UNIVERSITÉ MOHAMMED V – AGDAL
FACULTÉ DES SCIENCES
Rabat

N° d'ordre 2460

THÈSE DE DOCTORAT

Présentée par

Nom et Prénom : **DAOUDI Imane**

Discipline : Sciences de l'ingénieur
Spécialité : Informatique et Télécommunications

Titre : « Recherche par similarité dans les bases de données multimédia : application à la recherche par le contenu d'images »

Soutenue le **17 Juillet 2008**

Devant le jury

Président :
D. ABOUTAJDINE PES à la Faculté des Sciences de Rabat

Examinateurs :

A. Baskurt	PES à l'Institut National des Sciences Appliquées de Lyon-LIRIS
M. DAOUDI	PES à Télécom Lille 1-ENIC
H. Ibn El Haj	PH à l'Institut National des postes et télécommunication de Rabat
K. Idrissi	Maitre de conférence à l'Institut National des Sciences Appliquées de Lyon-LIRIS
S. Ouatik	PH à la Faculté des Science Dhar El-Mahraz de Fès

Avant Propos

Le travail présenté dans ce mémoire a été effectué dans le cadre d'une co-tutelle entre le laboratoire marocain LRIT (laboratoire de Recherche en Informatique et Télécommunications) de la Faculté des Sciences de Rabat sous la direction du Professeur Driss Aboutajdine et le laboratoire français LIRIS (Laboratoire d'Informatique en Image et Systèmes d'information) sous la direction du Professeur Atilla Baskurt.

Je remercie M. Driss Aboutajdine Professeur d'enseignement supérieur à la faculté des Sciences de Rabat pour son suivi et ses encouragements tout au long de ce travail de thèse de doctorat. Mes remerciements vont aussi à Mr Atilla Baskurt Professeur d'enseignement supérieur à l'INSA de Lyon pour son encadrement et son suivi rigoureux et patient durant ces années de thèse.

Ensuite, je tiens à remercier Mr Mohamed DAOUDI Professeur des universités à Télécom Lille 1 et Mr Ibn Lhaj El Hassan Professeur Habilité pour avoir consacré du temps à lecture de cette thèse ainsi pour avoir soumis leur précieux jugement sur la qualité et le contenu de ce travail.

Je voudrais exprimer ma profonde reconnaissance envers mon co-directeur Khalid Idrissi maitre de conférence à l'INSA de Lyon pour toute l'attention et le soutien qu'il m'a portés pendant ces années de thèse et pour sa très grande disponibilité durant toute cette période. Mes remerciements vont aussi à mon co-encadrant Mr Said Ouatik professeur Habilité à la faculté des Sciences de Fès pour son suivi.

Merci à mes parents et mes frères et sœurs pour leur soutien durant toutes ces années d'études: je ne saurais être qu'infiniment reconnaissante quant aux sacrifices qu'ils ont consentis. Merci à tous les collègues du laboratoire LRIT et le laboratoire LIRIS pour leur amitié et bonne humeur qui ont égayé ma vie au laboratoire.

Enfin merci à ceux et celles que je n'ai pas pu citer, mes sincères amitiés et remerciements.
.

Table des matières

Chapitre 5 : Expérimentations

1 Evaluation des performances de RA+-Blocks 106

2 Evaluation des performances du KRA+-Blocks 114

3 Synthèse 129

Conclusion Générale

Annexe : Algorithmes

Références de l'auteur

Références Bibliographiques

Tables des figures

Liste des tableaux

Introduction Générale

L'une des conséquences directes de la baisse des coûts des équipements informatiques, du développement des télécommunications et de la disponibilité des techniques de numérisation de haute qualité, est la création et l'échange de volumes de plus en plus importants de données multimédias numérisées. Ces données sont par essence *hétérogènes* et leur contenu prépondérant est visuel. Les développements récents dans les domaines du traitement du signal et des bases de données offrent tous les éléments nécessaires pour l'extraction, l'indexation et la recherche du contenu visuel des données multimédias, notamment des images. Dans cette thèse, nous nous intéressons aux techniques l'indexation multidimensionnelles et la recherche des images fixes par le contenu. Ces techniques sont complémentaires et fondamentales pour une recherche rapide et efficace dans un système de recherche d'images par le contenu.

Les techniques d'indexation d'image ont pour but d'organiser un ensemble de descripteurs (un descripteur étant un vecteur de réels décrivant le contenu visuel d'une image et pouvant être de très grande dimension) afin que les procédures de recherche soient performantes en temps de réponse. Cette organisation se traduit généralement par une structuration des descripteurs en petits ensembles et par l'application de stratégies de recherche capable de filtrer toutes les images non pertinentes qui seront évitées (non parcourues) pendant la recherche garantissant ainsi un temps de recherche acceptable par l'utilisateur.

Les techniques de recherche par le contenu quant à elles, consistent à développer et à appliquer des outils qui permettent de sélectionner les images les plus pertinentes par leurs contenus. Lors d'une interrogation, un (ou plusieurs) descripteur, généralement hétérogène, est tout d'abord extrait à partir de l'image requête. Ce descripteur requête est ensuite utilisé pour retrouver les descripteurs stockés dans la base qui lui sont les plus proches en terme de similarité. Les descripteurs trouvés permettent d'obtenir les images auxquelles ils sont associés et qui, de fait, sont censés être similaires à l'image requête.

La mise en œuvre de ces outils d'indexation et de recherche dans un contexte de très grande collection d'images fait appel à des techniques développées dans deux domaines différents : l'analyse d'images et les bases de données. Cette mise en œuvre s'effectue dans le cadre d'un Système de Recherche d'Images par le Contenu (SRIC ou CBIR pour Content Based Image Retrieval).

Objectif de la thèse

Les techniques d'indexation et de recherche d'images basée sur le contenu visent à extraire automatiquement des caractéristiques visuelles des images et à les organiser dans des index multidimensionnels pour ensuite faciliter la recherche dans les grandes bases d'images. Ces techniques ont une complexité particulière liée à la nature des données manipulées. La littérature fait état de diverses approches d'indexation et de recherche de données de caractère multidimensionnelles. Parmi ces approches, certaines souffrent de la malédiction de la

dimension [Web 98][Ams 01]. D'autres, par contre, sont spécifiques à une représentation particulière des données (distribution uniforme des données, espaces métriques …). Il parait donc nécessaire d'élaborer des techniques d'indexation multidimensionnelles qui soit adaptées aux applications réelles pour aider les utilisateurs à faire une recherche rapide et efficace.

L'indexation et la recherche basées sur le contenu comporte trois principales opérations relativement complexes 1. La description automatique consistant à extraire des signatures compactes du contenu visuel de l'image. 2. La structuration de l'espace de description (indexation), consistant à mettre en place une structure d'index multidimensionnelle permettant une recherche efficace pour des milliers, voire des millions d'images. 3. La recherche par similarité dans laquelle une distance est associée à chaque type de descripteur, puis une recherche des k plus proches voisins est effectuée.

Notre thèse consiste à traiter les différentes étapes citées ci-dessus en se focalisant essentiellement sur l'indexation multidimensionnelle et la recherche par le contenu dans les grandes bases d'images fixes. En fait, il s'agit de développer une méthode rapide et efficace d'indexation et de recherche des $k-ppv$ qui soit adaptée aux applications d'indexation par le contenu et aux propriétés des descripteurs d'images.

Nous nous intéressons dans un premier temps à l'indexation multidimensionnelle. En effet, la problématique se complexifie lorsque la taille de la base devient conséquente et que les descripteurs deviennent de grande dimension. La recherche est généralement effectuée d'une manière exhaustive sur la totalité de la base ce qui se traduit par un temps de réponse inacceptable par l'utilisateur. Dans la littérature, plusieurs techniques conventionnelles d'indexation multidimensionnelles ont été proposées pour l'optimisation du temps de réponse. Ces techniques permettent de réduire la recherche séquentielle à un sous ensemble de paquets de vecteurs en regroupant ces derniers dans des formes géométriques particulières (rectangle, sphère, etc.) et en utilisant des stratégies de filtrage. Ceci permet par conséquent de réduire le nombre d'E/S ainsi que le nombre de calculs de distance. Malheureusement, la performance des techniques conventionnelles d'indexation multidimensionnelle se dégrade dramatiquement lorsque la dimension des données augmente [Web 98], phénomène connu sous le nom de *la malédiction de la dimension qui* rend la recherche séquentielle exhaustive bien meilleure qu'une recherche sur les structures d'index conventionnelles. Pour cela, des techniques d'indexation multidimensionnelles basées sur l'approche approximation ont été proposées [SYUK 00][Ter 02], elles reposent sur la compression des données où un codage particulier des données est appliqué permettant d'améliorer la recherche séquentielle par des stratégies de filtrage. Les méthodes d'indexation basées sur l'approche *approximation* sont considérées comme efficaces pour gérer les vecteurs de grande dimension [Web 98], mais leur intégration dans un système de recherche et d'indexation basé sur le contenu (très grand volume de données, très grande dimension, aucune hypothèse sur la distribution des données, etc.) pose de sérieux problèmes. Notre objectif est d'améliorer l'efficacité de ces techniques d'indexation et d'apporter des réponses au problème du passage à l'échelle et de la malédiction de la dimension pour pouvoir ensuite intégrer ces techniques dans un système de recherche par le contenu.

Notre second objectif consiste à appliquer une méthode d'indexation multidimensionnelle basée sur l'approche approximation, à la recherche d'images basée sur le contenu. Rappelons qu'il s'agit de mettre en place des techniques permettant de sélectionner les images les plus pertinentes par leur contenu relativement à une requête donnée selon différents types de descripteurs (couleur, texture, forme). L'intégration de ces techniques dans un système SRIC est confrontée à de nombreux problèmes. Le premier se pose lors de l'étape d'indexation. En effet,

cette étape consiste à gérer les descripteurs caractéristiques des images auxquelles sont associés plusieurs types de données décrivant à la fois la couleur, la texture, la forme, etc. des images. Ces descripteurs possèdent généralement un très grand nombre de composantes (>100), donc une grande dimension difficile à gérer par les méthodes d'indexation existantes en raison du problème de la malédiction de la dimension. Le deuxième problème se présente lors de la structuration de l'espace de description. Il s'agit à ce stade de structurer et d'organiser en index des vecteurs multidimensionnels composés des différents types d'attributs qu'on désignera par *descripteurs hétérogène*. Cette structure d'index devrait regrouper les descripteurs dans des formes géométriques particulières de sorte que les descripteurs appartenant à la même forme soient similaires en termes d'une distance donnée. D'où le troisième problème qui consiste à définir une distance permettant d'une part de mieux approximer la proximité entre les vecteurs d'attributs hétérogène dans l'espace multidimensionnel et d'autre part d'estimer le plus fidèlement possible la similarité visuelle entre les images. Notre objectif est donc d'élaborer une technique d'indexation multidimensionnelle qui réponde efficacement à ces problématiques et permettant ainsi une recherche efficace et rapide dans un SRIC.

Synthèse des contributions

Notre travail a d'abord porté sur l'amélioration des méthodes d'indexation basées sur *l'approche approximation*, pour répondre aux problèmes de la malédiction de la dimension et réduire le temps de la recherche dans les espaces de grande dimension et au passage à l'échelle. D'abord, nous avons passé en revue les principales méthodes basées sur l'approche approximation et nous avons ensuite comparé leurs principales caractéristiques. Sur la base de cette étude, nous avons choisi d'améliorer les performances de la méthode RA-Blocks [Ter 02] en raison des avantages quelle présente. La méthode proposée (RA^+-Block) repose sur un nouvel algorithme de partitionnement qui permet d'améliorer notablement les performances de la structure d'index du RA-Blocks en terme de capacité de stockage et de temps de recherche en générant des régions compactes et disjointes. Les résultats de ces travaux ont été publiés dans [1][8]

La deuxième contribution est la proposition d'une mesure de similarité adaptée aux données réelles lors de l'indexation et de la recherche par le contenu. Nous avons opté pour une représentation de la similarité par fonction noyau. Ainsi, toutes les mesures de similarité et calculs de distance auxquels nous nous sommes intéressés sont entièrement basés sur ce formalisme. Nous avons étudié les différents paramètres de la fonction noyau et nous avons proposé une stratégie de sélection des paramètres qui permettent une meilleure estimation de la similarité entre descripteurs hétérogènes ainsi qu'une représentation discriminante des données dans l'espace de caractéristiques. Les résultats de ces travaux ont été publiés dans [4].

La troisième et principale contribution est la conception d'une méthode efficace d'indexation et de recherche par le contenu particulièrement adaptée aux données de nature hétérogènes (KRA^+-Blocks). Cette méthode permet d'accélérer considérablement le temps de la recherche et d'améliorer significativement la qualité des résultats retournés, particulièrement pour les grandes bases de descripteurs d'attributs hétérogènes. La méthode proposée combine une méthode non linéaire de la réduction de la dimension et une méthode d'indexation multidimensionnelle fondée sur l'approche approximation pour faire face au problème de la malédiction de la dimension et à celui de l'indexation des données hétérogènes. La réduction non linéaire de la dimension permet d'utiliser et d'exploiter les propriétés des fonctions noyau pour définir une mesure de similarité adaptée à la nature des données. Pour améliorer la qualité

de la recherche, nous avons également implémenté un schéma de bouclage de pertinence avec une approche statistique. Nous avons modélisé le problème de la recherche par une classification binaire, dans laquelle nous avons créé un modèle pour discriminer la classe des images pertinentes de celle des images non pertinentes, ceci à travers le calcul des probabilités des classes de vecteurs. Les résultats de ces travaux ont été publiés dans [5][6].
La quatrième contribution de cette thèse est l'intégration de la méthode d'indexation multidimensionnelle KRA$^+$-Blocks au moteur de recherche par le contenu des images fixes IMALBUM, développé au sein du LIRIS. Nous avons mené des expérimentations pour évaluer nos deux méthodes (RA$^+$-Blocks et KRA$^+$-Blocks) à très grandes échelle (base de 2.200.000 éléments) et avec des descripteurs visuelles de grande dimension (d=252), ce qui est rarement le cas dans la littérature. Cela nous a permis de montrer que l'utilisation de l'approche *approximation de régions* et *l'approche noyau* permettent d'être très robuste à l'augmentation de la taille de la base de données et de la dimension des descripteurs utilisés aussi bien pour la qualité que pour le temps de recherche. Cela a également montré que la combinaison de plusieurs types de descripteurs en une seule structure d'index est possible grâce à l'approche de similarité que nous avons proposée.

Description des chapitres

Ce document décrit l'ensemble des travaux menés dans le cadre de cette thèse sur la recherche par similarité dans les grandes bases de données multimédias: application à la recherche par le contenu dans les bases d'images. Il comporte cinq chapitres décrits comme suit :
Le premier chapitre propose un tour d'horizon des principales approches de la description de l'apparence visuelle. Nous commençons par la présentation des principales méthodes d'extraction automatiques des caractéristiques visuelles des images (couleur, texture, et forme) en précisant à chaque fois, les techniques mises en œuvre dans cette thèse pour la description. Nous présentons ensuite les principales mesures de similarité qui existent dans la littérature et nous énumérons celles utilisées par les systèmes de recherche d'images basée sur le contenu. Enfin, nous introduisons la notion de similarité par l'approche noyau, nous présentons quelque aspect de cette théorie dans notre contexte et nous proposons une généralisation de la notion de distance par cette approche.
Le deuxième chapitre est composé de trois paragraphes. Dans le premier, nous passons en revue les principales techniques conventionnelles d'indexation multidimensionnelle en détaillant respectivement les approches basées sur le partitionnement de données et sur le partitionnement de l'espace. Le deuxième présente brièvement les problèmes de la malédiction de la dimension qui perturbent le fonctionnement des techniques d'indexation. Nous présentons dans le même paragraphe les principales techniques de la réduction de la dimension qui ont été proposées dans la littérature pour contourner ces problèmes. Dans le dernier paragraphe de ce chapitre, sont présentées les nouvelles méthodes pour la recherche et l'indexation des données multidimensionnelles basées sur l'approche approximation ou filtrage
Pour dépasser les limites de la méthode d'indexation basée sur l'approche approximation RA-Blocks, particulièrement au niveau du découpage de l'espace de données, nous proposons dans le troisième chapitre une autre méthode que nous avons appelée RA+-Blocks. Nous présentons d'abord les méthodes de partitionnement de l'espace de données KD-Tree et KDB-Tree sur lesquelles sont basées respectivement les deux méthodes RA$^+$-Blocks et RA-Blocks. Nous présentons ensuite la méthode RA-Blocks. Enfin notre méthode d'indexation et de recherche basée sur l'approche approximation RA+-Blocks est détaillée.

Le quatrième chapitre est consacré à la présentation de notre méthode KRA^{+}-Blocks. Nous présentons d'abord les principales méthodes d'indexation basée sur l'approche noyau, puis nous détaillons notre nouvelle méthode d'indexation.
Dans le cinquième chapitre, destiné à la présentation et à la discutions des résultats expérimentaux, nous présentons le contexte des évaluations, puis nous effectuons une série d'expérimentations qui permettent de valider nos deux méthodes d'indexation et de recherche sur des bases d'images réelles et synthétiques.
La conclusion générale présente une synthèse des travaux effectués dans cette thèse. Elle décrit aussi les perspectives que nous proposons au prolongement de ce travail de recherche

Chapitre 1

Recherche par le contenu dans les bases d'images fixes

L'objectif de ce chapitre est faire un tour d'horizon des principaux concepts de base relatifs à la recherche d'images basée sur le contenu. Nous présentons d'abord les principales approches pour la description de l'apparence visuelle des images fixes permettant une recherche efficace par le contenu. Ensuite, nous introduisons les différentes approches de mesure de similarité proposées dans la littérature, nous intéressant particulièrement à la notion de similarité par l'approche noyau.

1 Introduction

Comme l'indique clairement leur nom, les "systèmes de recherche d'images par le contenu" (SRIC ou CBIR avec le vocable Anglais) ont pour fonction principale de permettre la recherche d'images en se basant non pas sur des mots clefs, mais sur le contenu propre des images.

Les applications de tels systèmes sont très nombreuses et assez variées. Elles incluent des applications judiciaires : les services de police possèdent de grandes collections d'indices visuels (visages, empreintes) exploitables par des systèmes de recherche d'images. Les applications militaires, bien que peu connues du grand public, sont sans doute les plus développées [Eak 99] : reconnaissance d'engins ennemis via images radars, systèmes de guidage, identification de cibles via images satellites en sont quelques exemples. Bien d'autres applications existent telles que le diagnostic médical, les systèmes d'information géographiques, la gestion d'œuvres d'art, les moteurs de recherche d'images sur Internet et la gestion de photos personnelles, etc. Le besoin en recherche d'images par le contenu est réel, et les problématiques sont nombreuses et variées. Dans le domaine militaire par exemple, la recherche d'engins ennemis dans les bases d'images radars ne présentera pas les mêmes difficultés que la recherche de voitures, voire d'une voiture en particulier, dans une base d'images généralistes. Toutefois, certaines phases relatives aussi bien à l'indexation de la base qu'à la recherche dans celle-ci vont être nécessaires dans tous les cas de figure.

Les SRIC ont pour vocation de répondre efficacement aux requêtes de l'utilisateurs, ils s'appuient généralement sur une représentation de bas niveau du contenu de l'image. La recherche se fait ainsi par comparaison des caractéristiques. Malheureusement, la conception

d'un système permettant d'assister des utilisateurs dans leurs tâches de recherche d'images est confrontée à des problèmes très divers. Parmi les difficultés pouvant être rencontrées [Eak 99] :
1. Comprendre les utilisateurs d'images et leurs comportements : de quoi les utilisateurs ont-ils réellement besoin ?
2. Identifier une manière "convenable" pour décrire le contenu de l'image. C'est une tâche rendue difficile par l'aspect sémiotique des images.
3. Comparer les requêtes et les images de la base de manière à refléter fidèlement les jugements de similarité humains. Cette comparaison s'effectue à travers une mesure de similarité, généralement explicité sous forme d'une distance
4. Fournir des interfaces conviviales : c'est la vitrine du système permettant la représentation des résultats et qui peut s'avérer fondamentale en présence d'un mécanisme de bouclage de pertinence.
5. Offrir des temps de réponse acceptables : cette contrainte requière une stratégie d'indexation et de recherche pour naviguer efficacement dans les grandes bases d'images.
Le schéma générique SRIC peut être représenté par le digramme de la figure 1.1. Deux processus principaux doivent exister. Un premier qui calcule les descripteurs des images de la base et un second qui, à partir des descripteurs et des paramètres de la requête, recherche les images positives et les fournit à l'utilisateur.

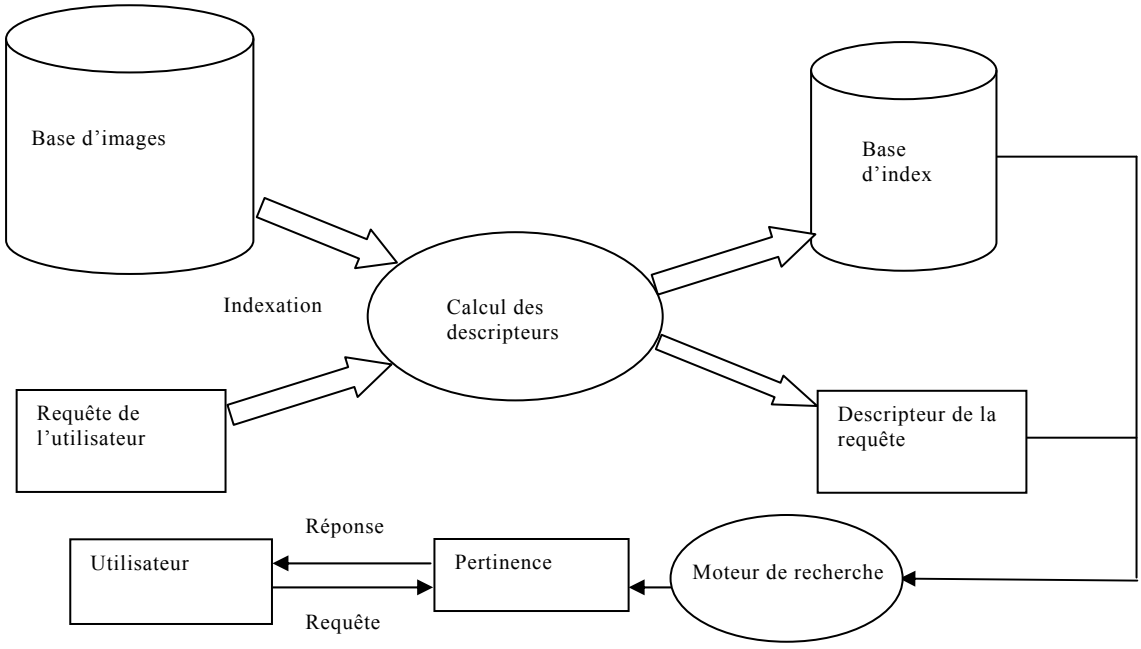


Fig 1.1 Schéma d'un système de recherche d'images par le contenu

Cette phase de recherche est souvent couplée avec une possibilité d'interaction du système avec l'utilisateur, ce qui permet de raffiner le processus de recherche en indiquant au système les résultats pertinents et ceux qui ne le sont pas. Les informations fournies sont alors exploitées pour améliorer la recherche dans une phase dite de bouclage de pertinence (*Relevance feedback*).
Toutes les images de la base sont décrites à l'aide des descripteurs. Ceux-ci doivent contenir des attributs discriminants permettant une bonne description du contenu de l'image et être associés à une mesure efficace de la similarité.

2 Représentation du contenu visuel des images

La performance des systèmes de recherche dépend pour une grande partie du choix des descripteurs employés et des techniques associées à leur extraction. De nombreux descripteurs sont utilisés dans les systèmes de recherche pour décrire les images. Ceux-ci peuvent être différenciés selon deux niveaux :

Les descripteurs de bas niveau : les plus utilisés dans les systèmes actuels sont la couleur, la texture et la forme, leur pouvoir de discrimination étant limité au contenu visuel de l'image.

Les descripteurs de haut niveau : tendent à se rapprocher du contenu sémantique de l'image, et peuvent être soit extraits automatiquement soit fournis par l'utilisateur sous forme de mots-clefs lors de l'indexation. Cependant, l'extraction automatique ne semble réaliste actuellement que sur des bases thématiques.

La normalisation MPEG7 [Man 02] a justement pour objectif de standardiser la description des contenus multimédia, en proposant pour chaque type de contenu (image, son, vidéo), les attributs à décrire ainsi que les descripteurs associés à chaque attribut.

Le but ici n'est pas du tout d'être exhaustif, mais plutôt de présenter brièvement quelques descripteurs proposés pour l'image, puis de revenir sur ceux que nous avons utilisés dans le cadre de ce travail.

2.1 Descripteurs couleur

Le fort pouvoir de discrimination de la couleur en fait un attribut omniprésent dans la grande majorité des systèmes d'indexation et de recherche par le contenu. De nombreux descripteurs sont proposés dans la littérature et nous pouvons considérer qu'ils forment 2 grandes catégories :

- Les descripteurs relatifs à l'espace couleur, où il s'agit de représenter les principales couleurs d'une image, tout en fournissant des informations sur leur importance, leur distribution colorimétrique, etc.
- Les descripteurs incluant des informations spatiales relatives à la distribution dans le plan image de la couleur, à la connexité entre couleurs, etc.

Ces classes de descripteurs sont complémentaires et généralement les systèmes font appel à ces deux aspects.

MPEG7 propose naturellement des descripteurs pour les 2 familles. Les couleurs dominantes, les quantifications des espaces couleur pour la première, et les descripteurs "Color Layout", "Color Structure" et "Scalable Color" pour la seconde. Nous ne détaillerons pas ces descripteurs ici.

Toute fois, l'approche la plus courante et la plus rencontrée dans la littérature est l'histogramme couleur. De très nombreux auteurs ont proposé diverses manières d'utiliser l'histogramme comme descripteur, ainsi que diverses distances associées qui permettent de mesurer la similarité entre deux histogrammes.

Dans [Swa 91], Swain et Ballard ont suggéré de décrire la couleur d'une image à l'aide de son histogramme couleur et ont défini l'intersection d'histogrammes couleur comme mesure de similarité entre une image d'histogramme I et un modèle d'histogramme M par :

$$\sum_{j=1}^{n} \min\left(I_j, M_j\right) \qquad (1.1)$$

qu'il est possible de normaliser par :

$$H_{sect}(I,M) = \frac{\sum_{j=1}^{n} \min(I_j, M_j)}{\sum_{j=1}^{n} M_j} \tag{1.2}$$

Cette mesure semble la plus référencée dans la littérature car ayant servi de base à bon nombre de travaux qui ont cherché à l'améliorer.

Stricker & Orengo [Str 95] montrent que l'utilisation de l'histogramme cumulé $\overline{H}(I) = (\overline{h_{c1}}, \overline{h_{c2}}, ..., \overline{h_{cn}})$ d'une image I défini par :

$$\overline{h}_{c_j} = \sum_{c_l < c_j} h_{c_l} \tag{1.3}$$

où $H(I) = (h_{c1}, h_{c2}, ..., h_{cn})$ représente l'histogramme couleur classique, permet de gagner en robustesse aux changements locaux de forme et aux petits décalages dans l'image. La mesure de similarité peut être effectuée par la distance L_1, L_2 ou L_∞. Les auteurs considèrent que $c_i < c_j$ si les valeurs des 3 composantes de la couleur c_i sont inférieures respectivement aux 3 composantes couleur de la couleur c_j.

Syeda [Sye 00] propose de décrire la couleur par le vecteur :

$$(C, Nb, \{\delta C_i\}, \{H(\delta C_i)\}) \tag{1.4}$$

C étant un paramètre pour indiquer l'espace de travail, Nb le nombre de couleurs quantifiées utilisées, δC_i la valeur de la $i^{ième}$ couleur et $H(\delta C_i)$ le pourcentage de pixels rattachés à cette couleur. Une distance de mesure de similarité est proposée, à partir de la définition d'une matrice A de similarité entre couleurs, où l'élément $a(i,j)$ représente la similarité entre la couleur i et la couleur j. La distance est donnée par :

$$\|V\| = V^T . A V \tag{1.5}$$

V étant un vecteur représentant la différence, élément par élément, des histogrammes des images à comparer. Cette distance tient compte aussi bien de la différence entre les couleurs, que de la quantité de présence d'une couleur dans les deux images.

Hafner [Haf 95] propose une distance pondérée entre histogrammes afin de tenir compte de la similarité entre couleurs :

$$dist_{hist}(H, H') = \sqrt{\sum_i \sum_j a_{ij} (h_i - h_j')(h_i - h_j')} \tag{1.6}$$

où h_i et h_j' désignent respectivement 2 bins des histogrammes H et H', et

$$a_{ij} = 1 - \frac{dist_{ij}}{\max_{ij}(dist_{ij})} \tag{1.7}$$

ou

$$a_{ij} = \exp\left(-cst \left(\frac{dist_{ij}}{\max_{ij}(dist_{ij})} \right) \right) \tag{1.8}$$

représente la similarité entre les couleurs i et j, $dist_{ij}$ étant la distance euclidienne entre ces 2 couleurs et cst une constante positive.

L'auteur fait appel à une autre distance $dist_{avr}$, plus souple et beaucoup plus rapide à calculer, afin de réaliser un premier filtrage sur les images de la base, éliminant ainsi une bonne partie de celles-ci, et n'effectuant le calcul de la distance $dist_{hist}$ que sur les images déjà présélectionnées.

La distance $dist_{avr}$ choisie représente en fait la distance euclidienne entre les vecteurs de couleur moyens de 2 images.
Une autre approche pour caractériser la couleur est l'utilisation des moments statistiques, l'histogramme étant une densité de probabilité. Stricker [Str 96] propose de calculer pour chaque canal la moyenne, la variance et le moment d'ordre 3, afin de caractériser la distribution de la couleur. Il définit ensuite une distance entre deux images par :

$$d(I,H)=\sum w_{i1}\left|\mu_i^I-\mu_i^H\right|+w_{i2}\left|\sigma_i^I-\sigma_i^H\right|+w_{i3}\left|s_i^I-s_i^H\right| \tag{1.9}$$

avec μ_i^I, σ_i^I et s_i^I représentent respectivement la moyenne, la variance et le moment d'ordre 3 du $i^{ième}$ canal de l'image I. Les w_{ij} étant des pondérations associées à chaque terme.
Au niveau des espaces couleur utilisés, de très nombreux travaux ont également été menés, notamment ceux cherchant à offrir des espaces perceptuellement uniforme, et dans lesquels une distance mesurée entre couleurs reflète bien la différence entre celles-ci. L'espace de base est l'espace RGB, cependant non seulement il ne présente pas cette propriété, mais en plus les trois composantes Rouge, Vert, Bleu sont très corrélées.
Parmi les espaces perceptuellement uniformes les plus utilisés nous avons les espaces Lab et Luv. Ceux-ci sont obtenus à partir de l'espace RGB par des transformations non linéaires.
Dans ce travail, nous avons opté pour les couleurs dominantes en limitant le nombre maximal de celles-ci à 5, le descripteur utilisé est celui utilisé dans le système IMALBUM [Idr 03], il contient les couleurs dominantes CD_i, leur pourcentage pr_i, leur variance σ_i, la cohérence spatiale $SCR(i)$, et le maximum de l'histogramme du gradient G [Idr 01]. Notons que CD_i, pr_i, et σ_i sont déterminés lors de la segmentation en considérant l'espace couleur Lab.
Rappelons que la cohérence spatiale est donnée par : $SCR(i)=\frac{H_s(c)}{H(c)}$, où H représente l'histogramme couleur classique, et $H_s(c)=\sum_{i=0}^{Y-1}\sum_{j=0}^{X-1}\delta(I(i,j),c).\alpha(i,j)$, où I est l'image segmentée de taille (X,Y), c est la couleur du pixel (i,j), $\delta(i,j)$ est le symbole de Kronecker et $\alpha(i,j)$ défini par :

$$\alpha(i,j)=\begin{cases}1 & si \quad \forall k,k' \in [-W,W] \quad I(i+k,j+k')=I(i,j)\\ 0 & \text{sinon}\end{cases} \tag{1.10}$$

En définitif, le descripteur couleur associé à chaque image de la base est de la forme :

$$(CD_1, pr_1, \sigma_1, SCR(1), ..., CD_L, pr_L, \sigma_L, SCR(L), \quad G \quad)$$

où L est le nombre de couleurs dominantes utilisées (dans notre cas L=5)
Nous avons également utilisé l'histogramme couleur des couleurs dominantes (Lab) en supposant une distribution Gaussienne des couleurs.

2.2 Descripteurs texture

La texture est une caractéristique fondamentale des images, car elle concerne un élément important de la vision humaine. Il n'existe pas de définition précise et consensuelle de la texture en vision par ordinateur. On parle fréquemment de répétition de motifs similaires, sans pour autant que cette notion soit exhaustive.
De manière générale, la texture se traduit par un arrangement spatial des pixels que l'intensité ou la couleur seules ne suffisent pas à décrire. Elle peut consister en un placement structuré

d'éléments mais peut aussi n'avoir aucun élément répétitif. On peut ainsi grâce à la texture faire la différence entre un coucher du soleil et une orange. Elle traduit donc l'aspect homogène d'une zone et peut être décrite selon ses propriétés spatiales et fréquentielles.

L'approche basée sur la configuration spatiale de l'image consiste à représenter la texture sous forme d'un histogramme en niveau de gris. Des moments statistiques sont alors calculés sur les matrices de cooccurrences pour donner des indications sur le contraste, la directionalité, la périodicité du motif, etc. [Har 73, Had 93]. Les matrices de cooccurrence définissent la probabilité jointe de l'occurrence de deux niveaux de gris quelconques dans une image. Chaque élément $i(x,y)$ de position (x,y) dans cette matrice, rend compte de la probabilité que deux niveaux de gris x et y apparaissent conjointement dans un certain voisinage. Plusieurs statistiques peuvent alors être calculées sur la matrice de cooccurrence. Citons, par exemple : l'énergie, l'entropie, le contraste et l'homogénéité. Néanmoins, de telles approches sont limitées du fait que la pertinence des résultats dépend des zones traitées de l'image, ainsi que des valeurs des paramètres liés à la méthode (orientation et pas).

L'une des méthodes de description de la texture les plus utilisées concerne les propriétés fréquentielles et s'appuie sur la transformée de Fourier, la représentation de Gabor, les ondelettes et la transformée en cosinus discrète [Man 96a, Nas 98]. Elle repose sur l'analyse d'une fonction de densité spectrale dans un domaine fréquentiel [Baj 73], les coefficients d'une transformation bidimensionnelle indiquant la corrélation des motifs dans une image. La granularité de la texture étant proportionnelle à la période spatiale, les textures de grande granularité ont une énergie spectrale concentrée aux fréquences basses alors que les textures plus fines concentrent leur énergie aux fréquences hautes.

Puisque les textures sont la répétition d'un motif, elles seront caractérisées par des pics dans la représentation fréquentielle de l'image (les textures fines dans les fréquences hautes et les textures grossières dans les basses fréquences). L'idée consiste donc à décrire une image $I(x,y)$ comme étant un signal à deux variables spatiales x et y. Ce signal est décomposé comme une somme pondérée de fonctions sinusoïdales. Le poids de chacune de ces fonctions dans la décomposition est calculé par le coefficient de Fourier $F(f_x, f_y)$ où :

$$F(f_x, f_y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} I(x,y)e^{-2i\pi(f_x x + f_y y)}dxdy, \qquad (1.11)$$

f_x et f_y sont les fréquences spatiales de l'image $I(x,y)$.

Les descriptions de type Fourier décomposent l'image sur des fonctions sinusoïdales, qui ont la propriété de s'étendre indéfiniment dans l'espace. Une telle description est donc adaptée à des signaux périodiques ou quasi-périodiques, mais trouve sa limitation pour des signaux plus complexes, présentant de fortes discontinuités. Or, les images présentent justement ce type de caractéristiques. C'est pourquoi d'autres décompositions ont été utilisées, comme celles dite de Gabor. L'idée fondatrice de la méthode de Gabor est de décomposer l'image sur des fonctions analysantes obtenues à partir d'une fonction sinusoïdale orientée sur l'axe des x et modulée par une enveloppe gaussienne dans les directions x et y. Ces fonctions sont ensuite générées par dilatation et rotation.

Les filtres de Gabor sont largement utilisés aujourd'hui, notamment du fait de leur pertinence en regard du système visuel humain. En effet, Marcelja [Mar 80] a montré que les cellules du cortex humain pouvaient être modélisées par des fonctions de Gabor à une dimension. Daugman [Dau 85] a élargi ce modèle à deux dimensions.

Cependant, une limitation provient du choix non trivial des paramètres pour déformer la fonction analysante. Une classe plus générale de méthodes de description espace - fréquence est ainsi couramment utilisée en traitement d'images : les analyses multi-résolution par ondelettes. Etant donnée la nature des bases d'images utilisées dans le cadre de notre travail, nous n'avons pas fait appel au descripteur de texture.

2.3 Descripteurs forme

La forme est l'un des attributs bas niveau également le plus utilisé pour décrire le contenu visuel des images. L'importance de la forme pour la recherche d'images peut être constatée par le simple fait que plusieurs systèmes SRIC incorporent d'une façon ou d'une autre des descripteurs de formes [Fli 95, Gev 00, Pen 96]. Ces derniers sont utilisés pour décrire la structure géométrique générique du contenu visuel. Zhang et al. [Zha 04] ont proposé de classifier les descripteurs de forme en deux familles :

- Descripteurs orientés région : qui décrivent les objets selon la distribution spatiale des pixels qui les constituent.
- Descripteurs orientés contour : qui décrivent les objets selon leur contour externe.

Pour chacune de ces approches (région ou contour), on peut ensuite distinguer deux sous familles :

- Celles qui décrivent globalement les objets.
- Celles qui décrivent les objets en les considérant comme un arrangement de sous parties (structurelles).
- La hiérarchie entière de cette classification et les principales méthodes couramment citées dans la littérature sont présentées dans la figure 1.2.

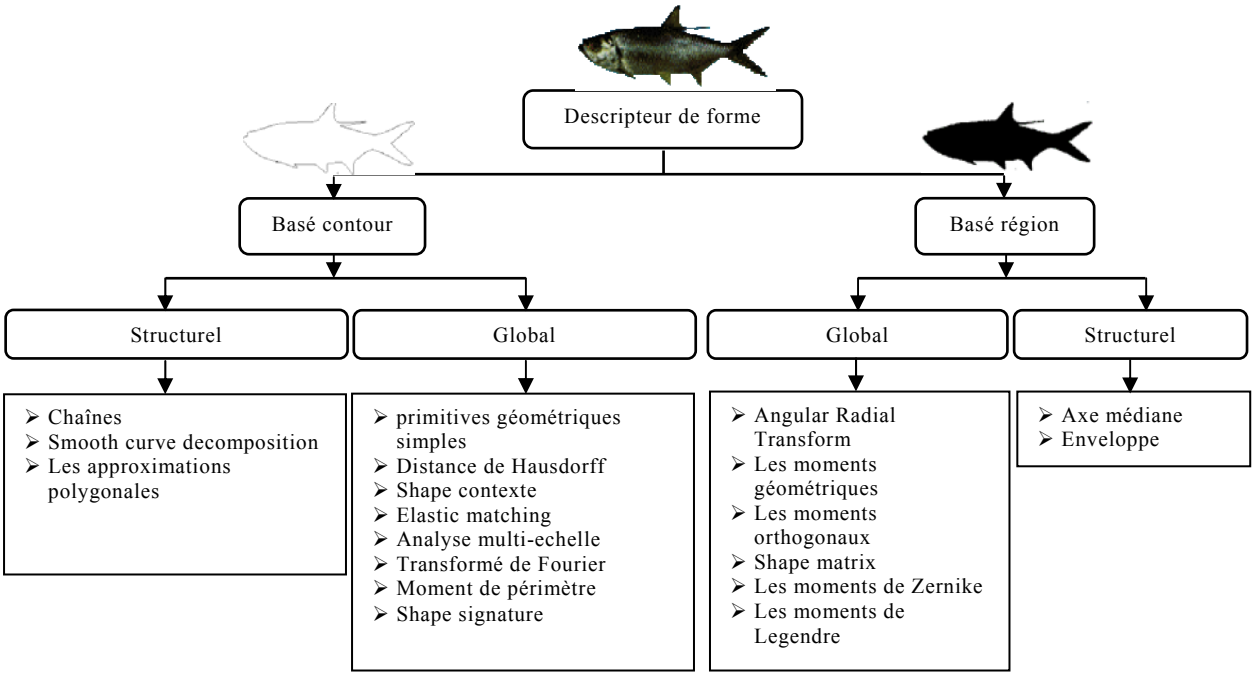


Fig. 1.2. Classification des descripteurs de formes 2D

Quelque soit la méthode utilisée, il demeure important que certaines propriétés d'invariance soient vérifiées, notamment pour la translation, le changement d'échelle et la rotation, du fait que l'être humain corrige instinctivement les effets de ces transformations lors de la recherche d'un objet. Dans certaines applications, la robustesse à l'occultation peut également être souhaitée.

Pour les descripteurs orientés contour, un exemple classique est l'histogramme des orientations des contours [Jai 98], qui prend en compte l'angle du gradient calculé pour des pixels de contours. Cet histogramme donne une description approximative de la forme dans une image, calculée sur les régions de l'image où il y a suffisamment d'activité de la couleur pour générer des contours. D'autres descripteurs sont basés sur les coins et les contours dans les segments de frontières [Pet 00]. En utilisant seulement l'information sur la frontière des objets, ces descripteurs ignorent les informations ponctuellement importantes contenues à l'intérieure des objets. Dans [Bel 02, Scl 95] les auteurs utilisent des configurations spatiales de points d'intérêt situés à l'intérieur des contours, mais la méthode est couteuse en temps de calcul, en raison de la nécessité de localier les points. De plus, l'absence de points d'intérêt peut affecter les performances. La théorie des moments géométriques, qui utilisent des développements en série pour représenter la forme des objets, évite ce type de problème [Ter 98]. [Hu 62] propose plusieurs fonctions non linéaires définies sur les moments géométriques qui sont invariantes à la translation, à la rotation et aux changements d'échelle. Ces descripteurs ont été appliqués avec succès à l'identification d'avions, de navires et de visage [Ter 98]. Dans cette catégorie nous trouverons également les descripteurs de Fourier qui décrivent le contour par ses composantes fréquentielles [Per 77, Zha 01], les moments de Zernike et Zernike modifié [Tea 80], qui ont été adaptés par de nombreux auteurs [Ter 98, Kim 00], les invariants différentiels [Gev 04], des descripteurs basés sur la transformée de Hough [Fer 05], et les descripteurs élémentaires, comme le rapport longueur/largeur du rectangle englobant, l'isotropie, et enfin le descripteur CSS (Curvature Scale Space) [Mok 92] recommandé par MPEG7. Daoudi et al [Dao 00] ont proposé d'utiliser la distance géodésique définie dans [Ebe 94]. Ainsi au lieu de calculer cette distance parmi l'ensemble de points de CSS (u,σ), les auteurs ont proposé l'utilisation des maximums des CSS calculés à partir d'un certain seuil σ pour indexer le contour de la forme

Les descripteurs orientés région ont également été beaucoup utilisés. Nous focalisons sur le descripteur ART (Angular Radial Transform) introduit par [Kim 99], et que nous avons exploité dans le cadre de ce travail. Le principe de ce descripteur est de projeter l'objet à étudier sur une série de fonctions de base (Fig1.3).

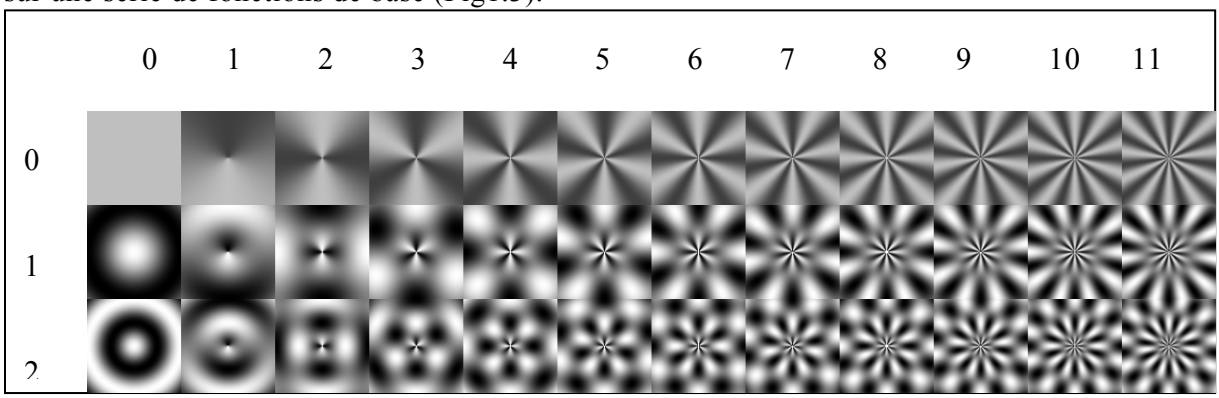


Fig 1.3 Parties réelles des fonctions de base d'ART

Le descripteur est constitué d'un vecteur de 33 coefficients obtenus par l'équation :

$$F_{nm} = \langle I_{nm}(\rho,\theta), f(\rho,\theta) \rangle = \int_0^{2\pi} \int_0^1 I^*_{nm}(\rho,\theta), f(\rho,\theta) \rho d\rho d\theta \qquad (1.12)$$

dans laquelle I représente l'image à décrire et f , l'une des 33 fonctions de bases utilisées.

Ce descripteur de formes possède de nombreuses propriétés : petite taille, robuste aux bruits, invariance aux changements d'échelle et à la rotation, possibilité de décrire des objets complexes.

2.4 Combinaison des descripteurs

Les attributs de bas niveau (couleur, texture, forme) décrivent les images par leur contenu visuel. Ces attributs peuvent être combinés pour caractériser de façon encore plus efficace le contenu. Cependant, dans de très nombreuses situations, un attribut seul ne suffit pas pour décrire le contenu de l'image. Il est donc intéressant de combiner ces différents attributs pour une recherche plus efficace et plus discriminante. Les problèmes qui se posent lors de la combinaison de ces différents attributs pour la recherche et l'indexation sont au moins de trois ordres :

La mesure de similarité : Celle-ci est une étape primordiale dans tout système de recherche. Dans le cas où les images sont décrites par différents attributs, une solution classique pour mesurer la similarité est de calculer séparément les mesures de similarité pour chaque attribut puis d'en déduire une mesure composée de la similarité globale entre les images. Cela suppose bien évidement que les différents attributs sont indexés séparément (avec des structures d'index séparées). Or, en base de données, Il existe peu de méthodes qui utilisent plusieurs index pour structurer les données. Une autre difficulté relative à la similarité est de déterminer comment combiner plusieurs mesures définies souvent sur des domaines différents, avec des dynamiques différentes, des degrés d'importance différents, notamment pour l'utilisateur, mais également de natures différentes.

L'espace de description : Le choix de l'espace de description consiste à chercher les attributs (descripteurs) visuels significatifs de la base d'images, l'ensemble de ces attributs étant représenté par un nuage de points dans un espace de dimension élevée. Un problème qui se pose, si les vecteurs contiennent plusieurs attributs, est celui de la dimension de l'espace de description. Ce problème est connu dans la communauté des bases de données par "la malédiction de la dimension" ou "dimensionality curse avec le vocable anglais"

La structuration : la phase de construction d'une structure d'index est une étape utile dans le cas où les données sont volumineuses et appartiennent à un espace de description de grande dimension. Cela revient à structurer les nuages de points relatifs aux descripteurs des images et à les stocker efficacement en machine. Cette tâche de structuration peut s'avérer difficile dans le cas où les données à structurer sont de nature hétérogène. La difficulté réside notamment dans le choix de la distance à utiliser pour la structuration (mise en place d'un index) et dans la normalisation des différents types de données.

Dans le paragraphe suivant nous abordons les différents aspects de la mesure de similarité. Les problèmes liés aux bases de données à savoir le problème de la malédiction de la dimension et celui de la structuration seront traités dans les chapitres suivants.

3 Mesure de similarité

Dans la majorité des SRIC, la recherche d'images similaires est basée sur la similarité des caractéristiques visuelles telles que la couleur, la texture ou la forme. La fonction distance

utilisée pour évaluer la similarité dépend des critères de la recherche mais également de la représentation des caractéristiques. L'idée principale est généralement d'associer à chaque image un vecteur multidimensionnel représentant les caractéristiques de l'image, et de mesurer la similarité des images en utilisant une fonction de distance entre les vecteurs. Plusieurs mesures de similarité existent dans la littérature, et sont basées soit sur la comparaison entre des descripteurs des images, soit sur la comparaison directe entres images. Nous nous intéressons dans cette étude au premier type de mesure de similarité. Nous présentons dans ce paragraphe d'abord le principe de la similarité attentive et pré-attentive avant de faire un rapide tour d'horizon sur les mesures de similarité basées sur le modèle métrique. Nous ferons le point sur les principales mesures de similarité utilisées dans le domaine des SRIC, et enfin nous introduirons le principe de la mesure de similarité par l'approche noyau.

3.1 Similarités attentive et pré-attentive

L'œil humain possède la capacité d'identifier visuellement les images similaires à partir de leur contenu. Il est prouvé [Lev 76] que la perception visuelle humaine opère selon deux modes : perception attentive et pré-attentive [Bim 99]. La perception attentive a un lien direct avec l'interprétation, elle intervient une fois que l'attention de l'utilisateur a été focalisée. Son utilisation est requise dans des domaines spécifiques tels que la détection de visages, l'imagerie médicale ou mécanique. Ces domaines comparent les objets et définissent des critères de similarités qui leur sont propres. La perception pré-attentive quant à elle, est basée sur la similarité entre les signaux perçus par l'œil, sans aucune forme d'interprétation, elle fait intervenir des mécanismes globaux sur l'image, et correspond à des traitements qui ont lieu avant le cortex dans le cerveau humain. Elle est utile dans des applications plus générales pour lesquelles la couleur, la texture, la forme ou la perception des relations spatiales est plus importante. Plusieurs modèles tentent de reproduire automatiquement la perception humaine de la similarité des signaux sensoriels, Parmi les plus importants, on cite le modèle de Tversky [Tve 77] et le modèle métrique [Tor 65].

3.1.1 Modèle de Tversky

Dans la théorie de Tversky [Tve 77], la définition de la similarité est inspirée de la théorie des ensembles. Dans le cas où l'on cherche à comparer une image I d'une base d'image avec une image requête R, Tversky propose que la similarité $Sim(R,I)$ entre I et R ait les propriétés suivantes :

* $Sim(R,I)$ est d'autant plus importante que I et R ont des caractéristiques communes.
* $Sim(R,I)$ est d'autant moins importante que I possède des caractéristiques propres que R ne partage pas.
* $Sim(R,I)$ est d'autant moins importante que R possède des caractéristiques propres que I ne partage pas.

En notant C_I et C_R les caractéristiques extraites de I et R respectivement. Il vient que $C_R - C_I$ représentent les caractéristiques propres de R, $C_I - C_R$ celles de I, et $C_R \cap C_I$ celles commune à I et R, Tversky propose alors :

$$Sim(R,I) = f(C_R \cap C_I) - \alpha f(C_R - C_I) - \beta f(C_I - C_R) \qquad (3.13)$$

autrement dit, la similarité entre deux images est obtenue par une combinaison linaire d'une fonction de deux types de descripteurs, α et β étant des pondérations.

Notons que cette mesure s'applique à des caractéristiques de prédicats logiques et que son application à des données numériques pose de nombreux problèmes. Santini et Jain [San 95] montrent que le modèle de similarité de Tversky peut être utilisé pour modéliser les distances entre les textures, mais cette approche reste inintéressante pour l'indexation.

3.2.1 Similarité par une fonction distance

La perception humaine de la similarité peut être modélisée par une mesure de distance appropriée dans un espace métrique multidimensionnel.
Mathématiquement, une distance normalisée $dist$ est définie comme une fonction à valeur dans [0 ;1] et qui vérifie les trois propriétés suivantes :

* $dist(V_1,V_2)+dist(V_2,V_3)\geq dist(V_1,V_3)$ (3.14)
* $dist(V_1,V_2)=dist(V_2,V_1)$ (3.15)
* $dist(V_1,V_2)=0\Leftrightarrow V_1=V_2$ (3.16)

La condition (3.14) est connue sous le nom d'inégalité triangulaire, (3.15) est appelée symétrie. Les distances sont nombreuses dans la littérature, définies pour des valeurs scalaires, ensemblistes, vectorielles, etc. (différence absolue, cosinus, Harman, Dice, Jacquard, degré d'inclusion, produit scalaire, Manhattan, Hamming, Euclidienne, Hausdorff…). Nous présentons dans la suite les distances les plus couramment utilisées.
Lorsque les données sont assimilées à des vecteurs, ce qui est souvent le cas, la distance de Minkowski est fréquemment employée. Elle est donnée par :

$$dist_m(V_1,V_2)=(\sum_{i=1}^{N}W_i\times\left|v_{1,i}-v_{2,i}\right|^m)^{\frac{1}{m}} \qquad (3.17)$$

m représente l'ordre et W est une matrice de pondération.
En faisant varier la valeur de m, on obtient différentes fonctions de distance. La distance de Minkowski du premier ordre ($m=1$) est une distance de Manhattan et la distance de Minkowski du deuxième ordre ($m=2$) est une distance Euclidienne. Le choix d'une valeur appropriée pour m dépend de l'importance que nous voulons accorder aux plus grandes différences. Ainsi les grandes valeurs de m donnent progressivement plus d'importance aux différences les plus grandes et quand m tend à l'infini la distance de Minkowski tend vers la distance de Tchebychev.

