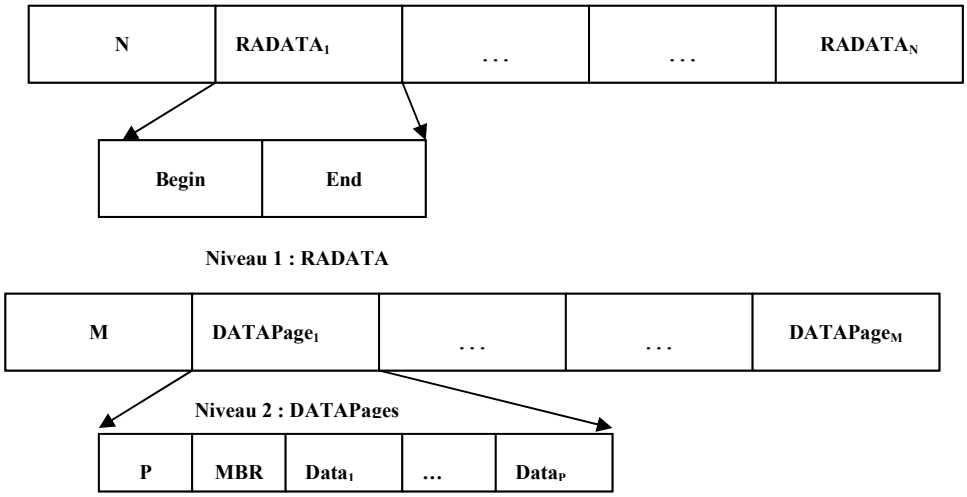


Fig. 14 : Structure d'index du RA-Blocks

A la différence du VA-File, RA-Blocks approxime les vecteurs par les régions qui les contiennent et non plus par des cellules hyper rectangle, ce qui permet d'une part de diminuer la taille du fichier d'approximation par rapport au VA-File. Et permet d'autre part, d'affiner les règles de filtrage, puisqu'on peut éliminer facilement toutes les régions ainsi que les vecteurs non similaires aux vecteurs requête. Cette méthode par contre présente un inconvénient aux niveaux du découpage de l'espace de donnée. En effet, le partitionnement avec la méthode K-D-B-Tree entraîne la création d'un très grand nombre de régions vides où presque vide ce qui diminue la performance du processus de

recherche. Pour résoudre ce problème, RA-Blocks a été étendu à RA-Blocks adapté [15].Son principe de base repose sur l'idée d'éliminer le partitionnement des nœuds internes (subdivision downward) de la structure K-D-B-Tree. L'inconvénient majeur de celle-ci est le fait qu'elle produit des restructurations importantes de la totalité de l'arbre pour préserver les propriétés de la structure K-D-B-Tree. Ces structurations provoquent souvent la création d'un très grand nombre de feuilles vides ou presque vides, ce qui ne peut garantir un taux optimal d'utilisation de l'espace alloué, et par la suite augmente le temps de recherche des k plus proches voisins. L'idée donc est de transformer la structure arborescente en une liste de régions. Par la suite, chaque région saturée sera subdivisée et remplacée par deux nouvelles régions (région gauche, région droite) ayant chacune une taille inférieure à la capacité des régions. Ceci permet d'une part, d'éliminer les régions triviales (vides) et de ne générer que des régions denses, et d'autre part, garantie l'existence, dans la plupart des cas, des plus proches voisins dans une même page disque, ce qui permet de réduire le nombre total des régions obtenues et par la suite le temps d'E/S. Un exemple de subdivision utilisée par la méthode RA-Blocks adapté est illustré dans la figure qui suit :

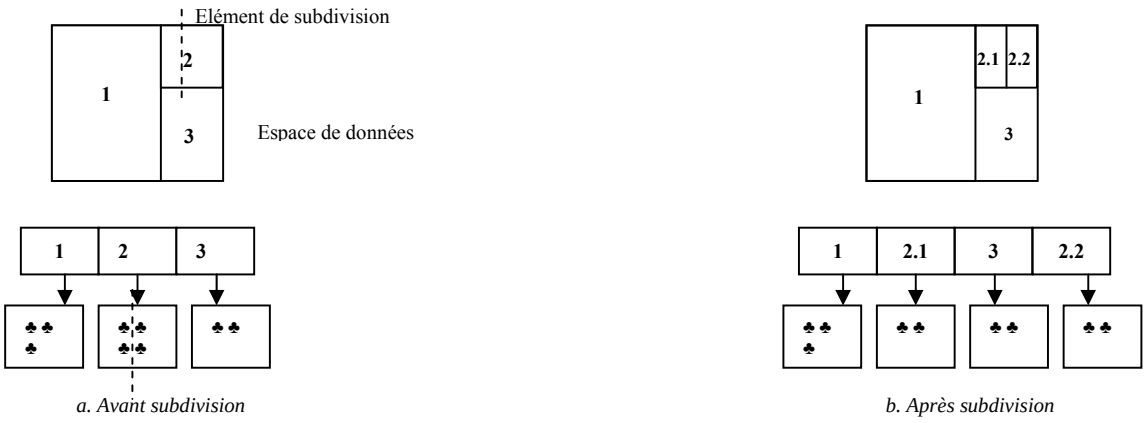


Fig. 15. Exemple de subdivision des régions RA-Blocks adapté

RA-Blocks adapté permet d'obtenir un fichier d'approximations tellement compact qu'on peut le stocker entièrement en mémoire principale sans avoir besoin d'accéder au disque. La stratégie de partitionnement de l'espace de données adopté par cette méthode permet de bien réduire le temps de recherche en éliminant les régions triviales et peu denses. Un autre avantage de cette technique est l'utilisation optimale des pages disques puisque les vecteurs voisins sont souvent stockés soit dans la même page disque, soit sur des pages

disque voisines. L'évaluation de cette méthode a montré de bonnes performances comparée à la méthode séquentielle, notamment dans les grandes dimensions (>60) et même pour un nombre de descripteurs important [14].

3.2.5 Synthèse

Les méthodes d'indexation basées sur l'approche approximation reposent sur l'idée d'approximer les données (vecteurs) pour réduire leurs taille dans le fichier de données et par la suite, réduire le nombre d'entrée/sortie au fichier de données réelles. Ces méthodes sont très adaptées aux applications réelles puisqu'elles présentent un temps de réponse assez réduits par rapport aux méthodes d'indexation conventionnelles, pour un volume important de données et pour des dimension d>10.

Le tableau ci-dessus présente une synthèse des méthodes d'indexation basées sur l'approche approximation citée dans les paragraphes précédents :

Index	Structure	Points faibles	Point forts
VA^{+}-File	Approximation des vecteurs	Compromis précision de l'approximation / taille du fichier d'approximation	Adapté aux données non uniforme
IQ-Tree	R*-Tree + approximation des vecteurs	Structure d'index complexe	Adapté aux grandes dimension, bonne sélectivité des régions.
A-Tree	R*-Tree + approximation des vecteurs/MBR	Taille importante des noeuds	Adapté aux données non uniforme, grande capacité des régions
RA-Blocks adapté	Approximation des régions	Non adapté aux données non uniformes	Adapté aux grandes dimension, bonne gestion des pages disque.

Tableau 3. Récapitulatif de quelques propriétés générales des méthodes d'indexation basées sur l'approche approximation

4 Système de recherche existant

Nos travaux se basent essentiellement sur le système de THIS [18] . Il s'agit en faite, d'appliquer une, des méthodes d'indexation citées dans le paragraphe précédant, pour améliorer le processus de recherche d'image par le contenu. Pour ce faire, il est donc nécessaire de comprendre l'architecture générale du système, les descripteurs mise en jeu et la méthode de comparaison, c'est pourquoi dans cette partie le système THIS est détaillé.

4.1 Architecture générale

Le système d'indexation THIS (Things oriented image retrieval System) est un système orienté objet :. Il considère que la base d'images est interrogée par un simple dessin de l'utilisateur (modèle). La comparaison d'une image de la base de donnée et le modèle se base sur une hiérarchie de segmentation. En effet, pour toutes les images de la base de donnée, le système THIS réalise une sur-segmentation suivie d'un groupement perceptuel des différentes régions obtenues lors de la segmentation. Enfin, pour chaque niveau de hiérarchie, des descripteurs sont extraits pour réaliser la comparaison entres les images de la base (requête) et le modèle.

4.1.1 Le modèle

Dans la conception du système THIS, le modèle est constituée d'un objet lui-même composé de différentes parties (sous partie du modèle), noté M_1, M_2,…,M_n. Chaque partie M_i est caractérisée par différents descripteurs. L'objet, appelée aussi modèle peut être reconnu par sa liste non ordonnée de sous parties. La figure qui suit représente un exemple simple de modèle.

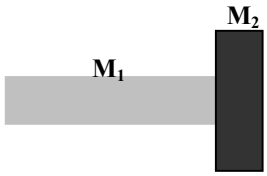


Fig. 16 Exemple de requête (modèle) composé de deux sous parties

4.1.2 Sur segmentation

La sur-segmentation se base sur une segmentation couleur, chaque image est partitionner en des régions homogènes en terme de couleurs. Un exemple de sur-segmentation est représenté dans la figure 17 :

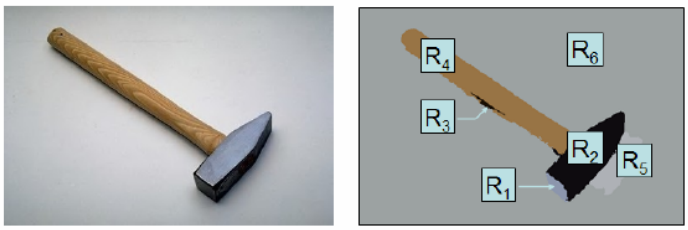


Fig 17. Exemple d'image et de sur-segmentation associé

Cette phase de segmentation est souvent entachée d'erreurs et elle est presque impossible à réaliser dans des domaines non contraints. Elle conduit souvent à des descripteurs extraits sur des régions non pertinentes. Par exemples, lorsqu'un objet comporte une ombre, la segmentation va conduire à séparer la zone d'ombre de l'objet et fournira alors des descripteurs pour chacune d'entre elles. Il est donc, nécessaire de regrouper ces régions pour essayer de tendre vers des objets sémantiques, ceci est possible donc en utilisant une notion beaucoup plus générale que la segmentation : le groupement perceptuel.

4.1.3 Groupement perceptuel

Le groupement perceptuel est une technique qui permet de réduire les problèmes de la segmentation en fusionnant les régions qui ont tendance à appartenir à un même objet par l'utilisation de propriétés psycho vusuelles telles que les propriétés de Gestalt, notamment la proximité, la similarité, la compacité, la continuité et la symétrie. A partir de ces fusions, on construit une hiérarchie de segmentation, ou l'image est représentée par un arbre de région qui décrit l'image avec un certains niveau de détail. Un exemple d'arbre de régions obtenues à partir de la sur-segmentation de la figure 17 est représenté sur la figure 18.

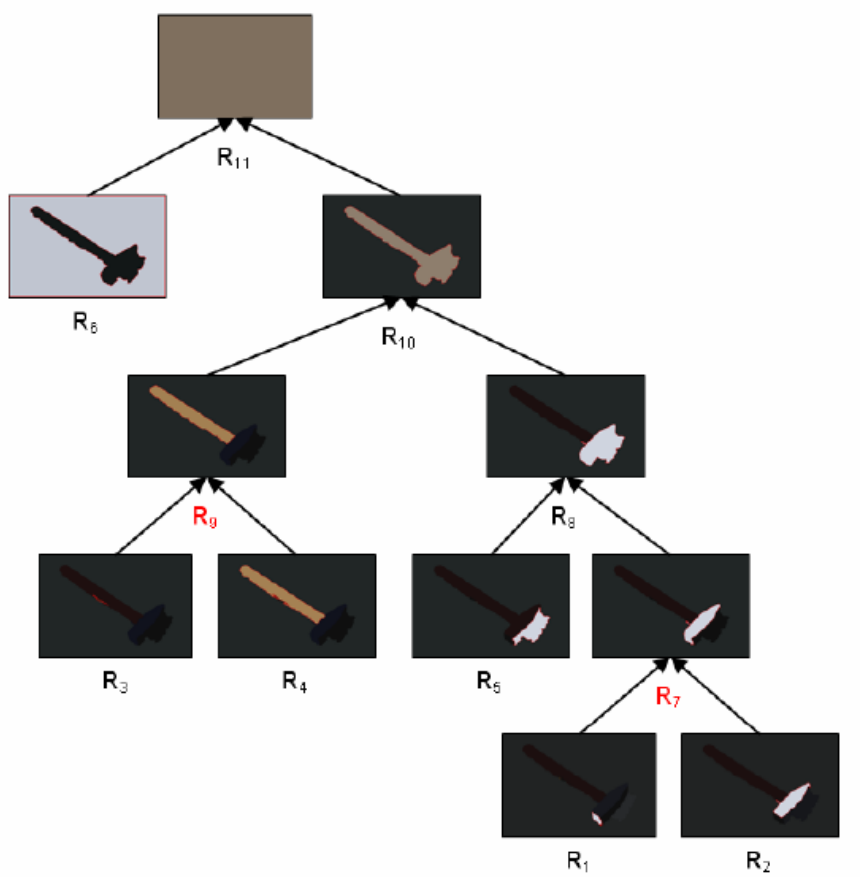


Fig. 18 Exemple d'arbre de régions, obtenus à partir de la sur-segmentation de la figure 17

Les feuilles de l'arbre représentent l'ensemble des régions retrouvées après une segmentation couleur, la fusion de deux régions permet de créer un père pour les deux régions fusionnées. Au fur et à mesure du processus, les régions seront fusionnées pour arriver à une région qui représente des objets réels de l'image, le nœud correspondant à ces objets est appelé nœud sémantique (R7 et R9 dans la fig 18).