Distance Euclidienne

Lorsque $m=2$, on obtient la distance Euclidienne :

$$dist(V_1,V_2)=\sqrt{\sum_{i=1}^{N}W_i\times(v_{1,i}-v_{2,i})^2} \qquad (1.18)$$

C'est la distance la plus fréquemment utilisée grâce à ses propriétés géométriques intéressantes. Cette distance à tendance à donner plus d'importance aux plus grandes différences sur une variable simple. Le contour de la même distance (Euclidienne) à partir d'un point donnée à une forme sphérique (cercle en deux dimensions). Elle est surtout recommandée pour des espaces isotopiques où les données ont les mêmes propriétés dans toutes les directions (dimensions).
Les clusters définis par une distance Euclidienne sont invariants à la translation et la rotation, par contre ils ne sont pas invariants aux transformations linéaires et plus généralement à d'autres transformations qui dégradent les rapports entre les distances. La figure 1.3 représente

un exemple de l'effet d'une transformation linéaire sur la classification basée sur une distance Euclidienne[1].
D'après la figure 1.4, on remarque que, lorsque l'axe vertical est multiplié par un facteur de 2.0 et l'axe horizontal est rétréci par un facteur de 0.5, le groupement de données est modifié (comme illustré à droite). Réciproquement, si l'axe vertical est rétréci par un facteur de 0.5 et l'axe horizontal multiplié par un facteur de 2.0, de nombreux petits faisceaux apparaissent. Dans les deux cas de figure les groupements des données diffèrent du groupement original

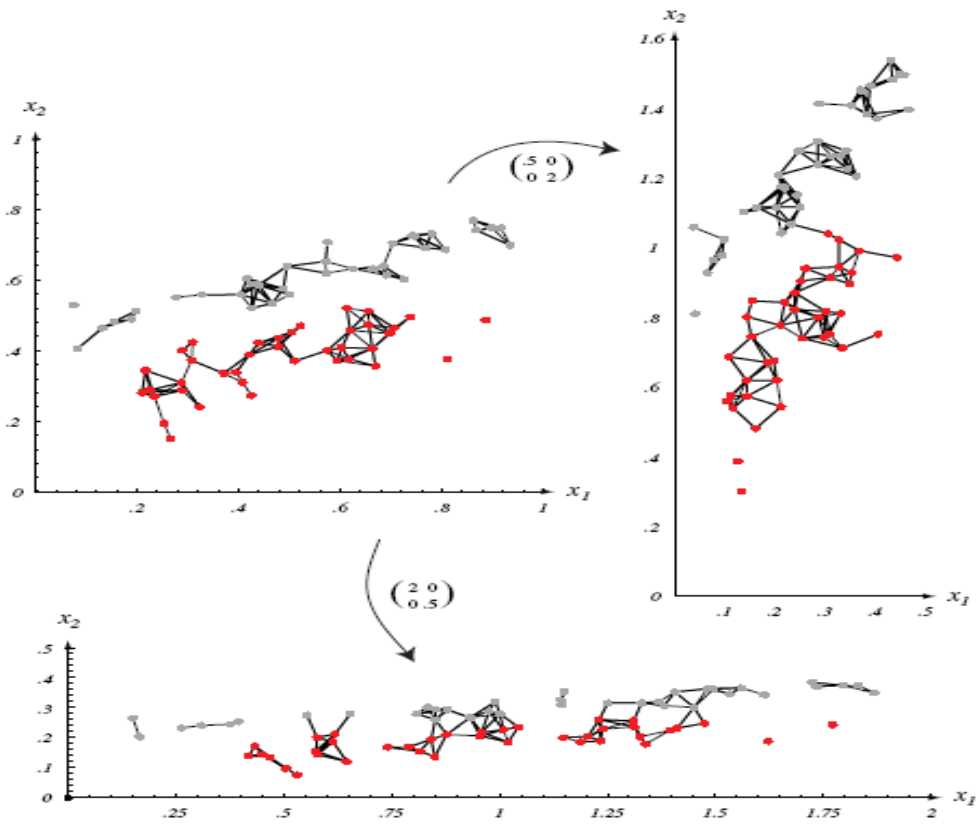


Fig 1.4. Effet d'une transformation linéaire à une classification basée sur une distance Euclidienne

Distance de Manhattan

Pour $m=1$, on obtient la distance de Manhattan :

$$dist(V_1,V_2)=\sum_{i=0}^{N}(\left|v_{1,i}-v_{2,i}\right|) \qquad (1.19)$$

[1] Exemple tiré de Pattern Classification de Richard O. Duda, Peter E. Hart et David G. Stork

Cette distance est aussi connue sous le nom de city-block. Elle est recommandée par le comité MPEG-7 [Jea 01] pour comparer deux formes décrites avec la méthode ART de Kim [Kim 99]. Ainsi, pour deux objets O_1 et O_2 dont les formes sont décrites respectivement par 2 vecteurs V_1 et V_2, constitués d'une série de N coefficients chacun, [Jea 01] propose d'évaluer une mesure de dissemblance entre les objets par la formule :

$$dist(O_1, O_2) = \sum_{i=0}^{N} \left| v_{1,i} - v_{2,i} \right| \qquad (1.20)$$

C'est cette formulation que nous utiliserons dans cette thèse.

Distance de Chebychev

Pour $m = \infty$, on obtient la distance de Chebychev :

$$dist(V_1, V_2) = \max_i (W_i \left| v_{1,i} - v_{2,i} \right|) \qquad (1.21)$$

Cette distance est adaptée aux données de grande dimension, elle est souvent employée dans les applications où la vitesse d'exécution est importante. Cette distance examine la différence absolue entre les différents pairs des vecteurs, elle est considérée comme une approximation de la distance Euclidienne mais avec moins de calcul.

Dans la formule de la distance, les variables $v_{1,i}$ et $v_{2,i}$ représentent respectivement les valeurs du $i^{ème}$ attribut des deux vecteurs/descripteurs V_1 et V_2, et W_i représente le poids associé à cet attribut.

Nous distinguons une autre famille de distances dites distances quadratiques dont l'expression est la suivante : $d(V_1, V_2) = \sqrt{(v_{1,i} - v_{2,j})^t W (v_{1,i} - v_{2,j})}$ (1.22)

En fonction de la matrice M on obtient différentes distances.

Avec $W = I$, telle que I est la matrice unité, on retrouve la distance Euclidienne classique.

Une illustration des contours des différentes distances présentées auparavant est donnée par la figure 1.5

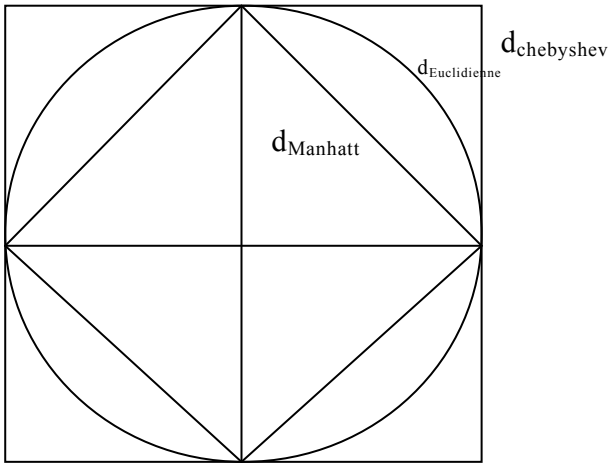


Fig.1.5. les contours de différentes distances

Distance de Mahalanobis

Avec $W = Cov^{-1}$, on obtient la distance de Mahalanobis. La matrice Cov correspond à la matrice de covariance entre l'ensemble des descripteurs d'images. A la différence de la distance Euclidienne où les composantes des vecteurs sont traitées de la même façon, cette distance prend en considération les dépendances (corrélations) entre les données.

Distance du Chi-2

Avec $W = D^{-1}$ telle que D est une matrice diagonale exprimant la fréquence relative d'apparition des mots/attributs dans les vecteurs de données, on obtient la distance du $Chi-2$. Cette distance est utilisée en analyse de données pour que ces données puissent apparaître dans un espace commun. Ce type d'analyse est souvent utilisé dans l'analyse en composantes principales permettant de mettre en évidence les combinaisons des attributs principaux. Cette distance est exprimée par la fonction suivante :

$$Chi-2(V_1,V_2) = \sum_{j=1}^{N} \frac{N}{f.j} \left(\frac{f_{V_{1,j}}}{f_{V_1}} - \frac{f_{V_{2,j}}}{f_{V_2}}\right)^2 \qquad (1.23)$$

où N est le nombre total d'attributs dans la base d'images, $f_{V_{1,j}}$ désigne la fréquence de l'attribut j dans le vecteur V_1, f_{V_1} désigne le nombre total d'attributs dans le vecteur V_1 et, $f.j$ désigne la fréquence de l'attribut j dans l'ensemble de la base des vecteurs.

Dans les mesures que nous avons présentées précédemment, pour un attribut donné, la distance est calculée à partir de la différence $|v_{1,i} - v_{2,j}|$ entre les vecteurs de données. Or, cette manière de procéder risque de fausser le résultat de la mesure de similarité notamment lorsque les attributs décrivant les images ont des domaines de définition différents. Par exemple, dans le cas de la distance de Manhattan, les attributs ayant une grande dispersion de valeurs sont implicitement favorisés, ce qui conduit à une augmentation considérable de la distance finale. Il est donc nécessaire de normaliser la valeur de similarité. Cela peut s'effectuer par l'intermédiaire d'une opération dite de recodage des données en réalisant un centrage pour tous les attributs (on retranche à toutes les valeurs la moyenne des valeurs de l'attribut pour tous les exemples) et une réduction (on divise toutes les valeurs par l'écart type de l'attribut). Ainsi, tous les attributs sont de moyenne nulle et d'écart type égal à 1. Toutefois, ce processus ne peut être effectué que si l'on connaît tous les vecteurs de données, ce qui n'est pas toujours le cas notamment dans les traitements incrémentaux. Une solution consiste plutôt à exprimer de manière explicite le domaine de définition des attributs et à intégrer cette information dans le calcul de la similarité. Par exemple, si l'on introduit une fonction Dom_i qui calcule la différence entre la borne haute et la borne basse du domaine de définition de l'attribut, la distance de Manhattan s'énonce de la manière suivante :

$$dist(V_1,V_2) = \sum_{i=1}^{N} W_i \times d(v_{1,i}, v_{2,i}) \quad \text{avec} \quad d(v_{1,i}, v_{2,i}) = \frac{(v_{1,i} - v_{2,i})}{dom_i} \in [0\,1] \qquad (1.24)$$

Avec cette fonction de distance, il est possible de traiter des vecteurs composés de données hétérogènes à l'aide d'une seule mesure générale que l'on utilise lors du calcul de la fonction $d(v_{1,i}, v_{2,i})$ correspondant au type de la donnée en cours de traitement. Le problème qui se pose avec cette distance réside dans le choix des pondérations.

En effet, la manière dont les attributs sont pondérés est tout à fait cruciale pour que la mesure effectuée soit pertinente et puisse refléter l'importance que l'on souhaite accorder à chaque attribut. Cette pondération n'est pas toujours facile à réaliser, notamment si l'on n'a pas d'expert du domaine capable de la fournir. Dans le cas des SRIC, la pondération est liée au besoin de l'utilisateur, susceptible de changer à chaque requête.

3.2 Similarité dans les moteurs recherche

Les performances des SRIC dépendent largement de la mesure de similarité/dissimilarité utilisée pour la comparaison des descripteurs des images. L'un des attributs les plus utilisés dans les SRIC est la couleur, et le descripteur le plus courant et le plus rencontré dans la

littérature est l'histogramme couleur. De très nombreux auteurs ont proposé diverses manières d'utiliser l'histogramme comme descripteur, ainsi que diverses distances associées qui permettent de mesurer la similarité entre deux histogrammes.
La distance Euclidienne est le plus souvent utilisée pour mesurer la similarité entre les histogrammes, mais malheureusement cette distance n'est pas optimale pour la comparaison de ce type de signature. Parmi les nombreuses techniques de calcul de similarités qui ont été proposées, on trouve l'intersection des histogrammes [Swa 91]. Cette mesure est équivalente à une distance L_1 si les histogrammes sont normalisés. Une deuxième manière de mesurer la similarité entre les histogrammes est de voir l'histogramme comme une distribution, et ainsi utiliser l'entropie comme similarité [Kul 59]. Une version symétrique de mesure par l'entropie est aussi proposée [Puz 97], ainsi qu'une mesure probabiliste au sens de $Chi-2$ [Set 95]. Malheureusement ces techniques comparent indépendamment les valeurs des histogrammes sans prendre en considération la corrélation entres les composantes. Ceci pose des problèmes en termes de robustesse puisqu'elles sont sensibles à certaines transformations, notamment la translation. Des mesures plus robustes sont proposées, comme les distances sur les distributions cumulées [Str 95], l'Earth Mover's Distance [Rub 99], ou les distances quadratiques généralisées. Ces méthodes sont efficaces pour la mesure de similarité mais elles ne traitent pas le problème de la non linéarité des données, de plus elles sont coûteuses au niveau temps de calcul particulièrement dans le cadre de l'apprentissage.
En fait, dans les SRIC, les images ne sont pas représentées par un unique vecteur relatif à un attribut, mais un ensemble de vecteurs se rapportant aux différents espaces de caractéristiques (couleur, texture, forme, etc.). Le problème qui se pose alors, est de savoir comment combiner plusieurs mesures définies souvent sur des domaines différents et éventuellement avec des importances variables différentes. Néanmoins, les résultats expérimentaux montrent, de manière évidente, que la prise en compte simultanée de plusieurs attributs donne toujours de meilleurs résultats par apport à l'utilisation d'un seul et unique attribut [Pic 96b].
Généralement, on va être amené à combiner des mesures établies indépendamment les unes des autres. Les difficultés sont au moins de trois ordres :
* Les domaines de définition des attributs sont différents.
* Les attributs ne sont pas toujours indépendants les uns des autres.
* Leurs importance n'est pas la même.
La littérature présente plusieurs formes de combinaison entre différentes mesures de similarité plus au moins justifiées. Nous citons uniquement les techniques les plus fréquemment utilisées dans les SRIC.
Une des techniques courante consiste à concaténer les différents types de vecteurs puis d'uniformiser les échelles avec une distance de Mahalanobis. Les systèmes CANDID [Kel 95], QBIC [Nib 93] et Netra [Man 96b] offrent la possibilité de combiner facilement les descripteurs

de texture et de forme en calculant séparément les mesures de similarité pour chaque type de descripteurs puis en dérivant une mesure composée de similarité globale. Les descripteurs couleur peuvent également être intégrés dans la combinaison. Ce type de distance demeure confronté aux problèmes du choix des pondérations et de la non linéarité des données.
Quand les composantes sont indépendantes, des mesures plus simples suffisent. VISUALSEEK [Smi 96] utilise une simple somme entre la distance spatiale (elle-même une simple somme) et la distance de couleurs (une distance quadratique adaptée). Cha et Chung [Cha 98] proposent encore une somme des différences absolues entre la moyenne et l'écart-type respectivement sur les quatre quadrants et trois canaux d'images RVB[2]. Une somme des différences quadratiques, pondérée par l'inverse de la variance correspondante, est utilisée pour la texture (granularité, contraste et direction) et la forme dans QBIC [Fal 94].
Il est généralement difficile de faire un choix parmi toutes les techniques de mesure de similarité. Le choix d'une mesure particulière dépend non seulement de l'application, mais également de plusieurs facteurs tels que la distribution des données, le temps de calcul, la complexité, etc. Chaque mesure présente des propriétés différentes, et l'intérêt d'une mesure de similarité par rapport à une autre, notamment dans un SRIC, réside dans sa capacité à s'adapter aussi bien à l'application visée, qu'à la nature des données utilisées pour la description des images.
A la différence des mesures de similarité présentée dans les paragraphes précédents, les fonctions noyaux offrent de nombreux avantages pour l'analyse et l'exploitation des descripteurs d'images. Comme nous allons le voir, les fonctions noyaux définissent, le produit scalaire dans l'espace multidimensionnel à travers une fonction de base ϕ par : $k(V_1,V_2) = <\Phi(V_1),\Phi(V_2)>$. L'intérêt de l'approche noyau vient du fait que seul le noyau k doit être défini alors que la fonction de base ϕ n'est jamais explicitement calculée. Ceci offre un grand choix d'utilisation des fonctions de bases non linéaires, possédant des capacités pour s'adapter à la diversité des caractéristiques des descripteurs. Avec l'approche noyau, on peut également obtenir un cadre mathématique formel à partir duquel une seule mesure de similarité pour différents types de données peut être établie. Ceci, permet ainsi de faciliter le processus d'indexation et de contourner les problèmes liés à la combinaison de plusieurs mesures de similarité exposés dans les paragraphes précédents.

4 Similarité par approche noyau

Dans le paragraphe suivant, nous nous intéressons à la notion de similarité par approche noyau. Nous présentons d'abord le principe du "Kernel Trick", qui permet de transformer une méthode basée sur le produit scalaire en une méthode à noyau. Nous aborderons ensuite les exemples de fonctions noyaux les plus utilisées dans la littérature. Enfin, nous détaillerons la généralisation de la notion de distance par des fonctions noyaux.

---

[2] $dist(V_1,V_2) = \sum_{t=1}^{4}\sum_{i=1}^{3} \left|moy_{t,i}(V_1) - moy_{t,i}(V_2)\right| + \left|ecart_{t,i}(V_1) - ecart_{t,i}(V_2)\right|$

4.1 Astuce du noyau « Kernel Trick »

Avant de présenter l'astuce du noyau, il est nécessaire de revenir sur quelques notions de base liées aux espaces à noyau. Commençons d'abord par définir formellement les espaces de Hilbert à noyau reproduisant (RKHS : Reproducing Kernel Hilbert Space).

Définition 1

Un espace de Hilbert à noyau reproduisant H *(RKHS) est un espace de Hilbert de fonctions pour lequel il existe une fonction* $k : \chi \times \chi \to \Re$ *(appelée noyau auto-reproduisant) vérifiant :*

* H *contient toutes les fonctions* $k(V_1,.)$ *pour* $V_1 \in \chi$

* *la propriété reproduisant est satisfaite :* $\forall f \in \mathrm{H},\ \forall V_1 \in \chi\ < f, k(V_1,.) >_H = f(V_1)$

Définition 2

Une fonction $k : \chi \times \chi \to \Re$ *est un noyau défini positif si et seulement si :*

* *Elle est symétrique :* $\forall V_1, V_2 \in \chi\ k(V_1,V_2) = k(V_2,V_1)$

* *Et définie positive :* $\forall\ N \geq 1, \forall (V_1,...,V_n) \in \chi^N, \forall (a_1,...,a_n) \in \Re^N,\ \sum_{i,j=1}^{N} a_i a_j k(V_i,V_j) \geq 0$

Le théorème de Moore-Aronszajn [Aro 50] indique qu'à tout noyau défini positif correspond un unique RKHS. Notons que ce résultat ne suppose aucune condition sur l'espace d'origine χ. Pour la suite, nous notons H_k le RKHS associé à la fonction noyau k

Pour mieux comprendre la définition formelle du RKHS, nous introduisons le théorème de Mercer [Mer 09] duquel vont découler de nombreuses et remarquables implications pratiques.

Théorème 1 : Théorème du Mercer

Toute fonction $k(V_1,V_2)$ *définie positive sur un domaine compact* $\chi \times \chi$ *peut s'écrire sous la forme d'une série uniformément convergente, i.e.*

$$k(V_1,V_2) = \sum_{i=1}^{N} \lambda_i \psi_i(V_1)\psi_i(V_2), \qquad N \leq \infty \tag{1.25}$$

où les $\{\lambda_i\}_{i=1}^{N}$ *et les* $\{\psi_i\}_{i=1}^{N}$ *sont des valeurs propres et les fonctions propres orthogonales de l'opérateur (linéaire auto-adjoint compact donc diagonalisable)* $T_k : L_2(\chi) \to L_2(\chi)$,

$$(T_k f)(V_1) = \int_{\chi} k(V_1,V_2) f(V_2) dV_2 \quad \forall f \in L_2(\chi) \tag{1.26}$$

Dans le cas où $N = \infty$ *, la série est alors presque partout absolument et uniformément convergente.*

Remarque 1

La séquence $\{\psi_i\}_{i=1}^{N}$ *est une base orthonormale de* H_k *. Il s'ensuit que toutes fonctions* $f \in H_k$ *s'expriment sous la forme :* $f(V_1) = \sum_{i=1}^{N} a_i \psi_i(V_1)$

Le théorème 1 nous apprend que tout noyau défini positif admet la représentation suivante :

$$k(V_1,V_2) = \sum_{I=1}^{N} \lambda_i \psi_i(V_1)\psi_i(V_2) = \sum_{i=1}^{N} \sqrt{\lambda_i}\psi_i(V_1)\sqrt{\lambda_i}\psi_i(V_2) =< \Phi(V_1), \Phi(V_2) > \tag{1.27}$$

Nous pouvons alors définir explicitement l'espace de description engendré par la fonction noyau k :

$$\Phi : \chi \to H_k$$
$$V_1 \to (\sqrt{\lambda_1}\psi_1(V_1), \sqrt{\lambda_2}\psi_2(V_1), ..., \sqrt{\lambda_N}\psi_N(V_1)) \qquad (1.28)$$

M représente la dimension de H_k, et peut être de dimension infinie.
A tout RKHS H_k, on associe un produit scalaire et une norme, définis par les équations (1.30) et (1.31) : soit $f(V_1) = \sum_{i=1}^{N} a_i \psi_i(V_1)$ et $h(V_1) = \sum_{i=1}^{N} b_i \psi_i(V_1)$, le produit scalaire dans H_k est défini par :

$$< f(V_1), h(V_1) >_{H_k} = \left\langle \sum_{i=1}^{N} a_i \psi_i(V_1), \sum_{i=1}^{N} b_i \psi_i(V_1) \right\rangle_{H_k} = \sum_{i=1}^{N} \frac{a_i b_i}{\lambda_i} \qquad (1.30)$$

La norme de H_k est définie par :

$$\| f \|^2_{H_k} = < f(V_1), f(V_1) >_{H_k} = \sum_{i=1}^{N} \frac{a_i^2}{\lambda_i} \qquad (1.31)$$

De l'équation (1.27), on peut voir que $k(V_1, V_1)$ correspond à l'évaluation d'un produit scalaire dans le RKHS H_k induit par k. Cette constatation a de grandes conséquences pratiques que nous explicitons au travers de la proposition 1.

Proposition 1 «L'astuce du noyau»(Kernel Trick)
Tout modèle ne nécessitant dans sa construction que la manipulation de produits scalaires entre observations (et non leur cordonnées explicites) peut être construit implicitement dans un espace de Hilbert en remplaçant chaque produit scalaire par l'évaluation d'un noyau défini positif sur un espace quelconque. Ce procédé de substitution a pris le nom de l'«astuce du noyau».

L'astuce du noyau est qu'elle permet de transformer de nombreux algorithmes "linéaires" (e.g. régression, classification, réduction de dimensionnalité) en algorithmes "non-linéaires" tout en conservant les mêmes propriétés d'optimisation. Nous verrons par la suite comment l'astuce du noyau peut être exploitée pour définir une mesure de similarité (distance) généralisée dans l'espace H_k. Dans le paragraphe suivant nous présentons les différentes fonctions noyaux les plus citées dans la littérature.

4.2 Fonctions noyaux classiques

Les fonctions noyaux les plus fréquemment utilisées sont : le noyau linéaire, polynomial, triangulaire et le noyau Gaussien.

o Le noyau linéaire :

$$k(V_i, V_j) = < V_i, V_j >,$$

Cette fonction exprime le produit scalaire usuel et elle nous permet de nous comparer à une utilisation sans approche noyau. Par exemple, elle exprimera la distance Euclidienne si la mesure de similarité est calculée à travers une distance Euclidienne.

o Gaussienne avec une distance Euclidienne

$$k(V_i, V_j) = e^{-\frac{\|V_i - V_j\|^2}{2\delta^2}}$$

C'est cette approche que nous utiliserons dans le cadre de ce travail

o Polynomiale de degré m :

$$k(V_i, V_j) = < V_i, V_j >^m$$

o Triangulaire :

$$k(V_i,V_j)=1-\frac{1}{r^2}\left\|V_i-V_j\right\|$$

Ce noyau [Ber 84] a été introduit pour la recherche d'images pour ces propriétés d'invariance aux échelles [Sah 03]. Afin d'avoir un noyau symétrique défini positif, les données sont placées dans une sphère de rayon $r^2/2$

o Gaussienne avec une distance $Chi-2$ (Gaussien $Chi-2$)

$$k(V_i,V_j)=e^{-\frac{dist(V_i,V_j)^2}{2\delta^2}}$$

Avec $dist$ une distance au sens du $Chi-2$: $dist(V_i,V_j)=\sum_{t=1}^{N}\frac{(V_{ti}-V_{tj})^2}{V_{ti}+V_{tj}}$.

Le noyau le plus utilisé dans la littérature est le noyau Gaussien :

$$k(V_i,V_j)=e^{-\frac{dist(V_i,V_j)}{2\delta^2}}$$

tel que δ est le paramètre de l'échelle. Ce paramètre permet de définir d'une manière implicite l'espace de projection à partir duquel on déduit la mesure de similarité. Généralement, une petite valeur de δ correspond à un espace de données riche dans le sens où les frontières entres les différentes classes de données sont bien distinguées, et une grande valeur de δ correspond à un espace de données pauvre dans le sens où les données sont moins discriminantes dans l'espace projeté. Le noyau Gaussien dépend de la distance $dist(V_i,V_j)$, ce qui nous permettra de choisir la distance la plus appropriée aux données traitées.

Une comparaison entre les différents noyaux qui existent dans la littérature en terme de classification et qualité de la recherche a été effectuée par Philippe Henri et al. [Phi 08]. La comparaison montre que la mesure de similarité avec un noyau linéaire (sans noyaux) produit de mauvais résultats, tandis que l'utilisation d'une mesure de similarité basée sur un noyau Gaussien avec une distance $Chi-2$ donne de meilleurs résultats. Par ailleurs, les fonctions Gaussiennes avec une distance Euclidienne et avec une distance triangulaire sont moins performantes qu'une Gaussienne $Chi-2$.

Finalement, le problème de la recherche par le contenu consistera à trouver la bonne distance pour mesurer la similarité entre les descripteurs d'images. Ceci est équivalent au problème de trouver une fonction de représentation $\varphi:V_1\rightarrow V_2$ qui permet de représenter le vecteur V_1 dans un espace de projection avec un autre vecteur V_2. Idéalement cette représentation devrait d'une part être informative et compacte (avoir le maximum d'information en ayant la dimension la plus petite possible) et d'autre part séparer au mieux les différentes classes de données. Le choix du noyau est crucial et correspond à un choix a priori sur la notion de similarité entre descripteurs.

Dans le reste du document, nous avons choisi de travailler avec le noyau Gaussien. Cette fonction comme nous allons le montrer est bien adaptée à nos objectifs. En effet, avec un noyau Gaussien, il est relativement aisé de composer (additionner, multiplier, …) des noyaux semi-définis positifs pour en obtenir d'autres, permettant ainsi une adaptation aux données. Nous verrons dans le chapitre 3 comment cette propriété sera exploitée dans le cadre de l'apprentissage pour déduire de nouvelles mesures de similarité adaptées à la nature des données et aux besoins de l'utilisateur.

4.3 Généralisation de la notion de distance à travers l'astuce du noyau

L'astuce du noyau peut être appliquée à une distance Euclidienne.

Dans l'espace d'entrée χ la distance Euclidienne entre deux vecteurs V_1 et V_2 est donnée par :

$$dist(V_1,V_2) = \|V_1 - V_2\|^2 = < V_1,V_1 > + < V_2,V_2 > -2 < V_1,V_2 >$$

On définit un noyau $k(V_1,V_2)$ dans le RKHS H_k tel que :

$$k(V_1,V_2) = < \Phi(V_1),\Phi(V_2) >$$

où $\Phi : \chi \to H_k$ est une fonction, qui à tout vecteur V_1 fait correspondre un nouveau vecteur dans l'espace H_k

L'astuce du noyau explicitée à travers le théorème de Mercer et la proposition 1 est exploitable puisque les vecteurs dans l'expression de la distance Euclidienne n'interviennent que par leur produit scalaire. Si l'on dispose donc d'un noyau k, alors on dispose d'une distance généralisée qui s'exprime en fonction des évaluations du noyau dans l'espace H_k de la manière suivante:

$$\begin{aligned} d(\Phi(V_1),\Phi(V_2))^2 &= < \Phi(V_1),\Phi(V_1) > + < \Phi(V_2),\Phi(V_2) > -2 < \Phi(V_1),\Phi(V_2) > \\ &= k(V_1,V_1) + k(V_2,V_2) - 2k(V_1,V_2) \end{aligned} \quad (1.32)$$

Les distances dans l'espace transformé s'expriment facilement à l'aide du noyau k et il est donc inutile de calculer Φ.

5 Synthèse

Dans ce chapitre, nous avons présenté certaines techniques utilisées dans le cadre de la recherche d'images par le contenu que ce soit pour la description ou pour la mesure de similarité . Dans la première partie, nous avons abordé la description de l'image par la couleur, la texture et la forme. Ces attributs bas niveau sont les descripteurs les plus largement rencontrés dans des SRIC. La mesure de similarité repose généralement sur un calcul de distances entre descripteurs dans un espace multidimensionnel. Dans la deuxième partie, nous avons passé en revue les différentes mesures qui permettent d'évaluer la similarité entre les différents ces descripteurs. Le choix d'une mesure de similarité par rapport à une autre dépend de l'attribut en question, par exemple dans le cas de la couleur, l'intersection d'histogrammes est la mesure la plus fréquemment utilisée tandis que pour la forme on fera souvent appel à la distance Euclidienne notamment si les descripteurs de type ART. Cependant, la similarité devient plus difficile à évaluer si plusieurs attributs (couleur, texture, forme) sont utilisés simultanément pour décrire l'image images. La littérature fait un état de diverses approches pour mesurer la similarité des descripteurs visuels., Celles-ci se trouvent confrontées à de nombreux problèmes telle que le choix des pondérations ou l'indépendance dans le traitement des différents attributs, diminuant ainsi leurs efficacité à traduire fidèlement la similarité visuelle entre les images. Dans la troisième partie de ce chapitre, nous avons abordé la mesure de similarité par l'approche noyau. Nous avons présenté nos motivations et nos arguments quant à l'utilisation des fonctions noyaux en tant que fonction de similarité. Cette approche permet l'utilisation d'un grand nombre de techniques d'analyse et d'exploitation de données. Deux paramètres que nous allons utiliser pour personnaliser ou orienter le choix du modèle de similarité, comme nous le verrons dans le chapitre 3, sont la dimension de l'espace de projection d et le paramètre de l'échelle δ de la fonction noyau Gaussienne.

Outre les différents avantages en termes d'analyse et d'exploitation de données, l'approche noyau présente un avantage particulièrement intéressant en indexation multidimensionnelle d'image du fait qu'elle permet de séparer le problème de structuration du problème de la représentation des données. Nous pouvons ainsi utiliser un seul espace de donnée, donc une

seule structure d'index, tout en ayant la possibilité de travailler avec plusieurs types d'attributs à la fois, sous réserve bien évidemment d'avoir construit une fonction noyau adaptée.

Chapitre 2

Indexation multidimensionnelle

Ce chapitre présente, un état de l'art sur les principales méthodes d'indexation multidimensionnelles utilisées pour la structuration de l'espace de description des images fixes. Nous présentons d'abord le principe général, les points forts et les points faibles de ces techniques. Ensuite, nous présentons le problème de la malédiction de la dimension, ainsi que les principales techniques de la réduction de la dimension qui permettent de contourner les problèmes liées aux espaces de grande dimension. Les techniques d'indexation basées sur l'approche approximation qui améliorent les performances en grandes dimension sont enfin présentées suivies d'une synthèse.

1 Introduction

L'émergence des technologies numériques dans le domaine du multimédia a mis en valeur l'importance des problèmes de l'indexation multidimensionnelle et de la recherche par le contenu dans grandes bases de données. Les méthodes conventionnelles d'indexation multidimensionnelles ont été proposés pour organiser les descripteurs ou aussi les vecteurs caractéristiques issus des images numériques afin d'éviter le parcours séquentiel exhaustif des grandes bases permettant ainsi un accès et une interrogation rapides lors d'une recherche par le contenu.

Généralement les descripteurs des images sont représentés par des vecteurs multidimensionnels dans un espace de grande dimension. Chaque composante de vecteurs correspond à une dimension différente de l'espace de données et représente un attribut spécifique d'un descripteur. Les données appartiennent alors à un espace multidimensionnel où chaque dimension représente un axe différent de l'espace de données. Ce format de représentation est souvent employé par les méthodes d'indexation multidimensionnelles grâce à sa flexibilité et sa simplicité pour représenter un grand nombre de différents types d'information. Dans ce contexte, le problème majeur des méthodes d'indexation multidimensionnelles est comment indexer efficacement une grande collection de données multidimensionnelles pour répondre rapidement et efficacement aux requêtes des utilisateurs. De ce fait, les méthodes d'indexation multidimensionnelles doivent répondre à un certain nombre d'exigences en ce qui concerne les aspects suivants :

- Problème de la dimension : répondre efficacement aux problèmes liés à la malédiction de la dimension.
- Passage à l'échelle : permettre le déploiement à grande échelle de la structure d'index.
- Performance de la recherche : assurer une bonne précision de la recherche en un temps raisonnable et acceptable par l'utilisateur.
- Dynamique : Supporter les insertions et suppression de données sans affecter significativement l'organisation de la structure d'index.
- Adaptabilité : Prendre en compte la répartition spatiale des données, et en tenir compte lors du processus d'indexation et de structuration.

La réponse à toutes ces conditions est un grand défi pour les méthodes d'indexation multidimensionnelles. Dans le paragraphe suivant nous passons en revue les index multidimensionnels permettant l'organisation des images fixes. Nous présentons leurs principes de base ainsi que leurs points forts et points faibles en ce qui concerne les performances en grandes dimension.

Il existe deux grandes familles de méthodes d'indexation multidimensionnelles : les méthodes d'indexation conventionnelles et les méthodes d'indexation basées sur l'approche *approximation* dite aussi *filtrage*. Le paragraphe suivant présente les principales méthodes de ces deux familles.

2 Méthodes d'indexation conventionnelles

Les techniques conventionnelles d'indexation multidimensionnelles reposent sur le principe de regrouper des vecteurs de la base en paquets, puis de les englober dans des formes géométriques simples à manipuler. L'idée est de réduire le parcours lors de la recherche à un sous ensemble de paquets en sélectionnant les plus pertinents et de n'accéder qu'un un nombre réduit de vecteurs. Les premiers travaux de recherche sur les algorithmes d'indexation multimédia dans un espace à plusieurs dimensions ont été réalisés entre 1979 et 1984. Une première classification des méthodes d'indexation multidimensionnelles a été réalisée par Gaede [Gae 98] En effet, Gaede a classifié ces méthodes selon les types de données qu'elles supportent. On distingue deux familles de méthodes :

- Méthodes d'accès par point (Point Acces Methods- PAM) où les données sont représentées par des points (vecteurs).
- Méthodes d'accès spatiales (Spatial Acces Methods - SAM) où les données sont représentées par des objets spatiaux complexes (segment, droite, polygones, etc.).

Une deuxième classification des méthodes d'indexation conventionnelles a été établie par la suite se basant sur la méthode de partitionnement et l'organisation des données dans l'espace de vecteurs multidimensionnel. En vertu d'un tel critère, deux grandes familles de méthodes peuvent être considérées :

- Méthodes basées sur le partitionnement de données qui regroupent les données en fonction de leur proximité dans l'espace. Elles sont toutes dérivées de la méthode R-Tree [Gut 84] où chaque groupe de vecteurs est englobé dans une forme géométrique particulière (hyper-sphère, hyper-rectangle, etc.). Le tout est structuré dans un arbre équilibré.
- Méthodes de partitionnement de l'espace. Ces méthodes sont toutes dérivées de K-D-Tree [Ben 75, Bfe 79] et Quad-Tree [Sam 84], elles découpent l'espace multidimensionnel en régions disjointes et ensuite, stockent les données selon ce découpage. La majorité des techniques d'indexation basées sur le partitionnement de l'espace de données sont des méthodes d'accès par point PAMs. Elles sont fréquemment appliquées sur un nuage de points dans un espace de vecteurs multidimensionnels.

Nous présentons dans la suite les principales caractéristiques et limites de ces deux familles de méthodes, en se focalisant sur les techniques d'indexation les plus connues dans la littérature. Un état de l'art exhaustif sur les méthodes d'indexation conventionnelles peut être trouvé dans [Gae 98] et [Ber 04].

2.1 Méthodes de partitionnement de données

L'indexation multidimensionnelle basée sur le partitionnement de données tient compte de la distribution des vecteurs dans l'espace multidimensionnel. Ces méthodes sont toutes dérivées du R-Tree [Gut 84] ou les données sont groupées dans des formes géométriques simples (hyper-rectangles, hyper-sphères, etc. selon leur proximité dans l'espace. Ces formes géométriques issues du partitionnement de données sont organisées sous forme d'un arbre dans lequel les vecteurs sont stockés dans les feuilles et les formes géométriques sont stockées dans les nœuds interne de l'arbre. Dans ce paragraphe, nous détaillerons quelques méthodes les plus connues dans la littérature

2.1.1 Famille du R-Tree

Les arbres de la famille R-Tree [Gut 84] indexent un espace multidimensionnel de points par un découpage hiérarchique équilibré en hyper-rectangles. Le R-Tree [Gut 84] est l'extension directe du B-Tree [Bay 71] aux espaces multidimensionnels. Le R-Tree, est un arbre équilibré dans lequel chaque nœud est associé à un rectangle minimum englobant (REM), ce dernier est le REM de tous les rectangles de ses fils. Les feuilles de l'arbre contiennent une liste d'entrées de type (REM, oid), où REM est le rectangle minimum englobant de l'objet identifié par son oid. La taille des nœuds et des feuilles est limitée et fixée a priori. Elle correspond à la taille d'une page disque. La figure 2.1 illustre un exemple simplifié dans lequel nous avons représenté les données par des points ronds et les régions par des rectangles.

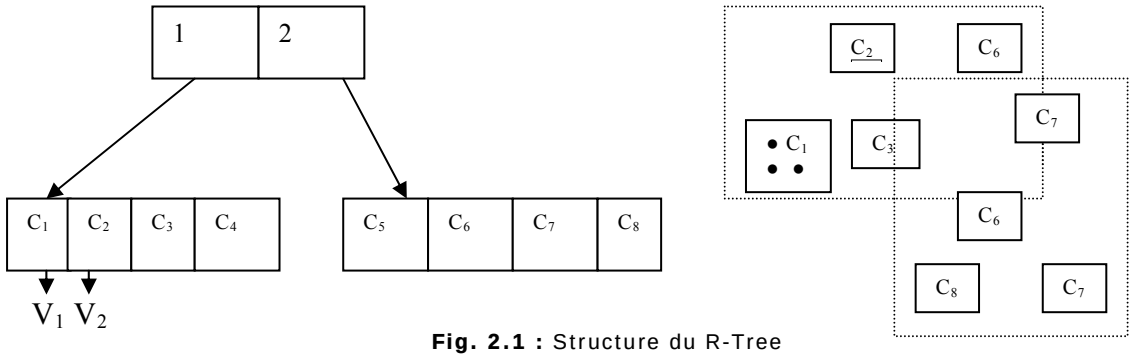


Fig. 2.1 : Structure du R-Tree

Le R*-Tree [Bec 90] est une variante du R-Tree. Pour améliorer les performances du R-Tree, Beckmann et al. Proposent la réinsertion de données stockées dans R-Tree. Au lieu de découper un nœud surchargé, les entrées supplémentaires sont supprimés et réinsérer dans l'arbre au même niveau. Dans plusieurs cas, ceci permet d'éviter le fractionnement envisagé, et, par conséquent augmente le taux d'exploitation de l'espace mémoire alloué. L'insertion d'un nouveau vecteur dans l'arbre R*-Tree se base sur deux critère : minimiser les volumes des REMs lors d'une insertion dans un nœud et minimiser le taux de chevauchement lors de l'insertion dans une feuille. Le partitionnent des REMs se base sur un critère qui prend en compte la minimisation du taux de chevauchement ainsi que le périmètre des hyper rectangle.

En Résumé, la famille des techniques d'indexation multidimensionnelle dérivées du R-Tree possède les propriétés suivantes :
– les cellules sont des hyper-rectangles englobant minimaux (REMs) ; elles sont organisées en hiérarchie où un niveau supérieur englobe les REMs du niveau inférieur ;
– les REMs permettent d'optimiser les règles de filtrage par l'utilisation d'une distance précise
– les formes englobantes se recouvrent, ce qui diminue l'efficacité des procédures de recherche; ce problème s'amplifie lorsque la dimension des données croît.
– les REMs se représentent par deux points multidimensionnels, ce qui pose des problèmes sur la quantité d'éléments que l'on peut stocker lorsque que la dimension augmente, ce qui peut entraîner la construction d'arbres de grande hauteur ;
– L'algorithme de construction du R-Tree par insertions successives est non déterministe : le même ensemble de données ne sera pas représenté par le même arbre selon l'ordre d'insertion dans l'arbre.

2.1.2 M-Tree

M-Tree [Cia 97] est une structure arborescente basée sur le partitionnement de données. Les vecteurs de données sont groupées d'une manière hiérarchique à partir de leur proximité dans l'espace de données et en se basant sur une distance métrique. C'est l'une des premières méthodes d'indexation qui vise à réduire, en plus du nombre d'entrée/sorties, le coût CPU des calculs de distances. En se basant sur des distances recalculées ainsi que l'inégalité triangulaire, cette technique évite certains calculs de distances inutiles lors de la recherche en éliminant des sous arbres non pertinentes à la requête. Dans un M-Tree, chaque nœud de l'arbre est composé d'un objet de la base appelé objet de redirection et un rayon appelé rayon de couverture. M-Tree est construit par insertions successives des objets (vecteurs de la base). Le critère du choix du chemin d'insertion vise à minimiser les rayons de couverture des objets de redirection. Le partitionnement des nœuds saturés permet de sélectionner deux objets du nœud puis de répartir les objets restant en minimisant le volume de deux régions obtenues ainsi que leurs taux de chevauchement. Un schéma représentatif de la structure du M-Tree est donné par la figure 2.2. La sélection de deux objets de redirection peut être effectué d'une manière exhaustive, aléatoire ou en considérant uniquement un échantillon d'objets en se basant sur les rayons de couverture.
M-Tree souffre du problème du chevauchement qui permet d'élargir le nombre de chemins à parcourir lors d'une recherche, le chevauchement augmente également le nombre de calculs de distance pour répondre à une requête de l'utilisateur ce qui diminue fortement la performance de la méthode. L'arbre Slim [Tra 00]est une méthode qui a été proposée pour améliorer le M-Tree en réduisant le chevauchement entre les REMs, elle est présenté dans le paragraphe suivant.

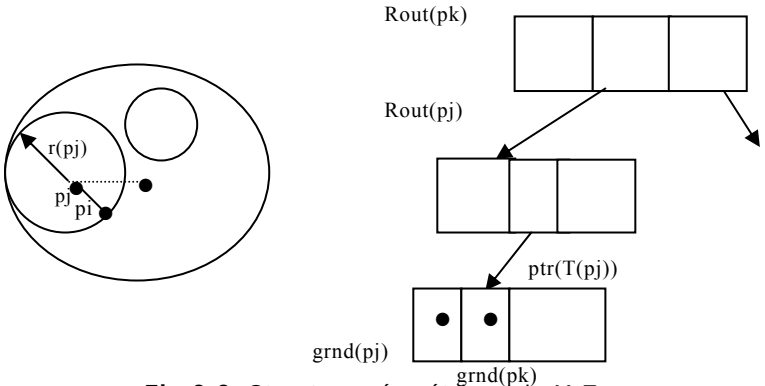


Fig.2.2. Structure géométrique du M-Tree

2.1.3 Slim-Tree

L'arbre Slim [Tra 00] est un arbre dynamique et équilibré qui regroupe les données dans des pages disques de taille fixe où chaque page correspond à un nœud de l'arbre, et les objets (vecteurs) sont stockés dans les feuilles de l'arbre. Il se base sur une technique simple pour quantifier le degré de chevauchement entre les nœuds de l'arbre. Il est bien connu que le degré de chevauchement touche directement la performance des index multidimensionnels, de ce fait l'arbre Slim est l'une des premières méthodes qui a été spécialement conçue pour réduire de taux de chevauchement. Cette technique organise les objets dans une structure hiérarchique à travers un représentant des données qui est le centre de la plus petite région qui couvre les objets dans un sous arbre. Tout comme la méthode M-Tree, la distance du représentant et l'inégalité triangulaire sont utilisées pour éliminer de la recherche les sous arbres non pertinents.
La construction de la structure d'index de l'arbre Slim se fait par insertions successives des différents vecteurs de la base de données. A partir du nœud racine, l'algorithme d'insertion cherche le nœud de l'arbre correspondant à une région de l'espace qui peut contenir le vecteur à insérer. Si aucun nœud n'est trouvé, le nœud dont le centre est plus proche du vecteur est choisie. Si l'algorithme d'insertion trouve plus qu'un nœud qui peuvent contenir le vecteur, l'algorithme d'insertion choisit le nœud soit d'une manière aléatoire, soit il choisit le nœud qui a une distance minimale entre le vecteur et le centre du nœud, ou bien le nœud qui possède le minimum de vecteurs.
Notons que le chevauchement est un des problèmes majeurs dont souffrent la majorité des techniques d'indexation, et qu'il est généralement difficile de le quantifier en raison de l'impossibilité de calculer le volume de l'intersection des régions. Pour résoudre ce problème, l'arbre Slim estime le taux de chevauchement par le nombre relatif des objets couverts par deux (ou plus) régions et puis applique un algorithme appelé "Slim-down" pour diminuer le taux de chevauchement entre les régions. Cet algorithme se compose de trois étapes. Dans la première étape, l'algorithme calcule le vecteur p_1 de plus loin de son vecteur représentatif p_0 (centre du nœud), pour chaque nœud. Dans la deuxième, l'algorithme identifie les nœuds qui couvrent le vecteur p_1, le déplace vers le nœud le plus proche et corrige le rayon du nœud résultant. La figure 2.3 illustre le fonctionnement de l'algorithme "Slim-down"

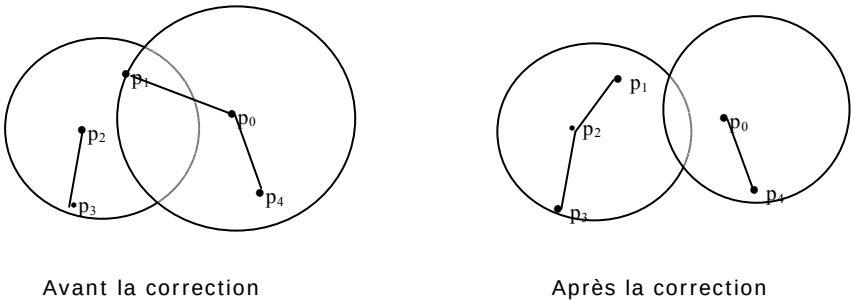


Fig.2.3 Fonctionnement de l'algorithme "Slim-down"

Notons que l'algorithme "Slim-Down" réduit bien le taux de chevauchement en comparaisons avec M-Tree, par contre celui-ci produit des nœuds triviaux contenant peu de vecteurs ou des nœuds vides ce qui réduit fortement les performances de l'index.

2.1.4 PM-Tree

PM-Tree (Pivot Metric Tree) [Sko 04] est une méthode d'indexation multidimensionnelle qui combine l'approche du pivot [Cha 01, Mic 94] et la méthode M-Tree dans l'objectif de réduire le volume de la région délimitant les vecteurs de données. Une telle réduction permet d'augmenter le taux de rejets des sous arbres non pertinents et par la suite augmente l'efficacité de l'index. La structure d'index du PM-Tree est construite de la même manière que M-Tree, par contre elle introduit des éléments de la base de données appelés *Pivot* pour identifier les plus petites régions contenant les vecteurs. Etant donné un ensemble de pivots $(Piv_1, Piv_2, ..., Piv_N) \in S$, avec S est l'ensemble de N vecteurs de la base, l'ensemble S est parcouru pour calculer un tableau de N' entrées. Le tableau de N' entrées contient les distances minimales $HR[i].\min$ et maximales qui délimitent chaque pivot (voir figure 2.4) et son objet correspondant. Pour chaque objet est définie une région appelée "hyper-ring" (MBR dans la figure 2.4) telle que son intersection avec l'hyperpère où se trouvent les données constitue la plus petite région délimitant un sous ensemble de vecteurs. Lors de la recherche d'une requête Q, la distance $dist(Q, Piv_j)$ est calculée pour chaque pivot, l'algorithme de la recherche élimine les objets O_i qui vérifient $|d(O_i, Piv_t) - d(Q, Piv_t)| \geq R_Q$ telle que R_Q est l'intervalle de la requête. Les objets non éliminés sont ensuite directement comparés avec la requête.

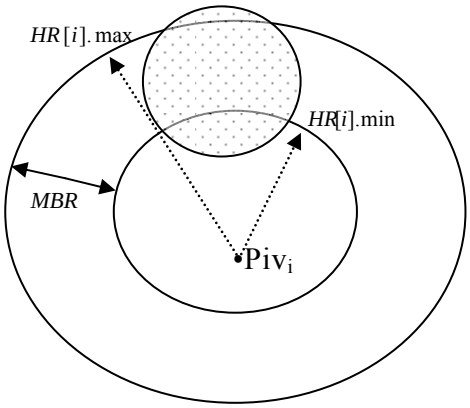


Fig. 2.4 La structure géométrique de la méthode du pivot métrique PM.

Notons que la différence entre PM-Tree et M-Tree réside dont le fait que PM-Tree utilise des régions "sphère-ring" au lieu des hyper sphères pour délimiter les zones qui contiennent les sous groupement d'objets stocké dans les feuilles de l'arbre, ceci réduit d'une manière considérable le volume des régions et augmente ainsi les performances de l'index .

2.1.5 MH-Tree

Récemment, MH-Tree est un index proposé par Guoren et al. [Guo 07] fondé sur le partitionnement des données par des hyperplans. L'arbre MH est dynamique et équilibré, il est conçu pour supporter les données dynamiques et ne nécessite pas de réorganisations périodiques de la structure, il se base sur l'utilisation d'un hyperplan de partitionnement au lieu d'une dimension clé pour le partitionnement des données ainsi que le filtrage de ces derniers. Notons que l'arbre MH convient uniquement aux espaces Euclidiens en raison des caractéristiques de l'hyperplan utilisé pour le partitionnement. L'idée d'utiliser un hyperplan de partitionnement pour diviser les données et non pas une seule dimension vient du faite que dans la plupart des applications, il existe plusieurs dimensions des données qui peuvent contenir une grande quantité d'information. Pour cela, il convient donc de garder les dimensions qui contiennent des

valeurs maximales de la variance pour construire un hyperplan de partitionnent en utilisant ces dimensions. L'arbre MH se base sur les étapes suivantes pour déterminer l'hyperplan de partitionnent des données : Premièrement l'arbre MH sélectionne deux points de références en se basant sur la méthode m-RAD-2 [Cia 97] pour déterminer le centre des deux sous espaces, ensuite calcule la valeur absolue de la différence entre les deux points et finalement choisit les dimensions qui ont la plus grande valeur absolue comme des dimensions clés. Le filtrage des régions lors de la recherche d'un vecteur requête s'appuie sur les distances entre un vecteur et l'hyperplan d'une part, et d'autre part, sur l'utilisation de l'inégalité triangulaire comme dans la méthode M-Tree.

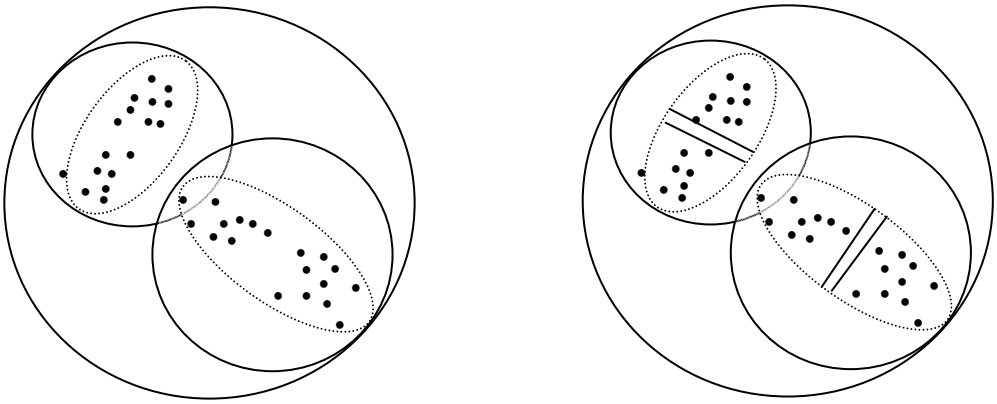


Fig.2.5 le partitionnement de données selon (a) M-Tree (b) MH-Tree

La figure 2.5 montre un exemple de partitionnement de données selon la méthode (a) M-Tree et (b) MH-Tree. M-Tree subdivise l'espace de données en deux sous espaces en s'appuyant sur les distances entre les vecteurs. L'arbre MH par contre utilise deux niveaux de partitionnement dans le premier niveau les données sont partitionnées selon la méthode m-RAD-2 en deux sous espaces et dans le deuxième niveau les deux sous espaces sont partitionnées en deux sous nœuds suivant les dimensions clés choisies. L'arbre MH utilise un hyperplan de partitionnent au lieu d'une dimension clé, l'hyperplan peut être différent des deux sous espaces de données et sa direction peut être modifiée selon la distribution de données dans l'espace considéré. La stratégie du partitionnement détermine un hyperplan de partitionnement en fixant un certains nombre de dimensions ainsi que leurs coefficients correspondant en s'appuyant sur la distribution des données.

2.1.6 Synthèse

Les méthodes conventionnelles d'indexation multidimensionnelles basées sur le partitionnement de données se basent sur l'utilisation des formes géométriques englobantes qui permettent de d'affiner le filtrage des régions pouvant contenir l'ensemble des résultats. Malheureusement ces méthodes souffrent des problèmes de la malédiction de la dimension, leurs performances se dégradent dès que la dimension augmente. Le principal inconvénient de ces méthodes et le chevauchement entre les formes géométriques englobant les vecteurs. En effet, le chevauchement rend les règles de filtrages incapables d'éliminer les formes géométriques non pertinentes. Par conséquent, un grand nombre de régions (rectangles, sphères, etc.) est visité au cours de la recherche entraînant ainsi une augmentation du temps de réponse. Pour cela le principal objectif de la plupart de des méthodes basées sur le partitionnent est de réduire autant que possible le taux de chevauchement entre les formes géométriques englobantes. Ci-dessous

le tableau 2.1 récapitule les principaux avantages et les inconvénients des différentes méthodes présentées précédemment.

Méthode	*Avantages*	*Inconvénients*
R-Tree	*Formes englobantes permettant d'affiner le de filtrage des candidats	*Chevauchement entre les formes géométriques * dégradation des performances au passage à l'échelle & grande dimension
R*-Tree	*Formes englobantes permettant d'affiner le de filtrage des candidats * Réduction du taux de chevauchement par rapport à R-Tree	*Chevauchement entre les formes géométriques *Dégradation des performances au passage à l'échelle & grande dimension
M-Tree	* *Formes englobantes permettant d'affiner le de filtrage des candidats *Réduction du temps de calcul par le stockage de le distance dans les nœuds. *Réduction du cout CPU par l'utilisation de l'inégalité triangulaire	*Chevauchement entre les formes géométriques *Dégradation des performances au passage à l'échelle & grande dimension *Méthode de partitionnement non adapté aux données fortement groupées
Slim-Tree	*Formes englobantes permettant d'affiner le de filtrage des candidats *Réduction du taux de chevauchement par rapport à M-Tree * Bonnes performances par rapport à M-Tree	*Possibilité de produire des nœuds triviaux lors du partitionnement des données * Dégradation des performances au passage à l'échelle &grande dimension
PM-Tree	*Formes englobantes permettant d'affiner le de filtrage des candidats *Réduction du volume des régions délimitant les données=> bonnes performances par rapport à M-Tree	*Possibilité du chevauchement. * Dégradation de performance au passage à l'échelle & grande dimension
MH-Tree	*Formes englobantes permettant d'affiner le de filtrage des candidats *Méthode de partitionnement tient compte de la distribution des données *Adapté aux données fortement groupées *Supporte les mises à jour (structure dynamique)	*Exploitée uniquement dans les espaces Euclidiens * Non adaptée aux données uniformes *Convient uniquement à l'espace Euclidien *Dégradation des performances au passage à l'échelle & grande dimension

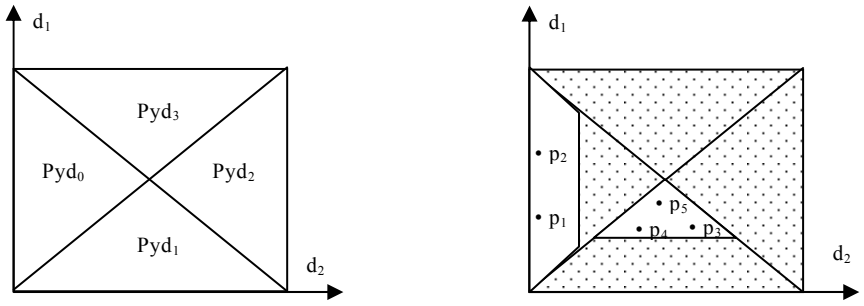
Tableau 2.1 Récapitulatif des avantages et inconvénients des différentes méthodes citées précédemment

2.2 Méthodes de partitionnement de l'espace

Contrairement aux méthodes d'indexation basées sur le partitionnement de données dont le principal inconvénient est le chevauchement entre les formes géométriques choisies pour le groupement des données, les méthodes de partitionnent de l'espace partitionnent l'espace de données en des formes géométriques (hyper-sphère, hyper-rectangle, etc.) disjointes. Plusieurs techniques ont été proposées. Nous citons par exemple : K-D-B-Tree [Rob 81], Grid-File [Nhs 84], LSD-tree [Hsw 89]. D'autres techniques plus récentes ont été proposées comme Pyramide-Tree [Bbk 98], iMinMax [Ooa 00] , P^+-Tree [Zot 04], ViTri [Soz 05], Kpyr [Thi 05]. Ces méthodes sont présentées dans le sous paragraphe suivant.

2.2.1 Pyramide

La technique de la pyramide [Bbk 98] est l'une des premières méthodes proposée ne subissant pas les problèmes de la malédiction de la dimension, elle divise l'espace en $2\times d$ pyramides, puis affecte aux données un numéro de pyramide et sa hauteur jusqu'au sommet de la pyramide. Chaque pyramide possède une base ayant une surface de $d-1$ dimensions, elle est découpée en tranches parallèlement à sa base. Les tranches près du sommet sont plus petites que celles qui sont proches de la base. Ce découpage de l'espace a la propriété de créer un nombre de cases linéairement croissant avec la dimension. D'un autre côté, les vecteurs multidimensionnels selon la méthode de la pyramide sont représentés par une clé composée du numéro de la pyramide et sa hauteur jusqu'au sommet, ces clés sont ensuite indexées par un arbre B^+-Tree [Bay 72]. Ainsi, les $2\times d$ pyramides formant un hyper cube, issues du découpage de l'espace de données sont représentés avec une même structure. La technique du pyramide transforme un point multidimensionnel en une clé à une seule dimension. La figure 2.6 représente un exemple de numérotation de quartes pyramides Pyd_1, Pyd_2, Pyd_3, et Pyd_4 dans deux dimension d_1 et d_2.
Les performances de la technique de la Pyramide se dégradent lentement lorsque la dimension des données augmente. Cependant, elles dépendent fortement de la distribution des données et de la position de la requête dans l'espace (requête intervalle : rang query). En effet, lorsque la distribution des données n'est pas uniforme, le choix du sommet des pyramides n'est pas significatif, ce qui détériore la qualité d'indexation et la rapidité de la recherche. De même, lorsqu'une requête est proche de la base d'une pyramide elle peut générer des accès inutiles à un ensemble de données n'ayant pas forcément de similarité avec la requête. Ceci risque d'affecter considérablement les performances de la recherche.
D'un autre côté, la performance de la pyramide dépend de la distribution des données. En effet, si les données sont regroupées dans un coin de l'espace, toutes les requêtes devraient être dans le même coin que les données puisque la répartition des requêtes souvent suit celles des données, c'est pour cette raison que la méthode de la pyramide est mois performante qu'une séquentielle pour les données fortement groupées.
La technique de la pyramide n'est pas efficace dans le cas où les requêtes ont une forme non hyper cubes comme dans le cas de la figure 2.7. D'après la figure si le rectangle de la requête est situé à la position p_1, le temps du parcours lors de la recherche peut être acceptable. Par contre si le rectangle de la requête est situé à la position p_2, le cout de la recherche la région accédée est plus élevé que celui du rectangle de la requête. Ceci rend également la technique de la pyramide inefficace.



(a) **(b)**

Fig. 2.6 La correspondance entre les régions (b) et les pyramides (a) en deux dimensions selon la technique de la pyramide.

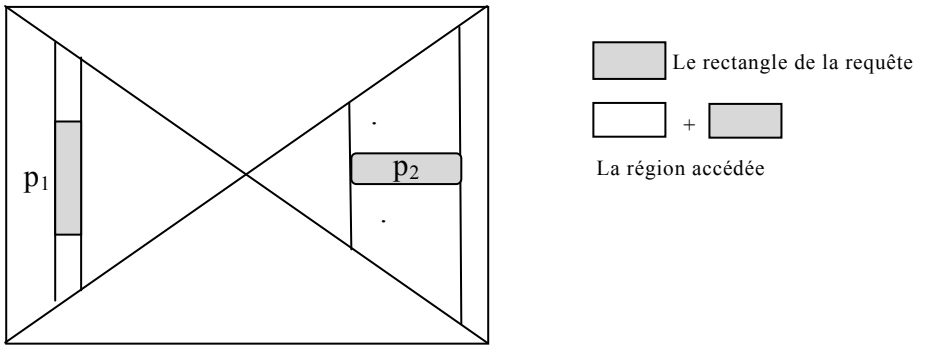


Fig. 2.7 Exemple de requête de forme non-hyper cube

2.2.2 IMinMax

Tout comme la technique Pyramide-Tree, iMinMax [Ooa 00] décompose l'espace multidimensionnel en $2\times d$ pyramides. La différence entre les deux méthodes réside dans la stratégie avec laquelle les données sont représentées. En effet, chaque vecteur multidimensionnel dans iMinMax est représenté par une clé composée de sa coordonnée la plus proche de la surface relatif à la dimension $d-1$ et sa magnitude. Cette méthode utilise une seule transformation pour projeter les vecteurs multidimensionnels dans un espace à une dimension (clé des vecteurs). Soient $x_{\min}$ et $x_{\max}$ respectivement la valeur minimale et maximale d'un vecteur p suivant la dimension d et $d_{\min}$ et $d_{\max}$ sont respectivement les dimensions relatifs aux coordonnées $x_{\min}$ et $x_{\max}$. Le vecteur p est projeté sur la dimension d_0 suivant la transformation ci-dessus:

$$d_0 = \begin{cases} d_{\min} + x_{\min} & si \quad x_{\min} + \theta < 1 - x_{\max} \\ d_{\max} + x_{\max} & \text{sinon} \end{cases} \qquad (2.1)$$

En modifiant la valeur du paramètre θ, on obtient différentes structures de iMinMax. Par exemple, pour $\theta \le -1$, les données sont projetées sur une dimension avec la valeur minimale, alors que pour une valeur de θ supérieur à 1, les données sont projetées sur une dimension avec la valeur maximale. Un exemple illustratif de l'influence du paramètre θ sur la création de la structure d'index est illustré dans la figure 2.8. Les clés des vecteurs multidimensionnels calculées à travers l'équation (2.1) et ils sont ensuite indexées à travers l'arbre B+.

Les performances de iMinMax et Pyramide-Tree sont identiques pour des données uniformes et pour $\theta = 0$. Pour une distribution non uniforme des données, les performances de iMinMax

dépendent largement du choix de la valeur de θ, le choix de cette valeur n'est pas évident et il peut être adaptés selon la distribution des données.

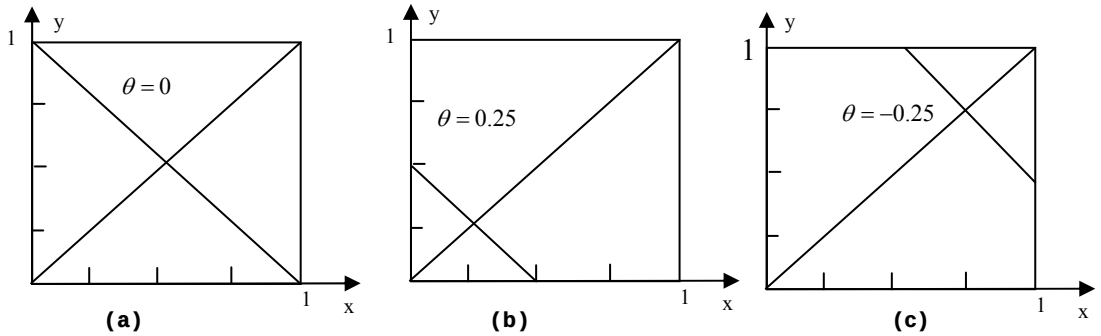
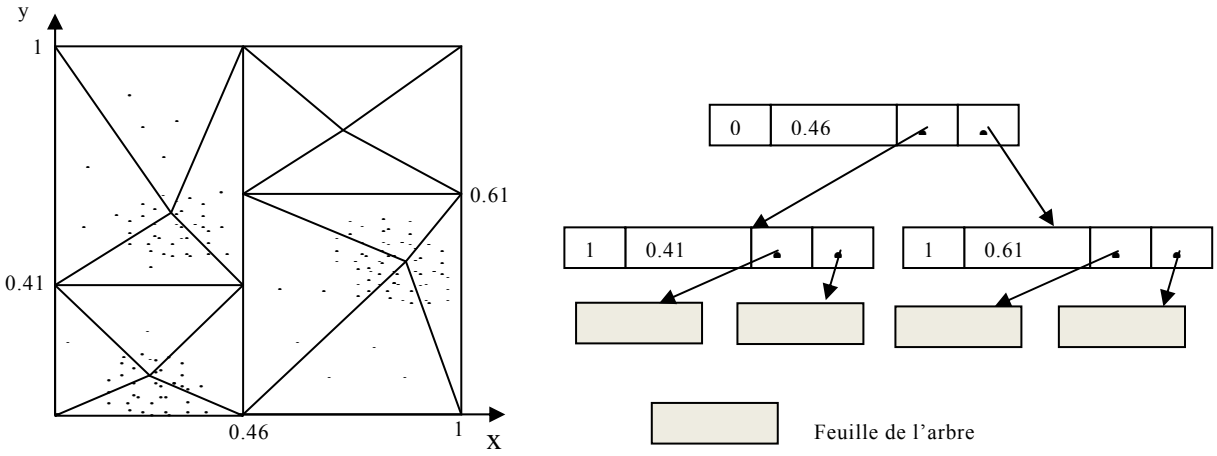


Fig. 2.8 Résultat du partitionnement d'un espace à deux dimensions selon la méthode iMinMax où (a) $\theta = 0$ (b) $\theta = 0.25$ (c) $\theta = -0.25$

2.2.3 P$^+$-Tree

P$^+$-Tree [Zot 04] est une amélioration de la technique Pyramidale. Cette approche associe une méthode de division de l'espace basée sur Bissecting K-means [Sbb 02] et la technique de la Pyramide. En effet, P$^+$-Tree divise l'espace en hyper rectangles, puis applique la technique de la Pyramide à chacun de ces sous espaces. Chaque sommet de la pyramide représente le centre d'un groupement de données. Dans le cas où les données se situent autour du sommet, et malgré l'existence de certaines requêtes au bord de la pyramide produisant bien évidemment lors du processus de la recherche un accès à une large zone. Les données parcourues ne sont pas nombreux du fait que la plupart d'entre eux se situent autours du sommet de la pyramide. De plus, la région parcourue est réduite par le processus de partitionnement, c'est ainsi que la méthode P$^+$-Tree réduit bien l'inefficacité de la pyramide. Notons que la construction du P$^+$-Tree est réalisée à travers une deuxième structure appelée "Space-Tree", cet un arbre dont les feuilles stockent les informations relatives à la transformation de l'espace de données, cet arbre est construit au cours du partitionnement de l'espace de données. Un exemple de partitionnent de l'espace de données ainsi que la construction de la structure "Space-Tree" correspondant est illustré dans la figure 2.9.



. **(a)** **(b)**

Fig. 2.9. Exemple de partitionnement de l'espace de données (a) et (b)de construction de la structure space-Tree

Les performances de cette méthode dépendant essentiellement de la distribution des données et de la méthode de partitionnement de l'espace. Ce dernier ne tient pas compte de la

distribution des données, ceci réduit d'une manière considérable la performance de la méthode P^+-Tree

2.2.4 ViTri (Video Triplet)

Shen et al. [Soz 05] ont proposé une nouvelle méthode d'indexation vidéo ViTri sous forme d'arbre B+. Une séquence vidéo est tout d'abord divisée en un certain nombre de groupes contenant des "frames" similaires. Chaque groupement est ensuite modélisé par une hyper-sphère dans un espace à d dimensions où d représente également la dimension du vecteur multidimensionnel relatif à chaque "frame". Chaque hyper-sphère est représentée par le triplet : (position, rayon, densité) qui représentent respectivement, la position du centre du groupement, son rayon et le nombre de "frames" dans le groupement. Le rayon du groupement (où l'hyper-sphère) est calculé par la moyenne des écarts types entre les différent "frames" et le centre du groupement. La similarité entre deux groupements est évaluée par l'estimation du nombre de "frames" similaires entre les deux groupes, ceci est calculé par l'intersection entre les deux hyper-sphères multipliées par la plus petite densité. Ainsi, l'information locale au niveau de chaque groupement est identifiée d'une manière efficace. La recherche exhaustive au niveau de la structure de ViTri pour un grand nombre de données est très couteuse. Pour cela, Shen et al. proposent de transformer les données multidimensionnelles en un espace unidimensionnel, et puis d'appliquer une analyse en composantes principale (ACP) sur les données unidimensionnelles afin de désigner par la suite un point de référence optimal (O dans la figure 2.10) pour lequel la distance originale entre les vecteurs multidimensionnels soit maximale après la transformation. Les données résultantes sont ensuite indexées par l'arbre B+.

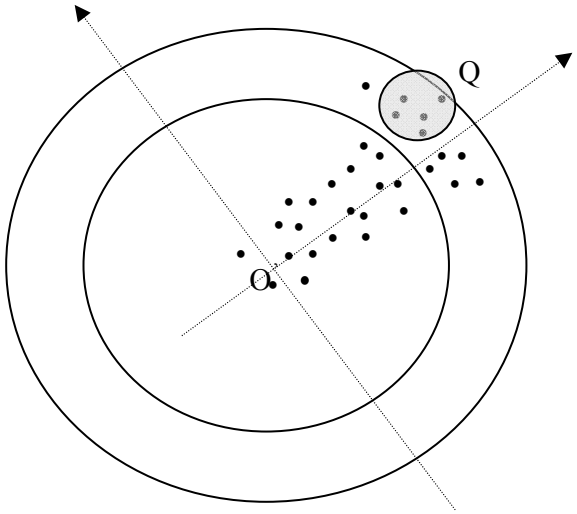


Fig.2.10. Représentation géométrique de l'indexation selon la méthode ViTri

2.2.5 Kpyr

Thierry et al. [Thi 05] ont proposé une méthode d'indexation qui associe une méthode de classification à une technique d'indexation multidimensionnelle. Kpyr procède d'abord par une étape de classification à travers l'algorithme K-Means [Mac 96] dans laquelle les données sont partitionnées en un certain nombre de groupements homogènes. Ensuite, à chaque groupement est effectué un changement de base du sous espace correspondant en un hyper cube unitaire sur

lequel est appliqué la technique pyramidale. Les données de chaque groupement sont indexées par un arbre B^+.
La structure du Kpyr est une structure arborescente binaire appelée "arbre espace" telle que les nœuds de l'arbre stockent les frontières calculées à partir des distances inter groupes alors que les feuilles de l'arbre sont représentées par les arbres B^+ des groupements. Un exemple de représentation géométrique de Kpyr est donné par la figure 2.11. La recherche s'effectue en deux étapes. La première consiste à déterminer les régions concernées par la recherche en s'appuyant sur les frontières des régions pour en déduire les groupements atteints par la requête. La seconde correspond à la recherche d'une requête à l'intérieur de la pyramide de chacun de ces groupements.

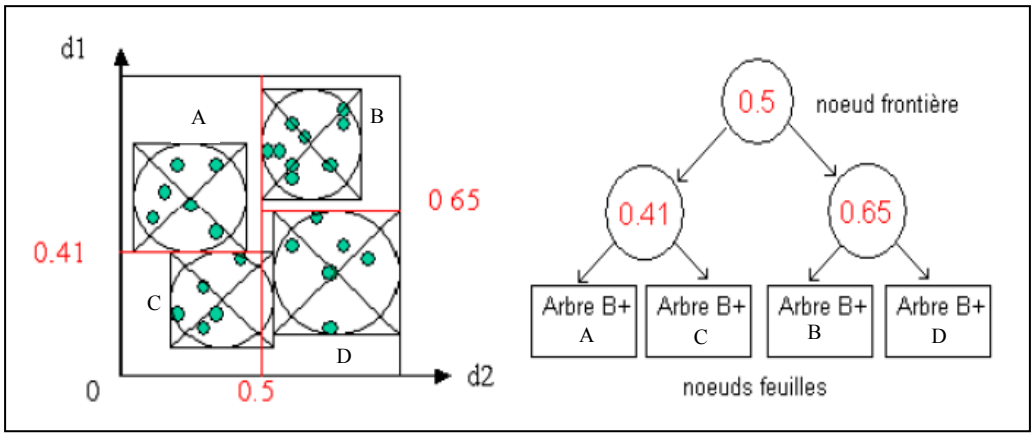


Fig. 2.11. La structure d'index du Kpyr[Thi 05]

Les performances de Kpyr sont affectées par le volume des données traitées lors de la recherche. Le fait d'accéder à un grand nombre de vecteurs non pertinents à une requête donnée dégrade considérablement le temps de la recherche. En effet, l'indexation par la technique de la pyramide ne tient pas compte du nombre important de vecteurs inutiles explorés au cours de la recherche, ce nombre augmente rapidement dans les espaces de très grandes dimensions. Plus la dimension augmente, plus les zones proches de la base de la pyramide sont volumineuses. Une requête appartenant à cette zone entrainera le parcours d'un nombre important de vecteurs non pertinents à la requête recherchée.
Pour contourner ce problème Tierry et al. proposent le KpyrRec [Thi 06] une nouvelle méthode d'indexation multidimensionnelle qui procèdent de la même manière que Kpyr. L'idée principale de cette méthode est d'effectuer un partitionnement efficace de l'espace de données pour réduire le nombre de données accédées lors d'une recherche. KpyrRec réalise un découpage de l'espace de données équilibré en volume de données en faisant une récurrence de la technique de la pyramide sur l'ensemble des données appartenant à la même tranche, où chaque récurrence réduit le nombre de dimension de 1. Un seuil correspondant au nombre de données maximum par tranche est fixé pour arrêter la récurrence. L'index récurent résultant induit un surcroit négligeable de mémoire par rapport au gain important en temps de réponse, cette technique garantit une réduction du temps de réponse de 20% en moyenne. Ces performances ne sont affectées ni par le volume de données, ni par leur distribution. Elles ne sont affectées que par les grandes dimensions.

2.2.6 Synthèse

Dans le tableau 2.2 nous récapitulons les avantages et les inconvénients des différentes méthodes d'indexation basées sur le partitionnement de l'espace de données présentées précédemment. Contrairement aux méthodes d'indexation conventionnelles basées sur le partitionnement de données, dont le principal inconvénient est le recouvrement des formes géométriques englobantes, ces méthodes partitionnent l'espace de données en des formes géométriques disjointes sans aucun chevauchement. Le partitionnement s'effectue sans prendre en considération la distribution des données, qui permet de générer des formes géométriques régulières simple à gérer. La plupart de ces méthodes ne supportent pas les données réelles de distribution non uniformes ou fortement groupées. Tout comme les méthodes conventionnelles basées sur le partitionnement de données, ces méthodes souffrent des problèmes de la malédiction de la dimension, leurs performances se dégradent notablement lorsque la dimension augmente.

Méthode	***Avantages***	***Inconvénients***
Pyramide	*Peu affecté par l'augmentation de la dimension des données *Performante pour les données uniformes * Nombre de cellules créer est linéairement croisant en fonction de la dimension	* Non adaptée pour les données fortement groupées (moins performante qu'une séquentielle pour les données fortement groupées) *Dégradation des performances au passage à l'échelle & grande dimension
iMinMax	* Peu affecté par l'augmentation de la dimension des données *Nombre de cellules créer est linéairement croisant en fonction de la dimension *Adaptée aux données uniformes et fortement groupées	*Paramétrage difficile pour les données fortement groupées *Dégradation des performances au passage à l'échelle & grande dimension
P^+-Tree	* Performante au passage à l'échelle et en grande dimension par rapport à la Pyramide	*Algorithme de division de l'espace de données peu performant *Dégradation des performances pour les données fortement groupées
ViTri	*Adaptée aux données corrélés linéairement * Réduction de la dimension des données	*Non adaptée aux données non linéairement corrélées
Kpyr	* Adaptée aux données fortement groupées * classification des données-> meilleure division de l'espace par rapport à la pyramide *réduction du temps de calcul par rapport à la pyramide * Indexation efficace par rapport à la pyramide (l'espace à indexer par	* Non adaptée aux données uniformes * Problème d'accès inutile * Dégradation des performances au passage à l'échelle & en grande dimension

	la pyramide est réduit)	
KpyrRec	* Adaptée aux données fortement groupées * classification des données-> meilleure division de l'espace par rapport à la pyramide * Performante au grand volume de données *Réduit le nombre de données à accédées lors de la recherche par rapport à Kpyr	*Non adaptée aux données uniformes * Occupe un espace mémoire plus grand que celui de Kpyr *Dégradation des performances au passage à l'échelle & en grande dimension

Tableau 2.2 Récapitulatif des avantages et inconvénients des différentes méthodes citées précédemment

Dans le paragraphe suivant, nous rappelons les problèmes liés aux espaces de grande dimension

3 Malédiction de la dimension

La malédiction de la dimension fait référence aux problèmes dont souffrent la majorité des méthodes d'indexation multidimensionnelles en grande dimension. Cette expression trouve ses origines dans les travaux de Bellman [Bel 61], elle indique que le nombre de vecteurs nécessaires pour estimer une fonction arbitraire avec un certain niveau de performance croit exponentiellement avec le nombre de dimensions. Par exemple, optimiser une fonction à plusieurs variables par une méthode exhaustive consiste à partitionner le domaine de définition de chacune des variables à intervalles réguliers. Chaque point d'intersection de ces intervalles et un optimums possible . Il s'agit donc d'évaluer la fonction coût en chacun de ces points et de calculer la valeur optimale. Le nombre d'évaluation nécessaire, croît exponentiellement avec le nombre de variable. Dans le cas de la recherche par similarité en particulier, la recherche des plus proches voisins, cela signifie que le nombre de cellules (formes géométriques) qui doivent être examinés croît de façon exponentielle avec la dimension. Par exemple, dans un cube unitaire de dimension deux, si chaque dimension est partitionnée en deux intervalle, 2^2 cellules sont générées est parcourues lors de la recherche. En dimension 10 dans les mêmes conditions 2^{10} cellules sont examinées lors de la recherche. Plusieurs travaux, par exemple ceux de weber et al. [Web 98] ainsi que la thèse de Berrani [Ber 04], ont étudié les difficultés rencontrés lors de l'utilisation les données multidimensionnelles. Une synthèse des différents problèmes cités dans la littérature est présentée dans le paragraphe suivant.

3.1 Problèmes d'indexation et de recherche dans les espaces de grande dimension

Les problèmes qui se posent dans les espaces de grandes dimensions sont nombreux et il est généralement difficile d'étendre dans ces espaces les techniques que l'on a dans les espaces à deux ou à trois dimensions. Un premier problème qui se pose lors de la manipulation des données de grande dimension est le phénomène des vecteurs équidistants. En effet, Beyer et al. [Bey 99], remarquent que sous certaines conditions sur la distribution des données, les vecteurs ont tendance à devenir équidistants lorsque la dimension augmente. Ce phénomène entraine

l'inutilité des index multidimensionnels pour la recherche par similarité et le manque de pertinence des résultats retournés. Un second problème qui rend les techniques d'indexation fondées sur le partitionnement de l'espace inefficaces est celui de la croissance exponentielle du nombre de cellules en fonction de la dimension. Cela pose de sérieux problèmes pour les procédures de recherche. En effet, dans le cas où un vecteur requête se trouve suffisamment près d'une frontière entre cellules, le processus de recherche des plus proches voisins doit nécessairement examiner la ou les cellules voisines ce qui augmente par conséquent le temps de la recherche. Les techniques d'indexation basées sur le partitionnement de données regroupent les données en des formes géométriques particulières (rectangles, sphères,…). Toutefois les propriétés géométriques de ces formes ne sont plus maitrisables en grande dimension. Par exemple, il est simple de montrer que la limite du volume d'une hyper-sphère lorsque le nombre de dimensions tend vers l'infini est nulle. Si les données ont une répartition homogène dans l'espace, l'observation précédente implique que la grande majorité des données se situent à l'extérieur de la sphère. Ce phénomène que nous retrouverons dans les volumes de recouvrement des méthodes telle que la famille du R-Tree, SS-Tree, SR-Tree etc. est appelé phénomène de l'espace vide [Sco 83], il complique le groupement des vecteurs en paquets car le nombre de vecteurs n'est pas suffisamment élevé par rapport au nombre de dimension. Ceci produit par conséquent des formes géométriques volumineuse qui se chevauchent et avec peu de vecteurs diminuant ainsi la performance des index en grande dimension. D'un autre côté, Weber et al. [Web 98], montrent que dans un espace de dimension supérieure à 16, la recherche séquentielle est meilleure que la plupart des techniques d'indexation basées sur le partitionnement de données et de l'espace. Dans le même ordre d'idées, Weber constate que la complexité des approches basées sur le partitionnement et le groupement de données tend vers $\theta(N)$ où N, le nombre de vecteurs dans la base lorsque la dimension augmente. Les auteurs ont montré également que pour toutes les méthodes de partitionnement et de groupement, il existe une dimension telle que toutes les formes géométriques qui englobent les données sont examinés lors de la recherche.
En conclusion, les techniques d'indexation restent adaptées à des vecteurs de petites dimensions, or dans des applications réelles les données sont généralement représentées par un grand nombre de caractéristiques ce qui rend l'utilisation de la plupart des index multidimensionnelles difficiles. Pour contourner les problématiques liées aux grandes dimensions des données manipulées, des techniques de réduction de la dimension sont par conséquent nécessaires. Dans la suite nous présentons les principales techniques de la réduction de la dimension qui existent dans la littérature, nous distinguons deux grandes familles : les méthodes dites linéaires de la réduction de la dimension et les méthodes non linéaires.

3.2 Techniques de réduction de la dimension

Les techniques de la réduction de la dimension ont été développés dans le domaine de la statistique et l'analyse de données pour transformer les données d'un espace appelée espace d'entrée χ "input space" dans un nouvel espace de caractéristiques F "feature space" de dimension réduite en gardant le maximum d'information portée par les données dans leurs espace original. Mathématiquement, le problème que visent à résoudre ces méthodes peut être formulé de la manière suivante :
étant donné un vecteur de dimension d', $p=(p_1,...p_{d'})^T$, il s'agit de trouver une autre représentation de dimension réduite $q=(q_1,...q_d)^T$ avec $d<d'$, qui exprime le plus fidèlement

possible l'information contenue dans les données initiales . Notons que la dimension correspond aux composantes des vecteurs multidimensionnels, dans ce manuscrit, la dimension est parfois appelée composante des vecteurs, attribut en encore caractéristique.
Les techniques de la réduction de la dimension ont été présentées dans plusieurs travaux [Wu 00, Cha 01, Bha 06], comme étant une solution palliative aux problèmes de la malédiction de la dimension. Ces travaux suggèrent de réduire la dimension de l'espace de données avec une méthode de la réduction de la dimension avant d'indexer les données avec une méthode d'indexation multidimensionnelle.
Dans la littérature, il existe plusieurs manières de classifier les méthodes de la réduction de la dimension:
1) une classification basée sur la méthode utilisée pour la réduction du nombre de caractéristique. On distingue deux grandes familles :
- Méthodes basées sur la sélection des composantes : l'idée est de réduire le nombre de caractéristiques utilisée dans la classification tout en gardant un taux de classification acceptable. L'objectif de ces méthodes est d'éliminer les informations non discriminantes tout en gardant un ensemble d'information suffisant pour la discrimination des classes.
- Méthodes basées sur l'extraction des caractéristiques : l'idée de cette famille de méthodes est de transformer les données originales dans un espace de dimension réduite de façon à générer un nouvel ensemble de données significatif. La transformation est basée sur certains critères, comme la maximisation de la variance ou la minimisation de l'erreur, etc.
2) une deuxième classification des méthodes de la réduction de la dimension est basée sur la méthode utilisée pour générer une représentation des données de dimension réduites. Nous distinguons deux grandes familles
- Les méthodes projectives : L'objectif de ces méthodes est de trouver les projections de dimension réduite qui permettent d'extraire des informations utile à partir des données originales, par une maximisation d'une fonction certaine fonction appelée fonction objective. La fonction objective est typiquement la variance de l'erreur de reconstruction.
- Méthodes basées sur la modélisation : Ces méthodes tentent de trouver les sous structures des données se trouvant dans des espaces complexes de grande dimension
3) Une troisième classification est basée sur le type de la fonction de transformation. Nous utilisons cette classification pour représenter les méthodes de la réduction de la dimension. Dans cette catégorie on trouve :
-Les méthodes linéaires : Ces méthodes réduisent la dimension des données originales on appliquant des transformations linéaires sur ces données. Ces méthodes sont plus efficaces si les données originales sont distribuées d'une manière uniforme dans l'espace d'entrée.
- Les méthodes non linéaires : lorsque les données sont corrèles d'une manière non linéaire, les méthodes linéaires de la réduction de la dimension ne sont plus efficaces. Les méthodes non linéaires sont plus adaptées dans ce cas, leurs objectifs est de la transformation non linéaire qui permet de réduire la dimension tout en gardant un maximum d'information possible.
Dans la section suivante, nous présentons les méthodes de la réduction de la dimension les plus citées dans le domaine de l'indexation multidimensionnelle et la recherche des k plus proches voisins. Nous classifions ces méthodes en deux grandes catégories : les méthodes linéaires de la réduction de la dimension et les méthodes non linéaires.

3.2.1 Méthodes linéaires

Ces méthodes permettent de transposer avec une transformation linéaire les données de l'espace d'entrée de dimension d' dans un nouvel espace de dimension réduite d . La transformation linéaire associe à chaque axe du nouvel espace un indice sur la quantité d'information portée par la composante associée. Elle permet de détecter toutes les corrélations linéaires entre les données. La réduction de la dimension est effectuée par la projection des vecteurs originaux dans un sous espace vectoriel de dimension plus réduite. La dimension de l'espace est réduite en éliminant les composantes qui portent peu d'information, elle est généralement fixée a posteriori. Toutes les techniques linéaires reposent sur le même principe. Elles se distinguent par la transformation linéaire utilisée dans la projection. Dans la suite, nous présentons deux méthodes principales pour la réduction linéaire de la dimension: l'analyse en composante principale (ACP), et le positionnement multidimensionnel (MDS)

3.2.1.1 L'ACP : l'analyse en composantes principales

L'analyse en composantes principales (ACP) est la méthode linéaire de la réduction de la dimension la plus utilisée, elle trouve ses origines dans les travaux de Hotelling [Hot 33], Karhumen [Kar 47] et de Loève [Loe 48]. L'ACP détermine des axes de projections orthogonaux, qui maximisent la variance de la projection dans l'espace de caractéristiques, ce qui est équivalent à minimiser l'erreur quadratique moyenne de reconstruction. La variance des données représente généralement la quantité d'information portée par chacun des axes : plus la variance des données selon un axe est grande, plus l'information portée par celui-ci est importante. En fait, ceci consiste à effectuer une translation suivie d'une rotation du repère de l'espace pour privilégier les axes de variance maximale par rapport à un ensemble de données. Les axes où la variance des données est faible sont éliminés pour atteindre une réduction de la dimensionnalité avec une perte minimale de l'information

Formellement l'ACP se calcule de la manière suivante : Etant donné un ensemble de N observations centrées p_i, $i = 1,...N$ telle que :

$$p_i \in \Re^N , \sum_{i=1}^{N} p_i = 0 .$$

L'ACP diagonalise la matrice de covariance Cov avec :

$$Cov = \frac{1}{N} \sum_{j=1}^{N} p_j p_j^t .$$

Ceci se ramène à résoudre l'équation :

$$\lambda V = Cov.V \qquad (2.2)$$

Où $\lambda \geq 0$. Les λ_i sont les valeurs propres et $\nu \in \Re^N / \{0\}$. Donc $Cov.\nu = \frac{1}{N} \sum_{j=1}^{N} (p_j.\nu) p_j$.

L'équation (2.2) est équivalente à :

$$\lambda (p_j.\nu) = (p_j.Cov.\nu) \text{ pour tout } j = 1,...N . \quad (2.3)$$

Le calcul de L'ACP se ramène donc à un calcul de valeurs propres et vecteurs propres de la matrice de covariance des observations d'entrée.

3.2.1.2 MDS : Positionnement multidimensionnel

Dans de nombreux cas, on connaît les distances entre les points d'un ensemble de vecteurs (généralement on utilise une distance euclidienne), et on cherche à obtenir une représentation de faible dimension de ces vecteurs. Le positionnement multidimensionnel MDS (Multidimensional Scaling) [Has 01] permet de construire cette représentation. En projetant

dans un espace métrique, un ensemble de points tout en conservant le plus exactement possible l'ordre des proximités des points.

MDS (Multidimensional Scaling) permet de construire une configuration de N points dans espace χ de dimension d à partir des distances entre N objets (vecteurs par exemple). Dans ce contexte, on n'observe plus les points directement dans l'espace d'origine mais plutôt les $N(N-1)/2$ distances correspondant à ces points. MDS produit un positionnent des objets dans un espace F avec une dimension $d < d'$, de façon à minimiser l'erreur quadratique moyenne entre les distances fournies à l'algorithme et celles des points générés. Les distances fournies à MDS ne doivent pas nécessairement être Euclidiennes et peuvent par conséquent représenter des similarités de tout genre entre des objets.

Formellement MDS cherche une configuration de points $q_i = (1,...N)$ dans un espace de dimension plus réduite qui conserverait les distances entre les points initiaux p_i, autrement dit, on cherche les $q_i \in \Re^d$ avec $d < d'$ tels que : $dist(q_i, q_j) \approx \Delta_{i,j}$ avec $\Delta_{i,j} = dist(p_i, p_j)$.

Ceci revient à optimiser un critère : $J = \sum_{i=1}^{l} \sum_{j=1}^{l} (\Delta_{ij} - dist(q_i, q_j))^2$

Définition 1

Une matrice de distance $\Delta = [\Delta_{ij}] \in \Re^{l \times l}$ est dite euclidienne si pour des points $p_i, (i = 1,...l)$, on a $\Delta_{ij}^2 = (p_i - p_j)^t (p_i - p_j)$

La méthode du MDS est basée sur le théorème suivant.

Théorème 1

Soit la matrice $A = \lfloor a_{ij} \rfloor$ avec $a_{ij} = -\frac{1}{2}\Delta_{ij}^2$, et la matrice $M = ZAZ$ où Z est une matrice de centrage des données définie par $Z = I_l - \frac{1}{l} 1_l 1_l^t$ avec $1_l = (11...1)^t \in \Re^l$. La matrice $\Delta = [\Delta_{ij}]$ est dite euclidienne ssi la matrice M est semi-définie positive.

Dans ce cas, la matrice des produits scalaires $U = X_c X_c^t$ calculée avec les données centrées $X_c = ZX$. Si les données sont centrées, les distances Δ_{ij} peuvent s'exprimer en fonction des produits scalaires $U_{ij} = p_i^t p_j$ et inversement. Connaissant uniquement les distances Δ_{ij}, il est possible de trouver une configuration de points dans un espace Euclidien vérifiant ces distances en réalisant la décomposition spectrale de M.

L'algorithme MDS se présente de la manière suivante :

- Contruire à partir de la matrice Δ, la matrice $M = ZAZ$ avec : $A = [a_{ij}]$ et $a_{ij} = -1/2\Delta_{ij}^2$
- Réaliser la décomposition spéctrale de $M \in \Re^{l \times l}$

$$M = W \wedge W^T \text{ avec } W = [v_1...v_l] \in \Re^{l \times l} \text{ et } \wedge = diag(\lambda_1,...\lambda_l)$$

La matrice M a au plus d valeurs propres non nulles (en supposant que $l > d'$).

- D'après le théorème, on a $X_c = ZX \equiv W \wedge^{1/2}$. Pour obtenir une configuration de points dans un espace de dimension réduite d, il suffit de considérer alors :

$$Y = \vee \wedge_d^{1/2}$$

avec $V = [v_1,...v_d]$, d' et $\wedge_d = diag(\lambda_1,...\lambda_d)$

Sachant que structure de la matrice Y est $Y = [y_1...y_l]^t \in \Re^{l \times d}$

3.2.2 Méthodes non linéaires

Les algorithmes de la réduction de la dimension peuvent être généralisés afin de considérer les relations non linéaires entre des données. Ces méthodes reposent sur l'idée de transformer les

données de grande dimension dans un espace de dimension réduite tout en préservant les proximités entre les vecteurs et leurs correspondants dans l'espace projeté. Parmi les méthodes, les plus connues de la réduction non linéaire de la dimension nous trouvons : l'analyse en composante principale KPCA [Sch 98], la Locally Linear Embedding (LLE) [Row 00] et la méthode Isomap [Ten 00]. Dans la suite, nous présentons les deux dernières méthodes, le KPCA sera présenté dans le chapitre 3.

3.2.2.1 Isomap

Isomap [Ten 00] est une généralisation non linéaire de la méthode MDS, cette technique de réduction de dimension se base sur la connaissance d'une matrice de dissimilarités entre les paires de vecteurs de données. Elle exploite le fait que pour des points proches, la distance euclidienne est une bonne approximation de la distance géodésique sur la l'ensemble des données.

Définition

Une distance géodésique entre deux points est une courbe localement de longueur minimale.

Une géodésique désigne le chemin le plus cout, ou l'un des plus court chemins (s'il en existe plusieurs), entre deux points d'un espace pourvu d'une métrique.
Les données 3D peuvent être vues comme un ensemble où chacune peut être considérée localement appartenir à un espace Euclidien 2D. Un exemples d'une distance géodésique entre deux point p_1 et p_2 est illustré dans le figure 2.12

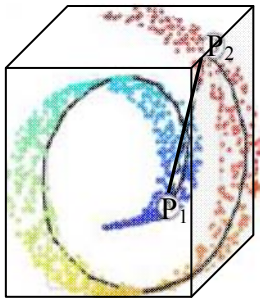


Fig 2.12. Exemple d'estimation d'une distance géodésique entre deux point p_1 et p_2

L'algorithme Isomap construit un graphe reliant chaque point à ses k plus proches voisins. Les longueurs des géodésique sont alors estimées en cherchant la longueur du plus court chemin entre deux points dans le graphe. La méthode de positionnement multidimensionnel (MDS) est appliquée aux distances obtenues afin d'obtenir un positionnement des points dans un espace de dimension réduite.
En résumé, Isomap consiste en trois étapes.

- Dans la première étape, l'algorithme détermine quels points sont voisins sur l'ensemble de vecteurs S en se basant sur les distances $dis(p_i, p_j)$ entre les paires de points i, j dans l'espace initiale χ afin de construire un graphe G approximant l'ensemble S.
- La seconde étape consiste essentiellement à approximer les longueurs des géodésiques $dist_G(p_i, p_j)$ entre toutes les paires de points i, j de l'ensemble S et de calculer le plus court chemin $dist_G(p_i, p_j)$.

o L'étape finale applique MDS à la matrice de distances DG afin de fournir un positionnement des vecteur de dimension d respectant le plus possible les longueurs des géodésiques.
Pour toutes les paires de points d'indice i, j la distance $dist_G(p_i, p_j)$ converge asymptotiquement vers la distance réelle entre ces points. La vitesse de convergence dépend de la courbure de l'ensemble des vecteurs et de leur densité. Si la densité des points n'est pas uniforme ou si la courbure de la variété est extrêmement forte, la convergence asymptotique est toujours garantie, mais la taille de l'échantillon nécessaire à l'estimation acceptable des longueurs des géodésiques peut être arbitrairement grande, donc impossible à obtenir en pratique. De plus, Isomap est non itérative, de complexité polynomiale et assez couteux en temps de calcul.

3.2.2.2 LLE

Le prolongement localement linéaire (locally linear embedding (LLE)) [Row 00], tente de résoudre le même problème que Isomap par une approche alternative. Chaque point est caractérisé par sa reconstruction par ses plus proches plus voisins. L'idée consiste à construire une projection vers un espace linéaire de faible dimension en préservant autant que possible le voisinage. Chaque point p_i est exprimé comme une combinaison linéaire de ses voisins, et l'image de p_i est construite dans l'espace de faible dimension en respectant cette combinaison linéaire.
Les différentes étapes de l'algorithme LLE sont :
o Trouver un certain nombre des plus proches voisin de chaque point p_i.
o Trouver pour chaque point p_i les poids w_{ij} sur les voisins p_j qui minimisent l'erreur de quadratique globale :

$$(p_i - \sum_j w_{i,j} p_j)^2$$

avec la contrainte $\sum_j w_{i,j} = 1 \ \forall i$. Cette contrainte assure l'invariance par translation des points et de ses voisins.
o Transformer la matrice de poids W en $M = (I - W)^T (I - W)$ symétrique.
Calculer k vecteurs propres v_j de plus petite valeur propre (excluant la plus petite, qui est 0, ce qui donne les coordonnées réduites des données $(v_{1i}, ..., v_{ki})$.
Notons que les vecteurs propres sont les coordonnées réduites q_i pour chaque vecteur p_i qui minimisent l'erreur :

$\sum_i \left\| q_i - \sum_j w_{i,j} q_j \right\|^2$ sous la contrainte $\sum_i q_{ij}^2 = 1$, (normalisation des cordonnées) et $\sum_i q_{ij} q_{ik} = 1_{j=k}$.

La figure qui suit résume les différentes étapes de cet algorithme.

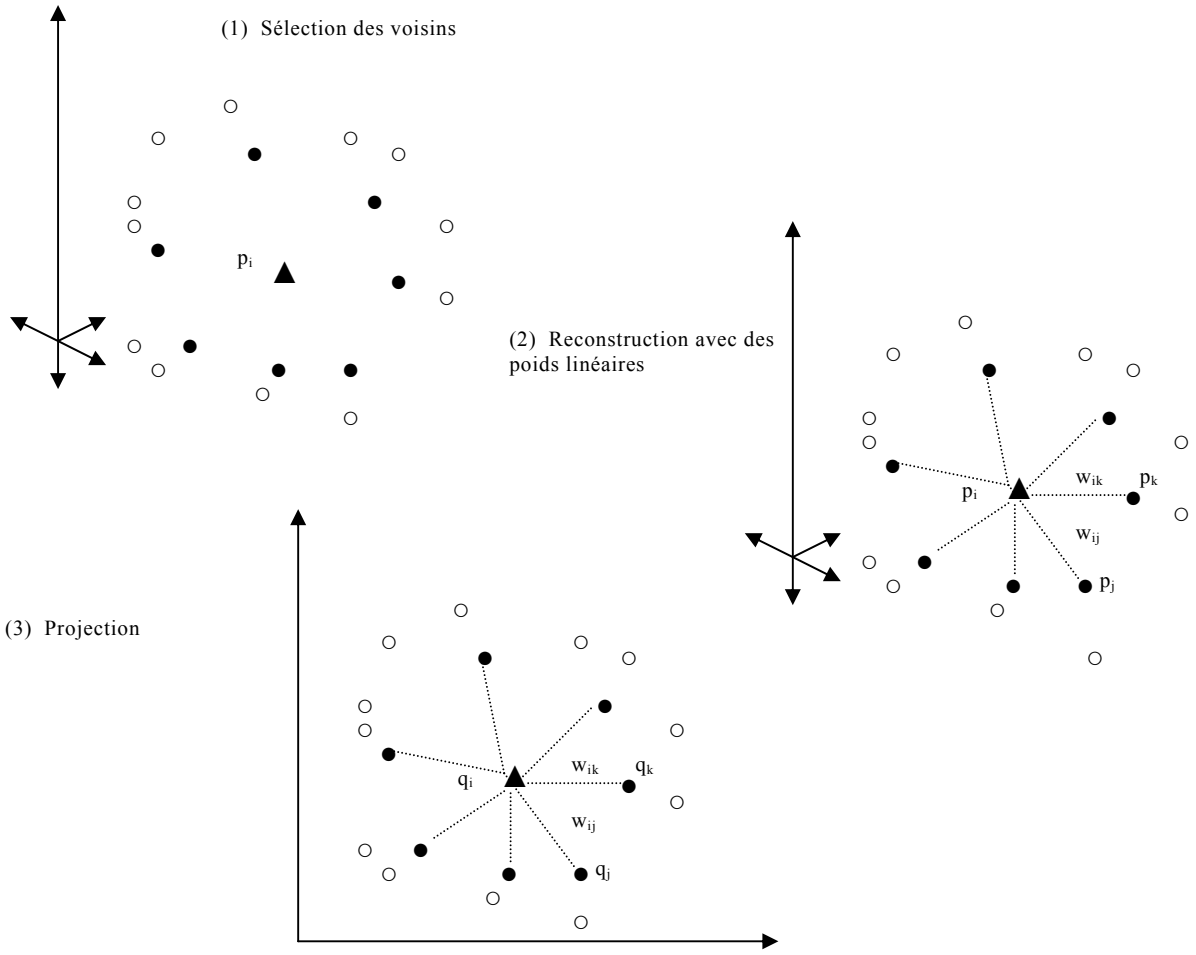


Fig.2.13.Principe de fonctionnement de l'algorithme LLE

3.2.3 Synthèse

Les méthodes de la réduction de la dimension ont pour objectif de réduire la dimension des données tout en préservant le maximum d'information. Ces méthodes sont utiles dans le domaine de l'indexation multidimensionnelle pour contourner les problèmes de la malédiction de la dimension exposés précédemment (paragraphe 3.1). Nous avons distingué deux grandes familles : la famille des méthodes linéaires et la famille des méthodes non linéaires de la réduction de la dimension.

Les méthodes linéaires transforment les données linéairement dans un espace de dimension réduite, elles considèrent les distances Euclidiennes entre les données. Si on suppose que les données supportent une dimension d plus faible que la dimension des données originales, l'ACP par exemple ne pourra trouver un système de coordonnées en d dimensions exact que si les dépendances entre les données sont linéaires. Dans le cas contraire, c-à-d si les dépendances entre les données sont non linéaires, l'ACP ne pourra pas créer une représentation en plus faible dimension respectant les distances entre les données.

Dans le même ordre d'idées, si les distances fournis à MDS sont les distances Euclidiennes, cet algorithme ne pourra pas non plus générer un positionnement des vecteurs qui présentent des corrélations non linéaires entre les différents attributs.

Généralement, les données observées dans la nature exhibant très souvent des caractéristiques hautement non linéaires. Pour cela, les méthodes linéaires de la réduction de la dimension ont

été généralisées pour supporter les données réelles, c-à-d dont les dépendances sont non linéaires.
L'idée principale des méthodes non linéaires de la réduction de la dimension est de trouver un nouvel espace de représentation de données de dimension réduite qui permet de préserver le plus fidèlement possible les distances entre les vecteurs. Contrairement aux méthodes linéaires de la réduction de la dimension, le choix de la dimension de l'espace réduit doit être effectué a priori. Ceci, constitue le principal inconvénient de ces méthodes. Le choix de la dimension optimale n'est pas évident et la fiabilité de la projection en dépend fortement.
Des méthodes plus efficaces ont été proposé dans la littérature, pour répondre aux problèmes de la malédiction de la dimension, elles sont présentées dans le paragraphe suivant.

4 Méthodes d'indexation multidimensionnelles basées sur l'approche approximation

Dans ce paragraphe, nous présentons les techniques basées sur l'approche *approximation* ou *filtrage*. Contrairement aux techniques conventionnelles d'indexation multidimensionnelles, ces techniques ont pour pas mal d'occasions prouvé de bonnes performances en grande dimension [Web 98][Ter 02]. L'idée principale repose sur la compression des vecteurs de la base et la gestion de deux ensembles de données lors de la recherche: un premier ensemble contenant des représentations compressées des données et un deuxième ensemble contenant les vecteurs réels de la base de données. Lors d'une interrogation, un premier parcours séquentiel du fichier d'approximations permet de sélectionner les approximations des données les plus pertinentes au vecteur requête au sens de la similarité. Cette étape n'est pas couteuse puisqu'elle est effectuée sur un ensemble de vecteurs de taille beaucoup plus réduite que l'ensemble des vecteurs original, elle permet d'élire rapidement les approximations des vecteurs pouvant figurer parmi les plus proches voisins. Ensuite, les vecteurs correspondant aux approximations sélectionnées dans la première phase sont visités séquentiellement pour calculer les $k - ppv$. Ainsi, le nombre d'opérations d'E/S est réduit et le coût de calcul CPU est petit par rapport à la recherche séquentielle qui, elle, analyse la totalité de la base.
Dans l'approche *approximation*, nous distinguons deux grandes familles de méthodes d'indexations: celles qui se basent sur des approximations locales et celles qui utilisent des approximations globales. Dans la suite, nous présentons les principales méthodes de chaque famille.

4.1 Approximation globale

Comme toutes les méthodes d'indexation multidimensionnelles, les méthodes basées sur l'approximation globale partitionnent l'espace de données en des formes géométriques hyper rectangulaires chacune est codée par un certain nombre de bits. L'approche *approximation* globale repose sur l'idée d'appliquer la même stratégie du partitionnement sur toutes les données de la base. La distribution des données, par conséquent, n'a aucune influence ni sur la stratégie de partitionnent ni sur le codage des formes géométriques issus du découpage. De plus, ces méthodes n'ont aucune structure particulière (arbre, graphe, etc.), les données et les approximations sont stockées dans des fichiers et l'accès s'effectue séquentiellement.
Il existe à notre connaissance uniquement deux méthodes principales basées sur *l'approximation* globales : la méthode VA-File et ses dérivées, et la méthode LPC-File. Ces méthodes sont détaillées dans les sous paragraphes qui suivent.

4.2.1 La famille VA-File

VA-File (Vector Approximation File) [Web 98] est une technique d'indexation qui améliore la recherche séquentielle, c'est la première technique efficace pour la recherche des $k-ppv$ dans l'espace de grande dimension.

Cette méthode est composée de trois étapes, une étape de compression de vecteurs pour laquelle des données sont codées par une chaine compressée de bits, une étape de filtrage qui permet de sélectionner les vecteurs candidats et une étape d'accès au données réelles qui permet de calculer les vecteurs les plus proches du vecteur requête.

-Etape de 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 $b=\sum_{i=1}^{d} b_i$ et un numéro de 0 à $2b-1$ selon chaque dimension. Les descripteurs de la base sont ainsi lus les uns après les autres, et l'approximation de chaque vecteur est déterminée par le numéro de la case qui le contient. Le fichier d'approximations est ainsi composé de paires (identifiant du vecteur, numéro de case). La figure 2.14 illustre un exemple de codage de vecteurs dans un espace de dimension deux.

Le nombre de cellules (2^b) produites lors de la quantification de l'espace de données est beaucoup plus grand que le nombre de vecteurs N. D'où, la majorité des cellules sera vide (Seules les informations liées aux cases contenant au moins un descripteur sont stockées, évitant ainsi le problème d'avoir à gérer un très grand nombre de cases vides). De plus, la probabilité pour que deux vecteurs appartiennent à une même cellule, et par la suite partagent la même approximation, est très faible [Web 98].

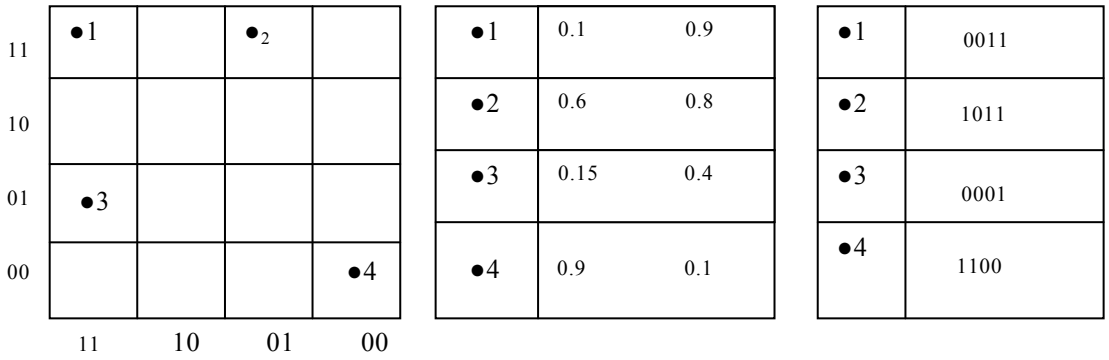


Fig. 2.14 Construction du VA-File

-Etape de filtrage :

Lors de la recherche des $k-ppv$, le fichier d'approximations est parcouru entièrement. Les distances maximales et minimales par rapport au vecteur requête peuvent être facilement déterminées en se basant sur les cellules rectangulaires représentées par l'approximation. Supposons que $D_{\min}$ est la plus petite distance maximale trouvée jusqu'à l'instant. Si une approximation est rencontrée telle que sa distance minimale est supérieure à $D_{\min}$, alors l'objet correspondant est éliminé puisqu'au moins un meilleur candidat existe. De la même manière, on peut définir une étape de filtrage lorsque $k-ppv$ doivent être retrouvés.