4.1.4 Les descripteurs utilisés

Pour comparer une région R de l'arbre de région et une partie du modèle M, des descripteurs sont extraits de chaque région de l'arbre et de chaque partie du modèle. On distingue deux types de descripteurs :

- Ceux qui dépendent uniquement de la région traitée, appelés descripteurs région
- Ceux qui dépendent de la région traitée ainsi que l'objet référence auquel elles appartiennent, ces descripteurs sont appelés : les descripteurs structurels.

Descripteurs régions :

Les descripteurs les plus robustes en terme de reconnaissance d'objets sont les descripteurs de formes. On distingue les descripteurs basés région et les descripteurs basés contour. Le premier type de descripteurs décrit une forme par sa distribution spatiale de pixels, alors que le deuxième type ne considère que le contour de l'objet pour caractériser celui-ci. Le système THIS utilise les deux descripteur ART (Angular Radial Transform) et CSS (Curvature Scale Space) comme descripteurs de forme basés région et contour respectivement.

ART : est un descripteur de forme robuste au changement d'échelle, à la translation, et à la rotation, il consiste à projeté l'objet à étudier sur une série de fonctions de base[16].

CSS : [17] : c'est un descripteur qui caractérise les contours des objets, il est robuste aux changements d'échelle, à la translation et à la rotation, il consiste à suivre les positions des points d'inflexion d'un contour, alors que celui-ci subit une série de filtrage gaussiens passe bas. Au fur et à mesure des itérations de filtrage, le contour deviennent de plus en plus lisse et les inflexions non significatives sont éliminées. Les points d'inflexion qui subsistent sont considérés comme étant caractéristiques du contour.

Descripteurs structurels :

Dans le système THIS chaque région est modélisée par trois descripteurs de structure :

- La position relative de la région par rapport à l'objet entier.
- La taille relative de la région par rapport à l'objet entier.
- L'orientation relative de la région par rapport à l'objet entier.

Pour estimer la taille et l'orientation de l'image, des ellipses englobantes sont utilisées. La comparaison se fait alors par le recalage des ellipses englobantes de la partie du modèle M et une région R de l'arbre des régions.

4.2 La recherche d'images

La recherche d'images dans le système THIS, consiste à chercher un model M dans un arbre de régions R. Il s'agit en faite de calculer une distance globale |M - R| entre le modèle M et l'arbre de régions R. Cette distance est calculée de la manière suivante :

On cherche d'abord un sous arbre optimal SA_k contenant le plus grand nombre de nœuds sémantiques sachant qu'un arbre de région R est composé d'un ensemble de sous arbre

($SA_{1,...,}$ SA_k,…, SA_n) contenant les nœuds les plus similaires au modèle (nœuds sémantiques). La recherche d'un tel sous arbre se fait en calculant les distances globales entre tous les sous arbres possibles et le modèle., et en séléctionnant la distance minimale.

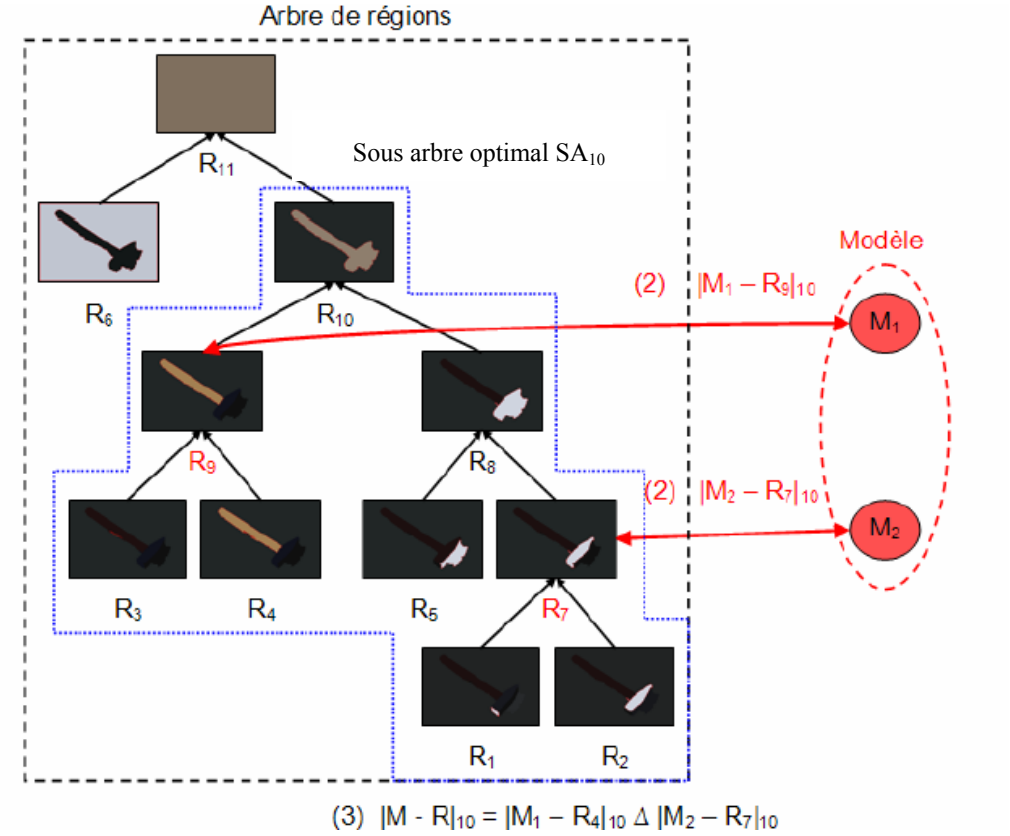


Fig. 19 Recherche de modèle M dans un arbre de régions

En effet, pour chaque sous arbre SA_k, on calcul une distance entre chaque partie du modèle M_i avec chaque région R_j du sous arbre SA_k. Cette distance $|M_i - R_j|_k$, est la somme pondérée des distances entre les descripteurs ART, CSS et les descripteurs structurels.. Chacune des distances $|M_i - R_j|_{k,}$ sert au calcul d'une distance globale $|M - R|_k$ entre le modèle M et le sous arbre ST_k. La combinaison de ces distances met en jeu la théorie de l'évidence [18]. La figure 19 illustre un exemple de recheche d'un modèle M dans un arbre de régions.