Un facteur critique de la performance est la sélectivité de cette étape de filtrage (% des vecteurs restants). Plus la quantité des approximations des vecteurs diminuent plus la performance augmente.

-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 distance 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 distance minimale rencontrée est supérieure à la plus poche distance jusqu'à cette étape, alors l'algorithme s'arrête. Un paramètre important de performance est le pourcentage des vecteurs visités (parmi les vecteurs restants) par rapport au nombre total des vecteurs de l'espace de données.

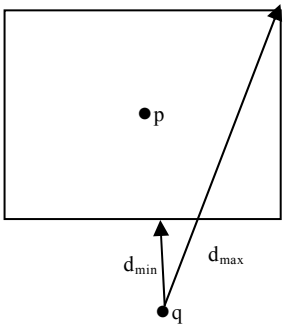


Fig. 2.15 Distance minimale est maximale par rapport au vecteur requête

Les performances du VA-File dépendent essentiellement de deux paramètres, la taille du fichier des approximations et du taux de filtrage. Si le fichier des approximations est très grand (dépasse la taille de la mémoire), les performances de la méthode VA-File se dégradent et deviennent plus mauvaise que la recherche séquentielle [Ams 01]. Le taux de filtrage quant à lui dépend du nombre de bits utilisés pour coder un vecteur. Le choix de ce dernier est critique : un grand nombre du bit de codage entraine une taille très grande du fichier d'approximation par contre produit une approximation fine et précise.

Plusieurs méthodes ont été proposées dans la littérature pour améliorer les performances du VA-File. Par exemple Cha et al. [Cha 02] remarquent que les performances du VA-File ne peuvent être améliorées que par une augmentation du nombre de bits utilisées lors du codage . Mais une telle augmentation peut entrainer un surcroit de la taille du fichier des approximations. Pour cela, Cha et al. [Cha 02] proposent le LPC-File où les informations codées sont enrichies par l'introduction des coordonnées polaires des vecteurs lors du calcul des approximations, cette techniques sera détaillée dans le paragraphe 4.2.2.

Rappelons que la méthode VA-File est basée sur le partitionnement de l'espace de données. Son efficacité diminue au passage à l'échelle, car le nombre de cellules crées durant le partitionnent croit exponentiellement en fonction de la dimension [Web 98], ceci rend cette méthode moins performante qu'une recherche séquentielle exhaustive en grande dimension. Pour résoudre ce problème Chen et al. [Cva 02] proposent C2-VA-File (Clustered Compact VA) qui permet d'éliminer les dimensions contenant les informations les moins importantes pour éviter de générer un grand fichier d'approximations pour des données de grande dimension et ainsi gagner au niveau temps de réponse durant la recherche. L'idée de base est de réduire les dimensions d'un vecteur dont les coordonnées sont inférieures à une valeur donnée et composer un nouveau système d'axe pour ce vecteur avec le reste des dimensions.

Par ailleurs, VA-file est particulièrement adaptée aux données uniformément distribuées dans l'espace. Or les données réelles sont souvent groupées en classes et leurs dimensions sont corrélées, de ce fait, Hakan et al. [Hak 00] proposent de transformer l'espace de donnés en utilisant la transformation de KLT (Karhunen-Loeve Transform) afin de réduire la corrélation des données pour les différentes dimensions. Un algorithme d'allocation non uniforme de bits et

un scalaire de quantification optimal sont appliqués indépendamment à chaque dimension pour construire le fichier d'approximation.
OVA-File [Lu 06] est une autre méthode qui améliore considérablement les performances du VA-File au passage à l'échelle et en grande dimension. En effet, un des inconvénients majeurs du VA-File est le parcours séquentiel exhaustif de tout le fichier d'approximation lors de la recherche, ce parcours est généralement très coûteux si le fichier est très grand et ne peut plus tenir en mémoire. Pour réduire l'ensemble des approximations à parcourir durant la recherche Lu et al. [Lu 06] ont proposé le OVA-File (Ordred Vector Approximation File), une structure d'index basée sur le groupement des vecteurs en fonction de leurs distances les un par rapport au autres. Cette structure permet d'éviter le parcours complet du fichier d'approximations en sélectionnant juste les régions les plus proches du vecteur requête. Selon OVA-File, les approximations des vecteurs sont ordonnées et ensuite partitionnées sur plusieurs fichiers appelés *slices*. Chaque *slices* étant caractérisé par le nombre d'approximation qu'il contient, son centre, et un rayon égal à la distance entre son centre et le vecteur le plus loin appartenant au slice. La structure du OVA-File est constituée de trois fichiers ; le fichier des données réelles, le fichier des approximations ordonnées, et le fichier contenant les sommaires des slices, c.à.d, le centre, le rayon, et le nombre d'approximation.
La recherche dans OVA-File se fait en deux étapes : premièrement les slices sont triés en fonction des distances de leurs centre par rapport au vecteur requête du plus proche au plus loin. Une recherche séquentielle est effectuée au sein de chaque slice dans l'ordre du tri. Deuxièmement, sur chaque slice visité une recherche des $k-ppv$ est effectuée séquentiellement. Soit S l'ensemble des vecteurs résultats et $S_{\max}$ le vecteur le plus loin appartenant à S, un slice n'est visité que si : $N_S \leq k$ ou $dist(q,O) \leq dist(q,S_{\max}) + \delta R \ \ \delta \in [0,1]$, telles que N_S et le nombre d'approximations, O le centre du slice, R son rayon et δ est un paramètre de qualité : plus sa valeur et grande, plus de slice seront visités et par conséquent plus on a une bonne qualité de la recherche.

4.2.2 LPC-File

LPC-File (Local Polar Coordinate File) [Gua 02a] est une méthode basée sur l'approche filtrage. Tout comme le VA-File, l'espace de données est partitionné en cellules rectangulaires, celles-ci sont utilisées pour générer l'approximation codée en bits pour chaque vecteur.
Pour améliorer le taux de filtrage du VA-File, Guang et al. [Gua 02a] proposent d'enrichir les approximations des vecteurs codés en introduisant les coordonnées polaires dans le calcul des approximations. Le codage des vecteurs ainsi que l'algorithme de recherche s'en trouvent ainsi modifiés par rapport à VA-File :

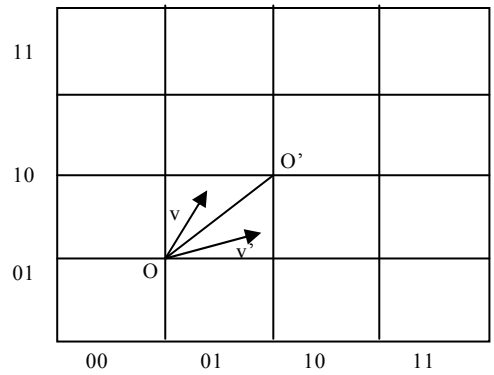


Fig. 2.16 Codage des vecteurs selon LPC-File

-Codage des vecteurs :

L'espace est tout d'abord partitionné en cases de taille identique et à chaque cellule est attribué un code binaire de la même manière que dans le VA-File. A chaque vecteur sont associées aussi ses coordonnées polaires (r et θ). Ces coordonnées sont calculées par rapport au coin inférieur gauche de la case et la diagonale reliant ce dernier et le coin supérieur droit. Le vecteur v de la figure 2.16 reçoit le code (cd, r, θ) où cd est le code de la case contenant le vecteur v : 0101 en l'occurrence, r la distance euclidienne entre v et o et θ l'angle entre la diagonale oo et le vecteur ov. Il est à noter que plusieurs vecteurs peuvent avoir le même code dans un espace de dimension 3, les vecteurs ayant le même code formant un cercle. Dans un espace de dimension supérieure à trois, les vecteurs ayant le même code forment une hyper sphère.

-Algorithme de recherche :

Tout comme le VA-File, en présence d'un vecteur requête, le fichier des approximations est d'abord parcouru pour effectuer une première phase de filtrage. Entre chaque approximation et le vecteur requête, une distance maximale $d_{\max}$ et une distance minimale $d_{\min}$ de la distance entre le vecteur de donnée et le vecteur requête sont calculées. Un vecteur est sélectionné si la borne inférieure de sa distance vers le vecteur requête est inférieure à la borne supérieure d'un autre vecteur.

Dans la deuxième phase, les vecteurs retenus sont lus dans l'ordre de leurs $d_{\min}$. La distance entre chaque vecteur lu et le vecteur requête est calculée et l'ensemble des résultats est mis à jour au fur et à mesure. La recherche s'arrête des que la distance $d_{\min}$ du prochain vecteur à lire est supérieure à la distance au plus proche voisin courant. Cette procédure est identique à celle utilisée par le VA-File. La nouveauté de cette technique réside dans le calcul des bornes $d_{\min}$ et $d_{\max}$ qui fait intervenir les coordonnées polaires.

Considérons un vecteur requête q et une approximation app d'un vecteur requête (voir figure 2.15), calculer les distances $d_{\min}$ et $d_{\max}$ entre le vecteur requête q et app revient à calculer la distance la plus proche et la distance la plus lointaine entre le vecteur requête q et l'hyper sphère ayant comme surface les points de code app. Si on note u le vecteur reliant o et app, et v le vecteur reliant o et q, alors $d_{\min}$ et $d_{\max}$ se calculent comme suit :

$$d_{\min} = \|u\|^2 + \|v\|^2 - 2\|u\|\|v\|\cos|\theta_1 - \theta_2|$$

$$d_{\max} = \|u\|^2 - \|v\|^2 - 2\|u\|\|v\|\cos(\theta_1 + \theta_2)$$

Où θ_1 et θ_2 sont les angles entre la diagonale de la case et les vecteurs u et v respectivement.

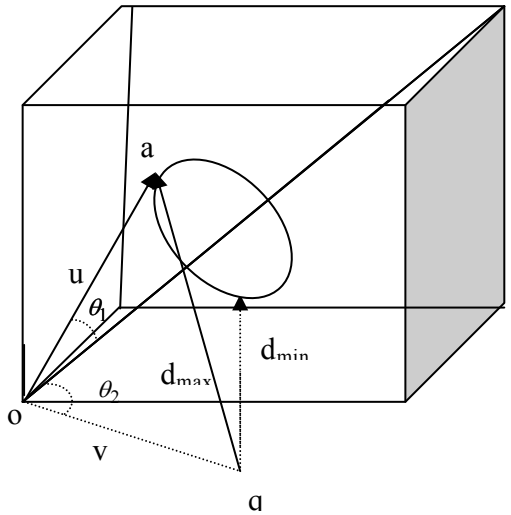


Fig. 2.17 Calcul de la distance minimale et de la distance maximale entre un vecteur requête et l'ensemble de vecteurs ayant la même approximation

Il est évident que le taux de filtrage de l'algorithme de recherche du LPC-File est meilleur que celui du VA-File car les approximations du LPC-File sont plus précises que celles du VA-File. Néanmoins, il est important de signaler que les calculs de $d_{\min}$ et $d_{\max}$ sont nettement plus coûteux dans le cas du LPC-File. Le choix du nombre de bits de codage reste un problème et les codes plus long que ceux du VA-File.

4.2 Approximation locale

Les méthodes basées sur des approximations locales reposent sur l'utilisation des formes géométriques (rectangles, sphères, ect.). Ces formes n'ont pas nécessairement les mêmes propriétés : taille différente, possibilité de chevauchement, codage différent, etc. Ces formes sont dans la plupart des cas indexés par une structure arborescente. Cette approche est plutôt adaptée aux données réelles dont la distribution est généralement non uniforme. Nous détaillons dans les sous paragraphes qui suivent les méthodes A-Tree, IQ-Tree, RA-Blocks, GC-Tree, Active Vertice , et PCR-Tree .

4.1.1 A-Tree

A-Tree [Syu 00] est une méthode d'indexation hiérarchique similaire au R*-Tree. La différence entre les deux méthodes est dans la stratégie d'approximation des MBRs (Minimum bounding rectangle). A-Tree introduit la notion des rectangles englobant virtuels (VBRs) qui repose sur l'idée d'approximer chaque MBR ou vecteur relativement par rapport à son MBR père. Cette approximation est plus précise que le R*-Tree et permet d'avoir un arbre avec une grande capacité de stockage.

En effet les VBRs sont des rectangles contenant une approximation des MBRS ou des données réelles selon le niveau de l'arbre à qui ils appartiennent. Ils se calculent de la manière suivante : Etant donné un rectangle multidimensionnel $Rect_P$ représenté par deux vecteurs multidimensionnels p et p' correspondant au côté bas gauche et haut droit tel que : $p=[p_1,...,p_d]$, $p'=[p'_1,...,p'_d]$ et $p_i \le p'_i$ pour $i=\{i=1,...d\}$ avec d est la dimension de l'espace multidimensionnel. Soit $Rect_q$ un rectangle dans P représenté par deux points q et q' telle que $p_i \le q_i \le q'_i \le p'_i$.

L'idée de l'approximation relative est de quantifier l'intervalle (q_i, q'_i) relativement par rapport à l'intervalle (p_i, p'_i) avec la fonction de quantification $Quant$ définie :

$$Quant(q_i) = p_i + \frac{(p'_i - p_i)h(q_i)}{nb}, \text{ avec } \quad h(q_i) = \begin{cases} nb-1 & si(q_i = p'_i) \\ \dfrac{q_i - a_i}{p'_i - p_i}.nb & \sin on \end{cases}$$

nb est un entier supérieur à 1. q'_i se calcul de la même manière que q_i.

Le rectangle virtuel englobant VBR est le rectangle $RectV_B$ représenté par les deux vecteurs v et v' telle que : $v=(Quant(q_1),...,Quant(q_d))$ et $v'=(Quant(q'_1),...,Quant(q'_d))$ sachant que Q est inclus dans $RectV_q$ et $RectV_q$ est inclus dans $Rect_P$. Soit U un vecteur de données contenu dans le rectangle P , ce vecteur de données est vu comme étant un rectangle de diagonal nulle, de ce fait son VBR peut être définie de la même manière que précédemment. La figure 2.18 montre un exemple de VBR.

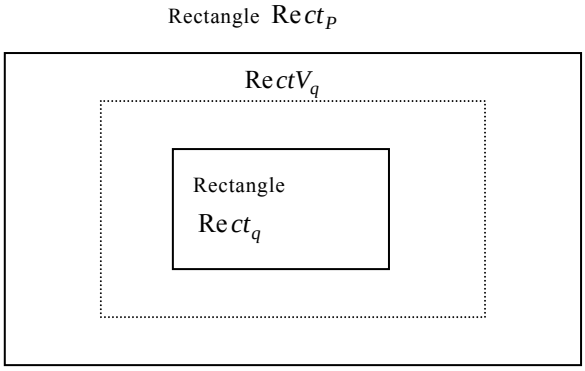


Fig. 2.18. Exemple de VBR

A-Tree est une structure arborescente, elle contient trois éléments : le nœud racine, les nœuds intermédiaires, et les feuilles. Le nœud racine représente l'espace de données, les nœuds intermédiaires contiennent les MBRs ainsi que les codes des VBRs relatifs aux MBRs des nœuds fils. Finalement les nœuds feuilles contiennent les vecteurs de données réels. Lors de la construction de la structure d'index, les codes des VBRs de chaque nœud de l'arbre sont calculés à partir des informations stockées dans les MBR des parents et des fils du même nœud. Lors de la recherche d'un vecteur requête, la position exacte du VBRs fils d'un nœud peut être calculée à partir du MBR père et des codes des VBRs stockés dans le nœud. Un exemple de cette structure est illustré dans la figure 2.19. Dans la figure 2.19, le rectangle Rect représente l'espace de données, M_1 et M_2 sont deux rectangle dans Rect. M_1 et le MBR de M_3 etM_4. M_3est le MBR du vecteur de données P_1 et P_2. Dans cette structure V_1, V_2, V_3, et V_4 sont les VBRs respectivement de M_1dans Rect, M_2 dans Rect, M_3 dans M_1, et M_4dans M_1. C_1 et C_2 sont les VBRs respectivement de P_1, P_2 et M_3

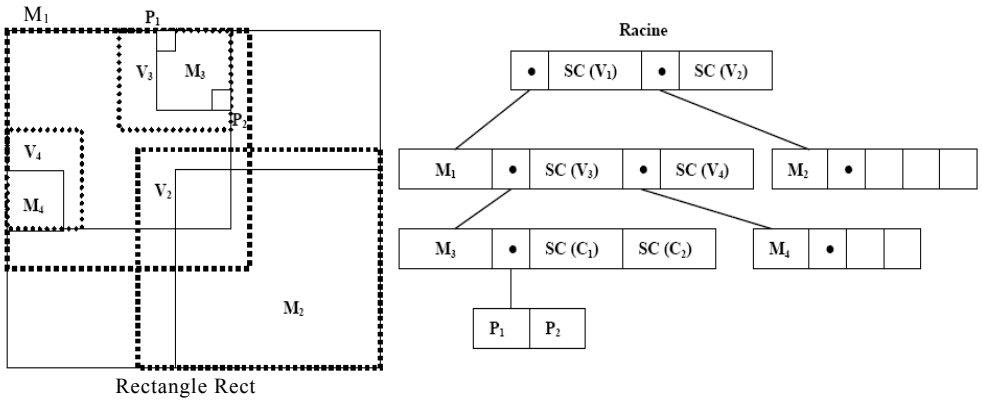


Fig.2.19 Structure d'index du A-Tree

L'algorithme de la recherche des *kppv* du A-Tree est une amélioration de l'algorithme présenté dans [Hja 95] . Ce dernier se base sur le calcul des distances MINDIST, c.à.d les distances minimales entre les MBRs d'un nœud et un vecteur requête donnée sont calculées et stockées avec un ordre décroissant dans une structure nommée queue de priorité. Les nœuds sont visités dans l'ordre décroissant de leurs distances MINDIST, et une liste des *kppv* est crée durant la recherche pour stocker les vecteurs les plus proches de la requête. Dans la queue de priorité les nœuds éliminés de la recherche sont ceux dont la distance par rapport au vecteur requête est supérieure au $k^{ème}$ vecteurs proche dans la liste. Dans A-Tree, la queue de priorité contient les

distances MINDIST et un pointeur sur le nœud, l'algorithme de recherche améliore le taux de filtrage en utilisant deux listes des *kppv*. La première appelée NNOVL, elle correspond aux vecteurs *kppv* usuels comme dans [Hja 95], elle contient les objets candidats stockées avec leurs distances par rapport au vecteur requête. La deuxième liste NNVL contient les distances maximales entre le vecteur requête et les VBRs des données réelles. En résumé deux conditions guident le processus de la recherche : les données sont retenues si la distance entre le vecteur requête et le VBRs contenant le vecteur réel est inférieur ou égale à la $k^{ème}$ distance dans NNOL. Si la distance entre le vecteur requête et le vecteur réel est inférieur au $k^{ème}$ vecteur dans NNOL ce vecteur est retenu et insérer dans NNOL. Le détail de l'algorithme de recherche est donné dans [Syu 00].
Les performances de A-Tree sont intéressantes par rapport à VA-File. En effet, à la différence du VA-File, qui se base sur des approximations exactes des vecteurs de données, A-Tree quant à lui utilise des approximations relative qui permettent de d'adapter à la distribution des données, d'un autre côté, l'erreur de l'approximation relative est beaucoup moins inférieure à celle d'une approximation exacte, ce qui rend cette technique bien adaptée aux distributions non uniforme des données.
Par ailleurs, les valeurs des approximations sont représentées d'une manière compacte diminuant ainsi la taille des entrées de chaque nœud, par conséquent, la capacité des nœuds augmente ce qui diminue par la suite le nombre des nœuds parcourus durant la recherche et ainsi le temps CPU.
Un autre avantage de la structure A-tree est chaque nœuds contient des informations relative à deux niveaux, celui du père et du fils. Cette configuration des nœuds est très utile pour l'efficacité des opérations de mise à jour.
A-Tree présente par contre une limitation au niveau de l'erreur d'approximation, cette dernière dégrade d'une manière considérable la qualité de la recherche particulièrement en grande dimension.

4.1.2 IQ-Tree

Berchtold et al. [Bbj 00] ont proposé le IQ-Tree (Independent Quantization Tree) un structure d'indexation constituée de trois niveaux d'indexation répartis sur trois fichiers différents: un premier niveau appelé "directory", répartit les données dans une structure hiérarchique de type R*-Tree [Bec 90], ce niveau contient les représentations exacte des MBRs. Le second niveau nommé "quantized data pages" conserve une approximation du style VA-File pour chaque rectangle minimum englobant (REM), enfin le dernier niveau correspond aux données réelles. Chaque MBR dans un "directory" contient un pointeur sur un "quantized data page" qui contient à son tour un pointeur sur une page de données réelle. Les "quantized data page" possède une taille maximale de blocks, c.à.d, étant donnée un nombre de bit de codage chaque "quantized data page" peut stocker un nombre maximale de points codés par une chaine de bit dans une seule "quantized data page".
Tout comme le VA-File, un nombre b_i de bits par dimension est utilisée pour approximer la localisation des points en divisant virtuellement les MBR suivant chaque dimension en 2^{bi} partitions de taille égale. Les points contenus dans une région sont représentés par une chaine de bits comme dans le VA-File. Le pas de quantification b_i (qui correspond au nombre de bit de codage) varie en fonction de la densité des points dans les MBRs. La structure de IQ-Tree est illustrée dans la figure 2.20.

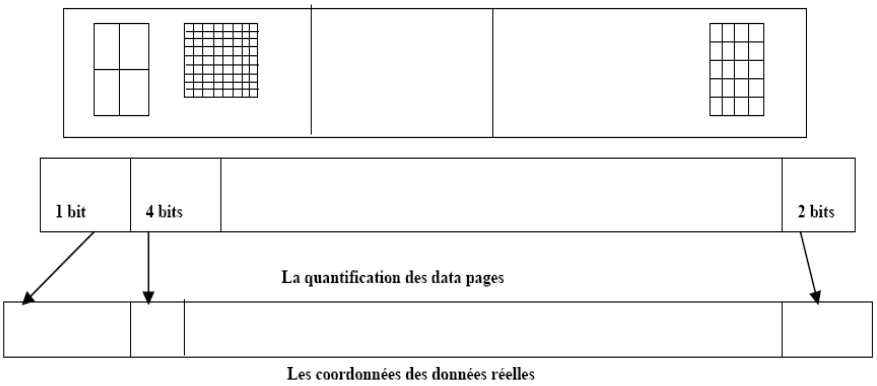


Fig.2.20 structure d'index d'IQ-Tree

La recherche dans IQ-Tree s'effectue de la même manière que dans le A-Tree. En effet, la condition générale pour guider le processus de la recherche est la suivante : si une région spécifique dans un certain niveau de l'arbre est non exclue de la recherche, le processus de la recherche suit le pointeur correspondant au MBR de la région courante et descend récursivement jusqu'à le niveau le plus bas. Cette condition peut être appliquée aux MBRs d'un ensemble de points ainsi aux approximations des MBRs relatif aux points. Cette condition permet également de limiter le nombre d'accès aux vecteurs réelles dans le sens où ceux-ci sont parcourus uniquement que s'ils sont les vrai réponses à la requête ou si la requête ne peut pas être évalué à la l'aide de la représentation compressée.
Etant donnée un requête, les approximations des MBRs sont insérées dans la queue de priorité comme dans le A-Tree. Les vecteurs réels correspondant à ces approximations sont parcourus uniquement dans le cas où leurs approximations deviennent des pages de la queue de priorité, c.à.d que leurs approximations possèdent les distances minimales $d_{\min}$ par rapport à un vecteur requête q donnée parmi tout les autres MBRs de la que de priorité. S'il existe un point telle que sa distance par rapport au vecteur requête q est inférieure à la plus petite $d_{\max}$ dans la queue de priorité, les coordonnées réelles de ce point sont examinées pour calculer la distance réelle. Dans le cas contraire, il existe aucun point qui vérifie la condition précédant, le point p est sélectionné car il constitue le voisin le plus proche du vecteur requête. Il n'existe en effet, aucune stratégie raisonnable qui permet d'éliminer ou de retarder la sélection du point p si son approximation est une page de la queue de priorité, puisque la vrai distance entre p et q est la plus petite possible.
IQ-Tree a de bonnes performances par rapport au VA-File : Le nombre d'accès aux données et le temps de recherche sont considérablement réduits. Par contre lorsque la dimension augmente la structure arborescente souffre des problèmes de la malédiction de la dimension, sa conception reste complexe par rapport aux autres structures.

4.1.3 GC-Tree

Le GC-Tree (Grid Cell Tree) [Gua 02b] est une technique qui combine les avantages de l'approche approximation des vecteurs permettant d'accéder à un nombre réduits de vecteurs réelles et les avantages des indexes multidimensionnels permettent de regrouper et d'organier les données dynamiquement dans l'index. GC-Tree repose sur une stratégie basée sur la densité pour le partitionnement de l'espace de données

-Partitionnement de l'espace basé sur la densité :

Pour approximer la densité des points, le GC-Tree partitionne l'espace de données en des cellules hyper carrées de même volume, ceci est effectué en divisant chaque dimension en des intervalles réguliers de même taille ce qui permet de produire par conséquent de cellules de même volume. Le nombre de vecteur dans chaque cellule est utilisé pour approximer la densité de la cellule. Pour une base statique, la densité d'une cellule est définie comme étant le quotient du nombre de points dans la cellule sur le nombre maximale de vecteurs qui peuvent être logés dans une page disque. Par contre, dans le cas d'une base de données dynamique, GC-Tree définit la densité d'une cellule comme étant le nombre de vecteur dans la cellule sur la capacité de la page disque sachant qu'un sous espace de l'espace de données correspond à un nœud dans l'arbre GC et il est physiquement stocké dans une page disque.

Une cellule est dense si sa densité est supérieure ou égale à un seuil τ. Sinon elle est dite éparse. Le partitionnent de l'espace de données basée sur la densité consiste en trois étapes : (i) L'espace de données est partitionné en se basant sur une fonction de densité identifiant les régions denses et les régions éparses. (ii) Le partitionnement est concentré sur les régions denses. (iii) Les vecteurs des régions éparses sont traités comme s'ils se trouvent dans une seule région. La figure 2.21 montre un exemple de partitionnent de l'espace de données basé sur la densité dans un espace à deux dimensions. Dans cet exemple, la capacité des cellules est fixée à 4. Si la cellule contient au maximum 3 points elle est appelée "cluster", sinon elle est nommée "outliers". La figure 2.21 montre le résultat de trois niveau de partitionnent selon GC-Tree. Lors du premier partitionnent les point 1, 2, 3, 4, 5 et 6 sont identifier en tant que "outliers", donc le "cluster" correspondant est subdivisé. Lors du second partitionnent, les points 7, 8 et 9 sont identifiés comme étant des "outliers" et les points 10, 11, et 12 sont également identifiés comme étant des "outliers" lors de du 3eme niveau de partitionnent. Finalement, nous obtenons, un seul "cluster" dans 3eme niveau, et trois "outliers" dans le niveau 2, 1 et 0.

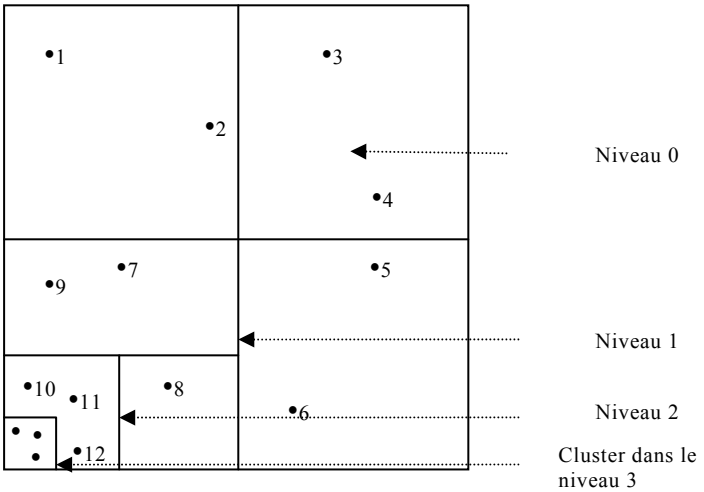


Fig. 2.21 Partitionnement de l'espace selon GC-Tree

Structure d'index

Le GC-Tree est organisé sous forme d'un arbre dynamique où les nœuds internes de l'arbre contiennent les cellules correspondant aux sous espaces et les feuilles contiennent les données réelles. L'entrée de chaque feuille de l'arbre contient l'approximation du vecteur réelle, cette approximation est calculée suivant la méthode LPC-File (voir paragraphe 4.2.2) et un pointeur vers un nœud de données dans lesquelles les données réelles sont stockées. L'entrée d'un nœud interne de l'arbre contient la cellule correspondant au sous espace couvrant les données de cette

cellule ainsi qu'un pointeur vers le niveau inférieur. Un exemple de construction de la structure d'index du GC-Tree est illustré dans la figure 2.22. La figure 2.22a représente un espace de données de deux dimensions divisé en 16 cellules avec un nombre de bits de codage fixé à 2. La densité maximale des cellules est fixée à 3/(capacité des nœuds). La racine contient trois entrées. La cellule (00,11) du second niveau représenté dans la figure 2.22a correspondant au nœud C_2 forme une partition fine du cluster correspondant au nœud C_4 et un "outlier". Le nœud C_4 à son tour forme une partition fine dans le "cluster" correspondant au nœud C_5 et un "outlier". La construction de l'index se fait par insertion successive des données réelles. La recherche des $k-ppv$ s'effectue en deux étapes : une première étape de filtrage et une seconde étape d'accès aux données réelles. L'algorithme de recherche utilise trois structures globales : deux listes bl_list et cand_list et un tableau k_element contenant les k vecteurs les plus proches. La liste bl_list contient les branches des nœuds internes de l'arbre avec leurs distances minimales et maximales par rapport au vecteur requête. La liste cand_list contient la liste des nœuds candidats, c-à-d ceux susceptible de contenir les vecteurs les plus proche à la requête, cette liste contient également les distances minimales et maximales qui séparent chaque candidats et le vecteur requête. Ces deux structures (bl_list et cand_list) sont calculées en utilisant l'algorithme min heap [Min 95]. Lors de la recherche si une approximation est rencontrée telle que sa distance minimale est supérieure à la plus petite distance maximale, le vecteur correspondant est éliminé de la recherche.

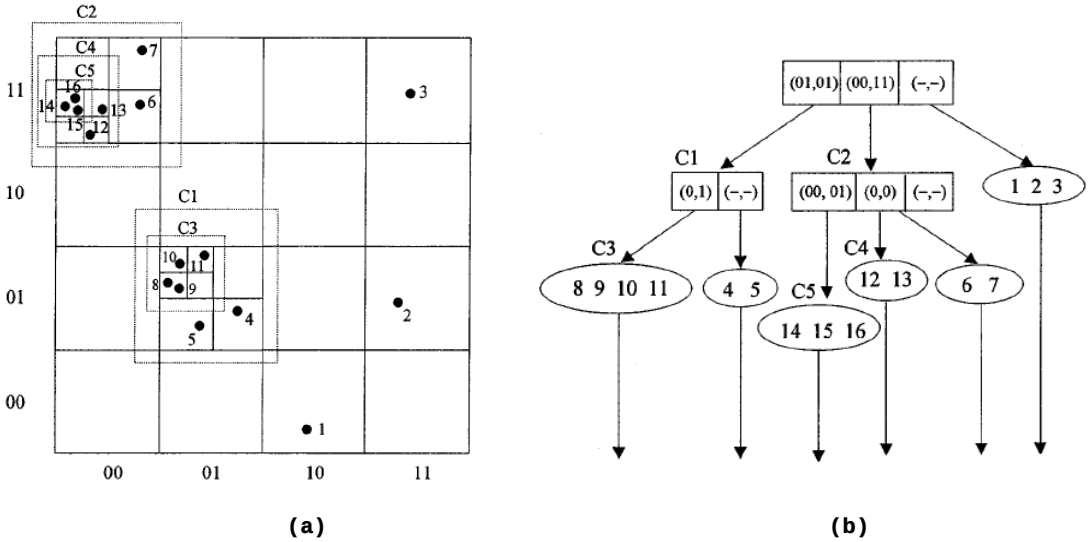


Fig. 2.22. Construction de l'index de l'arbre GC. (a)partitionnent de l'espace de données. (b) la structure d'index correspondante [Gua 02b]

GC-Tree améliore les performances de la majorité des techniques d'indexation basées sur l'approximation à savoir : IQ-Tree, LCP-File, VA-File et la recherche séquentielle. Par contre, cette méthode souffre des problèmes de la malédiction de la dimension. En effet, l'espace de données est partitionné suivant la méthode KD-Tree en des hyper cubes de petits volume. Or en grande dimension, les distances entre vecteurs ont tendance à converger ceci risque de regrouper tout les vecteurs dans le même hyper cubes ce qui diminue par conséquence la performance de l'index ainsi que celle de la recherche.

4.1.4 RA-Blocks

RA-Blocks [Ter 02] est une technique basée sur l'approximation des régions. L'espace de données est tout d'abord partitionné en cellules hyper rectangle de taille identique et à chaque cellule correspondant à un vecteur est attribué un code comme dans le VA-File.
L'espace de donnée est partitionné en des régions selon l'algorithme KDB-Tree, l'algorithme de partitionnent est guidé d'une part par les valeurs de quantification issues du découpage de l'espace de données en cellules et d'autre part par la capacité des régions préalablement fixé au cours de l'indexation. Les régions résultantes du découpage de l'espace de données sont codées par les deux cellules correspondant la cellule du bas à gauche et en haut à droite.
L'algorithme de la recherche des $k-ppv$ est composé de deux étapes : une étape de *filtrage* dans laquelle les régions candidates sont sélectionnées en se basant sur leurs distances minimales et maximales par rapport au vecteur requête. Dans la deuxième étape dite *accès au vecteurs*, les régions sélectionnées seront accédées et une recherche séquentielle est exécutée afin de calculer les vecteurs les plus proches au vecteur requête (cette méthode est présenté en détail dans le paragraphe 1.4 du chapitre 3)
Comme le VA-File RA-Blocks n'utilise aucune structure arborescente, elle repose sur la gestion de deux ensembles de fichiers, un fichier contenant les approximations des régions et un autre contenant les données réelles. À la différence des méthodes qui se basent sur une structure arborescente, L'insertion de nouveau vecteurs ne nésite aucune réorganisation de l'espace de données, sa structure est simple et dynamique.
D'un autre côté, RA-Blocks combine les avantages des méthodes basées sur une structure arborescente permettant de regrouper localement les données et les organiser en index, ainsi que les avantages des méthodes basées sur l'approximation globale permettant de réduire le temps de calcul des distances et d'accéder à un nombre réduits de vecteurs réelles.
La structure du RA-Blocks par contre, semble plus adaptée aux données de distribution uniforme, elle souffre aussi et comme la majorité des techniques d'indexation basées sur l'approximation des problèmes de la malédiction de la dimension.
Contrairement au VA-File, le nombre de bit de codage b_i n'a pas l'influence significative sur la taille du fichier d'approximations et les performances de la méthode. Un grand nombre de bits de codage produit une approximation précise par contre elle augmente légèrement la taille du fichier d'approximation puisque celui-ci contient uniquement les approximations des régions représentés par deux cellules.

4.1.5 Active Vertice

Balko et al. proposent une méthode appelée AV (Active Vertice) [Bal 04] pour l'indexation des données multidimensionnelle. Cette méthode reprend les techniques basées sur les arbres quaternaires [Sam 84] pour le partitionnement de l'espace, elle construit les approximations des vecteurs dans l'espace et associe à ces approximations des codes binaires B_i comme dans le VA-File. Cependant, les cellules régulières dans VA-File sont remplacées dans la méthode AV par un arbre équilibré pour refléter la distribution des vecteurs dans l'espace. Chaque nœud C_i de l'arbre détermine une région de l'espace décrite par un point représentatif, le nœud C_i est délimité par l'intersection d'un ellipsoïde et d'un hyper-rectangle. Le rayon maximum de l'ellipsoïde est un paramètre fixé au début de l'indexation. La taille de l'hyper-rectangle dépend du niveau du nœud délimité par l'hyper-rectangle. Elle est réduite de moitié lorsqu'on passe d'un nœud donnée aux nœuds de niveau directement inférieur en cas de distribution uniforme

des données. Le code binaire B_i est entièrement défini par le chemin parcouru pour aller de la racine de l'arbre au nœud contenant le vecteur, sa longueur est donc variable. La délimitation des régions par l'intersection d'une ellipsoïde et d'un hyper-rectangle dans la méthode AV permet d'obtenir, des approximations plus rigoureuses de la distance entre un vecteur requête quelconque et un vecteur donnée de l'espace. Un exemple d'approximation selon la méthode AV est illustré dans la figure 2.23.

La recherche des $k-ppv$ est similaire à celle du VA-File, elle s'effectue en deux étapes, dans la première étape le fichier des approximations est parcouru séquentiellement et la listes des approximations candidates est déterminée en se basant sur les distances minimales $d_{\min}$ et maximales $d_{\max}$ de vecteur requête par rapport à l'approximation. Ces distances sont calculées de la manière suivante (voir figure 2.23) :

$d_{\max}(q,B_i) = \max\{d_{\max,s}(q,B_i), d_{\max,c}(q,B_i)\}$, avec $d_{\max,s}(q,B_i) = \max\{\|q-p_i\|_2 - r, 0\}$

$d_{\min}(q,B_i) = \min\{d_{\min,s}(q,B_i), d_{\min,c}(q,B_i)\}$, avec $d_{\min,s}(q,B_i) = \|q-p_i\|_2 + r$

Et,

$LB_c(q,B_i) = \sqrt{\sum_{j=1}^{d} (\max\left\{(|q[j]-c_i[j]| - \frac{w_i}{2}), 0\right\})^2}$, w_i est la longueur d'arrête de l'hyper cube

$UB_c(q,B_i) = \sqrt{\sum_{j=1}^{d} (|q[j]-c_i[j]| - \frac{w_i}{2})^2}$

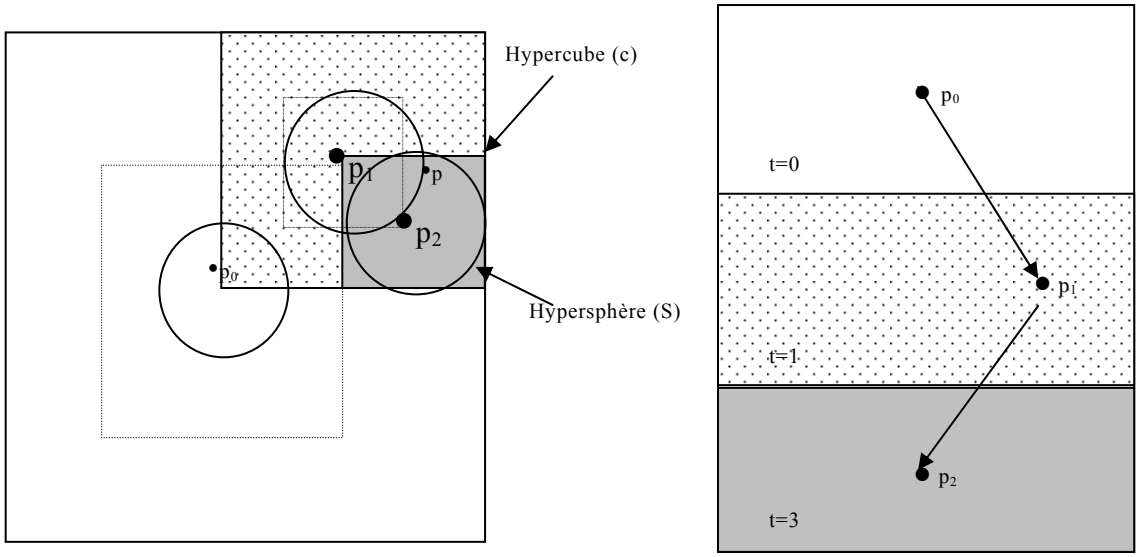


Fig. 2.23 le principe de l'approximation de la méthode AV

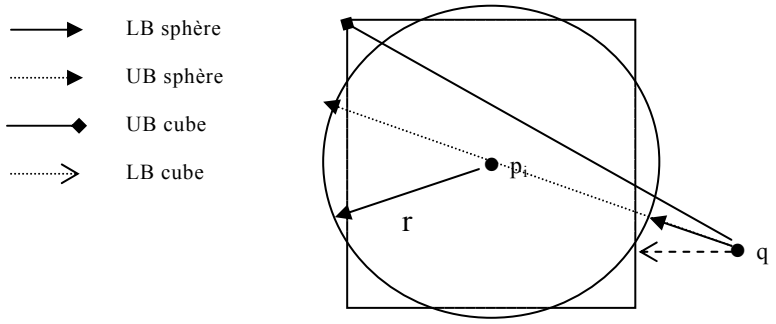


Fig 2.24. Exemple de calcul de la distance minimal et maximal du vecteur requête par rapport à l'approximation suivant la méthode AV

Notons que la méthode AV présente de bonne performance au passage à l'échelle et en grande dimension en comparaison avec VA-File et LPC-File, par contre cette technique exige un paramétrage assez complexe pour adapter cette méthodes aux applications réelles ce qui rend la méthode difficile à appliquer. D'un autre côté, l'utilisation de l'arbre quad-tree pour le partitionnement de l'espace de données et pour l'indexation des points de référence rend cette méthode inadaptée aux données uniformes.

4.1.6 PCR-Tree

Plus récemment, l'arbre PCR (Principal Component R-Tree) est introduit par Cui et al. [Cui 07] pour répondre aux problèmes de la malédiction de la dimension dont soufflement la majorité des méthodes d'indexation multidimensionnelles ainsi que le problème de la distribution non uniforme des données réelles. Cette méthode a été proposée pour supporter les données non uniformes dans un espace de vecteurs multidimensionnel. Elle combine l'approche filtrage ainsi que la compression des données pour accélérer la recherche séquentielle d'une requête donnée.
L'arbre PCR a une structure à trois niveaux : le premier niveau correspond aux nœuds de l'arbre contenant les rectangles englobant (REMs) et les approximations relatifs aux vecteurs. Le deuxième niveau correspond aux feuilles de l'arbre contenant les approximations des vecteurs réelles dans l'espace de dimension réduite. Le troisième niveau contient les vecteurs réels exprimés dans l'espace de grande dimension. La différence entre PCR-Tree et R-Tree se manifeste principalement dans la stratégie du groupement de vecteurs sous forme de rectangles. En effet, dans R-Tree les REMs sont construits en se basant sur les coordonnées réelles des vecteurs alors que dans l'arbre PCR, les REMs sont crées à partir des approximations des vecteurs codés en bits (voir figure 2.25). Notons que le principal inconvénient de la méthode R-Tree est le chevauchement entre les REMs produisant les dégradations rapides dans les performances en grande dimension. La structure de l'arbre PCR est construite par contre dans un espace de dimension réduite pour contourner les problèmes de la malédiction de la dimension.
La construction de la structure d'index de l'arbre PCR est résumée en 3 étapes :
Etape 1 : les données sont projetées dans un espace de dimension réduite en utilisant la transformé de KLT
Etape 2 : calculer les approximations des vecteurs projetés dans l'espace KLT en utilisant la méthode présentée dans VA^{+}-File [Hak 00]
Etape 3 : choisir le nombre des composantes principales (c.à.d la dimension de l'espace transposé)
Etape 4 : Construire l'arbre PCR selon les approximations des vecteurs dans l'espace de dimension réduite
A la différence du VA-File, la recherche dans l'arbre PCR s'effectue en une seule étape, elle se base sur une technique similaire à celle utilisée dans la méthode R-Tree. La condition générale pour guider la recherche est la suivante : si le REM d'un certain niveau de l'arbre n'est pas exclu de la recherche, l'algorithme de recherche suit le pointeur associé au REM et descend un niveau de l'arbre. Et si une approximation d'un vecteur n'est pas exclue de la recherche, l'algorithme de recherche accède aux vecteurs réels correspondant. Un tableau de $d_{\max}$ est utilisé pour stocker les k plus petites distance trouvées lors de la recherche.
Durant la recherche, les distances minimales $d_{\min}$ entre es candidats et le vecteur requête sont calculés. Si une approximation est rencontrée telle que sa distance minimale est plus grande que

la plus petite distance trouvée jusqu'à maintenant, le vecteur correspondant peut être éliminé de la recherche. La distance minimale entre le REM et le vecteur requête est aussi calculée. Si la distance minimale entre le REM et le vecteur requête est plus grande que la plus petite distance trouvée, tous les vecteurs englobés par le REM sont éliminés de la recherche.
L'arbre PCR présente de bonnes performances en terme temps de réponse par rapport à GC-Tree et VA^{+}-File, c'est une technique adaptée au données non uniforme de grande dimensions par contre sa structure est non dynamique. Le changement de la distribution des données nécessite la reconstruction de la structure d'index dans l'espace de dimension réduite.

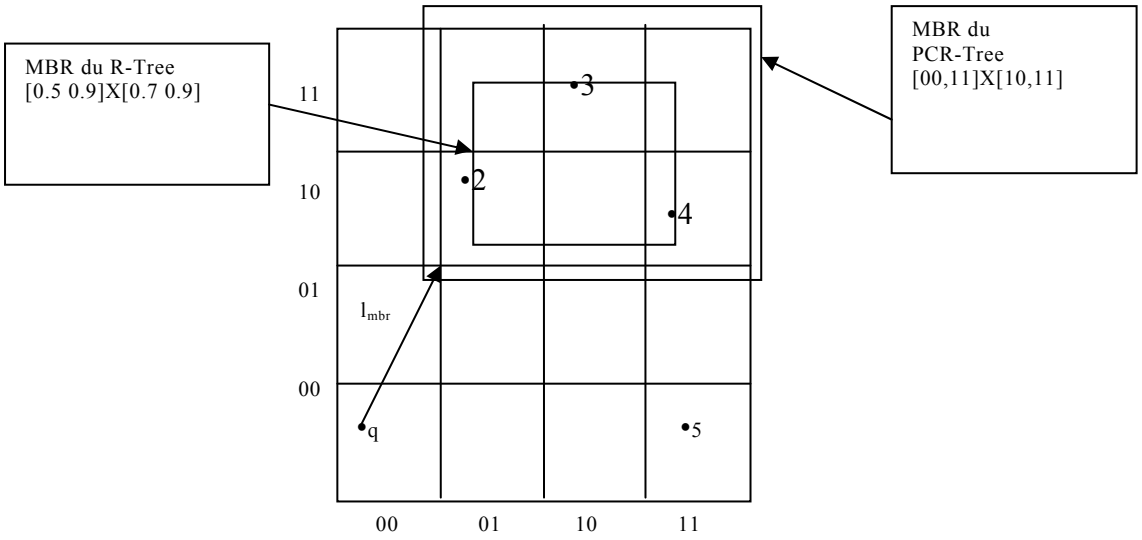


Fig. 2.25. MBRs de l'arbre PCR et du R-Tree

4.3 Synthèse

Ce paragraphe présente une synthèse sur les caractéristiques, les avantages et inconvénients des techniques d'indexation basées sur l'approximation pour la recherche des plus proches voisins présentées précédemment. Les tableaux 2.3, 2.4, 2.5, et 2.6 récapitulent les principales caractéristiques, avantages et inconvénients des méthodes présentées précédemment.
Nous avons distingué deux grandes familles : la famille des méthodes basées sur l'approximation globale (VA-File, VA+-File, C2-VA-File et LPC-File) et les familles des méthodes basées sur l'approximation locale. La première famille de méthodes se caractérise d'une part par le fait d'appliquer uniformément la même stratégie du découpage de l'espace de données et éventuellement le codage des formes géométriques (cellules) issu de ce découpage. En effet, la stratégie du découpage est guidé uniquement par le nombre de bits de codage fixée préalablement et aucune condition sur la distribution des données n'ai prise en considération lors de la création de la structure d'index. D'autre part ces méthodes ont la particularité d'utiliser uniquement deux fichiers pour la gestion des données : un premier fichier contenant les approximations des vecteurs et un deuxième contenant les données réelles. Ainsi aucune structure hiérarchique n'est adoptée par ces méthodes, ceci présente un premier point fort de ces techniques. En effet, la structure hiérarchique des données notamment la structure arborescente, exige certaines propriétés (équilibre, taille des nœuds, etc.). La prise en compte de ces propriétés rend les procédures d'insertion, suppression des données difficile, ces structures sont généralement non dynamiques et non adaptées à la mise à jour. Un deuxième point fort de ces techniques est le fait de ne plus utiliser des formes géométriques englobantes. Les méthodes

basées sur l'approximation globale procèdent par un codage des vecteurs de données, l'approximation est par conséquent exacte. Cette famille par contre est adaptée surtout aux données de distribution uniforme puisque les données sont indexées est codées de la même manière quelques soit leurs distribution.
La deuxième famille de méthode est basée sur l'approximation locale, son principe étant de partitionner et coder les données localement en se basant sur leurs distributions. Elle est basée sur l'utilisation des formes géométriques qui n'ont pas nécessairement les mêmes propriétés : taille différente possibilité de chevauchement, etc. Ces formes sont indexées par une structure arborescente, la structure arborescente, permet une certaine flexibilité lors de l'indexation des différentes régions de l'espace de données dans le sens ou chaque région peut être représentée et codée différemment. Par contre l'utilisation de cette structure pose des problèmes aux passages à l'échelle par exemple, la taille de l'arbre augmente au fur et à mesure que la taille de données évolue. Par ailleurs le fait d'utiliser les formes géométriques avec des propriétés différentes génère d'une part des approximations précises mais par contre pose de nombreux problèmes en grande dimension (possibilité d'avoir des régions vides, possibilité de chevauchent, problème d'optimisation de l'espace mémoire…). Cette famille de méthodes a l'avantage de pouvoir manipuler des données réelles (de distribution non uniforme) mais par contre elle reste limitée pour les données à grande échelle (grande dimension, grand volume de données).
Pour comparer les caractéristiques des différentes méthodes présentées précédemment, nous avons choisi des critères que nous avons jugé primordiales pour la comparaison des différentes structures d'index: la structure de méthode, sa capacité de supporter les grandes dimensions, le volume des données, l'adaptabilité par rapport aux données réelles, et la sélectivité.
Notons que ces critères ne sont pas exhaustifs. Nous n'avons pas inclus par exemple le critère temps de réponse, celui-ci se déduit de la sélectivité et de la structure de l'index.
Le passage en revue des principales méthodes basées sur l'approximation montre clairement que ces techniques ont de bonnes performances au passage à l'échelle et au grande dimension en comparaison avec les techniques conventionnelles d'indexation multidimensionnelles, leurs performances sont liées cependant selon la méthode à des paramètres comme le nombre de bits de codage, la taille du fichier des approximations ou la méthode du partitionnent de l'espace de données et parfois à la distribution des données. Ces méthodes restent néanmoins les plus utilisées et les plus efficaces à ce jour pour l'indexation des grandes bases de données réelles où la dimension dépasse 100 et la taille de la base de données est considérable. En se basant sur les différents tableaux comparatifs il est facile de montrer qu'aucune des méthodes citées précédemment ne peut être utilisée efficacement dans le domaine de la recherche d'images basée sur le contenu, ou nous avons un grand volume de vecteurs multidimensionnels de nature hétérogène (distribution non uniforme) et de grande dimension. Les méthodes basées sur l'approximation globale sont généralement adaptées aux données de distribution uniforme excepté la méthode C2-VA-File. Les performances de cette méthode sont par contre limitées aux grandes dimensions et au passage à l'échelle. Les méthodes basées sur l'approximation locale quant à elles sont adaptées aux données réelles. En revanche, certaines ont des structures d'index complexe (IQ-Tree), d'autres souffrent considérablement des problèmes de la malédiction de la dimension (A-Tree, PCR-Tree, GC-Tree, Active Vertice) .
Sur la base de cette étude comparative, nous avons remarqué que RA-Blocks possède des propriétés intéressantes en comparaison avec les autres méthodes. A la différence des techniques présentées dans ce chapitre, cette méthode combine les avantages de la famille des méthodes basées sur l'approximation globale en utilisant uniquement des fichiers pour la

gestion des données, et les avantages de la famille des méthodes basées sur l'approximation locale en utilisant des formes géométriques englobantes (des régions) pour regrouper les données. RA-Blocks possède les meilleurs caractéristiques (meilleur taux de sélectivité, structure d'index adaptée au grand volume de données, etc.). Comme toutes les méthodes d'indexation multidimensionnelle, RA-Blocks souffrent des problèmes de la malédiction de la dimension. Sa mise en place dans un système de recherche par contenu nécessite des améliorations. Le chapitre 3 présente en détail la méthode RA-Blocks ainsi que les améliorations apportées à cette méthode pour répondre aux problèmes de la malédiction de la dimension. Le traitement des problématiques liées à l'intégration de cette méthode dans un système de recherche d'images par le contenu sera présenté dans le chapitre 4.