4.3 Synthèse

On distingue deux grandes étapes dans le système existant : indexation et recherche d'images. L'indexation consitant à extraire les descripteurs carractéristiques des images de la base de données peut être résumée dans le shéma ci-dessus :

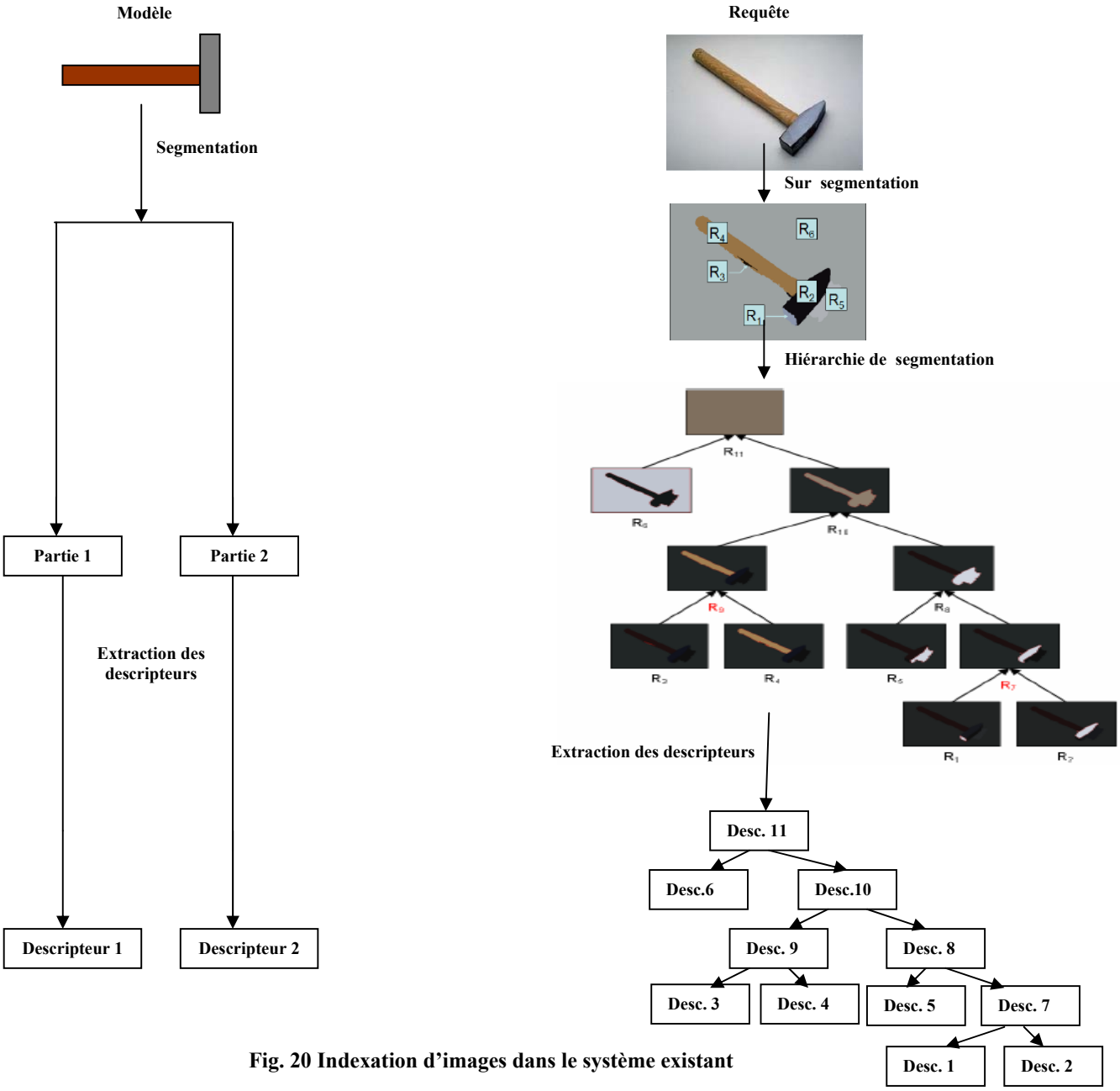


Fig. 20 Indexation d'images dans le système existant

La recherche d'images consiste à rechercher les descripteurs similaires, en terme d'une distance globale, à des descripteurs du modèle. Il s'agit de calculer des similarités partielles entre les différents descripteurs (liés aux parties) du modèle et tous les descripteurs de l'image requête. L'ensemble des similarités locales est combiné pour déduire une similarité globale entre le modèle et la requête. On distingue deux cas de figure dans la comparaison : selon que le modèle est composé d'une seule région ou de plusieurs régions.

1er cas : le modèle est composé d'une seule région

Dans ce cas, seuls les descripteurs ART et CSS sont utilisés pour le calcul des distances. Le processus de recherche parcourt tous les descripteurs de la base d'images et sélectionne les images ayant les distances les plus petites.

2ème cas : le modèle est composé de plusieurs régions.

Dans ce cas, le calcul de distance se fait sur la base des descripteurs ART, CSS et les descripteurs structurels. Le processus de recherche calcule, pour toutes les images de la base de données (les arbres de région), l'ensemble des similarités partielles entre chaque descripteur du modèle et une région de l'image requête. L'ensemble de ces distances est combiné ensuite pour déduire une distance globale entre le modèle et chaque image de la base de données. Un exemple de comparaison est illustré sur la figure 21.

En résumé, le processus de recherche effectue deux parcours séquentiels, un parcours séquentiel global et un autre locale. Le premier, parcourt toutes les images de la base de données pour en déduire une similarité globale entre le modèle et chaque image de la base de données. Le deuxième, parcourt toutes les régions des images requêtes pour calculer une similarité partielles. Sachant que la recherche séquentielle d'images nécessite une durée linéaire en fonction du nombre de données, le processus de recherche dans le système THIS n'est donc pas optimale en terme temps de réponse, il n'est plus adapté aux applications réelles (un volume important d'image, une dimension d>10). Ce processus présente une complexité exponentielle en fonction du nombre d'images dans la base de donnée. D'un autre côté, la structure arborescente dans le système THIS, n'est pratiquement pas utilisée pour la recherche d'images alors, qu'il y a plusieurs méthodes de recherche arborescente optimale pour des dimension <10 (voir chapitres précèdent).

Pour adapter ce système aux applications réelles, il est donc nécessaire d'organiser en

index l'ensemble des descripteurs de la base de données pour appliquer une recherche des k plus proches voisins. La question qui se pose, est comment et quelle structure d'index faut il choisir pour ce système ?