Méthode	***Avantages***	***Inconvénients***
A-Tree	* Adaptée aux données réelles *Flexible à la mise à jour *Grande capacité de stockage * Améliore les performances du VA-File *Petite taille des entrées d'index (arbre compact)	*Chevauchement entre les MBRs*Dégradation des performances aux passage à l'echelle & grande dimension * Erreur d'approximaion considérable en grande dimension
IQ-Tree	* Bonne selectivité des région * Adaptée aux données réelles * Améliore les performances du VA-File	* Structure d'index complexe * Non performante pour des données volumineux * Non adaptée au données uniformes *Dégradation des performances en grande dimension * Dégradation des performances pour un grand volume de données.
GC-Tree	* Adaptée aux données réelles * Améliore les performance du VA-File, LPC-File et IQ-Tree * Structue dynamique flexible à la mise à jour	* Dégradation des performances en grande dimension * Dégradation des performances pour un grand volume de données.
Active Vertice	* Améliore les performances du VA-File et LPC-File * Adaptée aux données réelles.	* Dégradation des performances en grande dimension * Dégradation des performances pour un grand volume de données. * Non adaptée au données uniforme * paramétrage complexe

PCR-Tree	* Adaptée aux données réelles * structure réduisant le chevauchement entre les MBR * Améliore les performances du GC-Tree et VA+-File	* Non performante au passage à l'echelle & grande dimension * Non adaptée au données uniforme *Structure non dynamique, non flexible à la mise à jour * Non adaptée aux données avec des correlations non linaires
RA-Blocks	*Structure dynamique * Améliore les performances des méthode conventionnelles et du VA-File * Aucune structure arborescente *Structure dynamique, flexible à la mise à jour * Bon taux de selectivité * Tolère de grandes valeurs du nombre de bits de codage=>bonne precision ** Améliore les performances des méthodes d'indexation conventionnelles et du VA-File	* Non adaptée au données réelles * Dégradation des performances en grande dimension * Dégradation des performances pour un grand volume de données.

Tableau 2.3 Récapitulatif des avantages et inconvénients des méthodes d'indexation basée sur l'approche approximation locale citées précédemment

Méthode	*Avantages*	*Inconvénients*
VA-File	* Adaptée aux données uniformes * Améliore les performances des méthodes d'indexation conventionnelles	* Non adaptée au données groupés (données réelles) * Approximations non précises pour un b_i faible, dégradation des performances pour un b_i grand * Dégradation des performances en grande dimension * Dégradation des performances pour un grand volume de données.
VA+-File	*Adaptée aux données réelles * Améliore les performances des méthodes d'indexation conventionnelles et du VA-File	* Approximations non précises pour un b_i faible, dégradation des performances pour un b_i grand * Dégradation des performances en grande dimension * Dégradation des performances pour

		un grand volume de données.
C2-VA-File	*Adaptée aux données réelles * Améliore les performances des méthodes d'indexation conventionnelles et du VA-File * Structure mieux adapté à l'espace de grande dimension en comparaison avec VA-File	* Approximations non précises pour un b_i faible, dégradation des performances pour un b_i grand * Dégradation des performances pour un grand volume de données. * Non adaptée aux données avec des corrélations non linéaires
OVA-File	*Adaptée aux données uniformes * Améliore les performances des méthodes d'indexation conventionnelles et du VA-File * Adaptée au grand volume des données et aux grandes dimensions	* Non adaptée au données groupés (données réelles) * Approximations non précises pour un b_i faible, dégradation des performances pour un b_i grand
LPC-File	* Adaptée aux données uniformes * Codage ou Approximation précise * Améliore les performances des méthodes d'indexation conventionnelles et du VA-File	* Non adaptée aux données réelles * Dégradation des performances pour un grand volume de données. * Non adaptée aux données avec des corrélations non linéaires

Tableau 2.4 Récapitulatif des avantages et inconvénients des méthodes d'indexation basée sur l'approche approximation globale citées précédemment

Méthode	*Structure*	*Dimension*	*Volume de données*	*Données*	*sélectivité*
A-Tree	R*-Tree +approximation des vecteurs/ MBR(VMBRs)	♦	♣♣	réelles	●
IQ-Tree	R*-Tree +approximation des vecteurs (VA-File)	♦	♣♣	réelles	●
RA-Blocks	Aucune structure + approximation des régions	♦	♣♣♣	uniformes	●●●
GC-Tree	arborescente+approximation des vecteurs selon LPC-File	♦	♣♣	réelles	●●●
Active Vertice	Intersection région/sphère+ structure arborescente + approximation du VA-File	♦	♣♣	réelles	--
PCR-	R-Tree+ PCA+	♦♦	♣♣	réelles	●

Tree	Approximation du VA-File

Tableau 2.5 Récapitulatif de quelques propriétés générales des méthodes d'indexation basées sur l'approximation locale citées précédemment

Méthode	*Structure*	*Dimension*	*Volume de données*	*Distribution des données*	*séléctivité*
VA-File	Aucune structure +Approximation des vecteurs	♦	♣♣	uniformes	●●
VA+-File	Aucune structure +Approximation des vecteurs	♦ ♦	♣♣	réelles	●●
C2-VA-File	Aucune structure +Approximation des vecteurs	♦ ♦	♣♣	réelles	--
LPC-File	Aucune structure +Approximation des vecteurs	♦	♣	Uniformes	●●

Tableau 2.6 Récapitulatif de quelques propriétés générales des méthodes d'indexation basées sur l'approximation globale citées précédemment

Légende

♦ : dimension faible
♦ ♦ : dimension moyenne
♣ : Base de données de petite taille
♣ ♣ : Base de données de taille moyenne
♣♣♣ : Base de données de grande taille
● : faible taux de séléctivité
●● : taux de selectivité moyen
●●● : meilleur taux de sélectivité

5 Synthèse

Dans ce chapitre nous avons présenté dans un premier temps les principales techniques conventionnelles d'indexation multidimensionnelles. Nous tenons à préciser que l'état de l'art que nous avons mené sur ces méthodes n'est pas exhaustif. Nous nous sommes limités aux méthodes les plus citées et les plus utilisées dans le domaine de l'indexation multidimensionnelle. Sur la base cette étude, nous avons remarqué d'une part que la performance de la plus part de ces méthodes se dégradent au passage à l'échelle et en grande dimension, et d'autre part, parmi ces méthodes certaines sont particulièrement adaptées à un espace Euclidien et d'autre sont convenables uniquement à des distributions de données particulières, notamment uniformes. Nous avons présenté par la suite les problèmes de la malédiction de la dimension qui se posent lors de la manipulation des données de grande

dimension et les méthodes de la réduction linéaires et non linéaires de la dimension qui ont été proposé dans la littérature pour faire face à ces problèmes. Nous avons montré que ces méthodes réduisent la dimension de l'espace de données, mais leur application reste liée à des contraintes comme la linéarité des données, le paramétrage, etc. Des méthodes d'indexation basées sur l'approche approximation plus efficaces ont été proposées pour pallier aux problèmes de la malédiction de la dimension. Leurs principe de base est la représentation des données par une approximation compactes afin de réduire le temps de parcours lors du processus de la recherche. Nous les avons classé en deux grandes familles : les méthodes basées sur l'approximation globale et celles basées sur l'approximation locale. La première famille est performante au passage à l'échelle et en grande dimension en comparaison avec les méthodes conventionnelles par contre elle est plutôt adaptée aux données de distribution uniforme, la seconde famille a l'avantage de pouvoir manipuler des données réelles mais par contre sa mise en place dans de applications réelles nécessite des améliorations. Nous avons finalement comparé les différentes méthodes que nous avons présentés. Sur la base de cette comparaison nous concluons que la méthode basée sur l'approche approximation local RA-Blocks présente les meilleurs caractéristiques, c'est une méthode que nous trouvons prometteuse pour faire face aux problèmes de la malédiction de la dimension et au passage à l'échelle, et c'est pour ces raisons que nous avons choisi de l'étudier de plus prêt pour améliorer ces performances et l'intégrer dans un système de recherche d'images par le contenu.

Chapitre 3

Méthode proposée pour l'indexation et la recherche dans les espaces multidimensionnels : RA$^+$-Blocks

Ce chapitre présente le RA$^+$-Blocks, une nouvelle technique d'indexation multidimensionnelle basée sur l'approche approximation (filtrage). Tout comme RA-Blocks, RA$^+$-Blocks repose sur le découpage de l'espace de données en régions compactes et disjointes, chaque région est approximée par deux chaines de bits. Notre nouvelle méthode apporte une amélioration au niveau de la phase de découpage des régions par rapport à RA-Block, qui se base sur l'algorithme de découpage KDB-Tree. Nous présentons d'abord la méthode KD-Tree et KDB-Tree, ensuite nous détaillons la méthode RA-Blocks suivie par notre nouvelle méthode d'indexation RA$^+$-Blocks.

1 Introduction

Comme nous l'avons présenté dans le chapitre 2, les performances des techniques conventionnelles d'indexation multidimensionnelles se dégradent dramatiquement lorsque la dimension de l'espace de données augmente. Contrairement à ces techniques, les méthodes d'indexation basées sur l'approche *filtrage* ont été proposées en prenant en compte explicitement les problèmes liés aux espaces de grandes dimensions. Plusieurs travaux ont montré que ces méthodes effectuent des recherches des plus proches voisins d'une manière efficace dans les espaces de grandes dimensions (>16) [Web 98]. A partir de l'état de l'art que nous avons mené dans le précédent chapitre sur les principales méthodes d'indexation qui existent dans la littérature. Nous avons pu nous positionner et proposer une nouvelle méthode d'indexation multidimensionnelle basée sur l'approche approximation (RA$^+$-Blocks) qui est une amélioration de la méthode RA-Blocks. Rappelons que RA-Blocks présente des propriétés intéressantes par rapport aux autres méthodes d'indexation multidimensionnelles. En effet elle améliore les performances en terme temps de réponse du VA-File, la méthode représentative de l'approche approximation. RA-Blocks possède une structure dynamique flexible à la mise à jour, sa structure n'est pas affectée par la modification de la taille de la base. Le point fort de cette méthode est qu'elle combine les avantages des méthodes basées sur l'approximation globale en utilisant uniquement des fichiers pour la gestion des données et les avantages des

méthodes basées sur l'approximation locale en utilisant des formes géométriques englobantes (des régions) pour regrouper les données. Ces avantages comme nous allons le voir, permettent d'une part d'augmenter la capacité de stockage des données par rapport aux autres structures d'index notamment le VA-File et d'autre part d'améliorer le temps de réponse en traitant des régions denses et compactes au lieu de considérer les vecteurs ou les approximations des vecteurs. Notre étude comparative montre finalement que RA-Blocks possède les meilleurs caractéristiques (meilleur taux de sélectivité, structure d'index adaptée au grand volume de données, etc.). En s'appuyant sur ces avantages, nous avons choisi d'améliorer les performances du RA-Blocks dans l'objectif de pouvoir l'appliquer au grand volume de données de dimension élevé. Malgré les bonnes performances de la méthode RA-Blocks, cette dernière se trouve confronté aux problèmes liées à la malédiction de la dimension notamment au niveau des régions obtenues lors de la phase du découpage de l'espace de données. En effet, le partitionnement de l'espace de données se base sur l'algorithme KDB-Tree [Rob 81], dont le principal inconvénient est la production d'un grand nombre de régions vides ou des régions contenant peu de vecteurs par rapport à leur capacité. Ceci entraine une augmentation considérable dans le temps de la recherche et une mauvaise exploitation de l'espace mémoire et par conséquent dégrade les performances de la structure d'index.

Pour remédier à ce problème, nous avons proposé la méthode RA$^+$-Blocks, une amélioration de RA-Blocks. Elle permet la recherche des $k-ppv$ d'un vecteur requête donné selon trois étapes: dans la première étape dite *phase de compression*, l'espace de données est partitionné sous forme de cellules hyper rectangulaires. Ensuite, en se basant sur une nouvelle technique de subdivision inspiré de l'algorithme de partitionnent KD-Tree [Ben 79], l'espace de données est partitionné en des régions compactes et disjointes chacune correspond à un ensemble de cellules, et chaque région est ensuite approximée par les deux cellules correspondant aux deux cellules en bas à gauche et en haut à droite de chaque région. La deuxième étape est appelée *phase de filtrage.* Elle consiste à sélectionner les régions qui ont une forte chance de contenir les voisins les plus proches du vecteur requête en se basant sur leurs distances minimales et maximales par rapport au vecteur requête. Enfin, dans la troisième étape *accès aux données,* un parcours séquentiel est effectué sur les vecteurs contenus dans les régions retenues dans la phase de filtrage pour calculer les voisins les plus proches de la requête en se basant sur les distances réelles.

Dans ce qui suit nous présentons d'abord les deux méthodes d'indexation KD-Tree et KDB-Tree avant de détailler les deux techniques RA-Blocks et RA$^+$-Blocks. Notons que KDB-Tree et KD-Tree sont deux méthodes d'indexation multidimensionnelles utilisées respectivement par les deux méthodes RA-Blocks et RA$^+$-Blocks dans un objectif de découpage de l'espace de données et non pas pour un objectif d'indexation des données, leurs structures correspondantes ainsi que leurs algorithmes de recherche ne sont pas exploités par les deux méthodes RA-Blocks et RA+-Blocks. De ce fait, nous détaillons dans le paragraphe suivant uniquement le principe de partitionnent de l'espace de données de KD-Tree et KDB-Tree.

2 KD-Tree

KD-Tree [Ben 79] est une méthode d'indexation multidimensionnelle permettant d'organiser les données dans un espace multidimensionnel de dimension d , c'est une structure arborescente binaire basée sur une subdivision récursive de l'espace en régions hyper-rectangulaires disjointes [Ben90][Bff 77], appelées cellules. Chaque nœud dans l'arbre est associé à une région ainsi qu'à l'ensemble des vecteurs se trouvant dans cette région. Le nœud racine de l'arbre est associé au cadre limite qui contient tous les vecteurs, les nœuds internes contiennent

l'hyperplan séparateur en plus des deux fils correspondant aux deux sous espaces générés par le partitionnent. Enfin, les feuilles de l'arbre contiennent l'ensemble des vecteurs contenus dans le sous espace correspondant. Un exemple de la structure KD-Tree est illustré dans la figure 3.1.
La construction de la structure d'index du KD-Tree se fait par des insertions successives des différents vecteurs de la base de données. Considérant un nœud arbitraire dans l'arbre, à chaque fois que le nombre de vecteurs associés à ce nœud est plus grand qu'une petite taille, qu'on appelle capacité de la feuille, le cadre associé à ce nœud est divisé en deux par un hyper plan orthogonal à un axe de coordonnées. Plusieurs algorithmes de partitionnement existent, ils différent dans le choix du plan séparateur, et dans la condition d'arrêt de l'algorithme. Dans le paragraphe suivant nous détaillons une stratégie générale de la construction du KD-Tree.

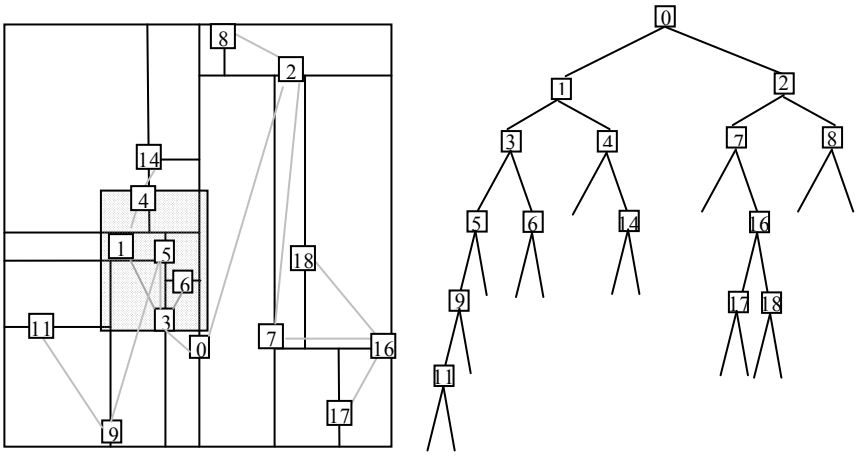


Fig.3.1. La structure du KD-Tree et ses partitions dans le plan

2.1 Construction d'un KD-Tree

Comme nous allons le présenter dans les sections suivantes, il existe différentes stratégies de division de l'espace de données, chacune faisant un choix différent pour l'hyperplan de division. Si un nœud est saturé, le sous espace relatif à ce nœud est divisé en deux sous espaces par un hyperplan orthogonal à un axe de coordonnées, les deux sous espaces résultants sont associés aux deux fils relatif à ce nœud. L'ensemble des vecteurs se trouvant dans le sous espace original est partagé entre les deux fils, selon leurs positions par rapport à l'hyperplan de division. Les vecteurs qui se trouvent sur l'hyperplan lui-même peuvent être associés aux deux fils simultanément (ou suivant la règle dictée par la méthode de division).
Quand le nombre des vecteurs associés au sous espace courant devient inférieur à la taille admise de la feuille (ou bien le nombre de partitionnent du nœud correspondant à cet espace dépasse un certain seuil), le nœud est déclaré feuille, et ses vecteurs sont mémorisés dans ce nœud.
La structure globale du KD-Tree contient des informations telles que le nombre de vecteurs, la dimension de l'espace d et la taille de la feuille. Toutes les feuilles de l'arbre contiennent le nombre de vecteurs associés à cette feuille (un nombre qui varie de 0 et la taille de la feuille) ainsi que leurs indices. Le nœud interne contient les informations suivantes :
1) une valeur entière qu'on appelle dimension de division (allant de 0 à $d-1$) indiquant l'axe de coordonnées orthogonal à l'hyperplan qui a réalisé la division.

2) une valeur réelle qu'on appelle la valeur de division indiquant la position à laquelle l'hyperplan coupe l'axe de coordonnées.
3) deux pointeurs, l'un sur son fils gauche et l'autre sur son fils droite.
Il existe plusieurs algorithmes de partitionnent de l'espace de données dans KD-Tree, chacun fait référence soit à un choix différent du plan séparateur, soit à un critère différent de partitionnent des nœuds. Pour le plan séparateur, le choix se base sur deux critères : l'orientation et la position. Pour l'orientation, ce choix correspond au plan perpendiculaire à chacun des axes du repère à tour de rôle alors que pour la position, ils existent plusieurs choix correspondant à plusieurs stratégies. Deux d'entres elles seront détaillés dans le paragraphe suivant.
En ce qui concerne le partitionnement de l'espace de données on distingue deux critères : le premier critère se base sur le nombre de vecteurs dans le sous espace. Si ce nombre est supérieure à une certaine valeur, le processus de partitionnement est déclenché. Le deuxième critère se base sur le nombre maximal de subdivision.
L'algorithme général de la construction du KD-Tree est donné dans l'annexe

2.2 Stratégies de subdivision

Plusieurs méthodes de partitionnent de l'espace de données selon la structure KD-Tree existent. Dans ce paragraphe nous allons en détailler uniquement deux, dont dérivent l'ensemble des autres techniques : la méthode standard, et la division du point médian.
Soit S l'ensemble des N vecteurs à indexer avec la méthode KD-Tree. On note C le nœud courant et $R(C)$ le sous espace limitant ce nœud. $R(S)$ dénote le rectangle limite dans lequel se trouvent tous les vecteurs de S. Au début, C est la racine de l'arbre, donc $R(C) = R(S)$

Définitions

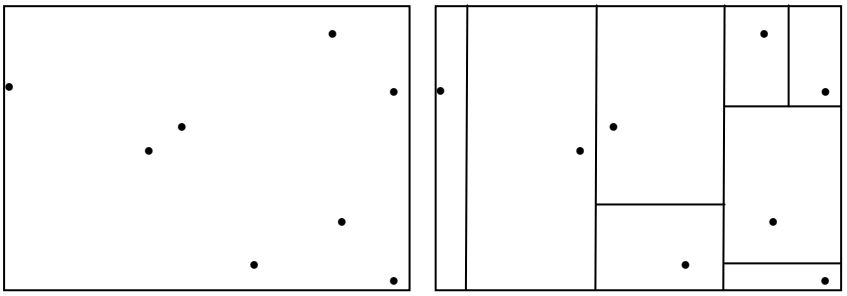
1.On définit le rapport hauteur/largeur d'un rectangle (sous espace), qu'on notera RHL, et qui est le quotient de la longueur de son plus grand côté sur celui de son plus petit côté.

2.Soit i une dimension donnée $0 \le i < d$. On définit la diffusion de S le long de la dimension i comme étant la différence entre la plus grande et la plus petite coordonnée suivant cette dimension.

3.Une partition médiane de N nombres de S, est une partition de S en deux sous ensembles, l'un avec $N/2$ éléments dont les valeurs sont plus grandes que la médiane de S, et l'autre avec les $N/2$ autres éléments dont les valeurs sont inférieures à la médiane de S.

2.2.1 La division standard

Dans cette stratégie, la dimension de division est celle qui correspond au maximum de diffusion de S et la valeur de division est la médiane des coordonnées de S suivant cette dimension. Cette procédure ne tient pas compte de l'aspect RHL des cellules résultantes. Un exemple de partitionnent avec la méthode standard est illustré ci-dessus :

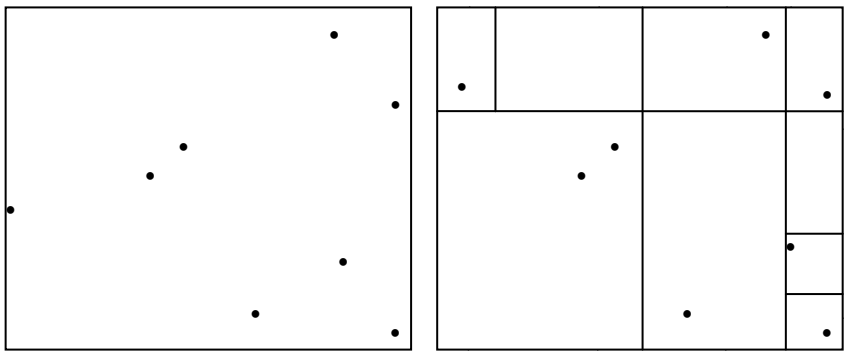


(a) Avant subdivision **(b)** Après subdivision

Fig. 3.2 La subdivision d'un espace de données par la méthode standard

2.2.2 La division du point médian

Cette stratégie divise une cellule par un hyperplan orthogonal à son plus grand côté suivant le point médian de ce dernier. Un hyper cube de côté x est partitionné en deux hyper cubes de côté $x/2$ chacun. Bien évidement cette procédure est simple mais elle a l'avantage de produire des cellules d'aspect RHL borné. Cependant, elle peut produire des cellules triviales, dans le sens où tous les points se trouvent d'un même côté de l'hyperplan de division. En conséquence, l'arbre peut avoir une taille arbitrairement grande et peut même dépasser N si les points sont très proches. Un exemple de partitionnent suivant cette méthode est donnée par la figure 3.3.
Finalement le KD-Tree permet une structuration spatiale de l'espace à d-dimensions, il adopte une structure arborescente binaire permettant d'accélérer le traitement des données multidimensionnelles. Il a l'avantage d'être plus ou moins précis dans la construction de la structure d'index puisque les stratégies de partitionnent utilisées suivent la concentration des données selon les différentes zones de l'espace, contrairement à KDB-Tree, comme nous allons le voir, qui ne prend pas en considération la distribution des données dans l'espace de données dans le processus du partitionnement.



(a) Avant subdivision **(b)** Après subdivision

Fig. 3.3 Application de a stratégie de division du point médian à l'ensemble de points de l'exemple précédent

3 K-D-B-Tree et ses variantes

K-D-B-Tree est une structure d'index multidimensionnelle se basant sur le partitionnement de l'espace en des régions disjointes organisées dans une arborescence équilibrée. C'est une structure qui décompose l'espace multidimensionnel comme K-D-Tree et équilibre l'arbre résultant comme B-Tree. Elle combine les propriétés de K-D-Tree et B-Tree [Bay 71] pour pouvoir manipuler les données multidimensionnelles dans une mémoire externe. Le nœud racine

de l'arbre correspond à l'espace de données qui contient les descripteurs des images. Les nœuds internes de l'arbre appelés " *region pages* " contiennent les pairs (RR, ID) avec RR est la région délimitant l'ensemble de données et ID est un pointeur sur les feuilles/nœuds internes. Les feuilles de l'arbre nommées "*pages points*" contiennent les vecteurs réels. Le nombre maximal de vecteurs pouvant être contenu dans chaque feuille est limité par une certaine taille de la feuille appelée capacité. La figure 3.4 représente la structure générale du KDB-Tree dans un espace de dimension deux.

La décomposition de l'espace de données s'effectue selon deux niveaux. Dans le premier niveau, les " *region pages* ", sont partitionnées suivant un hyperplan de subdivision parallèle aux différents axes de l'espace de données. Dans le deuxième niveau, les "*pages points*" correspondant aux " *region pages* " intersectés avec l'hyperplan de subdivision sont également partitionnées.

Le fractionnement d'une "*page point*" se produit lors de l'insertion d'un nouveau vecteur dans une "*page point*" saturée (c.à.d. le nombre de vecteurs de la "*page point*" est supérieur à la capacité). Dans ce cas, la "*page point*" est partitionnée en deux sous pages suivant un hyperplan de subdivision perpendiculaire à un axe de coordonnée. La position du partitionnement est fixée arbitrairement et la dimension est choisie cycliquement. La "*page point*" initiale est remplacée par les 2 nouvelles "*pages points*" résultantes du partitionnement, et une nouvelle entrée est insérée dans la "*pages points*" racine relative à la nouvelle "*page point*". La propagation de la subdivision "*pages points*" est dite **upward**. Un exemple d'une décomposition d'une "*page point*" est illustré dans la figure 3.4.

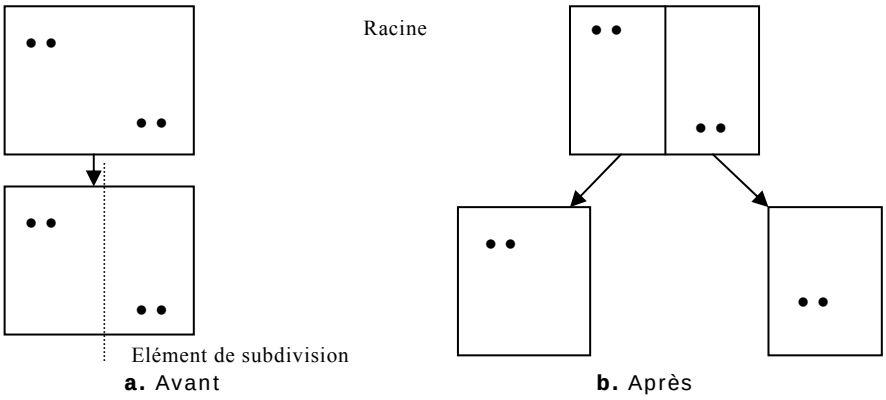


Fig.3.4 . Partitionnement d'une page point

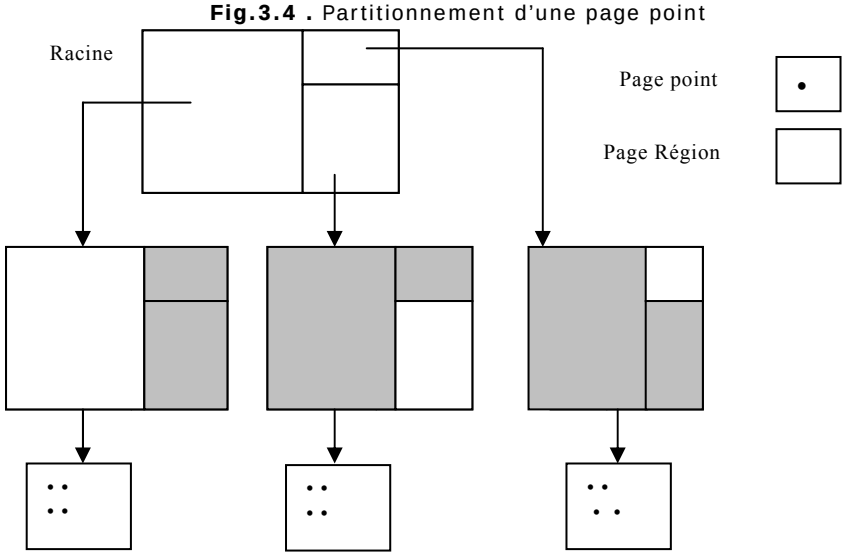


Fig.3.5. Structure d'un 2-D-B-Tree

Une "*region pages*" est partitionnée, si le nombre d'entrées ID dans cette page dépasse la capacité de la région. Dans ce cas, le nombre d'entrées de cette région est subdivisé en deux. L'hyperplan de subdivision décompose la "*region pages*" en deux sous régions telle que l'union des sous régions résultantes constitue la région mère. Par la suite, toutes les régions (RR,ID) du nœud interne ayant une intersection non nulle avec l'hyperplan de subdivision sont alors partitionnées de la même manière. Ceci entraîne par conséquent, le partitionnement des pages points correspondantes. Donc la propagation de la subdivision des "*region pages*" se propage du haut vers le bas, elle est dite **downward** (voir figure 3.6).
La construction de la structure d'index de KDB-Tree s'effectue par insertion successive des vecteurs de la base de données. Notant que l'ordre d'insertion des vecteurs est important dans la construction de la structure du KDB-Tree, et un ordre d'insertion différent produit des structures d'index différentes. Malheureusement il n'y a aucune étude sur le choix de l'ordre d'insertion des vecteurs, généralement l'insertion est effectuée de manière arbitraire. L'algorithme de la construction de la structure d'index est décrit dans l'annexe .
Le deuxième niveau de partitionnement correspondant à celui des "region *pages*" constitue le principal inconvénient de la méthode KDB-Tree. En effet, la structure de KDB-Tree vérifie deux propriétés :
- absence de chevauchement : les régions produites du partitionnement sont disjointes et leurs union constitue la région mère.
- arbre équilibré :. les chemins séparant toutes les feuilles et la racine sont égaux.

Pour préserver ses propriétés lors du partitionnement des nœuds internes saturés de l'arbre, KDB-Tree procède à une restructuration de la totalité de l'arbre entrainant le partitionnement , des nœuds internes non saturés , leurs descendants et les feuilles correspondant, produisant ainsi des régions vides où presque vides. Ceci augmente, le nombre de régions produites lors du partitionnent et par conséquent le temps de la recherche, et le taux d'occupation de l'espace mémoire. Notons que ces restructurations deviennent de plus en plus fréquentes lorsque la dimension des données augmente. D'un autre côté, la stratégie du partitionnent d'une page saturée ne dépend pas de la distribution des données : la dimension est choisie cycliquement et la position du partitionnent est fixée arbitrairement ce qui augment encore plus les chances d'avoir des partitionnements triviaux dans le sens où ils produisent les régions vides ou peu denses.

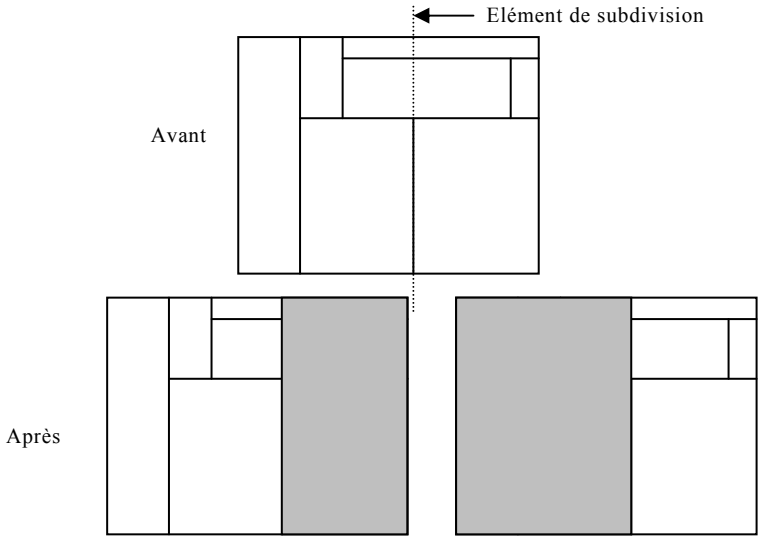


Fig. 3.6 décomposition d'une page région

Plusieurs méthodes ont été proposé dans la littérature pour contourner le problème du partitionnent des nœuds internes de la structure KDB-Tree. Une des méthodes consiste à reproduire le même partitionnement des niveaux inférieurs (les feuilles) dans les niveaux supérieures, le partitionnement des nœuds internes s'effectue suivant l'hyper plan de subdivision avec lequel les feuilles sont partitionnées pour la première fois. Cette stratégie est appelées *première division* ou "FD-splitting". Cette stratégie permet d'éliminer le partitionnement des nœuds internes de l'arbre ainsi que la subdivision "downward", elle garantit une occupation de l'espace mémoire inférieur à 60% [Hen 89, See 90]
Une deuxième méthode KDB_{HD}-Tree plus récente a été proposé par Orlandic et al. [Orl 02], elle se base sur la méthode "FD-splitting" pour éliminer la subdivision "downward" de la structure d'index de KDB-Tree, sa principale idée est de réduire la taille de la structure d'index en éliminant les informations redondantes de l'intérieure des nœuds. En effet, dans l'espace de grande dimension, chaque région est partitionnée le long d'un petit ensemble de dimensions. Les dimensions restantes de chaque région s'étendent sur toute la face de l'espace initiale. Dans la méthode KDB_{HD}-Tree, ces dimensions sont éliminées des vecteurs appartenant à la structure d'index. Une troisième méthode KDB_{KD}-Tree [Yu 03] est proposée dans la même optique, elle propose d'éliminer l'information redondante des nœuds internes de l'arbre par la modification de la structure des nœuds internes en un arbre KD-Tree ou la représentation de toutes les entrées de l'index dans un nœud, devient une séquence de partitions qui contient les vecteurs et les pointeurs sur les nœuds. Malheureusement, malgré les stratégies utilisées pour la suppression de l'information redondantes adopté par la méthode KDB_{HD}-Tree et KDB_{KD}-Tree, les problèmes de partitionnent inutiles de certaines régions ainsi que mauvaise répartition des vecteurs dans les régions sont toujours présents. Ces deux méthodes sont moins performantes de que KDB-Tree.
Yi et al [Yi 05] ont proposé une autre technique pour éviter le partitionnement des nœuds internes de la structure KDB-Tree. Dans cette méthode, les sous espaces correspondant au régions se trouvant au même niveau de l'arbre sont partitionnés selon une dimension spécifiée. Le partitionnement se produit lors de l'insertion d'un vecteur dans une feuille saturée. Contrairement aux autres méthodes, au lieu de partitionner la feuille saturée, l'espace non utilisé dans le nœud racine relatif à la feuille cible est exploité pour stocker la surcharge de données. Les nœuds fils de la région mère relatif à la feuille cible sont également exploités pour disperser la surcharge de données de la région mère. Cette stratégie permet d'éviter le fractionnement des nœuds internes de l'arbre, elle garantit l'existence d'un nombre minimal d'entrée dans chaque nœud. Les performances de cette méthode améliorent ceux de la méthode FD-splitting et KDB_{KD}-Tree [Yi 05]. Par contre, le risque de produire des régions vides ou presque vides est toujours présent notamment lorsque la dimension de l'espace de données augmente.
Dans le paragraphe suivant, nous présentons la méthode d'indexation basée sur l'approche approximation : RA-Blocks, cette technique se base sur la méthode KDB-Tree pour le partitionnent de l'espace de données en des régions compactes et disjointes.

4 RA-Blocks

Comme toutes les méthodes d'indexation multidimensionnelles, RA-Blocks est constituée de 3 étapes : Approximation ou codage au cours de laquelle les données sont compressées et codées par des chaines de bits, l'indexation où les descripteurs de la base sont organisés dans une structure d'index et finalement l'étape recherche dans la quelle un parcours de la structure

d'index est effectué pour calculer les plus proches voisins. Nous détaillons ces trois étapes dans la suite

4.1 Approximation

RA-Blocks (Region Approximation Blocks) [Ter 02] est basée sur le filtrage des régions. 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 d'abord en des cellules hyper rectangles, chacune est représentée par une chaîne de bits. Puis, il est subdivisé en des régions contigües et disjointes ayant la même capacité. Ces régions sont ensuite codées par deux chaînes de bits correspondant aux deux cellules en bas à gauche et en haut à droite de chaque région, formant ainsi le fichier d'approximations à gérer. On distingue deux ensembles de données : un fichier qui contient tous les vecteurs de la base de données et un autre de ″petite″ taille contenant les approximations codées en bits de chaque région.

Un exemple de codage des régions dans un espace de dimension deux est illustré dans la figure 3.7. Dans cet exemple, le nombre de bits par dimension est fixé à 3, chaque dimension est divisée en 8 segments de même taille, on obtient ainsi 2^3 intervalles suivant chaque dimension. Dans l'exemple, on suppose que chaque région contient uniquement 5 vecteurs, c.à.d. que le découpage récursif d'une région en sous régions est réalisé tant que le nombre de vecteurs est supérieure à 5. Les lignes en pointillées délimitent les cellules et les lignes en gras délimitent les régions. Le partitionnement en régions se fait selon l'algorithme KDB-Tree [Rob 81], l'élément de subdivision de chaque région correspond aux frontières des cellules. En effet, chaque dimension est décomposée en 2^3 segments (3 bits par dimension), et chaque cellule possède un indice i selon chaque dimension (dans l'exemple i varie de 0 à 7), par conséquent les bornes de chaque région sont représentées par : $x = i \times L/8$, $y = i \times h/8$ où L et h représentent respectivement la largeur et la hauteur de l'espace de données et i, l'indice des cellules qui délimitent la région. Chaque région est approximée par le codage de la cellule de début et celle de fin. Par exemple on utilise (000 000, 011 111) pour coder le début et la fin de la région qui contient les trèfles.

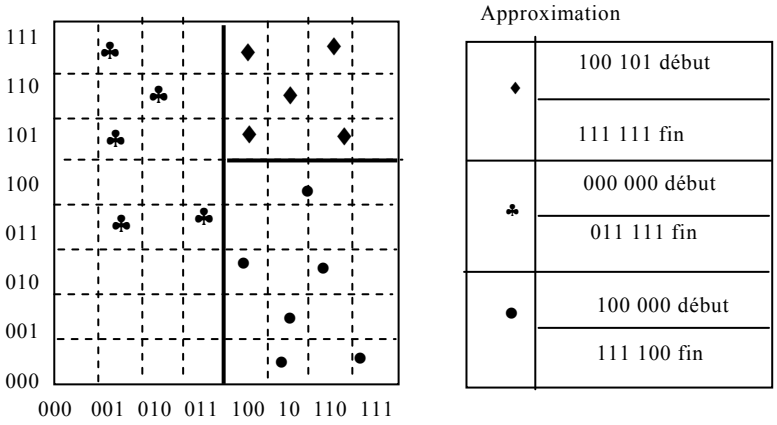


Fig. 3.7. Exemple de codage des régions dans un espace de dimension deux.

4.2 Indexation

L'étape indexation ou structuration consiste à organiser les descripteurs afin que les procédures de recherche répondent le plus rapidement possible. Cette organisation se traduit généralement par une structuration des descripteurs en de "petits" ensembles et par l'implémentation des stratégies de recherche capable de retourner un résultat en n'examinant qu'un "petit" nombre d'ensembles de vecteurs.

La structuration de l'espace de description selon la méthode RA-Blocks consiste dans un premier temps à quantifier l'espace de données avec un pas de quantification fixé au préalable et dans un deuxième temps à découper l'espace en régions, chacune étant approximée par deux chaines de bits. Ce découpage s'effectue suivant l'algorithme de découpage KDB-Tree que nous avons présenté dans le paragraphe 1.2. Il est basé d'une part sur la grille de codage issue du processus de quantification et d'autre part sur l'ordre d'insertion des vecteurs de la base de données dans la structure d'index du RA-Blocks.

A l'issu du partitionnement de l'espace des descripteurs, un certain nombre de régions est obtenu dont l'union constitue l'espace de données de départ. Ces régions sont utilisées pour construire la structure d'index du RA-Blocks.

RA-Blocks a une structure d'index dynamique, elle est composée de deux éléments principaux ; les "*RA-Data"* et les "*Data-Pages"*. Les "*RA-Data"* interviennent dans l'étape d'indexation alors que les "*Data-Pages"* sont utilisées pour stoker les données réelles. Une page région contient les approximations de la région qui correspondent en réalité à sa position réelle dans l'espace de données, en plus d'un pointeur sur une page point contenant les vecteurs réels de cette région. Chaque page région correspond à une région dans l'espace de données, elle est généralement stockée dans une page disque. Une page de données contient les vecteurs réels de la région correspondante.

Un exemple illustratif de la structure d'index du RA-Blocks est donné par la figure 3.8.

En résumé, la structure d'index du RA-Blocks est une structure à deux niveaux. Dans le premier niveau sont stockées les approximations des régions "region *pages*", ces dernières sont utilisées dans l'étape de filtrage pour sélectionner les régions susceptibles de contenir les voisins les plus proches (régions candidates). Dans le deuxième niveau sont stockés les données réelles "*pages points*" qui servent à calculer les voisins les plus proches du vecteur requête dans la phase d'accès aux données réelles.

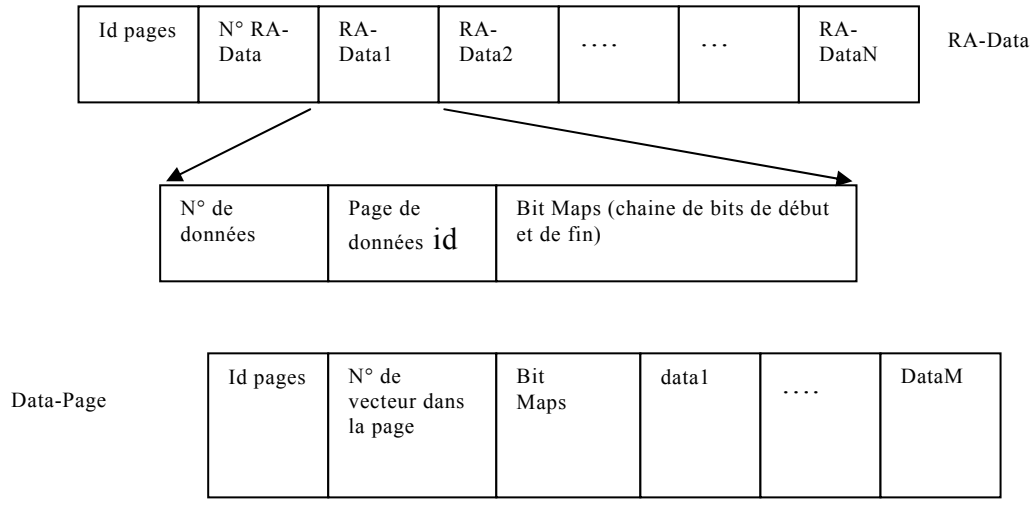


Fig. 3.8. Structure d'index du RA-Blocks

4.3 Recherche

La recherche des $k-ppv$ est effectuée en deux phases. Dans la première dite *de filtrage*, les régions issues du découpage de l'espace de données sont parcourues séquentiellement, et les régions candidates sont sélectionnées suivant leurs distances minimales et maximales par rapport au vecteur requête. Lors de la deuxième phase appelée étape *d'accès aux vecteurs*, les $k-ppv$ sont déterminés en calculant la distance réelle (Euclidienne) par rapport au vecteur requête.

Etape de filtrage

Pour calculer les $k-ppv$, le fichier des approximations est parcouru entièrement. Les distances $d_{\min}$ et $d_{\max}$ par rapport au vecteur requête sont calculées (voir figure 3.9) et la liste des candidats LC est initialisée par les $k/2$ régions ayant les plus petites $d_{\max}$ par rapport au vecteur requête. Cette liste est triée par ordre croissant de la distance $d_{\max}$. Ensuite, Le fichier des approximations est parcouru : si une approximation est rencontrée telle que sa distance minimale $d_{\min}$ est inférieure à la plus grande distance maximale $D_{\max}$ de la liste LC, la région correspondante est insérée dans LC en gardant la liste triée.

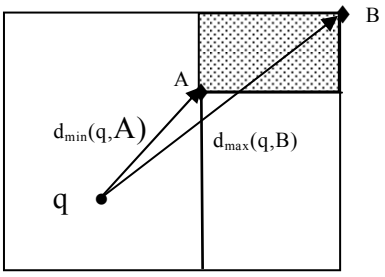


Fig. 3.9. Les distances minimales et maximales d'une région par rapport à un vecteur requête

Accès aux vecteurs réels

Soit LR, la liste des résultats devant contenir les vecteurs les plus proches de la requête. LR est d'abord initialisée par les k premiers vecteurs des premières régions de LC. Cette liste est ensuite triée selon la distance réelle (Euclidienne) au vecteur requête. Ensuite, l'algorithme de recherche parcours les régions retenues dans LC selon un ordre croissant de leur $d_{\min}$ en calculant la distance réelle entre les vecteurs de chaque région et le vecteur requête. Supposons que $D_{\max}$ est la distance réelle maximale trouvée. Si une région est rencontrée telle que sa distance minimale par rapport au vecteur requête est inférieure à $D_{\max}$, les vecteurs de cette région sont examinés séquentiellement. Notons que le coût de la séquentielle dans ce cas est faible puisqu'elle est effectuée sur un ensemble de vecteurs au maximum égal à la capacité des régions. Ensuite, si la distance réelle qui sépare un vecteur de cette région et le vecteur requête est inférieure à $D_{\max}$, alors le vecteur est inséré dans LR en gardant la liste triée, et le dernier vecteur de LR est supprimé. Par contre, si une région de LC est rencontrée telle que sa distance $d_{\min}$ est supérieure à $D_{\max}$ alors l'algorithme de recherche s'arrête. L'algorithme de recherche des $k-ppv$ est détaillé dans la figure 3.10.

Algorithm_de_recherhce_RA_Blocks *(LA : liste des approximations des régions de l'espace de données, NR : le nombre de régions dans LA, q : le vecteur requête) return LR : liste des $k-ppv$ du vecteur requête triée par la distance réelle par rapport à q.*

{
Variables :

// k : constante qui représente le nombre des plus proches voisins
// LC : liste des régions candidates
// NC : nombre de régions candidates dans LC
// $d_{\min}$ et $d_{\max}$: les distances minimale et maximale qui séparent le vecteur q et la région courante
// data : liste des données réelles

// Etape de filtrage

1: *Initialiser LC par les $k/2$ régions ayant les plus petites $d_{\max}$ par rapport à q triée par ordre croissant de $d_{\max}$ et NC par $k/2$*

2: $D_{\max} \leftarrow MAX(d_{\max}, LC)$ *// La plus grande $d_{\max}$ de LC*
3 *: Pour toutes les autres régions qui restent dans LA Faire {*
4 *: Soit i l'indice de la région courante*
5 *: Si $d_{\min}(LA[i]) \leq D_{\max}$ {*
6 *: Insérer $LA[i]$ dans LC*
7 *: $NC \leftarrow NC + 1$*
8 *: Trier LC par ordre croissant de $d_{\min}$*
9 *:}// FinSi*
}//FinPour
// Accès aux vecteurs
10 *: Initialiser la liste LR par les k premiers vecteurs des régions de LC triée par ordre croissant de leurs distances réelles par rapport à q*
11 *:Soit D_m la plus grande distance réelle trouvée dans LR*
12 *:Pour toutes les autres régions qui restent dans LC Faire {*
13 *: Soit i l'indice de la région courante*
14 *: Si $d_{\min}(LC[i]) \geq D_m$ alors arrêter*
15 *: Sinon*
16 *: Pour tous les vecteurs V_j de la région $LC[i]$ Faire {*
17 *: Si $dist(V_j, q) \leq D_m$ { // dist est la distance réelle*
18 *: Insérer V_j dans LR*
19 *: Supprimer le dernier vecteur de LR*
20 *: $D_m \leftarrow MAX(dist(V_j, q), D_m)$ // la plus grande distance réelle de LR*
} // FinSi
}// FinPour
}// Fin Pour
}// Fin

Fig. 3.10. L'algorithme de recherche des $k-ppv$ du RA-Blocks

4.4 Synthèse

RA-Blocks est une méthode d'indexation et de recherche des $k-ppv$ basée sur l'approche approximation des régions. Cette méthode a été proposée pour combler l'insuffisance des techniques conventionnelles et leur inefficacité dans les espaces de grande dimension. Elle combine les avantages de l'approche approximation globale en utilisant uniquement deux fichiers pour la gestion de l'index, un fichier des approximations et un fichier de données réelles, et les avantages de l'approximation locale en utilisant les formes géométriques sous

forme de rectangles pour l'approximation des données. Le principe de base de RA-Blocks est d'abord de quantifier l'espace de données comme le VA-File, puis de partitionner l'espace de données en un "petit" nombre de régions compactes et disjointes. Ensuite, parcourir un sous ensemble de données sélectionné dans la phase de filtrage par leur approximations et enfin accéder séquentiellement à l'ensembles des vecteurs qui correspond aux régions sélectionnées dans la phase de filtrage. Contrairement au VA-File qui approxime chaque vecteur par la cellule qui le contient, le RA-Blocks quant à lui approxime les régions, où chaque région peut contenir plusieurs vecteurs. Ceci permet d'une part d'augmenter la capacité de stockage des données en mémoire et d'autre part de réduire le temps de parcours par rapport au VA-File.
A la différence du VA-File, les performances du RA-Blocks dépendent essentiellement de la capacité des régions et non pas du nombre de bits de codage. En effet, une grande capacité des régions permet de regrouper les vecteurs de la base dans un nombre réduits de régions, ceci entraine l'exécution du parcours séquentiel sur un petit ensemble de vecteurs réduisant considérablement le temps d'E/S et le temps de la recherche. En revanche, une petite valeur de la capacité des régions produit un nombre important de celles-ci, les performances du RA-Blocks dans ce cas s'approchent ou deviennent identique (pour une capacité= 1) à celles du VA-File, i.e elles deviennent dépendantes du nombre de bits de codage et de la taille du fichier des approximations.
Par conséquent, la capacité des régions est un paramètre critique et difficile à fixer, il est étroitement lié au processus du découpage de l'espace de données. De notre point de vue, une bonne méthode de partitionnement devrait générer un "petit" ensemble de régions denses et compactes, ceci permettra une meilleure gestion de l'espace mémoire alloué et un temps de parcours optimal. Par ailleurs, le partitionnement de l'espace de données en régions par la méthode RA-Blocks, s'effectue suivant l'algorithme de découpage KDB-Tree, ceci constitue le principal inconvénient de la méthode RA-Blocks. En effet, comme nous l'avons déjà présenté, KDB-Tree produit un certain nombre de régions vides où presque vides lors du partitionnent des nœuds internes de l'arbre pour préserver les propriétés de la structure alors que celle-ci n'est pas exploitée dans le processus de la recherche. Ceci entraine une mauvaise gestion de l'espace de données alloué ainsi qu'une perte importante du temps de la recherche. Pour cela nous avons proposé une nouvelle méthode d'indexation multidimensionnelle basée sur l'approximation des régions RA$^+$-Blocks. Cette méthode apporte une amélioration considérable au niveau de la phase de découpage de l'espace de données, elle permet une meilleure gestion de l'espace mémoire et réduit le temps de recherche par rapport à la méthode RA-Blocks. Cette méthode est détaillée dans le paragraphe suivant.

5 RA$^+$-Blocks

Cette méthode a été proposé [Dao 08] pour répondre aux principales limitations de la méthode RA-Blocks, notamment les problèmes liés au découpage de l'espace de données. Notre méthode RA$^+$-Blocks repose sur l'approximation des régions, elle suit la même démarche du RA-Blocks dans la quantification de l'espace de données. En effet, chaque dimension d_i est partitionnée en $2b_i$ intervalles où chaque intervalle est codé sur b_i bits. Ensuite, l'espace de données est décomposé en régions compactes et disjointes de même capacité. Comme RA-Blocks, les régions obtenues sont approximées chacune par deux chaînes de bits formant ainsi le fichier d'approximations à gérer.
RA$^+$-Blocks se compose de deux étapes principales : une phase *hors ligne* de structuration de données dans laquelle l'espace de données est partitionné en régions et les vecteurs de la base sont organisés et stockés dans les régions, et une autre phase *en ligne* d'interrogation (la

recherche) de la base. Les phases de quantification qui consiste à subdiviser l'espace de données en des cellules hyper rectangulaires ainsi que la phase d'interrogation de la base sont identiques à celles du RA-Blocks, la principale différence entre les deux méthodes réside dans la phase de la structuration, c'est pour cela seule cette étape est détaillée dans le paragraphe suivant.

5.1 Structuration des données

La structuration de l'espace de description est une amélioration de la stratégie de structuration adoptée par la méthode RA-Blocks. Dans ce paragraphe, nous revenons succinctement sur cette phase et nous donnons ses points faibles par rapport à la tâche de recherche envisagée. Nous présentons ensuite les améliorations et les modifications que nous apportons à cette phase. Enfin nous décrivons de façon détaillée notre algorithme de découpage pour la formation des régions.

5.1.1 Retour sur la phase de structuration du RA-Blocks

La phase de structuration des données est une étape qui s'effectue *hors ligne* dont le but principal est d'organiser les vecteurs de la base en régions. Le nombre de régions ainsi que leurs compacités est basée d'une part sur la taille de la mémoire allouée à la construction de l'index et d'autre part sur la stratégie de l'algorithme du découpage. Ce dernier étant basé sur KDB-Tree, il présente trois limitations principales :

* Le choix de la dimension de partitionnement se fait indépendamment de la distribution des données, celui-ci s'effectue d'une manière cyclique pour toutes les directions de l'espace de données. Ce choix risque de provoquer le partitionnement des sous espaces correspondant aux régions selon les dimensions contenant peu de concentration de données par rapport à d'autres dimensions.

* Le choix de la position du fractionnement correspond au premier point qui génère deux régions non saturées. Ce choix est également indépendant de la distribution des données, il risque de produire des régions non équivalentes en nombre de vecteurs.

* Le fractionnement des nœuds internes de la structure arborescente du KDB-Tree, provoque le partitionnement d'un certain nombre de régions vides où presque vides (le nombre de vecteurs est inferieur à la capacité des régions).

Ces trois limitations influent mutuellement la performance de la stratégie du partitionnement et diminuent l'efficacité de la structure d'index notamment en grande dimension et au passage à l'échelle.

De notre point de vu, une bonne méthode de partitionnement devrait produire des régions vérifiant les propriétés suivantes :

* **Compactes et disjointes :** Cette propriété permet de rendre la phase de filtrage plus efficace et de réduire le nombre de vecteurs à parcourir lors de la recherche.

* **Le moins de chevauchement possibles :** le chevauchement est un événement nuisible au processus du filtrage ainsi que la recherche, l'absence du chevauchement entrainera un gain considérable dans le temps de la recherche.

***denses** (Un nombre de vecteurs > capacité/2) :Il est important d'avoir des régions denses lors du processus de partitionnement de l'espace de données. Si par exemple on obtient un grand nombre de régions peu denses lors du partitionnement, le processus de recherche devrait explorer un grand nombre de régions pour obtenir les plus proches voisins ce qui aura pour effet

d'augmenter le temps de la recherche. Le fait d'avoir des régions denses augmente les chances d'avoir les plus proches voisins dans la même région, et diminue donc le nombre de distances à calculer et par la suite le temps de la recherche, ce gain est d'autant plus important que le nombre de vecteurs et la dimension de l'espace de données sont grands.
Pour cela, nous avons proposé une nouvelle stratégie de découpage de l'espace de données vérifiant les critères cités auparavant . Notre algorithme de découpage est inspiré de l'algorithme de découpage KD-Tree pour le choix de la dimension et de la position de partitionnement. Notre algorithme repose sur l'amélioration des principales limitations de la méthode KDB-Tree adoptée par RA-Blocks, notamment au niveau du partitionnement des nœuds internes de la structure KDB-Tree. Les modifications ainsi que les améliorations que nous avons apportés sont détaillées dans le sous paragraphe suivant.

5.1.2. Modifications et améliorations apportées

Les principales modifications et améliorations que nous avons apporté à la phase de partitionnement de l'espace de données de la méthode RA-Blocks concernent les trois problèmes cités précédemment à savoir, le choix de la dimension de partitionnement, le choix de la position de partitionnement et le fractionnement des nœuds internes de la structure de KDB-Tree. Elles sont résumées dans les trois points suivants :
* Pour contrôler la taille des régions obtenues lors du partitionnent de l'espace de données, nous avons choisi un critère de sélection de la dimension du fractionnent se basant sur la distribution des données. En effet, la concentration des données suivant chaque dimension permet de guider le processus de découpage. Ainsi, les zones contenant peu de vecteurs seront moins partitionnées par rapport aux zones de l'espace où il y a plus de concentration de vecteurs. Ce choix permettra de garantir l'existence d'un nombre significatif de vecteurs dans les régions et par la suite évitera la création des régions triviales.
* Nous avons opté pour l'algorithme standard de subdivision de la méthode KD-Tree pour le choix de la position du fractionnement. A la différence de la stratégie du partitionnement du KDB-Tree, qui choisit la position d'une manière arbitraire (le premier vecteur qui provoque le fractionnement), cette technique a l'avantage de produire à chaque partitionnent deux régions avec le même nombre de vecteurs dans chaque région, évitant ainsi le problème de la création des régions vides ou presque vide lors de la construction de la structure d'index.
* Contrairement à la majorité des techniques d'indexation multidimensionnelles notamment le RA-Blocks qui organise les régions dans une structure arborescente, nous avons utilisé une structure d'indexation linéaire. En effet, la structure arborescente exige certaines propriétés que nous avons présentées auparavant. Ces propriétés entrainent dans la plupart des cas des restructurations de la totalité de l'arbre provocant ainsi le partitionnement des régions peu denses. Nous avons choisi de ce fait de remplacer la structure arborescente puisque celle-ci n'est pas exploitée lors du processus de la recherche, par une structure d'index à liste chainée. Cette structure nous permet de localiser les régions denses et par la suite, de réaliser un partitionnent local, et viser uniquement les régions qui nécessitent le partitionnent (les régions dont le nombre de vecteurs est supérieur à leur capacité).

5.1.3. L'algorithme de découpage de l'espace de données

Notre algorithme du partitionnent en régions est inspiré de la division standard de l'algorithme KD-Tree pour ce qui est de la recherche du côté susceptible d'être divisé, indiquant l'axe de coordonnées orthogonal à l'hyperplan réalisant la division (dimension de partitionnement), et du calcul de la valeur de subdivision suivant cette dimension. La décomposition d'une région saturée s'effectue suivant l'hyperplan de subdivision indiquant la position à laquelle l'hyperplan coupe l'axe de coordonnées en passant par la valeur de subdivision. Les choix de la dimension et de la valeur de la subdivision sont fixés de la manière suivante :

-Dimension de subdivision : la subdivision s'effectue suivant la dimension possédant le maximum de diffusion de données, c.à.d. la dimension qui correspond à la plus grande différence des composantes des vecteurs. Cette condition garantit le partitionnement des zones de l'espace ou il y a plus de concentration de vecteurs, elle permet ainsi la création des régions avec un nombre minimal de vecteurs.

-Valeur de subdivision : la valeur de subdivision est la valeur de quantification la plus proche de la valeur médiane suivant la dimension de subdivision. Le choix de cette valeur prend explicitement en compte la distribution des données, il garantit l'existence d'un nombre équivalent de vecteurs dans les régions produites lors du fractionnement.

D'un autre côté, notre algorithme de découpage repose sur le principe d'éliminer le partitionnement des nœuds internes (subdivision downward) de la structure K-D-B-Tree. Rappelons que 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 (équilibre, pas de chevauchement). 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. D'où l'idée de base de notre méthode qui est de partitionner uniquement les "*pages points*" saturées. Dans ce cas, seules les "*pages points*" non vides sont gérées par notre algorithme de découpage. En effet, notre algorithme élimine le partitionnement des nœuds internes de l'arbre en transformant structure arborescente en une liste de régions. Notre structure est composée uniquement de deux niveaux. Le premier niveau correspond au nœuds/nœuds internes de la structure de KDB-Tree, il contient les régions représentées par leurs approximations ainsi qu'un pointeur sur le niveau inférieur. Dans le deuxième niveau sont stockées les données réelles de chaque région. Les deux niveaux sont stockés dans deux fichiers binaires. Notre structure est construite par insertion successive des vecteurs de la base de données. Lors de l'insertion d'un vecteur dans une feuille saturée, celle-ci est subdivisée en deux feuilles contenant le même nombre de vecteurs, les régions correspondant à la feuille saturée sont également partitionner de la même manière, ils sont par la suite remplacées par deux nouvelles régions (région gauche, région droite) ayant chacune un pointeur vers la nouvelle feuille. Notre structure permet d'une part d'éliminer les régions triviales (vides) et de produire que des régions denses, et d'autre part elle garantit l'existence des $k - ppv$ dans une même page disque, ce qui entraine par la suite la réduction du temps de la recherche.

Notre algorithme de partitionnent de l'espace de données est détaillé dans la figure 3.11.

Algorithme_decoupage_RA$^+$-Blocks *(vecteur data[ ] : les vecteurs réels de la de la base de données, entier N : la taille des vecteurs) retourne ListReg : liste des régions contenant le résultat de la subdivision.*

{

Variables :

//Pos : entier
//Region Rg, Rd : les régions résultantes de la subdivision d'une région mère

***1**. Initialiser ListReg par la région correspondant à l'espace de données*
***2**. Pour i allant de 1 à N Faire{*
***2.1**. Pos= GetIdRegion(data [i]) // retourne l'indice de la région devant contenir le vecteur data[i]*
***2.2**. Soit Reg la région d'indice pos*
***2.3**. Inserer(data[i], Reg) // insérer data[i] dans Reg*
***2.4**. Si (saturer(Reg)) {*
// le nombre de vecteurs de la région Reg>capacité; la région Reg est saturée
***2.4.1**. split(Reg, Rg, Rd) // décomposer Reg en deux sous régions*
***2.4.2**. Si (nbredata(Rg)!=0) et (nbredata(Rd)!=0) {*
// le nombre de vecteurs de Rd et Rg est non nul
***a**. Remplacer(ListReg, Reg, Rg) // remplacer la région Reg par Rg dans ListReg*
***b**. InsererFin(listReg, Rd) // insérer la région Rd à la fin de ListReg.*
}// FinSi
} // FinSi
}// FinPour
}// Fin

Fig.3.11 L'algorithme de découpage de l'espace de données du RA+-Blocks

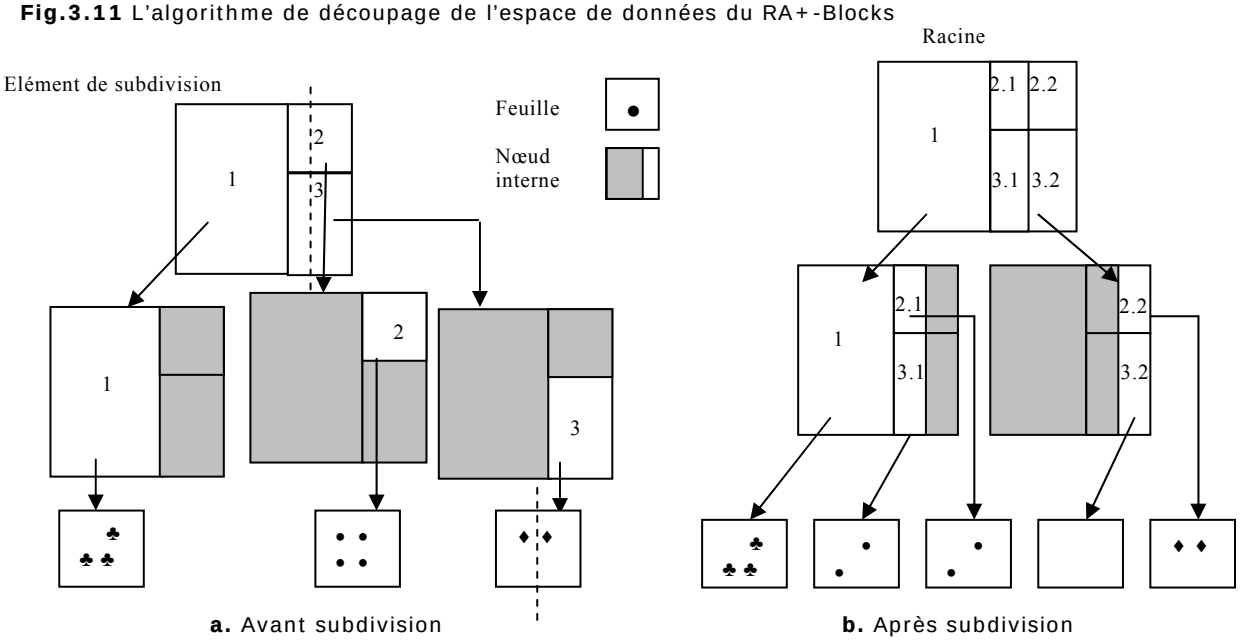


Fig. 3.12 Exemple de subdivision des régions selon K-D-B-Tree

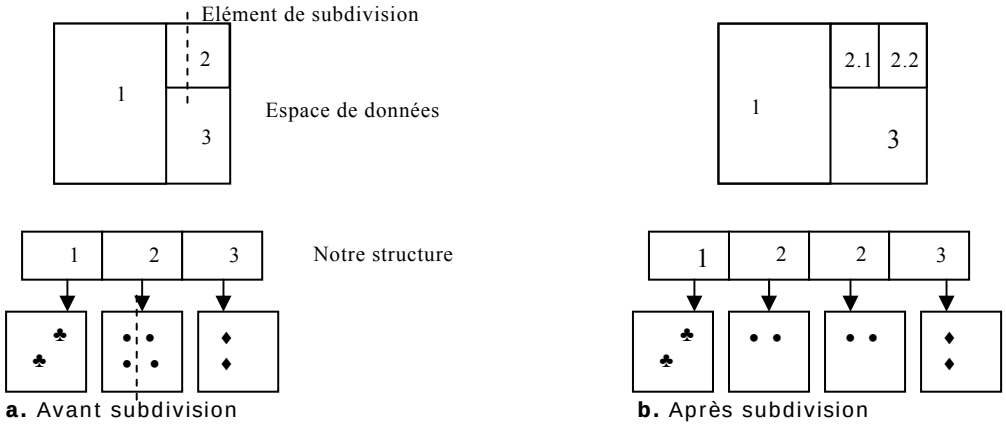


Fig. 3.13. Exemple de subdivision des régions selon notre méthode

Les figures 3.12 et 3.13 présentent un exemple de découpage d'un espace $2d$ par la méthode K-D-B-Tree et par notre méthode où la capacité des régions est fixée à 3. Selon la méthode K-D-B-Tree (fig.3.12), toutes les régions ayant une intersection avec l'élément de subdivision sont partitionnées en deux sous régions, y compris celles qui ne sont pas saturées (région 3). Par la suite, les pages points résultantes sont peu denses et parfois vides (régions 3.1 et 3.2). Par contre, selon notre méthode (fig. 3.13), l'élément de subdivision partitionne uniquement les régions saturées (région 2) évitant ainsi le problème de la création des régions vides ou peu denses.

5.2 Structure d'index

La structure d'index du RA$^+$-Blocks est composée de deux niveaux. Le premier niveau correspond à un fichier de petite taille contenant les approximations codées en bits de chaque région de l'espace de données (voir fig. 3.8). Le deuxième niveau correspond à un fichier où sont stockés les vecteurs réels.

Pour insérer un nouveau vecteur, on cherche la région devant le contenir en se basant sur les approximations. Si la région est saturée, c.à.d le nombre de vecteurs dépasse la capacité de cette région, alors elle sera décomposée en deux sous régions ayant pratiquement le même nombre de vecteurs. Dans ce cas, les approximations des deux nouvelles régions sont calculées et insérées dans le fichier des approximations, puis l'approximation de la région décomposée est supprimée.

La suppression d'un vecteur consiste à localiser la région le contenant. Si le nombre de vecteurs restant après la suppression est inférieur à la capacité/2, alors une restructuration des régions est effectuée.

5.3 Interrogation de la base de données

Tout comme la méthode RA-Blocks, la recherche des $k-ppv$ dans RA$^+$-Blocks est effectuée en deux phases. Dans la première (phase de filtrage), les régions sont parcourues séquentiellement, et les régions candidates sont sélectionnées en se basant sur leurs distances minimales et maximales par rapport au vecteur requête. Lors de la deuxième phase (accès aux vecteurs), les $k-ppv$ sont déterminés en calculant la distance réelle des vecteurs relatif aux régions candidates par rapport au vecteur requête. L'algorithme de recherche du RA$^+$-Blocks est identique à celui du RA-Blocks (voir figure 3.10).

6 Synthèse

Dans ce chapitre, nous avons présenté une nouvelle méthode d'indexation multidimensionnelle basée sur l'approche approximation RA$^+$-Blocks. Cette méthode combine les propriétés de l'algorithme de découpage KD-Tree et l'approximation des régions pour générer des approximations plus précises et compactes par rapport à celles générées par la méthode RA-Blocks. Nous avons tout d'abord présenté les méthodes du découpage de l'espace de données KD-Tree, KDB-Tree et ses variantes. Le découpage de l'espace de données selon KD-Tree s'effectue d'une manière récursif en prenant en considération la distribution des données dans le choix de l'hyper plan de partitionnement (la position et la dimension du partitionnement). La méthode KDB-Tree quand à elle choisie d'une manière arbitraire la position du partitionnement

et d'une manière cyclique la dimension de partitionnement. La distribution des données n'est pas incluse dans le calcul de l'hyper plan de partitionnement. KDB-Tree présente une limitation principale au niveau du processus de partitionnement, celui-ci produit le fractionnement inutiles des nœuds internes de l'arbre générant ainsi un grand nombre de régions vides ou peu denses. Plusieurs méthodes ont été proposé pour résoudre ces problèmes (KDB_{KD}-Tree, FD-splitting, etc). Malheureusement la performance de certaines méthodes sont inférieures à celles du KDB-Tree, et d'autres souffrent des problèmes de la malédiction de la dimension. Dans la deuxième partie de ce travail, nous avons présenté la méthode d'indexation multidimensionnelle RA-Blocks, et nous avons analysé l'influence des problématiques liées à la phase de découpage sur l'indexation et la recherche. Sur la base de cette analyse , nous avons proposé une nouvelle méthode RA^{+}-Blocks d'une part pour répondre aux problèmes liés à la malédiction de la dimension et au passage à l'échelle exposés dans le chapitre 2. Et d'autre part, pour répondre aux principales limitations de la méthode RA-Blocks notamment celles liées au processus du découpage de l'espace de données. Pour répondre à ces problèmes, nous avons proposé un algorithme de découpage de l'espace de données qui tient compte de la distribution des données dans le choix de la position de partitionnement ainsi que la dimension par laquelle passe l'hyper plan de partitionnement. Ces deux conditions nous permettent de garantir un nombre minimal dans les régions obtenues lors du partitionnement et d'éviter ainsi le problème de la création des régions vides ou peu denses.

Le deuxième principal problème dont souffre la phase de découpage de l'espace de données adoptée par la méthode RA-Blocks, est la propagation du processus du fractionnement du haut vers le bas (subdivision downward) qui produit le fractionnement des nœuds internes de l'arbre. Pour répondre à cette problématique, nous avons remarqué que la structure arborescente n'est pas exploitée dans le processus de la recherche. Donc, nous avons remplacé la structure arborescente par une structure linéaire à deux niveaux, où le premier niveau contient les régions et le deuxième contient les vecteurs réelles correspondant aux régions. Le partitionnement avec cette structure est ciblé. Seules les régions denses sont partitionnées et remplacées par deux nouvelles régions contenant le même nombre de vecteurs. Notre structure garantit une bonne gestion de l'espace mémoire alloué à la structure d'index ainsi qu'un meilleur taux d'occupation de l'index. Les régions produites lors du partitionnement sont moins nombreuses et plus denses que celles produites par la méthode de partitionnement adoptée par RA-Blocks. Les résultats expérimentaux confirmeront nos résultats.

Par ailleurs, bien que cette méthode répond aux limitations de la méthode RA-Blocks, RA^{+}-Blocks reste inadaptée aux applications réelles ou la distribution des données est hétérogènes , la taille des données et très grande et la dimension dépasse 100.. Dans le chapitre suivant nous proposons une amélioration de la méthode RA^{+}-Blocks . La méthode proposée prend explicitement en compte les différent problèmes liés à la recherche d'images par le contenu.

Chapitre 4

Nouvelle méthode multidimensionnelle par approche noyau pour l'indexation et la recherche dans des grandes bases d'images basées sur le contenu : KRA$^+$-Blocks

Ce chapitre présente, une nouvelle technique pour la recherche et l'indexation des bases de données multimédia adaptée aux moteurs de recherche d'images basés sur le contenu (SRIC). Contrairement à la plupart des index multidimensionnels, cette technique a été proposée en prenant en compte explicitement les problèmes liés à la nature des données utilisées pour décrire les images. KRA$^+$-Blocks est basée sur l'approximation des régions et la réduction non linéaire de la dimension par l'approche noyau. Nous présenterons d'abord le M-Tree à noyau qui est une méthode d'indexation métrique, mais qui utilise l'approche noyau pour définir de nouvelles distances pour l'indexation et la recherche. Nous présenterons également le KVA-File qui se base sur l'approximation des données et qui s'inspire du même principe que le M-Tree à noyau pour le calcul des distances à noyau. Enfin, nous présenterons notre nouvelle méthode d'indexation KRA$^+$-Blocks.

1 Introduction

La recherche d'images fixes par le contenu dans les grandes bases consiste à sélectionner les images les plus pertinentes à une requête donnée en se basant sur le contenu visuel des images. Ce contenu est généralement représenté par des descripteurs de bas niveau relatifs à la couleur, la texture et la forme.

Pour réaliser une recherche rapide est efficace sur ces descripteurs, des techniques d'indexation multidimensionnelles sont nécessaires. Elles consistent à limiter la quantité des descripteurs à comparer avec le descripteur requête et de diminuer ainsi le nombre d'opérations de calcul de similarité à effectuer. Malheureusement, ces techniques peuvent s'avérer inefficaces dans le domaine de la recherche basée sur le contenu. En effet, les descripteurs visuels (couleur, texture, forme) sont hétérogènes et leurs dimension est grande.

La dimension élevé des descripteurs pose de sérieux problèmes de performance aux méthodes d'indexation multidimensionnelles y compris celles basées sur l'approximation. Les propriétés particulières des espaces de grande dimension et le problème de la malédiction de la dimension font que la qualité de ces index diminuent lorsque la dimension de l'espace de données augmente.
Ensuite, le caractère hétérogène des descripteurs fait que la majorité des techniques d'indexation multidimensionnelles sont inadaptées aux applications de la recherche d'images par le contenu car la plupart de ces méthodes sont conçues et validées sur des données de distributions uniformes et sont liées à un espace Euclidien. De plus, dans d'autres méthodes, notamment RA^+-Blocks, aucune sémantique particulière n'est utilisée lors du groupement des données (ou lors du partitionnent de l'espace de données). L'évaluation de la plupart des méthodes d'indexation multidimensionnelles s'effectue uniquement par rapport à l'algorithme de la recherche des $k - ppv$ qui exploite les régions pour réduire la recherche à un sous ensemble de vecteurs. De même pour le choix de la capacité des éléments de l'index (rectangles), la seule contrainte qui guide le choix de ce paramètre est la réduction du temps de recherche.
Partant de ce constat, nous nous sommes intéressés dans ce travail aux problèmes liés à l'intégration de la méthode d'indexation multidimensionnelle que nous avons proposée dans le chapitre 2, RA^+-Blocks, dans un système de recherche d'images par le contenu. Le but de la méthode proposée est d'optimiser les performances de la recherche par le contenu d'une part en évitant le parcours systématique de toutes les images de la base, et d'autre part, de répondre au problème de similarité qui se pose dans le cas de données hétérogènes, en proposant une nouvelle approche pour mesurer la similarité. Notre technique combine deux méthodes : l'analyse en composantes principales "kernelisée" (ACPK) et l'approche approximation des régions pour améliorer le temps de réponse des systèmes SRIC tout en gardant une bonne qualité de la recherche. Dans notre approche, nous supposons que même si elles sont de nature hétérogène, les données peuvent être regroupées dans une même structure. Pour cela, d'une part la phase d'indexation permet à travers l'approche noyau une bonne représentation et une bonne organisation des données, et d'autre part, les mécanismes de structuration et de recherche font appel à des distances appropriées et adaptées à la nature des données. Nous détaillons dans ce chapitre les différentes étapes de cette méthode (la réduction de la dimension, l'indexation et la recherche par similarité)

2 Techniques d'indexation par approche noyau

Les méthodes à noyau forment une nouvelle famille d'algorithmes aux propriétés intéressantes pour la recherche d'information [Vap 99, Cri 00, Sch 02, Suy 02]. Leur simplicité est liée à leur efficacité, ce qui a fait de ces techniques un outil incontournable pour certains chercheurs et un sujet de recherche fondamental en apprentissage.
Dans ce paragraphe, nous nous intéressons aux méthodes d'indexation basées sur l'approche à noyau. Ces méthodes présentent la particularité de pouvoir introduire la notion d'apprentissage pour les index multidimensionnels et de définir des distances adaptées au besoin de l'utilisateur dans la phase de la recherche. Nous commençons par présenter les deux méthodes basées sur l'approche noyau qui existent dans la littérature, le M-Tree à noyau et le KVA-File.

2.1 M-Tree à noyau

La plupart des techniques d'indexation multidimensionnelles ont pour objectif de réduire le temps de réponse pendant la recherche sans nécessairement s'intéresser aux distances utilisées

pour l'indexation et la recherche. Cependant, il existe des distances à noyau pour lesquelles la majorité de ces techniques ne peuvent être appliquées. Le M-Tree à noyau [Peng 03] est l'une des premières méthodes d'indexation qui a été développée pour supporter d'une manière dynamique les distances à noyau. Cette technique utilise l'approche du noyau pour construire la structure d'index. En effet, le M-Tree à noyau est une structure arborescente similaire au M-Tree et R-Tree, qui se base sur l'inégalité triangulaire ainsi que sur des distances pre-calculées pour réduire le temps de réponses.

Dans un M-Tree à noyau, les nœuds internes de la structure arborescente sont composés d'un ensemble d'objets[3] de redirection (routing object), alors que les feuilles de l'arbre contiennent les données réelles. Un objet de redirection est composé de l'objet lui-même et d'un rayon de couverture (covering radius) qui est égal à la distance maximale entre l'objet de redirection et l'objet le plus lointain compris dans le sous-arbre correspondant à l'objet de redirection. Au niveau de l'arbre, à chaque objet est associée sa distance par rapport à son objet de redirection. Cette distance calculée lors de la construction du M-Tree permet d'appliquer l'inégalité triangulaire lors de la recherche afin d'éviter certains calculs de distance inutiles et ainsi réduire le temps de réponse. Lors de la recherche effectuée à la suite d'une requête, l'inégalité triangulaire est d'abord appliquée pour éliminer une partie des branches non pertinentes sans calculer la distance entre le vecteur requête et tous les objets stockés dans les nœuds. Cela est possible car la distance entre chaque objet stocké dans le nœud et son objet de redirection est stockée dans l'arbre et la distance entre le vecteur requête et l'objet de redirection du nœud a déjà été calculée.

La construction de la structure d'index du M-Tree dans l'espace à noyau se fait de la même manière que celle du M-Tree [Peng 03] en introduisant la distance $Dist(V_1,V_2) = k(V_1,V_1) - 2k(V_1,V_2) + k(V_2,V_2)$ tels que V_1 et V_2 sont deux vecteurs de la base de données. La même distance est utilisée pour la recherche des $k-ppv$.

Le M-Tree à noyau a été étendu à l'espace à noyau afin de réaliser l'apprentissage des distances en utilisant le bouclage de pertinence. Cette méthode a été particulièrement appliquée au cas du One-class SVM[Sch 01]. Cette dernière a pour but d'estimer le support qui inclut la plupart des éléments positifs de la base d'apprentissage, i.e. trouver une fonction positive dans une région qui englobe la plupart des données et négative ailleurs. Cette approche est équivalente à trouver la surface qui sépare les données positives et l'origine pour un certain seuil τ (figure 4.1) :

$f(p) = sign(w.\Phi(p) - \tau)$ tel que $w.\Phi(p) - \tau$ représente l'hyperplan séparateur optimal

Si N est le nombre d'élément (appartenant à la classe positive)de la base d'apprentissage $V_i (i = 1,...,N)$, le problème à maximiser est : $\min_{w,\tau,center} [\frac{\|w\|^2}{2} - \tau + \frac{1}{cst.N} \sum_i center_i]$, tels que

[3] Nous avons repris la terminologie utilisée dans [Cia 97]. Un objet est tout simplement un vecteur multidimensionnel appartenant à la base de données.

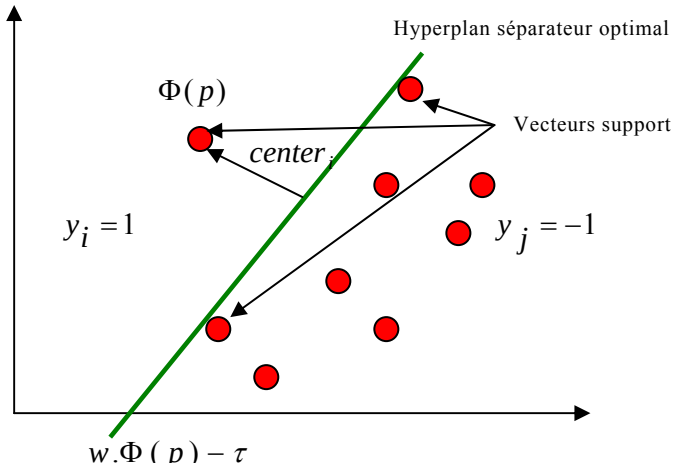


Fig. 4.1. One-Class SVM

$\|\Phi(V_i) - center\|^2 \leq w^2 + center_i$, $center_i \geq 0$ pour tout i et où $cst \in [0,1]$ contrôle le nombre des recherches pertinentes qui peuvent être incluses dans l'hypersphère.
En introduisant les multiplicateurs de Lagrange, on obtient comme solution au problème d'optimisation précédant :

$$center^* = \sum_i \alpha_i \Phi(V_i) \quad (4.1)$$

tels que $\sum_i \alpha_i = 1$ et $0 \leq \alpha_i \leq \frac{1}{cst.N}$. Notons que le centre $center^*$ (Eq. 4.1) est une combinaison convexe des recherches pertinentes dans l'espace à noyau. Les distances à noyau relatives à One-class SVM sont ainsi calculées comme suit :

$$D(V,q) = k(q,q) - 2\sum_i \alpha_i k(v_i,q) + \sum_{i,j} \alpha_i \alpha_j k(v_i,v_j) \quad (4.2)$$

Lors du processus du bouclage de pertinence utilisant les distances à noyau des One-class SVM, $center^*$ (Eq. 4.1) est initialisé par la projection du vecteur requête dans l'espace à noyau $\Phi(q)$.
Après chaque itération, $center^*$ est mis à jour dynamiquement par le bouclage de pertinence (Eq.4.1). Puisque Φ est une fonction non linéaire, nous avons $\sum \alpha_i \Phi(V_i) \neq \Phi(\sum \alpha_i V_i)$. Ainsi, le critère de la recherche des $k-ppv$ devient $\|\Phi(q) - \sum \alpha_i \Phi(V_i)\|^2$.
Notons que pour le M-Tree à noyau, les distances utilisées pour la recherche et la construction de la structure d'index sont les mêmes. Ceci constitue la principale limite de cette méthode et des méthodes métriques en générale. En effet, pour les approches telles que One-class SVM, qui ne modifie pas la fonction noyau pendant le processus du bouclage de pertinence, le M-Tree à noyau peut être appliqué efficacement. Par contre, si au cours du processus d'apprentissage le bouclage de pertinence modifie la fonction noyau, comme tel est le cas de l'AQK (adaptive Quasiconformal kernel), la structure d'index du M-Tree à noyau n'est plus valide. Une telle modification suppose la reconstruction de la structure d'index avec la nouvelle distance, or cette opération est très couteuse en terme de temps de calcul ce qui rend cette technique inapplicable. Le KVA-File a été proposé pour contourner ce problème. Son principe de base est décrit dans le paragraphe suivant.

2.2 KVA-File

KVA-File (Kernel Vector Approximation File) [Dou 05] est une méthode d'indexation constituée de deux étapes. Une étape de compression dans laquelle les données sont approximées par une chaine de bits et une étape de recherche des $k-ppv$ dans laquelle le fichier d'approximation est parcouru pour calculer les voisins les plus proches du vecteur requête. Nous verrons que ces deux étapes sont elles-mêmes composées de deux phases chacune.

En fait, le principe de base du KVA-File repose sur la compression des données. 2 niveaux de compressions sont utilisés pour générer le fichier compressé des approximations.

Le premier niveau consiste d'abord à projeter les données initiales dans un espace de dimension réduite par une application non linéaire

$$\Phi : \chi \to F$$
$$p \to \phi(p)$$

Ensuite chaque vecteur, sera représenté par sa projection dans l'espace de caractéristiques de dimension réduite, plus l'erreur associée à cette projection. Pour ce faire, Douglas et Jing Peng [Dou 05] utilisent une base orthonormée composée d'un certain nombre de vecteurs. Le nombre de vecteur de la base orthonormée est fixé à priori, et constitue la nouvelle dimension de l'espace de caractéristiques. La construction de la base orthogonale, les projections initiales des vecteurs ainsi que l'erreur de projection associée à chaque vecteur sont calculées au moyen du procédé d'othogonalisation de Gram Schmidt et l'ACPK. L'algorithme incrémentale d'orthogonalisation est décrit dans [Dou 05].

Les données dans l'espace de projection sont ainsi exprimées en fonction des vecteurs de la base orthogonale en plus de l'erreur d'arrondie.

Soit $e_i (i = 0,...,d-1)$ la base orthonormée de l'espace projeté, p un point dans cet espace exprimé par [Dou 05] :

$$P = \phi^{\perp}(x_p) + \sum_{t=0}^{d-1} \alpha_t v_t$$

avec d le nombre de vecteurs de la base. La distance entre deux vecteurs p et q dans l'espace de projection est donnée par [Dou 05]:

$$dist(Q,P)^2 = \phi^{\perp}(xq)^t \phi^{\perp}(x_q) + \phi^{\perp}(x_p)^t \phi^{\perp}(x_p) - 2\phi^{\perp}(x_q)^t \phi^{\perp}(x_p) + \sum_{t=0}^{d-1} (\alpha_t - \beta_t)^2, \qquad (4.3)$$

les α_t et β_t sont les composantes des vecteurs p et q dans l'espace de projection. Le terme $\phi^{\perp}(x_q)^t \phi^{\perp}(x_q)$ représente le module de l'erreur de projection, donnée par : $\phi^{\perp}(x_q)^t \phi^{\perp}(x_q) = \sqrt{\phi^{\perp}(x_q)^t \phi^{\perp}(x_q)} \sqrt{\phi^{\perp}(x_q)^t \phi^{\perp}(x_q)} \cos\theta$ où θ est l'angle entre les deux vecteurs p et q.

Le deuxième niveau de compression est similaire à celui du VA-File. Il consiste d'abord à partitionner l'espace de caractéristiques en cellules hyper rectangulaires contenant le même nombre de données réelles, puis à coder en bits l'ensemble des localisations des cellules obtenues. Le nombre de bits de codage est fixé à priori.

En résumé, l'étape d'approximation est représentée par deux phases, une phase dite de transposition de données réelles au cours de laquelle les données réelles sont projetées dans un espace de dimension réduite, et une seconde phase de compression, consistant à partitionner l'espace de projection en cellules, puis a approximer chaque cellule par une chaîne de bits.

L'étape de recherche des plus proches voisins s'effectue également en deux phases : d'abord le fichier des approximations est parcouru pour éliminer les cellules non pertinentes, c'est la phase

de filtrage. Cette phase se base sur les distances $d_{\min}$ et $d_{\max}$ entre le vecteur requête et les différentes cellules données par [Dou 05]:

$$d_{\min}(q,p)=\sqrt{G_d(q,q)+G_d(p,p)+2\sqrt{G_d(q,q)}\sqrt{G_d(p,p)}+\sum_{t=0}^{d-1}(\alpha_t-\beta_t)^2} \quad (4.4)$$

$$d_{\max}(q,p)=\sqrt{G_d(q,q)+G_{d}(p,p)-2\sqrt{G_d(q,q)}\sqrt{G_{d}(p,p)}+\sum_{t=0}^{d-1}(\alpha_t-\beta_t)^2} \quad (4.5)$$

avec $G_d(q,q)=\Phi^{\perp}(q)^t\Phi^{\perp}(q)$ représente le module de l'erreur de projection.

Les cellules sélectionnées sont ensuite ordonnées selon leur distance minimale, et des accès directs sont effectués au fichier de données selon les distances réelles (Eq.4.3)

Pour optimiser la qualité de la recherche, KVA-File exploite les propriétés de l'espace projeté (espace à noyau) afin d'apprendre, à partir des données, de nouvelles distances à noyau, notamment à l'aide de mécanismes de bouclage de pertinence. Contrairement à la méthode M-Tree, KVA-File peut être utilisée dans le cadre de l'apprentissage avec des approches qui modifie le noyau au cours du bouclage de pertinence. En effet, dans [Dou 05] l'auteur montre qu'en utilisant uniquement les approximations des vecteurs dans l'espace de projection, on peut facilement mettre en œuvre des approches d'apprentissage qui modifient d'une manière consécutive la fonction noyau. Particulièrement, la méthode AQK (Adaptive Quasiconformal kernel) modifie le noyau au cours du bouclage de pertinence, cependant la structure d'index de KVA-File reste intacte et les distances minimales, maximales et réelles sont mises à jours en se basant uniquement sur le bouclage de pertinence et sur les approximations des vecteurs.

Les performances du KVA-File, en terme de qualité, dépendent essentiellement de la pertinence de la projection et des distances utilisées dans les étapes d'approximation et de recherche, alors que les performances en terme de temps de réponse dépendent de la taille du fichier des approximations et du taux de filtrage.

Concernant la qualité de la recherche, celle-ci va notablement se dégrader si la projection des données initiales n'est pas pertinente. En effet, dans KVA-File la projection non linéaire des données initiales dans un espace de dimension réduite se fait au moyen de l'algorithme d'orthogonalisation de Gram Schmidt. Ce dernier, réalise une transposition de manière itérative, il est facile et rapide dans le cas d'un très grand nombre de vecteurs, en revanche, il présente des limitations au niveau de l'erreur associée à chaque localisation. Cette erreur augmente au fur et à mesure que la dimension de l'espace des composantes augmente. Le procédé d'othogonalisation est non stable pour les grandes dimensions, cependant il est plus pratique en terme de complexité et de temps de réponse lors du passage à l'échelle.

Le taux de filtrage quant à lui est lié aux règles de filtrage et à la taille du fichier des approximations, sachant que celui-ci dépend du nombre de bits de codage et du nombre de vecteurs de la base orthogonale de l'espace de projection. En effet, pour des valeurs importantes de ces paramètres, il sera généré d'une part une grille de codage fine et d'autre part un espace de projection de grande dimension. Ce qui aura pour conséquence de créer un fichier d'approximation de taille importante et donc d'augmenter fortement le temps de réponse. En réalité, ces deux facteurs agissent de manière contradictoire sur la qualité de l'approximation. Attribuer une grande valeur au nombre de bit de codage permet d'avoir des approximations précises, alors qu'une grande valeur du nombre de vecteur de la base orthogonale permet de générer une approximation mois précise (plus de variance entraîne plus de bruit).

3 KRA$^+$-Blocks : structure d'indexation multidimensionnelle pour la recherche par le contenu

KRA$^+$-Blocks est basée sur l'approximation des régions et la réduction non linéaire de la dimension. De part sa conception, cette méthode est très bien adaptée aux SRIC travaillant sur des bases de très grandes tailles. Elle réduit la dimension, indexe la base pour réduire les temps d'accès et mesure la similarité à l'aide de distances prenant en compte la nature hétérogène des données et donc améliorant la qualité de la recherche. Enfin, elle supporte un mécanisme de bouclage de pertinence, ce qui permet d'inclure l'utilisateur dans la boucle. Nous décrivons d'abord, le principe de base de l'ACPK et nous montrons ses propriétés et son intérêt dans le cadre de la classification et la structuration des données. Nous présentons ensuite, le modèle de similarité adopté par cette méthode, ensuite nous montrons comment l'organisation des descripteurs des images en régions et l'utilisation de la fonction noyau s'intègrent dans le processus de l'indexation et de la recherche et améliorent, comme nous le prévoyons, la qualité et le temps de la recherche.

3.1 Réduction de la dimension

Kernel PCA [Sch 98] ou ACP kernelisée (ou à noyau) est une méthode non linéaire de la réduction de la dimension. C'est une généralisation de l'ACP qui considère les relations non linéaires entre les données.

Il suffit de remplacer chaque produit scalaire $< p,q >$ par un noyau $k(p,q)$ dans l'algorithme de l'ACP. Le choix de k permettra alors de considérer des produits scalaires dans un espace de données possiblement non linéaire plutôt que dans l'espace original d'observation. En effet, le calcul de l'ACPK ne fait intervenir que des produits scalaires entre les points (pour le calcul de la matrice de covariance) et ne considère jamais les coordonnées d'un point isolé. Si l'on remplace le produit scalaire par un noyau, on calcule donc les composantes principales dans l'espace de caractéristiques H , et on peut ainsi accéder à des corrélations d'ordre supérieur entre les variables. Si $\Phi(p)$ est la représentation d'un vecteur p dans l'espace de données, le noyau calcule le produit de deux vecteurs dans l'espace de données de dimension réduite.

$$k(p,q) =< \Phi(p), \Phi(q) >$$

Formellement, l'ACPK peut s'exprimer de la manière suivante :

Etant donnée un ensemble de vecteurs S appartenant à un espace d'entrée χ L'idée est de projeter l'ensemble des vecteurs S dans un espace H, appelé espace à noyau, au moyen d'une application non linéaire ϕ, associée à un noyau k, telle que $k(p,q) =< \Phi(p), \Phi(q) >$, p et $q \in S$,

l'ACPK calcule l'ensemble des vecteurs propres e_i , et leurs valeurs propres correspondantes λ_i, en utilisant l'équation :

$$Cov.e_i = \lambda_i e_i \qquad (4.6)$$

Cov étant la matrice de covariance donnée par:

$$Cov = \frac{1}{N} \sum_{p \in S} \Phi(p)\Phi(p)^t \qquad (4.7)$$

Avec un noyau k, on peut utiliser l'astuce du noyau pour calculer les produits scalaires dans l'espace transformé sans jamais avoir à faire la transformation explicitement :

$$\Phi^t(p)\Phi(q) = k(p,q)$$

où N est le nombre de vecteurs de l'ensemble S. Comme montré dans [Sch 98], le problème de l'Eq. 4.6 peut être formulé de la manière suivante:

$$KZ_i = \lambda_i Z_i \qquad (4.8)$$

tel que $Z_i = (z_{i,1},...,z_{i,N})$ est un vecteur dont les composantes vérifient:

$$e_i = \sum_{p \in S} Z_{i,rank(p)} \Phi(p) \qquad (4.9)$$

et K est une matrice symétrique définie par: $K_{p,q \in S} = k(p,q)$

Les Z_i sont obtenus en résolvant l'Eq.4.8. Les projections α_i^p du vecteur p suivant la i^{th} composante principale sont données par:

$$\alpha_i^p = \langle e_i, \Phi(p) \rangle = \sum_{q \in S} Z_{i,rank(q)} k(p,q) \qquad (4.10)$$

Le calcul de KACP se ramène à une diagonalisation de la matrice $K_{i,j}$

Cet algorithme est valable pour des observations centrées $\sum_{i=1}^{n} \Phi(p_i) = 0$. Le centrage des données est facile à assurer dans l'espace d'origine, beaucoup plus difficile dans l'espace H puisque l'on ne peut pas calculer explicitement la moyenne des observations projetées dans cet espace. Il existe cependant une solution à ce problème il s'agit de calculer la matrice de Gram à la place de la matrice $K_{i,j}$. La reformulation de l'algorithme de l'ACP à noyau en tenant explicitement en compte du terme de la moyenne est détaillée dans l'annexe.

Notons que le choix de la dimension de l'espace réduit ainsi que les paramètres de la fonction noyau est critique et doit être effectué a priori. Ceci constitue le principal inconvénient de cette méthode. Malheureusement il n'existe aucune méthode universelle permettant de trouver la valeur optimale de la dimension de l'espace réduit ainsi que le paramètre du noyau. Notons qu'il est toutefois possible de se servir de ces différents paramètres pour la classification ou la formation des classes dans le domaine de l'indexation et de la recherche des $k-ppv$ comme nous allons le présenter dans le paragraphe suivant.

3.2 Propriétés de l'ACPK

Nous nous intéressons dans ce paragraphe à l'intérêt de l'ACPK dans le cadre de la classification et la structuration des données, l'objectif est de monter que la dimension de représentation des données et le rayon de la fonction noyau (gaussienne) peuvent modifier d'une manière considérable la dispersion des données dans l'espace de projection ainsi que la structure des données transposées.

Nous commençons par montrer expérimentalement l'influence du rayon de la gaussienne sur la distribution des données, ensuite nous montrons l'existence d'un lien entre la dimension de l'espace de projection et le rayon de la gaussienne. Pour cela, nous avons pris trois ensembles de vecteurs appartenant à des classes différentes, chaque classe étant composée de 40 vecteurs de dimension 10 chacun. Notant que la 3ème classe est composée de deux sous classes adjacentes.

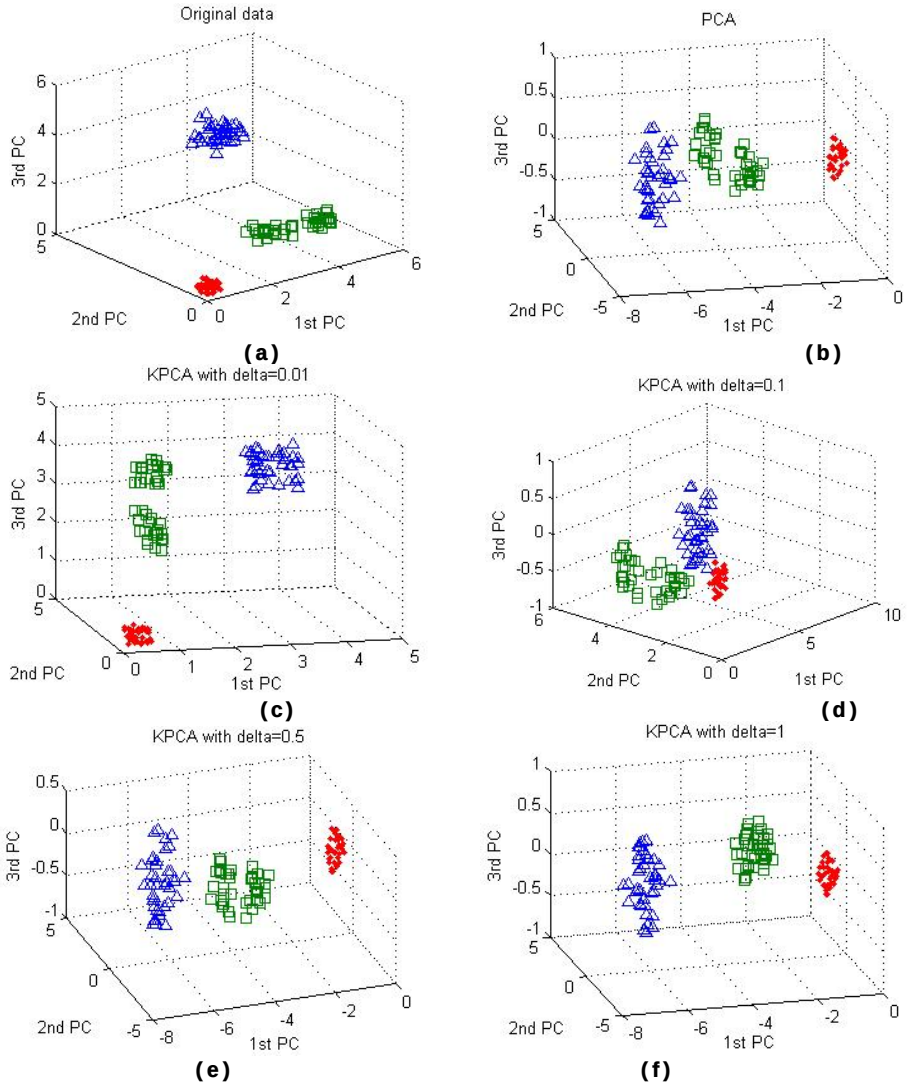


Fig. 4.2. PCs for different δ values. (a) Original data. (b) PCA. (c) KPCA, $\delta = 0.01$. (d) KPCA, $\delta = 0.1$. (e) KPCA, $\delta = 0.5$. (f) KPCA, $\delta = 1$

Outre la réduction de la dimension, l'ACPK permet de séparer un ensemble de vecteurs en utilisant la proximité des vecteurs dans l'espace, l'objectif étant d'effectuer une séparation pour laquelle les vecteurs proches dans l'espace se trouvent dans le même groupe. Dans notre exemple, une bonne méthode de séparation de données aurait tendance à mettre les 3 classes de vecteurs dans 4 groupes différents (les 2 premières classes, et les 2 sous classe), alors que d'autres méthodes de classification non optimales ne seraient pas capables d'identifier et d'isoler les deux sous classes adjacentes du 3^{eme} groupe de vecteurs.

Pour illustrer l'intérêt de l'ACPK et le choix de ses paramètres dans le cadre de la classification, nous avons appliqué l'ACP (l'analyse en composante principale linéaire) ainsi que l'ACPK sur l'ensemble des 3 classes décrites précédemment et nous avons ensuite projeté les données sur les 3 premières composantes principales en utilisant différentes valeurs de δ. La figure (4.2.a) montre les données originales dans un espace de dimension trois, la figure 4.2.b, représente les 3 premières composantes principales de l'ACP, et les figures 4.2.c- 4.2.f représentent les trois composantes principales obtenues par l'ACPK avec des rayons respectivement égaux à 0.01, 0.1, 0.5 et 1.

D'après les figures, on remarque que le rayon de la gaussienne δ agit sur la structure des données transposées dans l'espace de représentation. Plus le rayon de la fonction gaussienne est

petit, plus l'ACPK produit une bonne séparation des données. En choisissant la valeur du rayon de la gaussienne, on choisit de manière implicite une transformation qui va modifier la structure des données dans l'espace de représentation. L'avantage potentiel de cette propriété dans la recherche d'images basée sur le contenu, particulièrement dans le cas ou la recherche d'images s'effectue à travers un index multidimensionnel, réside dans le fait d'avoir une certaine flexibilité dans la construction de la structure d'index. En effet, les différentes projections des vecteurs permettent de produire différentes distributions des données, générant ainsi des structures d'index différentes avec les mêmes données initiales. Le choix du rayon de la gaussienne peut modifier de manière importante la structure d'index ainsi que les résultats de la recherche. Il est donc possible de se servir de cette propriété de l'ACPK pour mieux séparer les données avant l'étape d'indexation.

Prenons à présent le cas où le rayon de la fonction gaussienne est fixe et la dimension de l'espace de représentation est variable. Supposons que les vecteurs propres de l'ACPK sont stockés par ordre décroissant de leurs valeurs propres. Généralement, les premières valeurs propres, correspondant aux premiers vecteurs propres, contiennent la majorité de l'information. En revanche, la quantité d'information contenue dans les premières composantes principales dépend aussi du rayon de la gaussienne comme le montre le tableau 4.1. Pour une grande valeur du rayon, nous atteignons les 99% avec peu de dimensions, alors que pour une petite valeur du rayon nous avons besoin d'une dimension plus grande pour capter un maximum de variance. Ainsi, la dimension de l'espace de projection et le rayon de la gaussienne affectent largement la pertinence de la projection utilisée pour représenter les données dans l'espace projeté.

δ	PC1	PC1 to PC2	PC1 to PC3	PC1 to PC4	PC1 to PC5	PC1 to PC6
0,01	0,11	0,22	0,33	0,44	0,56	0,67
0,10	0,11	0,22	0,33	0,44	0,56	0,67
0,50	0,23	0,40	0,53	0,63	0,72	0,82
1	0,57	0,73	0,82	0,87	0,91	0,95
2	0,85	0,92	0,95	0,97	0,98	0,99
3	0,93	0,96	0,98	0,99	0,99	0,99

Tableau 4.1. La variance cumulée dans les premières composantes principales en fonction de δ. Les zones en couleur grise correspondent à une variance cumulée supérieure ou égale à 98%

Finalement, en peut conclure de ces expérimentations que l'ACPK possède des propriétés intéressantes qui peuvent être exploitées pour mieux gérer la distribution des données dans l'espace de projection.

De notre point de vu, les propriétés souhaitables pour qu'une méthode d'indexation puisse être utilisée efficacement dans ce contexte peuvent être résumées dans les points suivants

Compacité : Il est important d'avoir des groupements de données très compacts qui se chevauchent le moins possible, afin d'accroître l'efficacité de la méthode de partitionnement et les règles de filtrage, et de confiner ainsi la recherche à un sous ensemble de vecteurs similaires.

Bruit : Il est important de supprimer les données redondantes, autrement ces données seront associées à des groupements et produiront des partitionnent inutiles, rendant ainsi le processus du filtrage complètement inefficace

L'équilibre : L'équilibre des groupements en nombre de vecteurs est une propriété importante qui peut affecter la performance du processus de recherche. Une grande variabilité du nombre

de vecteurs dans chaque groupement peut entrainer la lecture d'un nombre important de vecteurs de la base même si le taux de sélection des groupements dans la phase de filtrage est réduit.

Espace de grande dimension : les données manipulées dans le domaine de l'indexation sont des vecteurs de grande dimension. Par conséquent, il est important que la méthode de séparation de données prenne explicitement en compte le facteur de la dimension.

Le choix des bons paramètres de l'ACPK, permet de moduler ces propriétés de manière à obtenir la bonne projection des données dans l'espace de projection.

3.3 Indexation

L'objectif de l'étape d'indexation est de combiner l'approche noyau avec une méthode d'approximation de régions pour établir une représentation compacte est informative des différents descripteurs de la base d'images, et par conséquent, réduire le temps de parcours lors de la recherche. Le processus d'indexation du KRA$^+$-Blocks est composé de deux niveaux de compression. Le premier niveau consiste à projeter les vecteurs multidimensionnels dans un espace de dimension réduite par une fonction noyau, alors que le deuxième consiste à regrouper les vecteurs projetés dans des régions compactes et disjointes. Par la suite, chaque région est codée par une chaine de bits formant une représentation compressée des données

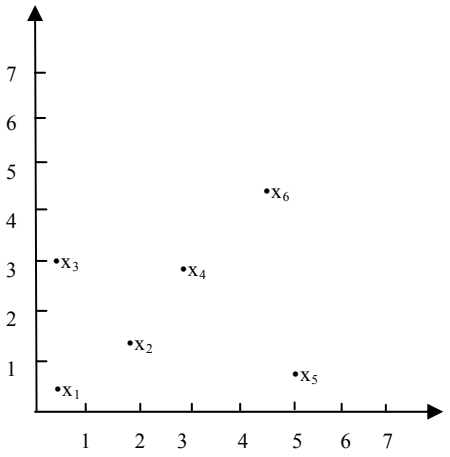
Pour calculer les approximations géométriques, les vecteurs descripteurs des images sont d'abord projetés dans un espace de dimension réduite par une ACPK en utilisant un noyau Gaussien. L'équation 4.9 sera utilisée pour calculer la base orthogonale de vecteurs propres nécessaire à une telle projection. Ainsi chaque vecteur p dans χ, correspond une projection $\Phi(p)$ dans F, qui peut être exprimée par:

$$\Phi(p) = \sum_{i=0}^{d-1} \alpha_i^p e_i + \alpha_d^p e^{\perp} \qquad (4.11)\ ,$$

avec α_i^p la $i^{ème}$ composante du vecteur p dans H (Eq. 4.11) et $e^{\perp}$ une composante orthogonale à la base des vecteurs propres "e", représentant la direction de l'erreur de projection, et dont la magnitude $(\alpha_d^p e^{\perp})^T (\alpha_d^p e^{\perp})$ est donnée par :

$$(\alpha_d^p e^{\perp})^T (\alpha_d^p e^{\perp}) = \frac{\sum_{i=d}^{d'} \lambda_i}{\sum_{i=0}^{d'} \lambda_i} \qquad (4.12)$$

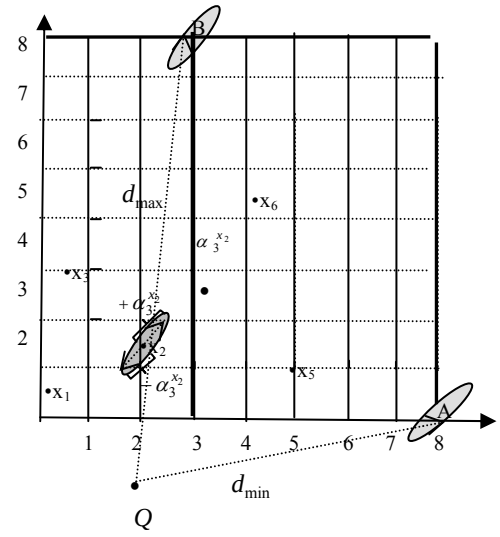
avec d' et d respectivement la dimension de l'espace χ et F, et λ_i les valeurs propres de KPCA. En se basant sur la représentation réduite des vecteurs, chaque dimension d_i de l'espace de projection est partitionnée en 2^{b_i} intervalles et où chaque intervalle est codé par b_i bits. L'espace de donnée est ensuite partitionné en régions suivant la même stratégie de partitionnement que celle utilisée par RA$^+$-Blocks. Chaque région est ensuite codée par deux chaines de bits formant le fichier d'approximations des régions à gérer dans la phase de la recherche. La figure 4.3 représente un exemple d'approximation d'un ensemble de 6 vecteurs de dimension deux selon la méthode KRA$^+$-Blocks.