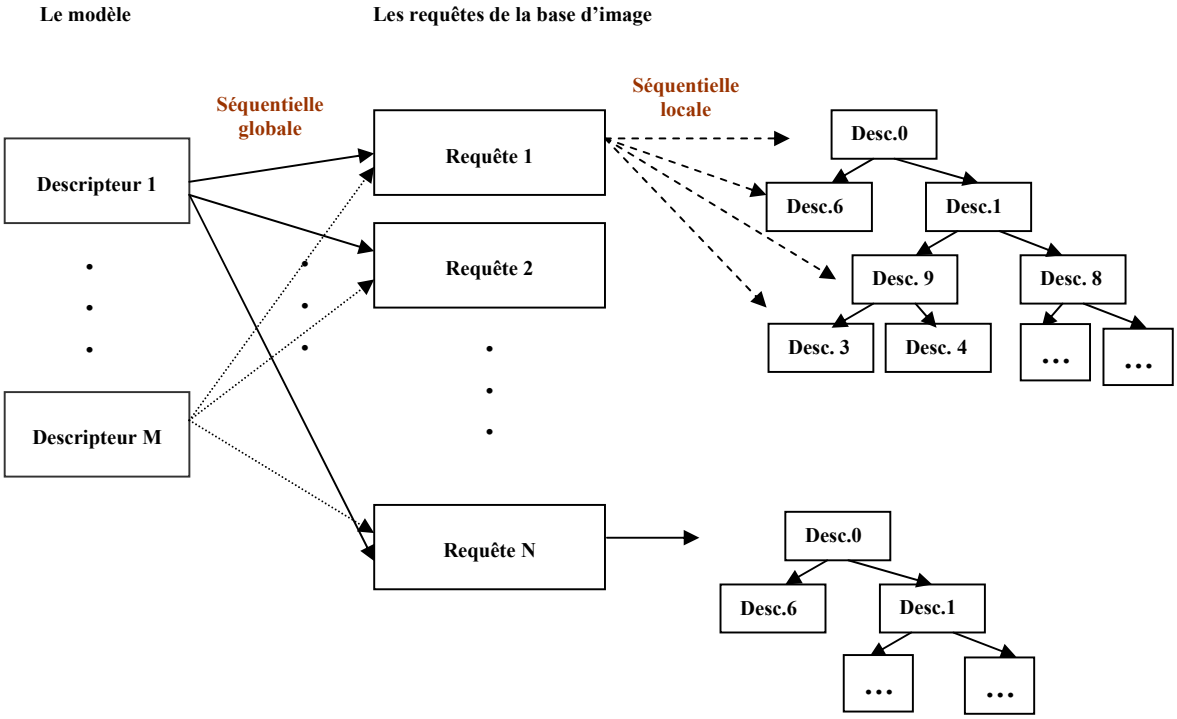


Fig. 21 la recherche d'images dans le système existant

5 Les structures d'index proposés pour le système existant

5.1 Positionnement du problème

Il s'agit en faite, d'appliquer une méthode de recherche d'images basée sur le contenu, en tenant compte des différents descripteurs utilisés pour décrire les images de la base de données. Il s'agit d'une part, de rechercher dans une même image les différents objets les plus similaires des objets du modèle, et d'autre part de rechercher l'image la plus similaire en terme d'une distance globale entre les sous objets du modèle et ceux de la requête. Pour, ce faire il faut appliquer une structure d'index qui tient compte de la nature des descripteurs utilisés (distribution, espace de données, dimension) et qui ensuite, préserve la hiérarchie utilisée décrivant les images à différents niveaux de détail. La structure d'index utilisée doit être adaptée à une recherche multicritères.

5.2 Le choix de la méthode

En se basant sur l'étude menée dans le paragraphe 3, nous constatons que, les méthodes d'indexation basées sur l'approche approximation (dite aussi filtrage) tel que VA-File et RA-Block sont performantes et répondent efficacement au problème de passage à l'échelle, contrairement aux méthodes d'indexation conventionnelles qui sont toutes dépassées par la recherche séquentielle. En effet, la recherche séquentielle a un coût linéaire en fonction de la dimension de l'espace de données et le nombre d'images dans la base de données, ce qui rend ce mode de recherche et par la suite toutes les méthodes d'indexation conventionnelles inadaptées aux grandes bases d'images

Pour cela, nous avons décidé de nous placer dans le contexte de la recherche par approximation d'images. La méthode qui semble la plus approprié au système existant est la méthode RA-Block adapté puisque d'une part, c'est une méthode très efficace au passage à l'échelle (grande dimension, volume très important d'images). Et d'autre part, elle possède une structure d'index dynamique et facile à implémenter. De plus, La méthode RA-Blocks adapté améliore les méthodes VA-File et RA-Blocks et son code existe déjà (nous l'avons déjà implémenté dans nos premiers travaux sur l'indexation d'images).

5.3 Les solutions proposées pour l'implémentation

Comme toutes les méthodes d'indexation multidimensionnelles basées sur l'approche approximation, La méthode RA-Block adapté repose sur le partitionnent récursif de l'espace de donnée en des régions/cellules ayant le même nombre de données. Cette méthode considère que les descripteurs sont des points multidimensionnels dans un espace de dimension fixe. D'un autre côté, les images dans le système THIS sont décrites par un ensemble de descripteurs de nature différentes (CSS, ART, descripteur structurel) et de cardinalité non fixe. Le premier problème qui se pose, est comment représenter ces descripteurs et quel espace de données utiliser ?

Pour traiter ce problème, deux solutions ont été proposées et implémentées.

5.3.1 Solution 1

Consiste à concaténer les descripteurs ART, CSS, et le descripteur structurel de chaque région de l'image en un seul vecteur. Chaque descripteur est représenté par un vecteur multidimensionnel de valeurs normalisées. Le choix de la valeur de normalisation dépend de la distribution des composantes des descripteurs ART, CSS, et le descripteur structurel, elle correspond à la valeur maximale moyenne de leurs composantes. L'ensemble de ces descripteurs est organisé dans un seul espace vectoriel dont la dimension est le nombre de composantes de chaque vecteur (voir figure 22).

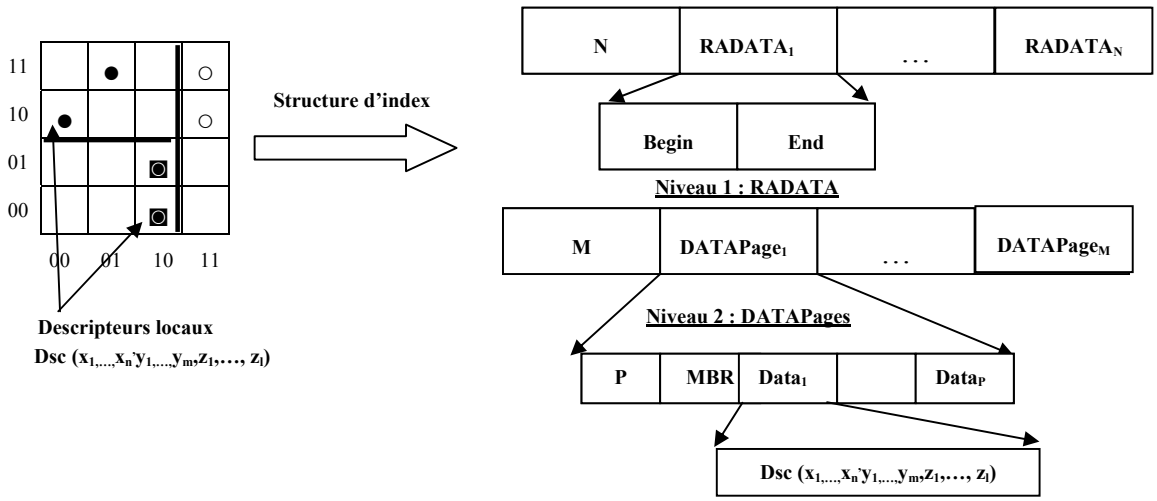


Fig. 22 : Exemple de structuration de données dans un espace de dimension 2.

La recherche d'images avec cette méthode se fait en deux étapes. Dans la première étape dite filtrage, on sélectionne les régions (RADATA) qui ont le plus de chance de contenir les descripteurs les plus proches du descripteur du modèle. La sélection se fait sur la base des distances euclidiennes entre le descripteur du modèle et les deux vecteurs bas gauche et haut droit de chaque région, ces vecteurs sont généralement le résultat de la subdivision de l'espace de données en des régions disjointes. Dans la deuxième étape, on sélectionne les k plus proches voisins en se basant sur les distances euclidiennes entre le descripteur du modèle et les descripteurs des régions parcourues.

Les résultats expérimentaux sur une base de 100 images, montrent que cette méthode réduit bien le temps de réponse (31 ms) par rapport la méthode de recherche d'images du système THIS (1.3 min). Par contre, on remarque que les images résultats sont différentes de celles du système THIS. Ceci peut être expliqué d'une part, par le faite que notre méthode de conception met en jeu des distances différentes de celles utilisées par THIS, et d'autre part, cette méthode dépend de la valeur choisie pour la normalisation des descripteurs, le choix de cette valeur reste un problème crucial du faite que les descripteurs utilisés sont de natures différentes. D'un autre côté, la structure d'index de la méthode proposée, regroupe les descripteurs jugés similaires en terme d'une distance euclidienne, sous forme des régions disjointes dans un espace de vecteurs multidimensionnel. Ces descripteurs représentent généralement les régions de toutes les images de la base de donnée. En appliquant le processus de recherche décrit précédemment, on va sélectionner les images qui possédant les/ le descripteur le plus similaire au descripteur modèle, alors que l'objectif est de sélectionner les images les plus similaires au modèle. Avec cette conception on néglige la hiérarchie utilisée par le système THIS, ainsi que la notion de la distance globale utilisée dans la comparaison.

5.3.2 Solution 2

Elle consiste à organiser les descripteurs de chaque image requête dans une structure d'index différente à deux niveaux. Dans le premier niveau sont stockées les approximations codées en bits des régions contenants les descripteurs similaires, et dans le deuxième niveau sont stockés les descripteurs ART, CSS et le descripteur structurel de chaque région d'une image. La structure d'index est représentée dans la figure 23.

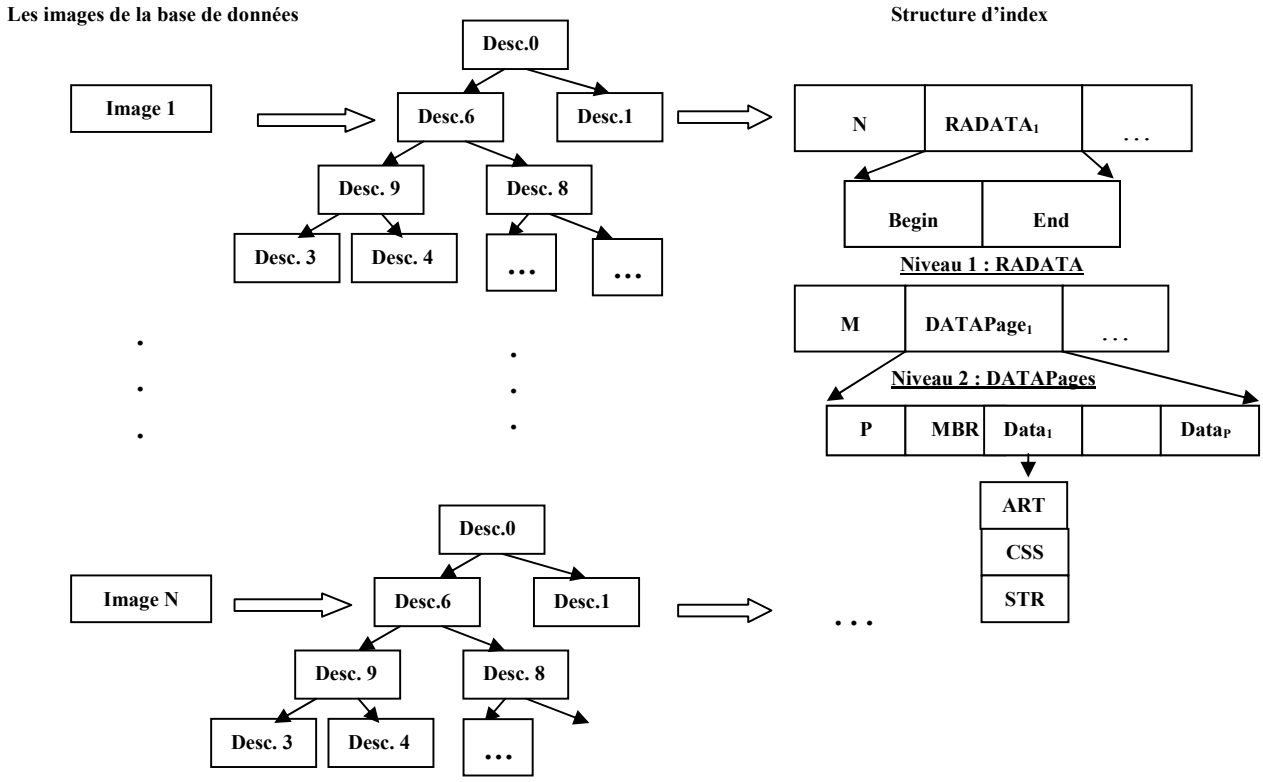


Fig. 23 Exemple de structuration de données.

Notons que la structure d'index est générée à partir des descripteurs ART. En effet, on considère que l'espace de données contient uniquement les descripteurs ART, ces derniers sont organisés dans des régions disjointes ayant pratiquement le même nombre de vecteurs. Pour chaque région on stocke également le descripteur CSS et le descripteur structurel associé.