(a)

	x	Φ $\alpha_1^{x_1}$ $\alpha_2^{x_1}$ $\alpha_3^{x_1}$
1	0.5 0.5	0.73 0.46 0.20
2	2 1.5	2.22 1.88 0.21
3	0.5 3	4.26 0.26 0.01
4	3 3	4.40 2.76 0.09
5	5 1	4.75 2.74 0.06
6	4.5 4.5	6.62 4.64 0.07

(b)



(c)

Fig. 4.3 KRA+-Blocks approximations. (a) données originales. (b) les données projetées avec ACPK. (c) les bornes minimales et maximales

3.4 Mesure de similarité

La recherche des contenus visuellement similaires est une étape primordiale dans les moteurs SRIC. De nombreux algorithmes regroupent les caractéristiques calculées sur une image dans un vecteur multidimensionnel appelés "descripteurs". Généralement, les descripteurs se présentent sous forme de vecteurs numériques de grande dimension auxquels est associée une mesure de similarité (une distance). Cette mesure permet de quantifier la proximité entre descripteurs, et par conséquent, la similarité visuelle entre images.

Parmi les techniques de similarité qui ont été introduites et popularisées par les supports à Vaste Marge (SVM) [Bos 92], certaines utilisent des fonctions noyaux [Sch 02]. Celles-ci offrent, comme nous l'avons déjà vu, un cadre théorique qui permet, grâce à la théorie du « Kernel Trick », la transformation d'un espace de caractéristiques en injectant cet espace initial dans un espace de plus grande dimension.

Ce paragraphe reprend la notion de la mesure de similarité par une approche noyau que nous allons ensuite appliquer à l'indexation et la recherche par le contenu dans les bases de données hétérogènes. Avant de présenter l'approche de similarité par fonction noyau, rappelons d'abord la notion du noyau.

Définition 1 :

L'ensemble de définition D_f d'une fonction f dont l'ensemble de départ χ et l'ensemble d'arrivée H, est l'ensemble des antécédents de f, i.e l'ensemble des éléments de χ que f met en relation avec des éléments H ; c'est donc l'ensemble des éléments x de χ pour lesquels $f(x)$ existe: $D_f = \{x \in \chi / \exists\, y \in H / y = f(x)\}$

Définition 2 :

Soit R et S deux relations binaires sur χ et H respectivement et $f : \chi \to H$ une application de χ dans H. f est un morphisme de (χ, R) dans (H, S) si et seulement si :

$\forall (x, y) \in \chi^2,\ x\, R\, y \Rightarrow f(x)\, S\, f(y)$

Le noyau d'un morphisme est interprété comme une relation d'équivalence sur l'ensemble de définition (plus particulièrement, la relation qui relie les éléments qui ont la même image par le morphisme). Avec l'approche noyau, on peut obtenir un cadre mathématique formel à partir duquel une mesure de similarité pour des données hétérogènes peut être établie et démontrée. Une mesure monomorphe unique peut être établie pour tous les types de données plutôt que d'avoir une mesure associée à chaque type. Cela permet d'optimiser le temps de recherche dans un espace multidimensionnel hétérogène

Rappelons que l'approche de la similarité par fonction noyau consiste à utiliser un produit scalaire comme fonction de similarité. Soit p et q les deux descripteurs correspondant à deux images I_p et I_q. La similarité entre I_p et I_q peut être exprimée par le produit scalaire de p et q : $s(p,q) = < p, q >$

Supposons que nous ayons une injection $\phi : X \to \mathrm{H}_k$, qui a tout vecteur p fait correspondre un vecteur $\Phi(p)$ dans un espace Hilbertien H_k associé à un noyau k, telle que la fonction de similarité soit donnée par le produit scalaire dans H_k : $sim(p,q) = < \Phi(p), \Phi(q) >$. Ce produit scalaire sur les projections des vecteurs par une injection est une fonction noyau $k(p,q)$ sur X (théorème de Mercer [Hei 01]) :

$$k : X \times X \to \Re$$
$$p, q \to k(p,q) = < \Phi(p), \Phi(q) >$$

Le noyau le plus utilisé dans la littérature est le noyau Gaussien :

$$k(p,q) = e^{-\frac{\|p-q\|^2}{2\delta^2}}$$

où δ est le paramètre de l'échelle. Ce paramètre permet de définir d'une manière implicite l'espace de projection à partir duquel on déduit la mesure de similarité. Généralement, une petite valeur de δ correspond à un espace de données riche dans le sens où les frontières entres les différentes classes de données sont bien séparées, tandis qu'une grande valeur de δ correspond à un espace de données pauvre dans le sens où les données sont moins discriminantes dans l'espace projeté. La mesure de similarité en terme noyau peut être formalisée de la manière suivante :

$$dist(\Phi(p), \Phi(q))^2 = k(p,q) + k(p,q) - 2k(p,q) \qquad (4.13)$$

Sachant que les approximations des deux vecteurs p et q dans l'espace transposé H_k sont respectivement données par :

$$\Phi(p) = \sum_{i=0}^{d-1} \alpha_i^p e_i + \alpha_d^p e^{\perp},\ \Phi(q) = \sum_{i=0}^{d-1} \alpha_i^q e_i + \alpha_d^q e^{\perp}.$$

L'expression de la distance entre p et q dans H_k devient :

$$dist(\Phi(p),\Phi(q))^2 = \left\| (\sum_{i=0}^{d-1} \alpha_i^p e_i + \alpha_d^p e^{\perp}) - (\sum_{i=0}^{d-1} \alpha_i^q e_i + \alpha_d^q e^{\perp}) \right\|^2 \quad (4.14)$$
$$= \sum_{i=0}^{d-1} (\alpha_i^p - \alpha_i^q)^2 + \alpha_d^p e^{\perp}(\alpha_d^p e^{\perp})^T + \alpha_d^q e^{\perp}(\alpha_d^q e^{\perp})^T - 2\alpha_d^p e^{\perp}(\alpha_d^q e^{\perp})^T$$

Grâce à l'approche noyau il est possible de définir une distance généralisée qui permet de prendre en compte différents types de descripteurs et de gérer les corrélations entre les attributs de ces vecteurs. Toute fois, bien que l'idée de la similarité par une approche noyau soit simple, sa mise en œuvre reste difficile. En effet, le nombre de composantes des vecteurs pour les projections ainsi que le paramètre de l'échelle sont variables, difficiles à fixer et critiques, puisque la précision de la recherche en dépend.

3.5 Recherche

L'algorithme de recherche des $k-ppv$ utilise les approximations des régions issues de la phase de découpage pour accroitre les performances de la recherche. Le principe de base consiste à utiliser les règles de filtrage qui, d'une part permettent d'éviter le parcours séquentiel inutile des approximations, et d'autre part d'arrêter la recherche dès que toutes les approximations des régions restant à parcourir ne sont pas pertinentes. La recherche est effectuée en deux phases. Dans la phase de filtrage les approximations des régions sont parcourues séquentiellement et les régions candidates sont sélectionnées. Lors de la deuxième phase, dite d'accès, les $k-ppv$ sont déterminés en calculant les distances réelles entre les vecteurs des régions candidates et le vecteur requête.

Notons que l'algorithme de recherche des $k-ppv$ du KRA$^+$-Blocks est similaire à celui du RA$^+$-Blocks et RA-Blocks, la différence résidant dans le calcul des distances utilisées dans le processus du filtrage et la mesure de similarité.

Etant donnée un vecteur requête q, et en se basant sur les équations 4.11 et 4.13, la distance entre $\Phi(p)$ et $\Phi(q)$ dans l'espace de projection est donnée par:

$$\|\Phi(p)-\Phi(q)\|^2 = \left\| (\sum_{i=0}^{d-1} \alpha_i^p e_i + \alpha_d^p e^{\perp}) - (\sum_{i=0}^{d-1} \alpha_i^q e_i + \alpha_d^q e^{\perp}) \right\|^2 \quad (4.15)$$
$$= \underbrace{\sum_{i=0}^{d-1} (\alpha_i^p - \alpha_i^q)^2}_{} + \underbrace{\alpha_d^p e^{\perp}(\alpha_d^p e^{\perp})^T + \alpha_d^q e^{\perp}(\alpha_d^q e^{\perp})^T - 2\alpha_d^p e^{\perp}(\alpha_d^q e^{\perp})^T}_{}$$

Afin de calculer la distance entre deux vecteurs p et q dans l'espace de projection H, nous utilisons les coordonnées des vecteurs dans H (le premier terme dans l'équation 4.15), en plus de l'erreur de projection (le second terme dans l'équation 4.15). Ces deux éléments sont obtenus par l'algorithme de l'ACPK. Le seul terme inconnu dans l'expression de la distance est $2\alpha_d^p e^{\perp}(\alpha_d^q e^{\perp})^T$. Il est équivalent à $2\sqrt{\alpha_d^p e^{\perp}(\alpha_d^p e^{\perp})^T}\sqrt{\alpha_d^q e^{\perp}(\alpha_d^q e^{\perp})^T}\cos\theta$, $\cos\theta$ représentant l'angle entre les deux vecteurs p et q dans l'espace de projection. Les valeurs extrêmes de cette expression sont utilisées pour calculer les bornes maximales et minimales des différentes régions par rapport au vecteur requête. Ces bornes sont calculées en considérant $\cos\theta$ égal à 1 pour la borne supérieure et -1 pour la borne inférieure. Par conséquent, les bornes maximales $d_{\min}$ et minimales $d_{\max}$ sont estimées respectivement avec leurs valeurs maximales et minimales.

Ces bornes ont pour expression:

$$d_{\max}^2(q,B) = \sum_{i=0}^{d-1}(\alpha_i^q - \alpha_i^B)^2 + \alpha_d^q e^{\perp}(\alpha_d^q e^{\perp})^T + \alpha_d^B e^{\perp}(\alpha_d^B e^{\perp})^T + 2\sqrt{\alpha_d^q e^{\perp}(\alpha_d^q e^{\perp})^T}\sqrt{\alpha_d^B e^{\perp}(\alpha_d^B e^{\perp})^T} \quad (4.16)$$

$$d_{\min}^2(q,A) = \sum_{i=0}^{d-1}(\alpha_i^q - \alpha_i^A)^2 + \alpha_d^q e^{\perp}(\alpha_d^q e^{\perp})^T + \alpha_d^A e^{\perp}(\alpha_d^A e^{\perp})^T - 2\sqrt{\alpha_d^q e^{\perp}(\alpha_d^q e^{\perp})^T}\sqrt{\alpha_d^A e^{\perp}(\alpha_d^A e^{\perp})^T} \quad (4.17)$$

B et A représentant respectivement les cellules en bas à gauche et en haut à droite de la région concernée (voir figure 4.3.c) . Ces deux vecteurs s'expriment en fonction de la base orthogonal "e" dans l'espace projeté par :

$$B = \sum_{i=0}^{d-1} \alpha_i^B e_i + \alpha_d^B e^{\perp} \quad , \quad A = \sum_{i=0}^{d-1} \alpha_i^A e_i + \alpha_d^A e^{\perp} \quad , \quad \Phi(q) = \sum_{i=0}^{d-1} \alpha_i^q e_i + \alpha_d^q e^{\perp}$$

Ainsi, le processus de recherche des $k-ppv$ sélectionne dans la phase de filtrage les régions pertinentes en utilisant les distances minimales (Eq.4.16) et maximales (Eq. 4.17) entre le vecteur requête et les différentes régions. Ces distances sont facilement calculables à partir des approximations des régions qui permettent de déduire les coins de chaque région. Les régions sélectionnées sont ordonnées suivant $d_{\max}$ et $d_{\min}$, puis des accès directs au fichier de données sont effectués pour sélectionner les $k-ppv$ en se basant sur la distance réelle (mesure de similarité) donnée par l'équation 4.15.

3.6 Bouclage de pertinence

En indexation d'images par le contenu, le bouclage de pertinence offre la possibilité à l'utilisateur d'affiner sa requête. Cette phase, qui permet d'améliorer l'adaptabilité du système aux besoins exprimés, peut être réalisée par un classifieur dont l'algorithme d'apprentissage se base sur le marquage par l'utilisateur des images similaires et/ou non similaires à l'image requête. Les deux approches classiques utilisées dans les systèmes SRIC consistent à modifier en fonction du marquage de l'utilisateur soit l'image requête, soit la mesure de similarité.

Dans notre approche, nous proposons la création d'un nouvel espace pour lequel la distance entre les vecteurs similaires est contractée et celle entre les vecteurs non similaires est étendue. Ceci peut être réalisé par la méthode Adaptive Quasi-Conformal Kernel (AQK) [Hei 01]. Etant donnée deux vecteurs p and p' relatifs à deux images I_p et $I_{p'}$, l'idée de AQK est de créer des fonctions noyaux successives à partir du model initial en utilisant l'équation suivante :

$$\tilde{k}(p,p') = f(p) f(p') k(p,p') \tag{4.18}$$

où $f(p)$ est une fonction semi définie positive telle que $f(p) = \dfrac{P_r(I_p / I)}{P_r(I_p / R)}$, avec :

$$P_r(I_p / I) = \frac{1}{nI} \sum_{I_q \in I} ((nI + nR) - rank) e^{\frac{-\|p-q\|^2}{2\delta^2}} \quad \text{et} \quad P_r(I_p / R) = \frac{1}{nR} \sum_{I_q \in R} ((nI + nR) - rank) e^{\frac{\|p-q\|^2}{2\delta^2}}$$

où R et I représentent respectivement l'ensemble des images marquées pertinentes et non pertinentes par l'utilisateur lors la phase du bouclage de pertinence, nR et nI sont respectivement le nombre d'images pertinentes et non pertinentes, et $rank$ représente le rang de l'image I_q.

En se basant sur les coordonnées des vecteurs dans l'espace de projection et en utilisant l'équation 4.15, on peut déduire les distances à noyau issues de AQK.

La distance réelle entre deux vecteurs p et q est donnée par :

$$\begin{aligned} dist_{AQK}^{2}(p,q) = \sum_{i=0}^{d-1} (f(q)\alpha_i^q - f(p)\alpha_i^p)^2 + f(q)^2 (\alpha_d^q e^{\perp})^T \alpha_d^q e^{\perp} + f(p)^2 (\alpha_d^p e^{\perp})^T \alpha_d^p e^{\perp} \\ - 2 f(p) f(q) (\alpha_d^p e^{\perp})^T \alpha_d^q e^{\perp} \end{aligned} \tag{4.18}$$

Les distances minimales et maximales du vecteur requête q par rapport à une région sont données par les équations suivantes

$$d^2_{\max}(q,B) = \sum_{i=0}^{d-1}(f(q)\alpha_i^q - f(B)\alpha_i^B)^2 + f(q)^2\alpha_d^q e^{\perp}(\alpha_d^q e^{\perp})^T + f(B)^2\alpha_d^B e^{\perp}(\alpha_d^B e^{\perp})^T + 2f(q)f(B)\sqrt{\alpha_d^q e^{\perp}(\alpha_d^q e^{\perp})^T}\sqrt{\alpha_d^B e^{\perp}(\alpha_d^B e^{\perp})^T} \quad (4.19)$$

$$d^2_{\min}(q,A) = \sum_{i=0}^{d-1}(f(q)\alpha_i^q - f(A)\alpha_i^A)^2 + f(q)^2\alpha_d^q e^{\perp}(\alpha_d^q e^{\perp})^T + f(A)^2\alpha_d^A e^{\perp}(\alpha_d^A e^{\perp})^T - 2f(q)f(A)\sqrt{\alpha_d^q e^{\perp}(\alpha_d^q e^{\perp})^T}\sqrt{\alpha_d^A e^{\perp}(\alpha_d^A e^{\perp})^T} \quad (4.20)$$

Les noyaux générés à partir du noyau initial utilisent les informations fournies par l'utilisateur lors de la phase du bouclage de pertinence pour créer de nouvelles distances à noyau. Celles-ci sont d'abord utilisées dans la phase de filtrage pour sélectionner les régions ayant le plus de chance de contenir des vecteurs pertinents pour la recherche et d'autres part pour mesurer la similarité entre le vecteur requête et les vecteurs des régions retenues lors de la phase de filtrage.

4 Synthèse

Dans ce chapitre, nous avons présenté notre méthode pour l'indexation et la recherche par le contenu KRA$^+$-Blocks basée sur l'approche noyau. Pour commencer, nous avons passé en revue les deux méthodes qui existent dans la littérature basée sur l'approche noyau M-Tree à noyau et KVA-File. Le M-Tree à noyau est une extension de la méthode M-Tree dans un espace non linéaire à noyau, c'est l'une des premières méthodes développées pour supporter les distances à noyau. Cette méthode a été particulièrement appliquée au cas du One class SVM. Malheureusement, elle souffre des problèmes de la malédiction de la dimension liés aux espaces de grande dimension. De plus, le processus de la structuration est fortement lié à celui de la recherche. En effet, ces deux processus sont basés sur la même distance, ce qui fait que la méthode M-Tree à noyau ne peut pas supporter les différentes techniques d'apprentissage qui modifient la distance au cours de la recherche. KVA-File a été spécialement conçu pour répondre à cet objectif. Cette méthode se base sur l'utilisation des approximations des vecteurs en se basant sur leurs coordonnées dans l'espace non linéaire à noyau. Ces vecteurs sont ensuite codés et stockés dans un fichier d'approximation. Contrairement à la méthode M-Tree à noyau, KVA-File permet d'utiliser les techniques d'apprentissage des distances pour améliorer la qualité de la recherche. En revanche, KVA-File se trouve confronté également aux problèmes liées à la malédiction de la dimension, ces performances dépendent fortement de la taille du fichier des approximations et du nombre de bits utilisé pour le codage des vecteurs. Si la taille du fichier des approximations est très grande, les performances du KVA-File se dégradent considérablement.

Pour cela, nous avons proposé dans ce chapitre une nouvelle méthode d'indexation multidimensionnelle basée sur l'approche noyau KRA$^+$-Blocks. Cette méthode a été proposée d'une part pour combler l'insuffisance des techniques d'indexation multidimensionnelle et leur inefficacité dans les espaces de grandes dimensions, et d'autre part pour surmonter les problèmes qui se posent lors de la manipulation des données de nature hétérogène, en particulier dans le cas des méthodes basées sur l'approximation. En effet KRA$^+$-Blocks, exploite les propriétés des méthodes basées sur l'approximation et les techniques de la réduction de la dimension pour faire face au problème de la malédiction de la dimension et au passage à

l'échelle. Notre méthode permet d'améliorer la précision de la recherche à travers l'utilisation d'une mesure de similarité basée sur les fonctions noyaux.
En Hors ligne, cette méthode utilise une technique non linéaire de la réduction de la dimension qui permet d'une part de réduire la dimension des vecteurs, et d'autre part de regrouper les données en paquets à travers le choix des paramètres optimaux de la fonction de projection.
En ligne, lors de la recherche, notre méthode utilise des distances basées sur la fonction choisie dans la phase hors ligne pour la projection. Le processus le filtrage est alors exécuté sur les régions en se basant sur les distances pour sélectionner les régions ayant le plus de chance de contenir les plus proches voisins. Ces régions sont représentées par les approximations des coordonnées des deux vecteurs délimitant la région dans l'espace de dimension réduite.
Nous avons également proposé un mécanisme de bouclage de pertinence qui permet d'apprendre de nouvelles distances basées sur les fonctions noyau et tendant à réduire le fossé entre la description des images et les attentes de l'utilisateur. Contrairement à la méthode M-Tree à noyau, notre méthode permet de séparer la phase de la structuration des données de celle de la recherche. Les distances utilisées lors de la recherche peuvent facilement être modifiées et améliorées par le processus du bouclage de pertinence sans que la structure d'index du KRA+-Blocks soit modifiée.
Par ailleurs, tout comme le RA+-Blocks/RA-Blocks les performances de KRA$^+$-Blocks en terme de temps de réponse dépendent essentiellement de la capacité des régions et du taux de filtrage. Concernant la capacité des régions, si celle-ci est très petite (voisine de 1 par exemple), alors les performances du KRA$^+$-Blocks deviennent similaires à celles du VA-File. D'un autre côté, si la capacité est très grande, le nombre de régions devient petit, ce qui diminue le nombre de comparaisons lors de la phase de filtrage. En réalité la capacité des régions est un paramètre critique et difficile à fixer, une grande valeur de ce paramètre réduit bien le temps de réponse par contre elle a tendance de regrouper les éléments non similaires dans les mêmes régions ce qui dégrade notablement la qualité de la recherche, car certaines régions risquent de ne pas être retenues, bien qu'elles contiennent des vecteurs pertinents.
Le taux de filtrage est un bon paramètre pour estimer la performance de la méthode KRA+-Blocks, il est lié, d'une part à l'efficacité de l'approximation qui dépend essentiellement du nombre de bits de codage et de la distribution des données, et d'autre part aux règles de filtrages liées aux distances maximales et minimales utilisées pour la sélection des candidats.
Enfin, à la différence du RA$^+$-Blocks, la technique de codage de KRA$^+$-Blocks est particulièrement adaptée aux données hétérogènes de distribution non uniforme. Le codage de la méthode KRA$^+$-Blocks proposée est effectué à deux niveaux : le premier procède à la projection des vecteurs vers un nouvel espace de dimension réduite à l'aide d'une ACPK afin d'éliminer les éventuelles corrélations non linéaires entre les dimensions des vecteurs et de réaliser une meilleure séparation des données. Ceci permet d'une part de réduire la dimension des vecteurs, et d'autre part de modifier leurs distributions de manière à rendre plus facile et plus efficace le processus de partitionnent en régions. Le second niveau applique une technique de quantification et de compression similaire à RA$^+$-Blocks afin d'effectuer le groupement des données en régions et de coder celles-ci.

Chapitre 5

Expérimentations

Le but de ce chapitre est d'évaluer nos deux approches pour l'indexation et la recherche par le contenu d'images fixes proposées aux chapitres 3 et 4. Les évaluations concernent à la fois la qualité des résultats retournés à l'utilisateur et le temps de réponse. Nous présentons d'abord les expérimentations relatives à la méthode RA+-Blocks avant de détailler celles de KRA+-Blocks.

1 Evaluation des performances de RA$^+$-Blocks

Ce paragraphe décrit les évaluations de performance de la méthode d'indexation multidimensionnelle RA$^+$-Blocks. Ces évaluations concernent le temps de réponse, le nombre de régions obtenues lors du partitionnement de l'espace de données et le taux de remplissage des régions. En ce qui concerne le temps de réponse, nous comparons notre méthode avec la recherche séquentielle, VA-File, et RA-Blocks. Nous avons choisi ces méthodes car elles possèdent un coût linéaire en fonction de la dimension contrairement à la plupart des techniques d'indexation dont les performances se dégradent exponentiellement lorsque la dimension augmente [Web98][Ams01]. En ce qui concerne le nombre de régions obtenues lors du partitionnement de l'espace de données et le taux de remplissage des régions, ces deux paramètres donnent une idée sur l'efficacité de l'index en terme de gestion de l'espace mémoire et compacité des régions formées. Dans la première expérimentation, nous avons comparé, le nombre de régions obtenues lors du partitionnement de l'espace de données avec les deux méthodes RA-Blocks et RA$^+$-Blocks en fonction de la dimension de l'espace et de la taille de la base de données, dans la deuxième expérimentation, nous avons comparé le taux de remplissage des régions des deux méthodes RA-Blocks et RA$^+$-Blocks et finalement dans la troisième expérimentation nous avons évalué et comparé le temps de réponse des quatre méthodes RA$^+$-Blocks, RA-Blocks, VA-File et la recherche séquentielle.

1.1 Environnement expérimental

Les structures d'index utilisées dans nos tests sont implémentées en Visuel C++ version 8.0. L'évaluation des performances est faite sur un ordinateur XP de Microsoft Windows de 2.3GHz

avec 1 Go RAM, et 160 Go de disque local. Deux paramètres sont utilisés pour mesurer les performances de RA$^+$-Blocks, le temps d'exécution et le nombre de région obtenues lors du partitionnement de l'espace de données. Les temps d'exécutions des algorithmes de recherches sont mesurés à l'aide des fonctions iGo() et iStop() de la class Chrono . Le nombre de régions formées est calculé hors ligne après la phase du découpage de l'espace de données.

1.2 Description des données

Les structures d'indexation multidimensionnelles sont évaluées sur deux bases de vecteurs multidimensionnels. La première est une base synthétique de 60.000 vecteurs de dimension 250, créée aléatoirement avec la fonction rand (). Les composantes de chaque vecteur sont des valeurs réelles uniformément distribuées sur l'intervalle [0 0.01]. La seconde est une base de 60.000 descripteurs de dimension 252 correspondant à la concaténation du descripteur couleurs dominante (couleurs dominantes, cohérence spatiale et le pourcentage de la couleur) de 25 composantes, de l'histogramme couleur quantifiés des trois canaux couleur (Lab) sur 192 composantes, et du descripteur de forme ART sur 35 composantes. Ces descripteurs sont extraits à partir des images réelles.

Les deux bases de données sont stockées en mémoire pour évaluer les performances des différentes structures en utilisant une moyenne de plusieurs requêtes. Les requêtes utilisées sont tirées aléatoirement des deux bases et l'ensemble des requêtes est identique pour chaque structure d'indexation.

1.3 Expérimentation 1 : Nombre de régions obtenues

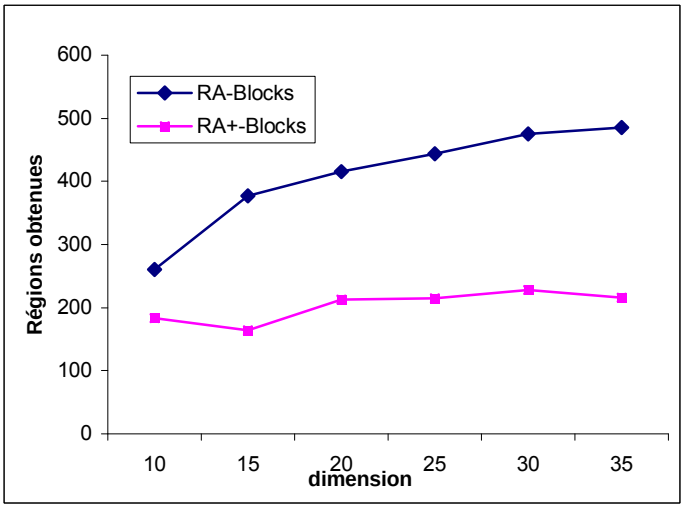
Les structures d'index du RA-Blocks et RA+-Blocks sont construites en utilisant les deux bases de vecteurs décrites dans le paragraphe précédent. Pour ces expérimentations nous avons fixé le nombre des plus proches voisins à 5, le nombre de bits par dimension à 8, la capacité des régions à 50 et la page disque à 16ko.

1.3.1 Influence de la dimension

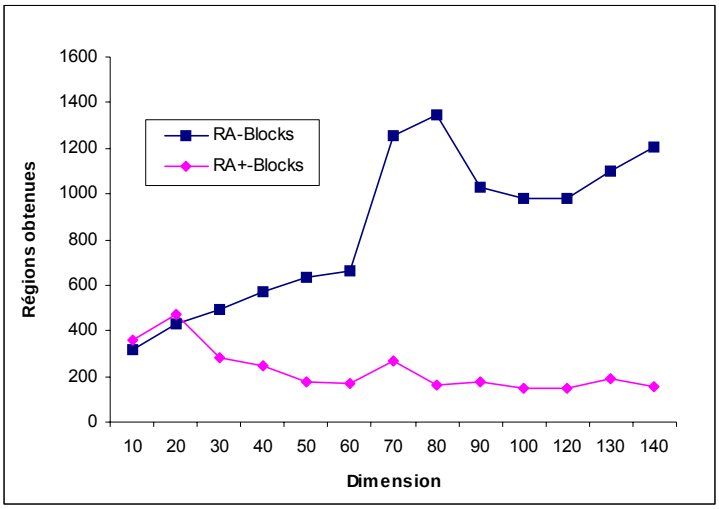
Dans cette expérimentation, nous calculons le nombre de région obtenues par le partitionnement de l'espace de données en utilisant la stratégie de partitionnent du RA$^+$-Blocks et RA-Blocks en fonction de la dimension. Pour cela, nous avons fixé le nombre de vecteurs à 10.000, alors que la dimension a été tronquée pour mesurer le nombre de région obtenues en fonction de différentes dimensions.

Les expérimentations menées sur les données uniformes et réelles montrent que le nombre de régions obtenues lors du partitionnement de l'espace de données par la méthode RA-Blocks est plus grand que celui obtenu en utilisant notre stratégie de découpage. Ceci signifie que les régions obtenues par notre méthode sont plus compactes et denses en comparaison avec celles obtenues par la méthode RA-Blocks. Rappelons que notre stratégie de partitionnement se base sur l'algorithme KD-Tree, ou le choix de la dimension du découpage est basé sur la distribution de données dans l'espace selon chaque dimension alors que le choix de la position de partitionnement est effectué de façon à garantir un nombre équivalent de vecteurs dans les régions formées. A la différence des méthodes conventionnelles d'indexation multidimensionnelles ou le nombre de partitions croit exponentiellement en fonction de la

dimension, le nombre de régions obtenues par notre stratégie de découpage croit plus lentement qu'une fonction exponentielle. Cela vient du fait que l'hyperplan de partitionnement subdivise uniquement les régions contenant un nombre de vecteurs supérieur à la capacité alors que dans la méthode RA-Blocks l'hyperplan de partitionnement subdivise toutes les régions ayant une intersection non nul avec l'hyperplan de partitionnement pour préserver les propriétés de la structure. De ce fait, le nombre de régions formées croit rapidement en fonction de la dimension en comparaison avec notre méthode. Remarquons qu'avec notre stratégie du découpage, à partir d'une certaine dimension (par exemple 20 dans le cas des données uniformes et 30 dans le cas des données réelles), le nombre de régions obtenues se stabilise. Ceci peut être expliqué par le fait qu'à partir d'une certaine dimension, l'hyperplan de partitionnement passe par les frontières entre cellules, ce phénomène est appelé phénomène de saturation. Dans ce cas de figure nous obtenons des régions avec un nombre de vecteurs supérieure à la capacité et pour lesquelles le partitionnement en deux sous régions n'est plus possible. Ce phénomène se produit uniquement en grande dimension, particulièrement pour cette stratégie de partitionnement. Il est équivalent au phénomène de l'espace vide qui se produit aussi en grande dimension suite au découpage récursif de l'espace de données. Il se produit également dans le cas du RA-Blocks où le nombre de région croit rapidement en fonction de la dimension.



(a) données réelles

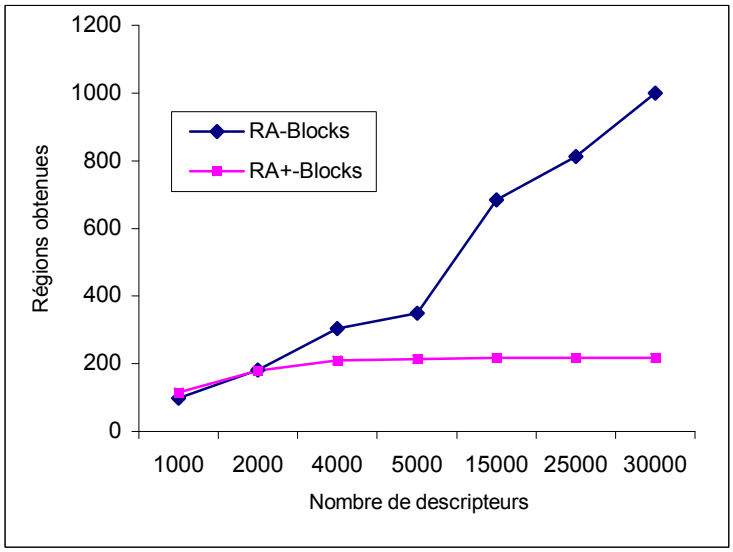


(b) données uniforme

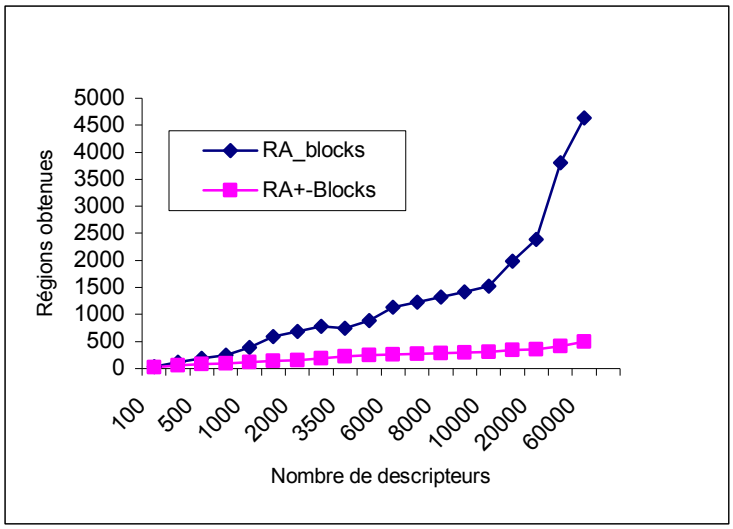
Fig.5.1 Le nombre de régions obtenues en fonction de la dimension (a) pour des données réelles (b) et uniformes

1.3.2 Influence du nombre de vecteurs

Dans cette expérimentation, nous évaluons le nombre de régions obtenues par le partitionnement de l'espace de données en utilisant la stratégie du RA^{+}-Blocks et RA-Blocks en fonction du nombre de vecteurs de la base de données. Pour cela, nous avons fixé la dimension des descripteurs à 12 et nous avons pris plusieurs ensembles de descripteurs de chaque base (réelle et uniforme). Les figures 5.2 (a) et 5.2 (b) représentent nombre de régions obtenues du partitionnement de l'espace de données en fonction du nombre de vecteurs respectivement pour la base de données réelle et uniforme.



(a) données réelles



(b) données uniformes

Fig.5.2 Nombre de régions obtenues en fonction de la taille de la base de données

Nous remarquons que le nombre de régions obtenues lors du partitionnement de l'espace de données avec la méthode RA-Blocks croit plus rapidement en fonction du nombre de vecteurs par rapport à celui obtenu par notre stratégie de partitionnement. Ceci est dû, comme nous l'avons déjà expliqué plus haut, à notre stratégie de découpage qui permet d'éviter la création des régions vides ou presque vide en partitionnement uniquement les régions saturées et en remplaçant la structure arborescente par une structure linéaire. Le fait d'avoir un grand nombre

de région augmente la taille de l'index et diminue la capacité de stockage et par conséquent diminue les performances de la structure d'index. Par conséquent, notre stratégie de découpage est plus adaptée aux données multidimensionnelles de taille importantes en comparaison avec la méthode RA-Blocks.

1.4 Expérimentation 2 : Taux de remplissage

Cette expérimentation présente une comparaison entre la méthode RA-Blocks et RA$^+$-Blocks en terme de taux de remplissage des régions sur les deux bases de descripteurs (la base réelle et uniforme). Le taux de remplissage est défini comme étant le rapport entre le nombre de vecteurs dans la base de données et la capacité totale de la structure d'index, c.à.d la capacité totale des régions obtenues lors du partitionnement de l'espace de données. Ce paramètre est un bon estimateur de performance du fait que la performance des structures d'index multidimensionnelles dépend largement de leur capacité de supporter le problème du passage à l'échelle des données dû au grand volume de données. En effet une grande valeur de ce paramètre (100%) garantit une exploitation optimale des pages disques associées aux régions et diminue par conséquent le nombre de distances à calculer lors du processus de la recherche et par la suite le temps de réponse.

La figure 5.3 représente la capacité de stockage (le taux de remplissage) des deux méthodes RA-Blocks et RA$^+$-Blocks, en fixant le nombre de vecteurs à 10.000, la dimension à 16, la capacité des régions à 50 et le nombre des $k-ppv$ à 5. Nous remarquons d'après ces expérimentations que le taux de remplissage de notre méthode est plus grand que celui du RA-Blocks par un facteur de 5. Notre méthode de découpage produit relativement peu de régions et celles-ci sont majoritairement denses alors que la stratégie du découpage du RA-Blocks produit un grand nombre de régions vides ou presque vide. La capacité de stockage de notre structure est plus intéressante que celle du RA-Blocks , c.à.d pour la même taille de la base de données notre structure nécessitera une taille mémoire 5 fois plus petite que celle exigée si on index les données avec la méthode RA-Blocks.

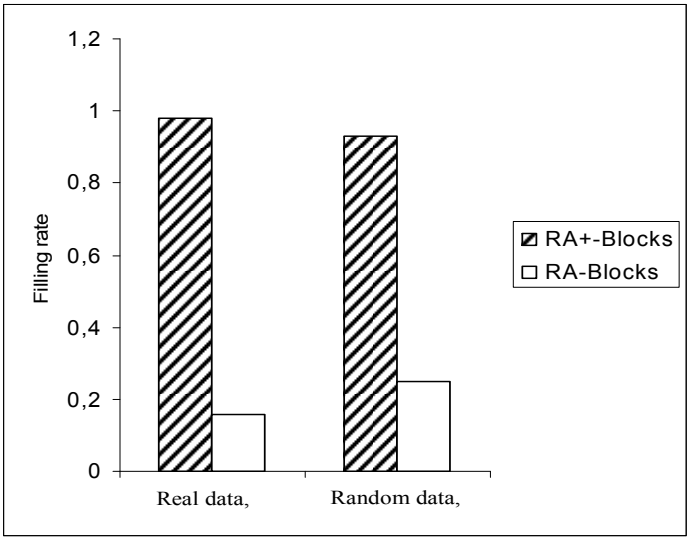


Fig.5.3 La capacité de stockage du RA-Blocks et RA+-Blocks

1.5 Expérimentation 3 : Temps de réponse

Ces expériences montrent l'influence de la dimension de l'espace de données et la taille de la base sur le temps de réponse. Nous comparons dans ces tests le temps de réponse de notre méthode par rapport à VA-File, la recherche séquentielle et le RA-Blocks. Pour la première expérimentation nous avons fixé la taille de la base de données à 10.000 et nous faisons varier la dimension de 2 à 200. Dans la deuxième expérimentation, la dimension de l'espace de données est fixée à 12 et la taille de la base de données varie de 100 à 60.000.

1.5.1 Influence de la dimension

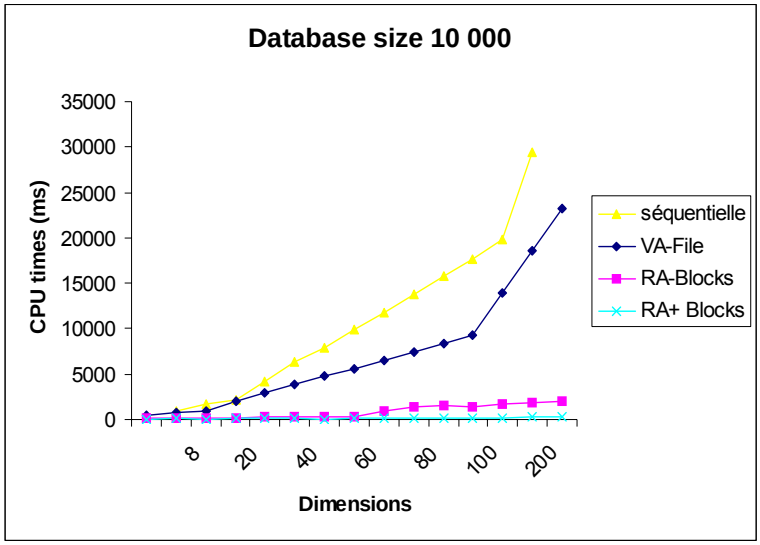
D'après la figure 5.4, on remarque que les temps de réponse de la recherche séquentielle, de VA-File et de RA-Blocks croissent linéairement en fonction de la dimension. Notons que la croissance du temps de réponse de la recherche séquentielle en fonction de la dimension est plus rapide que pour le VA-File et celle du temps de réponse du VA-File est également plus rapide que pour le RA-Blocks. Quant au temps de réponse du RA^+-Blocks, il croit plus lentement par rapport aux autres méthodes.

A titre d'exemple la recherche de 5 plus proches voisins dans la base de 10.000 vecteurs réels de dimension 125 s'effectue en 20s avec une recherche séquentielle, 15s avec le VA-File, 2s avec le RA-Block et quelques ms avec le RA^+-Blocks. Le gain de temps de réponse de notre méthode devient de plus en plus intéressant en grande dimension. Pour de faibles dimensions (inférieures à 10 pour les deux bases de vecteurs) les temps de réponse des 4 méthodes sont quasiment identiques.

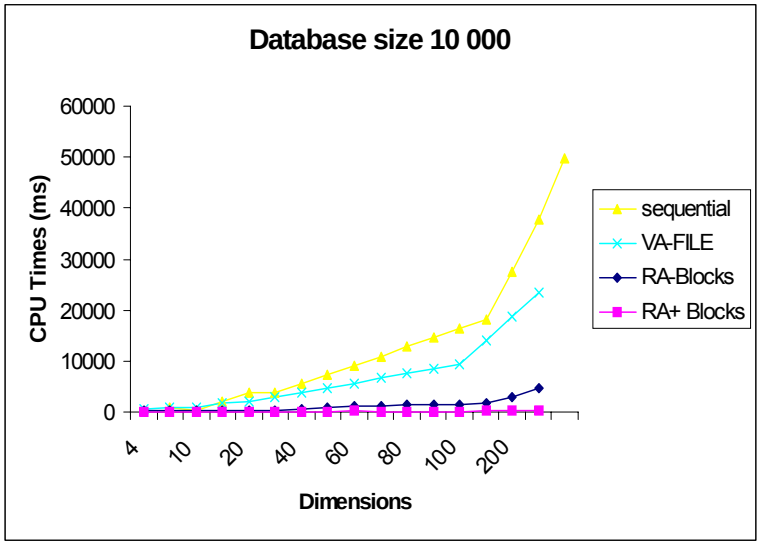
En grande dimension, et comme nous l'avons déjà présenté, le nombre de régions obtenues par notre méthode est inférieur à celui obtenu avec la méthode RA-Blocks. Ceci a une influence directe sur le temps de calcul des distances minimales et maximales par rapport au vecteur requête lors de la phase de filtrage. D'un autre côté, le nombre de régions parcourues et explorées lors de la phase d'accès est également plus faible, réduisant ainsi le nombre de comparaison et par la suite le temps de la recherche.

Concernant le VA-File, il est évident que les temps de réponse de RA-Blocks et de RA+-Blocks soient meilleurs de par la stratégie de structuration adoptée par chaque méthodes. En effet, étant donné que le VA-File approxime les vecteurs par des cellules, ce qui est une amélioration de la recherche séquentielle, le temps de la recherche est celui du parcours de tout le fichier des approximations associées aux cellules contenant les vecteurs alors que dans le cas du RA-Blocks et RA^+-Blocks le temps de la recherche est réduit à celui du parcours et l'exploration des régions.

Pour montrer la différence en temps de réponse entre RA-Blocks et RA+-Blocks, nous avons présenté sur des figures séparées le temps de la recherche sur les deux bases d'images précédentes. Les résultats sont montrés sur la figure 5.5

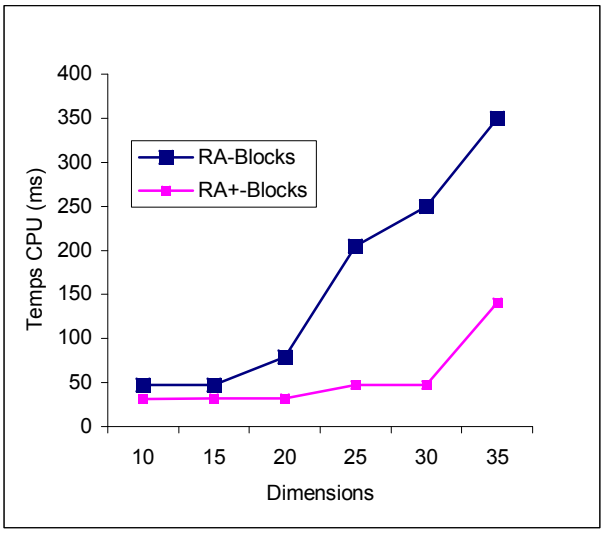


(a) données réelles

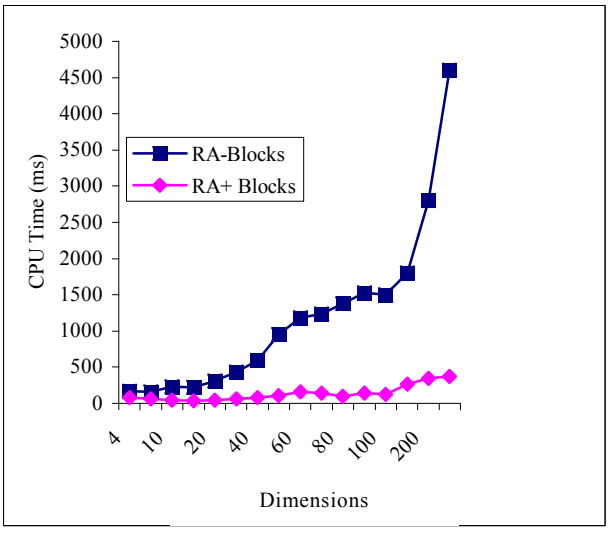


(b) données uniformes

Fig.5.4. Temps de réponse en fonction de la dimension



(a) données réelles

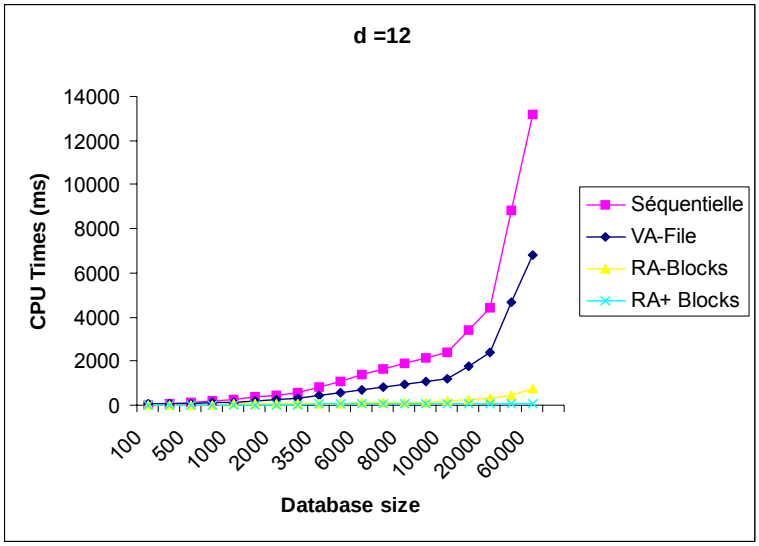


(b) données uniformes

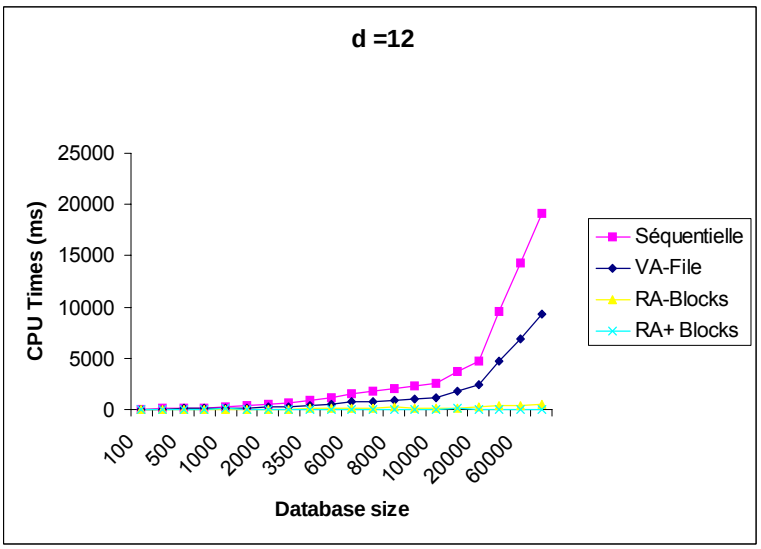
Fig.5.5. Temps de réponse en fonction de la dimension

1.5.2 Influence du nombre de vecteurs

Dans ces tests, nous évaluons le temps de réponse de notre méthode par rapport à la recherche séquentielle, VA-File et RA-Blocks lors du passage à l'échelle. Nous remarquons comme précédemment que le temps de réponse de la recherche séquentielle croit plus rapidement que celui du VA-File, RA-Blocks et RA$^+$-Blocks en fonction du nombre de vecteurs de la base données, et que les performances du RA$^+$-Blocks dépassent celles du RA-Blocks et VA-File. De nouveau notre méthode présente de bonnes performances en terme de temps de réponse au passage à l'échelle par rapport aux autres méthodes. Le gain en temps de réponse devient de plus en plus intéressant lorsque le nombre de vecteurs croit.



(a) données réelles

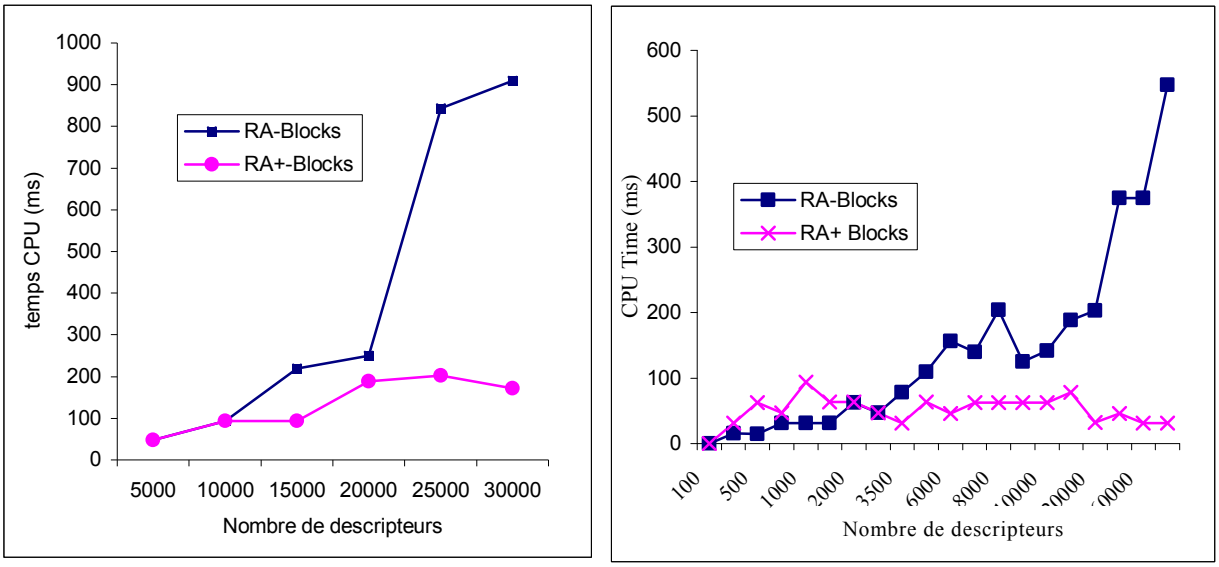


(b) données uniformes

Fig. 5.6 Temps de réponse en fonction de la taille de la base de données

Rappelons que la construction de la structure d'index des différentes méthodes d'indexation se fait par insertions successives des vecteurs de la base de données. Lorsque la taille de la base de données augmente, la construction produit le partitionnent des régions denses dans le cas du RA$^+$-Blocks et des régions denses et parfois peu dense où vides dans le cas de la méthode RA-Blocks. Comme nous l'avons déjà vu dans les expérimentations précédentes, le nombre de régions augmente rapidement en fonction du nombre de vecteurs avec la méthode RA-Blocks.

Le nombre de régions obtenues avec la méthode RA$^+$-Blocks par contre croit lentement en fonction du nombre de vecteurs de la base et se stabilise à partir d'une certaine taille. Le paramètre a une influence directe sur le temps de réponse. En effet, le temps de la recherche est équivalent à celui du parcours des régions (dans le cas du RA-Blocks et RA+-Blocks) / des vecteurs (dans le cas du VA-File), plus le temps d'accès. La structure d'index proposée réduit le temps de la recherche par rapport aux autres méthodes en optimisant la phase du découpage et la création des régions. Cette structure est bien adaptée aux bases de données de grande taille.
Pour montrer la différence en temps de réponse entre RA-Blocks et RA$^+$-Blocks, nous avons présenté sur des figures séparées le temps de la recherche sur les deux bases d'images précédentes en fonction de la taille de la base de données. Les résultats sont montrés sur la figure 5.7. Ces résultats montrent que le temps de réponse de la méthode RA+-Blocks est inferieur à celui du RA-Blocks, le gain en temps de réponse devient de plus en plus intéressant avec un grand nombre de vecteurs. Par exemple, avec la base de données aléatoires, on obtient un temps de réponse pour le RA-Blocks qui est quatre fois plus grand que celui de RA+-Blocks, pour un nombre de vecteur de 60.000.



(a) données réelles **(b)** données uniformes

Fig. 5.7 Temps de réponse en fonction de la taille de la base de données

2 Evaluation des performances du KRA$^+$-Blocks

Après avoir présenté l'environnement expérimental et décrit nos données, nous nous intéressons dans un premier temps, à l'étude des paramètres du noyau à savoir la dimension de l'espace de projection et le paramètre de l'échelle. Nous évaluons ensuite la qualité de la recherche de notre approche sur des bases de données réelles. Ensuite, nous montrons l'intérêt de l'utilisation des fonctions noyaux dans l'apprentissage et dans la combinaison de descripteurs hétérogènes. Nous évaluons enfin les performances de KRA+-Blocks en terme de temps de réponse lorsque nous traitons des vecteurs de grande taille et lors du passage à l'échelle.