La recherche d'image se fait de la manière suivant : pour chaque image de la base de données, on sélectionne d'abord les régions ayant le plus de chance de contenir les descripteurs (ART) les plus similaires au modèle. Ensuite on parcoure séquentiellement les vecteurs (ART, CSS, STR) des régions résultats pour sélectionner le vecteur le plus proche du modèle, en se basant sur la somme des distances ART, CSS, et STR. Ainsi, pour chaque image de la base de donnée, on obtient une distance globale entre le modèle

et l'image requête (la même démarche peut être appliquée pour les autres descripteurs du modèle), ces distances sont triées suivant un ordre croissant, et les k premières distances (images) sont sélectionnés.

Bien que les deux solutions présentées précédemment structurent en index les descripteurs de la base de données, elles présentent les inconvénients suivants :

La 1ère solution permet de rechercher les sous objets des images les plus similaires aux sous objets du modèle en terme d'une distance euclidienne tel que, les descripteurs de l'espace de données représentent les descripteurs caractéristique de tous les sous objets des images de la base de données. Avec cette conception, on a négligé la structure hiérarchique décrivant les images utilisée par le système THIS, cette structure hiérarchique permet d'obtenir des résultats plus précis par rapport à notre conception. Quand à la 2ème solution, elle respecte la hiérarchie du système THIS en représentant chaque images par une structure d'index différente, les descripteurs de la même structure d'index représentent les descripteurs des sous objets de la même image. De plus, cette conception permet d'utiliser la même notion de similarité du système THIS (la somme pondérées des distances entre les descripteurs ART, CSS et le descripteur structurel du modèle et les images de la base de données). Par contre elle favorise les descripteurs ART dans la sélection des régions similaires au modèle ce qui peut limiter les résultats de la comparaison. Un deuxième inconvénient est que la dimension de l'espace de données est pratiquement figée, elle correspond aux nombre de composantes utilisées dans le descripteur ART, ce qui peut limiter les expérimentations.

5.3.3 Résultats expérimentaux

Pour illustrer l'intérêt de notre propos, nous avons testé notre méthode de comparaison et celle utilisée par le système THIS sur une base de 400 images, nous avons pris la dimension de l'espace de données d=35. Les deux méthodes ont été exécutées sur une station de travail DELL (2.8 GHZ, et 2Go de RAM). Les courbes de rappel et de précision pour ces deux méthodes sont représentées sur les figures suivantes.

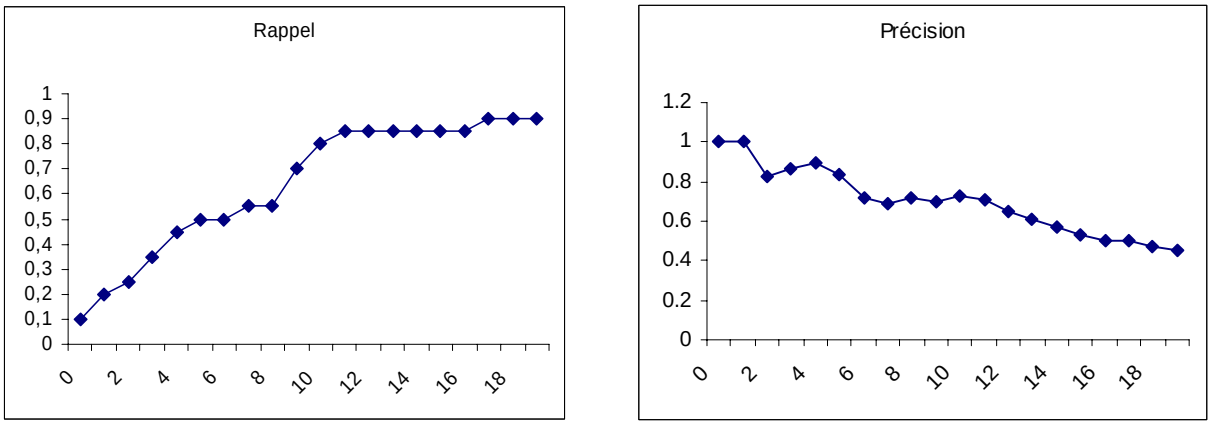


Fig 24. Les courbes de rappel et précision en utilisant notre méthode de comparaison

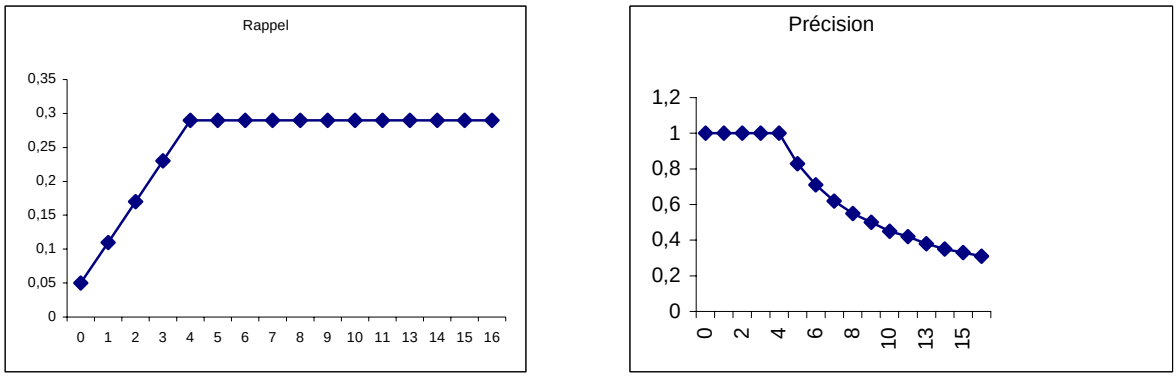


Fig 25. Les courbes de rappel et précision en utilisant la comparaison du système THIS

Notons que pour sélectionner les images similaires, le système THIS met, un temps t_1= 937.5s alors que notre méthode à un temps de réponse t_2 =5.2s.

Notre méthode de comparaison réduit bien le temps de réponse par rapport au système THIS, par contre on obtient des résultats différents. Ceci peut être traduit de la manière suivante : D'abord, les distances utilisées par notre méthodes sont différentes de celles du système THIS, qui lui utilise la théorie de l'évidence pour combiner l'ensemble des descripteurs (ART, CSS, le descripteur structurel) . D'un autre côté, notre conception se base sur la sélection des régions contenant des descripteurs ART similaires, ceci donne un certain poids à ce descripteur ce qui ne peux être adapté à un certain nombre d'application.

D'un autre côté, le système THIS nécessite de nombreux paramétrages difficile à implémenter dans nos solutions, ces paramétrages sont relatifs à la nature des images utilisées dans la comparaison, ce qui limite d'une part les applications de ce système et d'autre part, rend ce système instable dans le cas d'une grande base d'images (>100) non adaptée. C'est un système adapté aux images contenant des objets simples (avec un fond uniforme de type logo par exemple). La complexité des images de la base est limitée, il est particulièrement adapté au cas où l'utilisateur connaît exactement ce qu'il cherche , mais ne dispose pas d'image pour réaliser un modèle, et propose donc un croquis.

6 Conclusion

Pendant cette première période passée en France, nous nous somme intéressés principalement à la comparaison d'images dans le système THIS. Nous avons appliqué une méthode pour la recherche des k plus proches voisins sur le système existant et nous l'avons ensuite évalué sur une base de 400 images.

Pour atteindre cet objectif, nous avons tout d'abord étudié d'une manière approfondie les méthodes existantes et leurs limites en prêtant une attention particulière au problème de passage à l'échelle. Nous avons ensuite étudié le moteur de recherche THIS, nous avons focaliser notre étude sur la méthode de comparaison utilisée dans ce système.

Sur la base de cette étude, nous avons décidé de nous placer dans le contexte de la recherche par approximation. La méthode de recherche d'images proposée pour ce système repose sur le groupement des données dans des régions puis l'approximation de chaque région par deux cellules correspondant aux deux cellules bas gauche, et haut droit. Cette approximation sera ensuite exploitée pour sélectionner les régions candidates éliminant ainsi de la recherche, toutes les régions et par la suite tous les descripteurs n'ayant aucune chance d'être les voisins les plus proches du modèle. Notons que la sélection des régions tient compte uniquement du descripteur ART, vu que celui-ci a des composantes normalisées et une dimension fixe. Par contre les régions contiennent les descripteurs ART, CSS et le descripteur structurel.

Les résultats expérimentaux montrent que notre méthode est performante en terme temps de réponse comparé à la méthode de comparaison dans le moteur existant, mais par contre, elle nécessite des améliorations. En effet, notre méthode tient compte uniquement du descripteur ART, ceci réduit les chances de trouver des images similaires en terme d'une distance globale entre descripteurs.

Parmi la perspective nous proposons d'améliorer la structure d'index proposée en tenant compte des descripteurs de natures différentes. Il s'agit en faite, de concevoir un index multidimensionnel qui regroupe dans une forme particulière (rectangle, sphère…) les descripteurs hétérogènes similaires aux descripteurs du modèle en terme d'une mesure de similarité donnée. Pour traiter de tels cas, il faut soit indexer dans l'espace vectoriel les

descripteurs locaux et calculer ensuite un score d'appariement (comme dans la solution 2), soit envisager une projection de ces descripteurs dans un espace vectoriel métrique particulier, l'idée ici est d'étudier une méthode qui permet de transposer une configuration de descripteurs locaux hétérogènes dans un espace vectoriel unique de dimension fixe, de sorte que les configurations similaires possèdent des images similaires.

Bibliographie

[1] : R. Weber, H.Schok, S. Blott.***A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Space***.Proceedings of the 24th VLDB Conference New York, USA 1998.

[2] : Beyer, R. &McCreight, E. M. (1972). Organization and maintenance of large ordered indices. Acta Informatica, 1(3): 173-189.

[3] : Guang- Ho Cha, Chen-Wan Chung. The GC-Tree : ***A High-Dimensional Index Structure for Similarity Search in Image Databases.*** IEEE Transactions On Multimedia Vol 4, N°1 March 2002.

[4] : T. Sellis, N. Roussopoulos, and C. Faloustos. ***The KS-tree: A dynamic index for multi-dimensional objects***. In Proc. Of the Int. Conference on Very Large Databases, pages 507-518, Brighton, England, 1987.

[5] : N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. ***The R*-tree: An efficient and robust access method for points and rectangles***. In Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data, pages 322-331, Atlantic City, NJ, 23-25 May 1990

[6] : White, D. A. & Jain, R. (1996). Similarity indexing with the ss-tree. In Proceedings of the 12th International conference on Data Engineering, New Orleans, Louisiana, USA, pages 516-523.

[7] : Katayama, N. & Satoh, S. (1997). The SR-Tree: An index structure for high dimensional nearest neighbour queries. In Proceedings of the ACM SIGMOD International Conference on Management of Data, Tucson, Arizona, USA, pages 369-380.

[8] : J. Robinson. The k-d-b-tree: ***A search structure for large multidimensional dynamic indexes***. In Proc. of the ACM SIGMOD Int. Conf. on Management of Data, pages l0-18, 1981

[9] : Henrich, A., Six, H.-W., & Widmayer, P. (1989). The lsd tree: Spatial access to multidimensional point and nonpoint objects. In Apers, P. M. G. & Wiederhold, G., editors, Proceedings of the 15th International Conference on Very Large Data Bases, Amsterdam, The Netherlands, pages 45-53. Morgan Kaufmann

[10] : Henrich, A. (1998). The lsd^h-tree: An access structure for feature vectors. In proceedings of the 14th International Conference on Data Engineering, Florida, USA, pages 362-369

[11] : Hakan, A (2005)High dimensional nearest neighbour searching.

[12] : Berchtold, S., BOHM, C Jagadish, H., Kriegel, H.-H., and Sander, J. 2000a. Independent quantization: An index compression technique for high dimensional data spaces. In Proc. 16th Int. conf. on data engineering.

[13] : Yasushi Sakurai, Masatoshi Yoshikawa, Shunsuke Uemura, and Haruhiko Kojima: "The A-tree: An Index Structure for High-Dimensional Spaces Using Relative Approximation", in Proc. of the 26th International Conference on Very Large Data Bases (VLDB), pp. 516–526, Cairo, September 2000.

[14] : T. Chen, M. Nakazato, S. Huang. ***Speeding up the similarity search in multimedia database***.C2002 IEEE.

[15] : I. DAOUDI, An Indexing Method for Similarity Search in Hgh Dimensional Image Databases, Second International Symposium on Communication, Control and signal Processing, 2006

[16] : KIM, W.-Y. et Kim, Y-S "A New region Based Shape Descriptor" dans Mpeg Meeting, TR 15-01. Pisa. 1999. pages 25, 26, 39, 100.

[17] : Mokhtarian, F. et Mackworth, A. "A theory of multiscal, curvature-based shape representation for planar curve" IEEE Transaction on Pattern Analysis and Machine Intelligence, 14: 789-805. 1992 pages ix, 25, 27, 28, 100.

[18] : Nicolas Zlatoff, Indexation d'images 2D Vers une reconnaissance d'objets multicritères. Thèse préparé au sein du laboratoire LIRIS sous la direction de Atilla Bascurt et Bruno Tellez, soutenu le 12 juillet 2006.