2.1 Environnement expérimental

Les algorithmes d'indexation et de recherche sont implémentés en visuel C++ (version 8.0) sur une machine sous Windows, disposant d'un processeur de 2.3 Ghz, d'une mémoire centrale de 2Go et d'un disque local de 250Go. Les temps d'exécutions des algorithmes de recherches sont mesurés à l'aide des fonctions iGo() et iStop() de la class Chronoi .
Par ailleurs, pour mesurer la qualité de la recherche, nous avons utilisé le rappel et la précision dont la définition est rappelée ci-après.

$$Rappel = \frac{Q}{P} \times 100, \ \Pr écision = \frac{Q}{R} \times 100$$

où P est le nombre de toutes les images de la base de donnée pertinentes à la requête, Q est le nombre des images pertinentes retournées par le système et R est le nombre totale des images recherchées. Le rappel définit donc la proportion d'images pertinentes et effectivement retournées en réponse à une requête soumise au système alors que la précision est la proportion d'images retournées par le système et pertinentes pour la requête considérée

2.2 Description des données

Pour tester notre méthode, nous avons utilisé quatre bases de données. La première est composée de 7200 descripteurs de dimension 252. Ces descripteurs sont calculés à partir de la base d'images COIL-100 de l'université de Columbia [Cou], ils correspondent à la concaténation du descripteur "couleur dominante" sur 25 valeurs (5couleurs dominantes, leurs cohérences spatiales et leurs pourcentages de couleur), l'histogramme couleur des trois composantes (Lab) quantifié sur 192 valeurs (3 x 64), et le descripteur de forme ART [Kim 99] sur 35 valeurs. Cette base est composée de 100 classes de 72 images chacune et chaque classe est constituée d'un objet auquel sont appliquées des rotations successives de 5 degrés. Un exemple de cette base est illustré sur la figure 5.8.

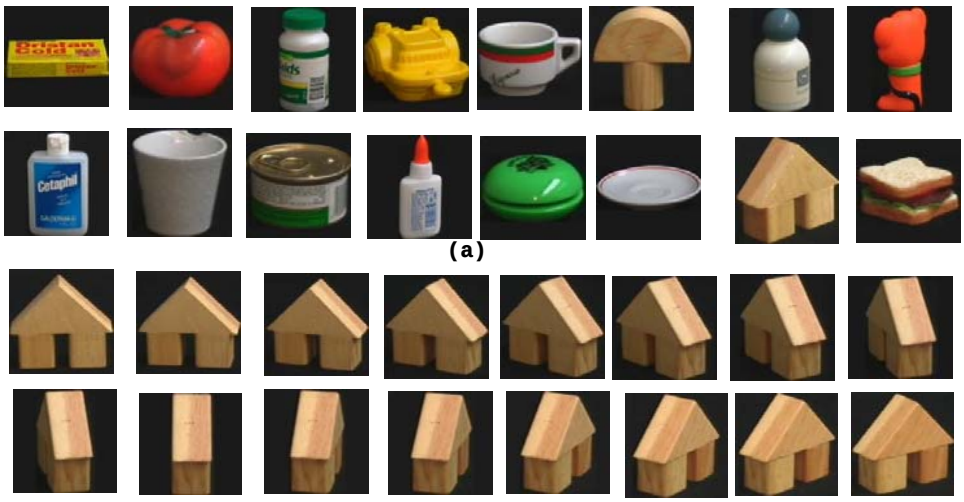


Fig. 5.8 Un exemple (a) d'images de la base COIL-100 (b) classes de la base d'images

La deuxième base est une extension de la base COIL-100 composée de 40.000 descripteurs générés de manière calculatoire à partir des descripteurs initiaux, issus de la base COIL-100 , de façon à ce qu'ils appartiennent aux mêmes classes que celles de la base d'origine. Au final, cette base est toujours composée de 100 classes, chacune ayant un cardinal de 400 images (ou

descripteurs). Nous avons eu recours à ce procédé car nous ne disposions pas d'un ensemble d'images fixes de cette taille, et pour lequel les classes sont déjà labélisées. Ces deux bases d'images (COIL-100 et COIL-100 étendue) serviront davantage à évaluer la performance de notre méthode d'indexation et de recherche en terme de qualité de la recherche, et donc de pertinence des réponses retournées.
La troisième base est une base de 2.200.000 descripteurs de dimension 252 générés selon une distribution uniforme sur l'intervalle [0 1] avec une fonction rand(). Cette base est utilisée pour comparer et évaluer la performance de différentes méthodes en terme temps de réponse.
La dernière base est une base de 300.000 descripteurs de dimension 60 créés aléatoirement avec une fonction rand() également distribuée sur l'intervalle [0 1]. Cette base a été conçue pour comparer notre méthode aux autres techniques qui existent dans la littérature et qui utilisent ces mêmes paramètres de taille et de dimension.
Nous utilisons les notations suivantes dans la suite :
BD1 la base COIL-100 de 7200 images/descripteurs de dimension 252.
BD2 la base COIL-100 étendue de 40.000 descripteurs de dimension 252.
BD3 la base de données aléatoires de 2.200.000 vecteurs de dimension 250.
BD4 la base de données aléatoires de 300.000 vecteurs de dimension 60.
Notons que les expérimentations reportées dans ce chapitre ne mesurent en aucune manière la capacité de reconnaissance des descripteurs, bien que dans le cas de BD1.et BD2, les descripteurs choisis sont bien adaptés à la nature des images contenues dans ces bases. Les tests ont pour vocation d'étudier les performances de notre méthode de recherche en termes de temps de réponse et de précision pour le calcul des plus proches voisins retrouvés.

2.3 Expérimentation 1 : Estimation des paramètres du noyau

Dans cette section, nous présentons une étude relative aux paramètres du noyau. Le but de cette étude est d'estimer les bons paramètres de la fonction noyau utilisée pour la projection des données dans un espace de dimension réduite. De manière générale, il n'existe pas de critères universels permettant de choisir ces paramètres. Ce choix dépend de l'application. Ici, dans le cas de la recherche des plus proches voisins, l'objectif est de trouver les paramètres qui donnent de meilleur taux de précision de la recherche. Malheureusement, la relation entre les paramètres du noyau et la qualité de la recherche ne peut être exprimée analytiquement, car elle dépend de plusieurs critères : la distribution des données, la technique du groupement/ partitionnent des données, la méthode d'indexation, etc.
Le but ici est donc de rechercher les paramètres optimaux du noyau de manière empirique. Rappelons que nous avons adopté une fonction noyau gaussienne de la forme :

$$k(p,q) = e^{-\frac{\|p-q\|^2}{2\delta^2}}$$

Avec p et q deux vecteurs de l'espace de données. La fonction noyau intervient dans l'algorithme de l'ACPK pour la réduction de la dimension de l'espace projeté. Deux paramètres doivent être fixés au préalable : la dimension de l'espace de projection d et le paramètre d'échelle δ de la Gaussienne.
Dans cette section, nous présentons les résultats de l'estimation des paramètres d et δ relative à la base BD1.
Pour estimer les valeurs optimales de d et δ, nous utilisons deux paramètres, l'indicateur de performance γ et la variance σ.

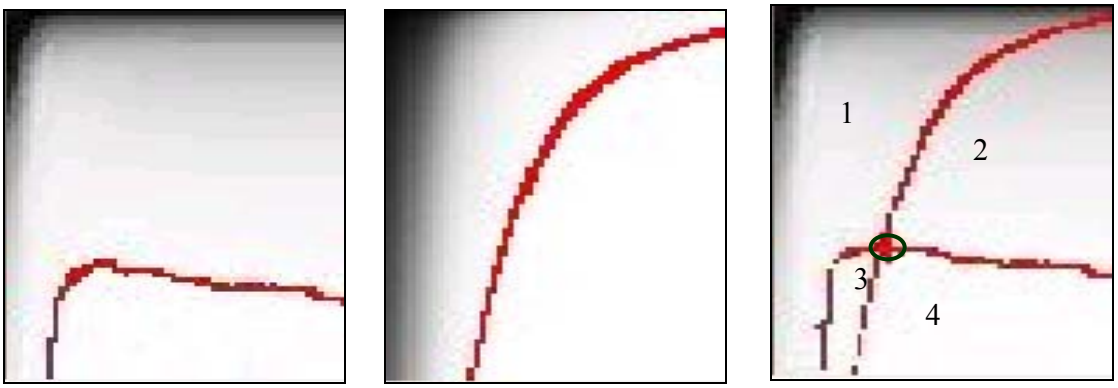
γ est donné par :

$$\gamma = \frac{Intercluster}{Intracluster}$$

où $Intercluster$ et $Intracluster$ représentent respectivement les distances interclasse et intra-classe. La distance interclasse représente la capacité de séparation de classe et peut être calculée par la distance entre les deux vecteurs les plus proches appartenant à deux classes différentes. La distance intra-classe, quant à elle, représente la capacité de classification. Elle peut être calculée par la moyenne des distances entre les vecteurs d'une classe et son centre de gravité.
Le second paramètre σ, représente la variance totale des données sur les axes principaux, elle est calculée à partir de l'ACPK et a pour expression :

$$\sigma = \sum_{i=0}^{d-1} eign(i)$$

où d est la dimension de l'espace projeté et $eing()$ sont les valeurs propres de la matrice de covariance des données initiales. Nous rappelons qu'une grande valeur de γ et σ est obtenue respectivement pour une bonne séparation de données et pour une faible perte d'information (paragraphe 3.2 du chapitre 3).
Pour illustrer l'influence des deux paramètres d et δ sur γ et σ, nous effectuons des tests sur les descripteurs de la base BD1 en calculant γ et σ pour différentes valeurs de d et δ. Dans des expériences non reportées ici, nous avons d'abord évalué les différentes plages de valeurs que peut prendre le paramètre δ et pour lesquelles nous obtenons une séparation significative des données. Sur la base de cette expérimentation, nous avons fixé un intervalle de variation de δ. Dans nos expériences les valeurs de δ varient d'une manière logarithmique entre les deux valeurs 1 et 100 alors que la dimension d varie entre 2 et 252 de manière logarithmique (d varie entre la dimension minimale et maximale des descripteurs). Les figures 5.9 (a) et 5.9 (b) représentent respectivement l'évolution des deux paramètres γ et σ pour différentes valeurs de d et δ.



(a) **(b)** **(c)**

Fig. 5.9. (a) $\gamma(d,\delta)$ (b) $\sigma(d,\delta)$ (c) valeurs optimales des paramètres du noyau

Dans les figures, les colonnes représentent la dimension d de l'espace de données et lignes le paramètre δ de la fonction noyau. Les pixels sombres correspondent aux faibles valeurs de γ et σ, et les pixels clairs correspondent à des valeurs importantes de ces deux paramètres (en fait, là où les conditions sont meilleures).
Il est évident qu'une bonne représentation de données devrait pouvoir séparer les différentes classes de la base tout en garantissant une très faible perte d'information, c.à.d avoir de grande

valeurs de γ et σ. Or d'après les courbes, ceci n'est possible qu'en fixant de grandes valeurs pour d et δ.
D'une part, si la dimension de l'espace de projection d est très grande, il sera difficile d'indexer efficacement les données à travers une structure d'index multidimensionnelle en raison du problème de la malédiction de la dimension. D'autre part, le paramètre de la gaussienne δ agit quant à lui d'une manière directe sur la séparation des données. Comme nous l'avons déjà présenté dans le paragraphe 3.2 du chapitre 3, une très grande valeur du paramètre du noyau δ entraîne le chevauchement entre les différentes classes de la base de données, ce qui est non souhaitable pour les données en vu de l'indexation. En conséquence, un compromis doit être trouvé lors de l'estimation de ces deux paramètres.
Pour cela, nous avons adopté une stratégie de sélection à deux étapes. D'abord nous procédons par une première étape de filtrage en sélectionnant les couples (d, δ) qui possèdent une valeur de l'indicateur de performance supérieure à 98% de valeur maximale de γ, c.à.d :

$$\gamma_{d,\delta} \geq 98\% \gamma_{\max} \qquad (5.1)$$

où $\gamma_{\max}$ est la valeur maximal de γ. Dans la figure 5.9 (a), la zone vérifiant cette condition se situe au dessus de la courbe qui vérifie l'équation : $\gamma_{d,\delta} = 98\%.\gamma_{\max}$
Dans la seconde étape, parmi les couples restant après la première phase de filtrage, nous sélectionnons les couples (d, δ) dont la variance est supérieure à 98% de la variance maximale, c.à.d.

$$\sigma_{d,\delta} \geq 98\% \sigma_{\max} \qquad (5.2)$$

où $\sigma_{\max}$ est la valeur maximal de σ. Dans la figure 5.9 (b), la zone vérifiant cette condition se situe au dessus de la courbe qui vérifie l'équation $\sigma_{d,\delta} = 98\%.\sigma_{\max}$
Dans le tableau suivant, nous indiquons pour les différentes régions correspondant aux γ et σ en fonction de d et δ si les équations 5.1 et 5.2 sont vérifiées ou pas.
D'après la figure 5.9 et le tableau 5.1, nous remarquons que les grandes valeurs de γ et σ se situent dans la région 4. De notre point de vu, les valeurs optimales de γ et σ sont les valeurs minimales qui vérifient les équations 5.1 et 5.2. Ces points se situent autour du point d'intersection des deux courbes de la figure 5.9 (a) et 5.9 (b), c.à.d le cercle de la figure 5.9 (c)

Région	Equation 5.1	Equation 5.2
1	Non	Non
2	Non	Oui
3	Oui	Non
4	Oui	Oui

Table 5.1 Validation des équations 5.1 et 5.2 pour les régions de la figure 5.9

Pour valider notre choix, nous avons mené des expérimentations sur la qualité de la recherche en utilisant différents paramètres correspondant aux différentes régions de la figure 5.9. Nous avons tracé les courbes de rappel et de précision en fixant 5 différentes valeurs pour le couple (d, δ). Les trois valeurs ($d = 52, \delta = 14$), ($d = 20, \delta = 33$) et ($d = 16, \delta = 9$) appartenant à 3 régions différentes de la figure 5.9. et les deux valeurs ($d = 62, \delta = 11$) et ($d = 65, \delta = 10$)

correspondent respectivement au point d'intersection des deux courbes et à un point voisin. D'après la figure 5.10, Nous voyons clairement que les meilleurs résultats de la recherche sont obtenus pour les couples ($d = 62$, $\delta = 11$) et ($d = 65$, $\delta = 10$)
Ce test permet d'un part de valider notre stratégie de sélection des paramètres du noyau et d'autre part de montrer l'influence du choix de ces deux paramètres sur la qualité de la recherche. En effet, une bonne représentation des données dans l'espace de dimension réduite est obtenue avec l'utilisation des paramètres optimaux de la fonction noyau qui permettent de séparer au mieux les différentes classes de la base de données et de réduire la dimension de l'espace projeté. Ceci permet de faciliter le processus du partitionnent de l'espace de données et le groupement de ces derniers en index et par conséquent d'améliorer la qualité de l'indexation et de la recherche.
Notons que nous avons mené d'autres expérimentations non reportées ici sur une autre base d'image [Aloi] pour valider le choix de notre stratégie de sélection des paramètres optimaux du noyau. Les résultats sont similaires, dans le sens où les meilleures performances de la recherche sont obtenues lorsque le couple des paramètres (d , δ) est pris dans la zone d'intersection.

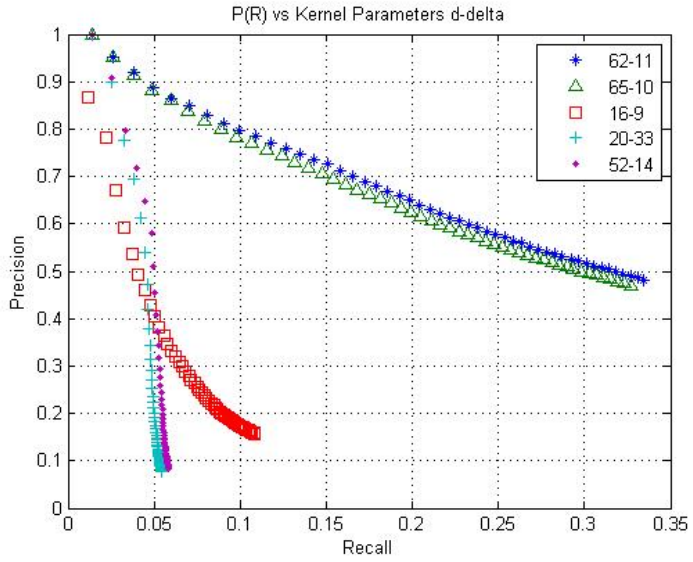


Fig. 5.10 les courbes de rappel et de précision pour différentes valeurs du paramètres du noyau.

2.4 Expérimentation 2 : Qualité de la recherche par similarité

Nous proposons dans cette section d'évaluer les performances de la recherche par le contenu basée sur notre méthode KRA$^+$-Blocks. Pour cela, nous fixons les valeurs des paramètres du noyau aux valeurs optimales trouvées précédemment ($d = 62$, $\delta = 11$). Nous montrons, de façon comparative, l'efficacité de la méthode proposée KRA$^+$-Blocks. Nous montrons également, dans le cadre du passage à l'échelle, que l'utilisation de la structure d'index KRA$^+$-Blocks conduit à de meilleures performances par rapport à d'autres méthodes, et ceci pour différentes tailles de la base d'image. Les évaluations sont essentiellement menées sur la base BD_1 de 7200 descripteurs de dimension 252 et BD_2 de 40000 descripteurs de dimension 252. Pour toutes ces expérimentations, le protocole d'interrogation des deux bases est identique. Ainsi, pour chaque base, 600 images tirées aléatoirement sont considérées comme requête, puis nous étudions les réponses retournées jusqu'à 2*cardinal des classes (144 pour BD_1 et 800 pour BD_2). Les

résultats sont enfin évalués et comparés pour les méthodes résumées dans le tableau 5.2. Les graphiques sont tracés sur la base de la moyenne des valeurs obtenues pour les 600 images requêtes tirées de chaque base.

	SCAN: recherche séquentielle	PCA: RA^+ Blocks +PCA	RA^+-Blocks	KRA^+-Blocks
Distance Euclidienne		X	X	
Distance noyau	X			X

Table 5.2. Les quatre méthodes utilisées pour la comparaison de la qualité de la recherche

Trois méthodes ont été utilisées pour comparer la performance de la structure proposée (Tab 5.2).
La recherche séquentielle exhaustive avec une distance à noyau. En effet, pour se mettre dans les mêmes conditions que KRA^+-Blocks, nous avons d'abord projeté les données initiales dans un espace de dimension réduite en utilisant la même fonction de projection et les mêmes paramètres du noyau utilisés dans KRA^+-Blocks ($d = 62$, $\delta = 11$). Puis une recherche séquentielle exhaustive est exécutée sur les données projetées de dimension réduite en utilisant les mêmes distances à noyau que dans KRA^+-Blocks.
Les deux autres méthodes utilisées dans la comparaison sont RA^+-Blocks et RA^+-Blocks avec une ACP. Le choix de ces méthodes n'est pas pris au hasard, mais possède au moins deux justificatifs. Tout d'abord RA+-Blocks est l'une des méthodes récentes qui contourne les problèmes de la malédiction de la dimension, elle a montré de bonnes performances au passage à l'échelle et dans les grandes dimensions, et enfin elle améliore VA-File qui est la méthode représentative de l'approche par approximation. Ensuite, RA+-Blocks combiné avec une ACP est, quant à elle, utilisée dans la comparaison pour mettre en évidence l'impact de la réduction linéaire de la dimension sur la représentation des données, sur l'indexation et par la suite sur la recherche. Cette méthode est mise en place pour comparer l'influence d'une réduction linéaire ou non linéaire de la dimension sur les données utilisées. La figure 5.11 donne les courbes de rappel et de précision pour les quatre méthodes.
Comme le montre cette figure, la structure d'index proposée améliore la méthode RA+-Blocks et RA+-Blocks avec l'ACP. Ces performances sont dégradées par rapport à une recherche séquentielle puisque KRA^+-Blocks est une méthode approximative de la recherche. En comparant KRA^+-Blocks, qui se base sur une distance à noyau, et RA^+-Blocks, qui utilise une distance Euclidienne, nous pouvons conclure des résultats de nos expérimentations que l'utilisation de distances à noyau dans notre cadre applicatif, permet d'améliorer significativement la performance de la recherche. Nous pouvons également conclure à partir de la comparaison KRA^+-Blocks / RA^+-Blocks avec ACP, que l'utilisation d'une méthode non linéaire ne peut être que bénéfique pour les processus d'indexation et de recherche, notamment dans le cas de données hétérogènes.

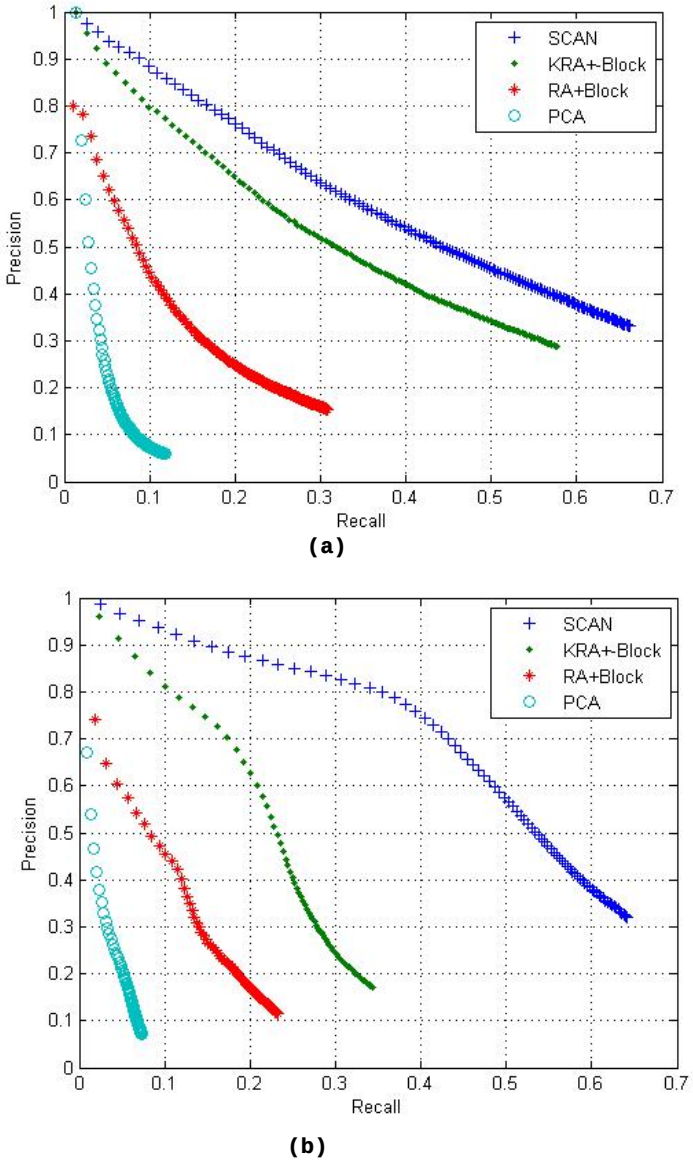


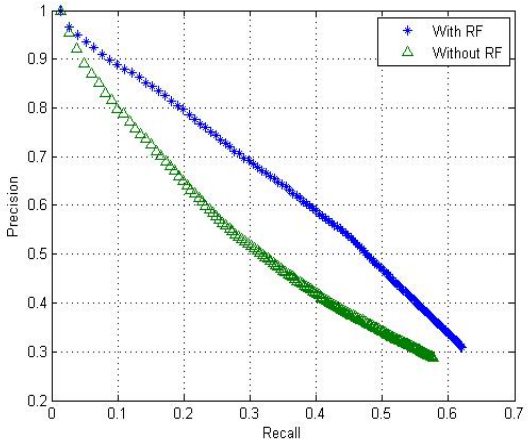
Fig. 5.11. Les Coubes de rappel et de précision en utilisant (a) la base B1 (7200) et (b) la base B2 (40000)

Notons que la méthode proposée garde une bonne qualité de la recherche au passage à l'échelle en comparaison avec RA+-Blocks et RA+-Blocks associé à une ACP (figure 5.11.b), les conclusions sont du même ordre, c.à.d KRA$^+$-Blocks améliore RA$^+$-Blocks et RA$^+$-Blocks avec une ACP. Un exemple des résultats retournés par notre méthode est illustré dans la figure 5.12

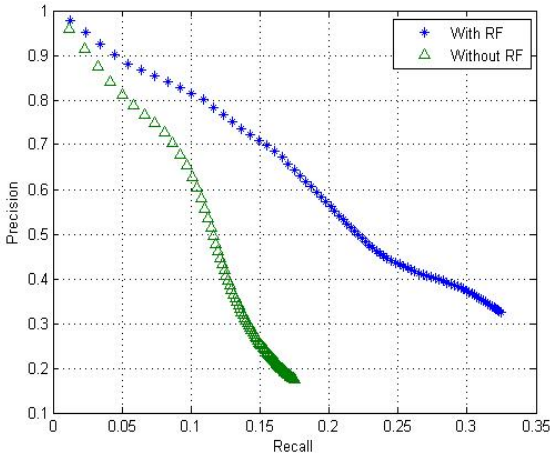
Fig.5.12. Les résultats de la recherche avec la méthode KRA+-Block dans la base COIL- 100 : la première image de chaque ligne représente l'image requête et le 11 images représentent les résultats triés par ordre croissant de similarité

2.5 Expérimentation 3 : Bouclage de pertinence

Le but de cette expérience est de montrer l'efficacité des fonctions à noyau dans le cadre de l'apprentissage. Il s'agit d'évaluer et de comparer la précision des résultats obtenus par rapport à ce que l'utilisateur a spécifié à travers le retour de pertinence. Nous avons utilisé les deux bases de données BD_1 et BD_2. Pour interroger les deux bases, nous avons utilisé 600 vecteurs requêtes tirés aléatoirement des deux bases, le nombre des k plus proches voisins est fixé à 144 pour la base BD_1 et à 400 pour la base BD_2. Un seul cycle du bouclage de pertinence est exécuté pour chaque requête. Notons que le marquage des images pertinentes et non pertinentes est effectué d'une manière automatique par le système en se basant sur les classes de la base d'images, celle-ci étant déjà labellisée. La figure 5.13 illustre les résultats de la recherche en terme de rappel et de précision avec et sans bouclage de pertinence. Un exemple des résultats retournés par notre méthode est illustré dans la figure5.14.



(a)



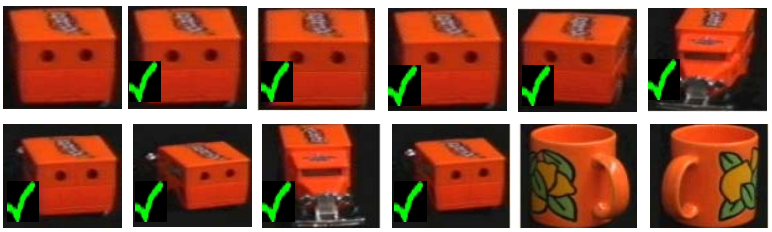
(b)

Fig. 5.13. Les Coubes de rappel et de précision en utilisant (a) la base BD_1 (7200) et (b) la base BD_2 (40000)

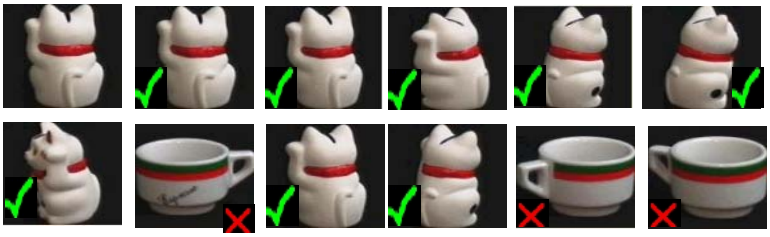
Comme nous pouvons le voir, le mécanisme du bouclage de pertinence améliore significativement la qualité de la recherche, ce qui prouve la pertinence des distances utilisées dans le processus de recherche. Rappelons que le mécanisme du bouclage de pertinence apprend à partir des informations introduites par l'utilisateur (dans notre cas le système) et recalcule, en se basant sur la fonction noyau de base (la Gaussienne avec les paramètres optimaux), de

nouvelles fonctions noyau. Celles-ci sont obtenues en multipliant la fonction noyau de base par de nouveaux noyaux déduits du retour de pertinence. Les nouvelles distances à noyau sont alors utilisées pour projeter les données dans un espace avec une nouvelle distribution. En effet, les données sont classées en deux groupes, les images pertinentes et les images non pertinentes par rapport au vecteur requête. Cette projection à tendance à maximiser la distance autour des images non pertinentes à la requête et minimiser les distances autours des images pertinentes. Ceci a pour conséquence de faciliter le processus du filtrage des régions et par la suite d'améliorer le processus de la recherche.

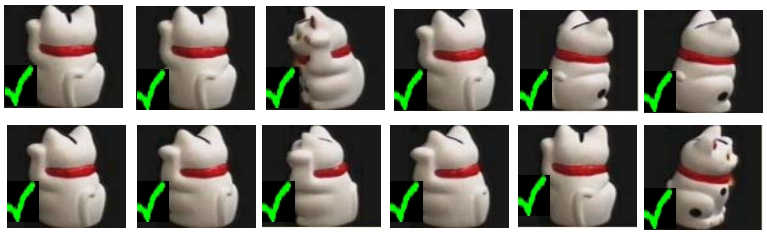
a. Recherche initiale



b. Une itération du BP



a. Recherche initiale



b. Une itération du BP

Fig.5.14. Résultat de la recherche en utilisant a. KRA+-Block avec les paramètres optimaux et b. en utilisant une itération du bouclage de pertinence: la première image de chaque ligne représente l'image requête et les autres 11 images sont les résultats retournés

2.6 Expérimentation 4 : Intérêt sur la combinaison des descripteurs globaux

Nous évaluons dans cette section l'intérêt de la combinaison de différents types de descripteurs du contenu de l'image par rapport à l'usage d'un seul type de descripteur, dans le contexte de la recherche par le contenu. Nous montrons également l'efficacité des distances à noyau pour mesurer la similarité globale entre des descripteurs dont les composantes sont hétérogènes et de nature différente, en comparaison avec une simple distance Euclidienne. La base BD1 est utilisée comme base de recherche. Les descripteurs de cette base on d'abord été normalisés par composantes avant d'être indexés à l'aide de RA$^+$-Blocks. D'un autre côté, ces descripteurs ont été projetés dans un espace de dimension réduite à travers une fonction Gaussienne utilisant les paramètres optimaux pour être indexés par KRA$^+$-Blocks. La recherche des $k-ppv$ est effectuée avec l'index du RA$^+$-Blocks au moyen de la distance Euclidienne et avec l'index KRA$^+$-Blocks au moyen de la distance à noyau. Le même algorithme de recherche des $k-ppv$ est appliqué dans les deux cas. 600 images requêtes de la base BD_1 sont soumises comme requêtes au système. Nous avons choisi de retourner 144 plus proches voisins (2*cardinal de la classe). Les résultats de précision utilisant les descripteurs couleur, et la combinaison entre la couleur et la forme sont illustrés dans le tableau suivant :

N° Images recherchées	Descripteur couleur (couleur dominante,Histogramme)		Descripteurs couleur et forme (couleur dominante, Histogramme, ART)	
	RA+-Blocks	KRA+-Blocks	RA+-Blocks	KRA$^+$-Blocks
10	70.27	86.60	72.13	87.75
20	60.44	74.79	61.15	76.81
40	46.49	52.53	48.34	62.43

Tableau 5.3. Comparaison de la précision (en %) utilisant la couleur et la couleur + la forme

Le tableau montre que la combinaison de deux types de descripteurs de nature différente (couleur et forme) donne un taux moyen de précision plus élevé que celui obtenu en utilisant uniquement le descripteur couleur. Nous passons ainsi de 52% à 62% dans le cas où la recherche est effectuée avec les distances à noyau (KRA$^+$-Blocks) et de 46% à 48% pour la distance Euclidienne (pour $k-ppv=40$).

On peut conclure qu'il est plus avantageux de combiner plusieurs types de descripteurs pour arriver à un meilleur taux de précision. Cette combinaison est d'autant plus intéressante si nous utilisons une distance à noyau avec des paramètres adaptés plutôt qu'une simple distance Euclidienne.

Par ailleurs, pour montrer l'intérêt des fonctions noyau crées lors du bouclage de pertinence et éventuellement utilisées pour mesurer la similarité entre ces différents types de descripteurs, nous avons mené les mêmes expérimentations que les précédentes en utilisant le bouclage de pertinence. Les résultats du pourcentage de la précision en fonction des descripteurs utilisés sont montrés dans le tableau suivant.

N° Images recherchées	Descripteur couleur (couleur dominante,Histogramme)		Descripteurs couleur et forme (couleur dominante, Histogramme, ART)	
	Sans bouclage de pertinence	Avec bouclage de pertinence (it1)	Sans bouclage de pertinence	Avec bouclage de pertinence (it1)
10	72.13	82.85	87.68	96.84
20	61.51	75.44	75.55	94.25
40	48.34	63.68	59.53	78.83

Tableau 5.4. Comparaison de la précision (en %) utilisant la couleur et la couleur avec la forme

Les résultats obtenus confirment nos conclusions concernant la combinaison de plusieurs types de descripteurs ainsi que les distances que nous pouvons utiliser pour une telle combinaison. Notons que dans ce test, l'amélioration du taux moyen de la précision est significative en comparaison avec les résultats précédents. Par exemple, pour $k - ppv = 40$, le taux de précision obtenu avec un descripteur couleur (couleur dominante, histogramme) est de 63.6% alors qu'avec les deux types de descripteurs (couleur et forme) on obtient 78.8%, soit un taux d'amélioration de 15% contre un taux d'amélioration de 10% sans bouclage de pertinence. Ceci peut être expliqué par le fait que les distances à noyau utilisées lors du bouclage de pertinences sont plus adaptées à la nature des données.

2.7 Expérimentation 5 : Temps de la recherche

Le but ici est d'étudier le comportement de notre méthode de recherche lorsque le nombre d'images ainsi que la dimension de l'espace de données varient. Pour cela, nous avons évalué notre méthode sur deux bases de données de grandes tailles, BD_3 et BD_4. Nous comparons d'une part notre méthode (KRA^+-Blocks) avec la méthode KVA-File [KVA] et la méthode de recherche séquentielle sur la base BD_4. Et d'autre part, nous étudions le comportement de notre approche lors du passage à l'échelle (grand nombre de vecteurs) ainsi qu'en grande dimension (grande taille des vecteurs), en calculant les temps de réponse sur la base BD_3 que nous avons conçue spécialement pour cet objectif.

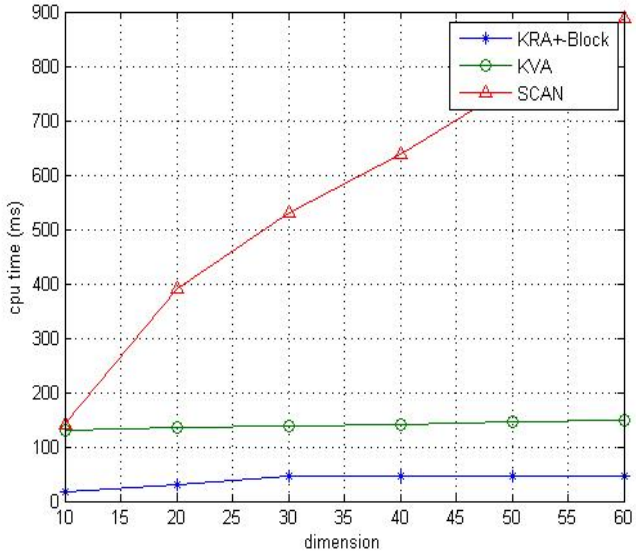
2.6.1Influence de la dimension

Cette expérience étudie l'influence de la dimension des descripteurs sur les performances de la recherche. Nous utilisons pour cela les deux bases BD_3 et BD_4. Rappelons que la dimension de BD_3 et BD_4 est respectivement 250 et 60, ce qui fait que la dimension peut varier respectivement pour ces deux bases entre 2 et 250 et entre 2 et 60. Notons que nous avons fixé la capacité des régions à 10.000, la taille de la page disque à 200 KB, le nombre de bits par dimension à 8

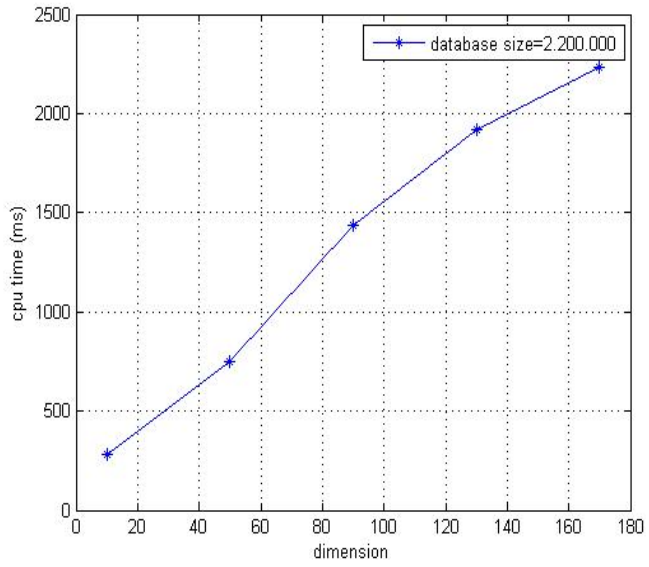
Ainsi, pour différentes valeurs de la dimension, nous avons cherché les 10 plus proches voisins de chaque vecteur requête d'une part en utilisant KRA^+-Blocks, KVA-File et la recherche séquentielle sur 100.000 descripteurs de la base BD_4 (Fig 5.15.a), et d'autre part en utilisant

uniquement notre méthode KRA$^+$-Blocks sur la totalité des descripteurs de la base BD_3.(Fig 5.15.b).

Comme nous pouvons le voir sur la figure 5.15.a, la recherche séquentielle garde un coût linéaire lorsque la dimension augmente (de 10 à 60). La courbe montre aussi que l'évolution du temps de réponse de la méthode KVA-File est linéaire aussi, mais avec une pente beaucoup moins importante que la recherche séquentielle. Le temps de réponse de notre méthode quant à lui évolue légèrement en fonction de la dimension. A titre d'exemple, lorsque la dimension est égale à 35, le temps de réponse de la recherche séquentielle est de 600ms, alors que celui du KVA-File est de 150 ms et celui de notre méthode est de 40 ms. Pour cet exemple, KRA$^+$-Blocks est 3 fois plus rapide que le KVA-File et 15 fois plus rapide que la séquentielle. Notons que ce rapport évolue linéairement avec une pente assez grande en fonction de la dimension pour la recherche séquentielle et avec une pente moins importante dans le cas du KVA-File.



(a)



(b)

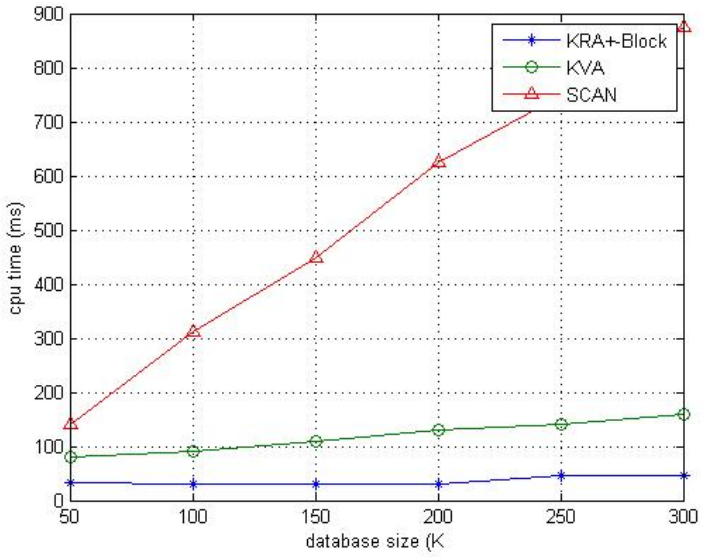
Fig.5.15. le temps de réponse en fonction de la dimension pour la base (a) BD_3. (b) BD_4

Dans la deuxième expérimentation (fig.5.15.b), nous avons étudié le comportement de la méthode KRA$^+$-Blocks sur la base BD_3 de 2.200.000 descripteurs de dimension 250. Nous

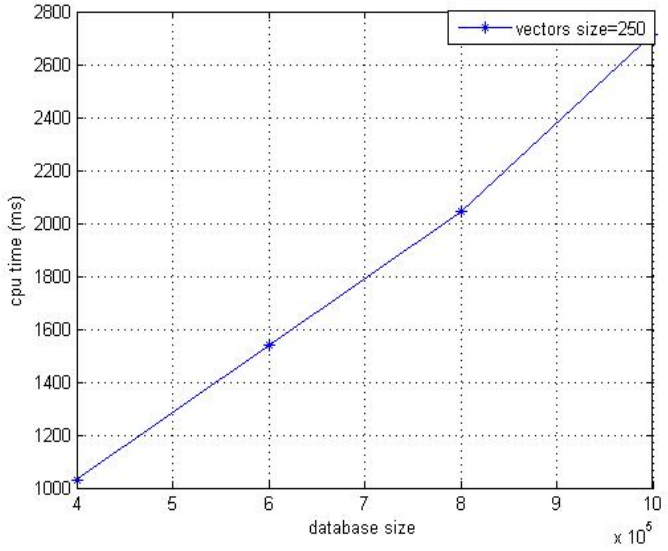
montrons sur cette figure le temps de réponse de la structure KRA$^+$-Block lorsque la dimension varie de 10 à 180. Nous avons arrêté les tests à la dimension 180 du fait que le processus d'indexation, même s'il est hors ligne, devient très couteux au delà de la dimension 100. D'après la figure, on remarque que le temps de réponse suit une fonction linéaire croissante en fonction de la dimension. L'évolution du temps de réponse est relativement faible en fonction de la dimension par rapport à la recherche séquentielle. A titre d'exemple, entre la dimension 20 et 40, on a une pente de 1.5 pour un nombre de vecteurs égal à 2.200.000 alors que pour la recherche séquentielle, dans le même intervalle (entre la dimension 20 et 40), on trouve une pente de 1.5 mais pour un nombre de vecteurs égal à 100.000. On peut conclure que KRA$^+$-Blocks est performante en grande dimension.

2.6.2 Influence du nombre de vecteurs

Ce test a pour but d'étudier l'évolution du temps de réponse en fonction du nombre de vecteurs dans la base. Pour cela nous avons utilisé les deux bases BD_3 et BD_4 et nous avons fixé la dimension à sa valeur maximale (250 pour la base BD_3 et 60 pour la base BD_4). Les résultats sont illustrés dans la figure 5.16. Celle-ci montre que le temps de réponse de notre approche évolue légèrement en fonction de la dimension en comparaison à KVA-File et à la recherche séquentielle. Ainsi, le temps de réponse obtenu par notre méthode pour 200.000 vecteurs de dimension 60 est de 40 ms alors que pour la même dimension et le même nombre de vecteurs, nous obtenons 130 ms pour KVA-File est 630 ms pour la recherche séquentielle, soit un facteur d'accélération de 3 par rapport à KVA-File et de 15 par rapport à la recherche séquentielle. Ceci montre bien que notre approche améliore le temps de réponse par rapport aux autres méthodes lors du passage à l'échelle.
Dans la figure 5.16.b, nous montrons le comportement de KRA+-Blocks au delà de 300.000 descripteurs. On remarque que le temps de réponse évolue linéairement en fonction de la taille de la base de données. En revanche, l'évolution est relativement faible en comparaison avec la recherche séquentielle



(a)



(b)

Fig.5.16. Evolution du temps de réponse en fonction de la dimension pour la base (a) BD3. (b) BD4

3 Synthèse

Nous avons évalué dans un premier temps dans ce chapitre les performances en terme temps de réponse, nombre de régions obtenues lors de la phase de découpage, et taux de remplissage de la méthode d'indexation multidimensionnelle RA$^+$-Blocks. Les performances ont été comparées avec la recherche séquentielle, VA-File et RA-Blocks sur une base de descripteurs réels de 60.000 vecteurs de dimension 252 et sur une base de 60.000 descripteurs uniformes de dimension 250. Dans notre approche, les bases d'images sont représentées par des vecteurs de caractéristiques de grande dimension. Les résultats obtenus en terme temps de réponse, nombre de régions obtenues ainsi que temps de remplissage montrent que la structure d'index RA+-Blocks est performante en grande dimension et aux passage à l'échelle. Notre méthode améliore la recherche séquentielle, le VA-File et RA-Blocks en optimisant la phase de découpage de l'espace de données et en créant des régions compactes et disjointes réalisant ainsi une bonne gestion de la mémoire et augmentant la capacité de stockage de la structure d'index.

Dans la seconde partie de ce chapitre, nous avons évalué la méthode d'indexation et de recherche proposée, appliquée au cas de la recherche d'images par le contenu KRA+-Blocks. Nos résultats liées aux performances aussi bien en temps de réponse du système qu'en terme de précision sont comparés à ceux de la recherche séquentielle, de KVA-File et de RA$^+$-Blocks sur des bases de tailles 7200, 40000, 300 000 et 2200000 avec deux types de descripteurs, un pour la couleur et un pour la forme. Dans notre approche, les bases d'images sont représentées par des vecteurs de caractéristiques de grande dimension. Ces vecteurs sont projetés à l'aide d'un noyau dont les paramètres sont calculés hors ligne, dans un espace de dimension réduite à travers une ACPK. Nous construisons la structure d'index KRA$^+$-Blocks dans l'espace de projection, puis nous établissons les distances à noyau correspondantes. La recherche des $k-ppv$ à travers la structure d'index du KRA$^+$-Blocks permet d'avoir rapidement de meilleurs résultats en comparaison avec RA$^+$-Blocks et RA$^+$-Blocks associé à une ACP linéaire. En effet, la projection non linéaire dans notre approche permet d'une part d'établir une bonne séparation

des données dans l'espace de projection, mais également de proposer une mesure de similarité à noyau adaptée à la nature des données. Les propriétés des fonctions noyau sont également exploitées dans la phase d'apprentissage pour définir des distances flexibles qui répondent au besoin des utilisateurs. D'un autre côté, notre approche est plus rapide que la recherche séquentielle exhaustive ainsi que la méthode KVA-File. Notre algorithme de recherche induit un ordre qui rend prioritaire le traitement des régions les plus intéressantes en se basant sur des distances à noyau adaptées. D'autre part, l'utilisation de l'approche du filtrage permet de réduire le nombre de régions à explorer et par la suite réduit le temps de la recherche.
Les expérimentations conduites sur les bases BD_1, BD_2, BD_3, et BD_4 montrent que notre approche améliore considérablement le temps de la recherche ainsi que la précision de la recherche par rapport à KVA-File et RA^+-Blocks. En recherche, le gain important de temps obtenu par rapport à KVA-File et à la recherche séquentielle du à l'utilisation de l'approche approximation des régions est pénalisé par une perte relative de qualité de la recherche, notamment en comparaison avec la recherche séquentielle exhaustive. L'étude de l'impact du nombre et de la taille des vecteurs a montré que notre méthode s'adapte bien aux grandes bases d'image de vecteurs multidimensionnels en comparaison à d'autres méthodes. Nos expérimentations ont montré également que la combinaison des descripteurs hétérogènes dans la même structure d'index est possible avec les distances à noyau, celles-ci améliorant notablement la précision de la recherche en comparaison avec la distance Euclidienne.

Conclusion

Dans le cadre de ce travail, nous avons d'abord amélioré les performances de la méthode d'indexation et de recherche d'images basée sur l'approximation RA-Blocks. Ensuite nous avons étendue cette méthode à l'espace à noyau (KRA+-Blocks) puis nous l'avons intégré dans un système de recherche d'images par le contenu IMALBUM. Nous commençons par résumer point par point les travaux effectués ainsi que les résultats obtenus pour les deux méthodes (RA+-Block et KRA+-Blocks), puis nous présentons nos perspectives pour le prolongement de ce travail.

Synthèse des travaux effectués

La recherche et l'indexation basée sur l'approximation

Notre objectif a été tout d'abord d'améliorer les performances des méthodes d'indexation multidimensionnelles pour ensuite pouvoir développer une approche rapide et efficace d'indexation et de recherche, particulièrement adaptée aux applications de recherche d'images fixes par le contenu. Pour atteindre cet objectif, nous avons commencé par un état de l'art sur les méthodes d'indexation multidimensionnelles. Les travaux présentés dans cet état de l'art montrent que les performances des méthodes d'indexation conventionnelles se dégradent lorsque la dimension de l'espace de données et/ou la taille de la base de données augmente. Cette étude montre aussi que certaines méthodes sont liées à des contraintes concernant par exemple la distribution des données (distribution uniforme) et la distance/Espace utilisé (Euclidien).

Ensuite, nous avons mis en avant le problème de la malédiction de la dimension et rappelé les principaux problèmes liés à l'indexation multidimensionnelle des images (le problème de l'espace vide, vecteurs équidistants en grande dimension, etc.). Des méthodes de la réduction de la dimension ont été proposées pour contourner certains problèmes liés aux espaces de grande dimension. Ces méthodes possèdent cependant plusieurs inconvénients. Soit elles ne fonctionnent que pour les données qui appartiennent à un espace linéaire (méthodes linéaires de la réduction de la dimension), soit leur mise en place nécessite un paramétrage difficile à réaliser par un utilisateur non spécialiste.

Par ailleurs, nous avons remarqué que les travaux présentés dans la littérature contournent le phénomène de la malédiction de la dimension par un recours aux méthodes basées sur l'approximation (filtrage). C'est pour cette raison que nous avons décidé de nous placer dans ce contexte. Nous avons cherché parmi les méthodes d'indexation existantes celles qui peuvent être adaptées à nos besoins. Sur la base de cette étude, nous avons choisi de travailler avec la méthode RA-Blocks [Ter 02] pour les avantages qu'elle présente. C'est une technique possédant une structure dynamique se basant sur l'approximation des données, de plus, elle améliore la performance à la fois des méthodes conventionnelles (R-Tree [2], X-Tree [3], etc.), et du VA-File [web 98], qui est la méthode représentative de l'approche "approximation". En revanche,

RA-Blocks présente des limitations au niveau de la phase de découpage de l'espace de données. En effet, le partitionnement de celui-ci s'effectue selon l'algorithme KDB-Tree [Rob 81], dont le principal inconvénient est la génération d'un très grand nombre de régions vides ou ne contenant que très peu de vecteurs par rapport à leur capacité. Par conséquent, ceci entraîne une mauvaise gestion de l'espace mémoire, ce qui diminue la capacité de stockage de l'index et augmente le temps de la recherche. Pour pallier à ces problèmes, nous avons proposé le RA+-Blocks qui améliore nettement la stratégie du découpage de l'espace de données. Notre stratégie cherche à partitionner uniquement les régions contenant un nombre de vecteurs supérieur à la capacité des régions. Ceci est possible en remplaçant la structure arborescente du KDB-Tree par une structure linéaire, et en utilisant un algorithme de découpage inspiré du KD-Tree pour subdiviser les régions denses. Les régions ainsi obtenues sont compactes est disjointes et leurs nombre est beaucoup plus faible que celui obtenu par l'algorithme de découpage KDB-Tree. Plusieurs expérimentations ont été menées en comparant notre approche à d'autres méthodes d'indexation existantes. Les résultats de ce travail ont été soldés par un article publié dans la revue Engineering Letters [2].

Mesure de similarité pour les données hétérogènes

Disposant d'une méthode efficace d'indexation basée sur l'approximation, nous nous sommes intéressés par la suite à son intégration à un système de recherche d'images par le contenu. Il est évident que l'efficacité de la recherche d'images dépend largement de la mesure de similarité utilisée. Particulièrement, si les données sont indexées à travers une structure d'index multidimensionnelle, il est important de faire appel à une mesure de similarité globale aux différents types de descripteurs, afin de pour pouvoir les indexer avec la même structure. C'est pour cette raison que nous nous sommes intéressés au problème de la mesure de similarité entre les descripteurs des images.
Pour établir une mesure de similarité adéquate, nous avons passé en revue les principales approches pour la description de l'apparence visuelle des images fixes (descripteurs), puis les différentes approches proposées dans la littérature pour mesurer la similarité entre ces descripteurs. Une attention particulière a été accordée au problème de l'hétérogénéité des données ainsi qu'à la combinaison de différents types de descripteurs dans l'objectif de les indexer plus tard à travers une même structure d'index. Sur la base de cette étude, nous avons remarqué que la notion de similarité par approche noyau offre un cadre théorique qui permet de combiner les mesures de similarité de plusieurs types de descripteurs en limitant autant que possible les problèmes qui se posent lors de la combinaison des descripteurs hétérogènes (normalisation, pondération, etc.) Cette approche nous a permis de généraliser la notion de distance et de définir un modèle à partir duquel, une seule mesure de similarité est établie pour les différents types de descripteurs.

La recherche et l'indexation basée sur l'approximation : application à la recherche d'images par le contenu

La deuxième partie de notre travail concerne la conception et l'intégration d'une méthode d'indexation multidimensionnelle basée sur l'approche approximation dans un système de recherche d'images par le contenu. A ce stade, plusieurs problèmes ont été soulevés : Le premier se pose lors de l'étape d'indexation. En effet, cette étape consiste à générer, à partir de la base d'images, un espace de vecteurs de caractéristiques ayant généralement un très grand

nombre de composantes, donc une très grande dimension, difficile à gérer par les méthodes d'indexation existantes. Le deuxième problème se présente lors de l'étape de la structuration de l'espace de données. Il s'agit d'organiser en index, des vecteurs multidimensionnels composés de descripteurs de différentes natures « descripteurs hétérogènes ». Le troisième problème consiste à définir une distance appropriée pour mesurer la similarité entre ces descripteurs hétérogènes.

La méthode proposée apporte une solution à l'ensemble de ces problèmes. Elle permet de réduire la dimension de l'espace de données à l'aide de l'analyse en composantes principale à noyau, ACPK, et utilise des distances adaptées, dérivées de l'approche à noyau. En effet, l'ACPK permet de projeter les données initiales dans un espace de dimension réduite par un noyau avec un double intérêt : d'une part cette projection élimine les corrélations non linéaires entres les caractéristiques des vecteurs, ce qui permet de générer une représentation compact est informative des données, et d'autre part, grâce à la fonction noyau, elle défini une mesure de similarité adaptée aux données hétérogènes.

Il est bien établi que l'efficacité de la recherche des plus proches voisins dépend fortement de la compacité et de la séparabilité des classes de vecteurs. Si ces deux propriétés ne sont pas satisfaites, le chevauchement entre les classes de vecteurs est important. Par conséquent, la phase de filtrage utilisée lors de la recherche devient inefficace ce qui ne permet pas de sélectionner correctement les plus proches voisins. Ainsi, le bon choix des paramètres du noyau est primordial pour assurer une recherche efficace des $k-ppv$. Afin de garantir une bonne représentation de nos données, nous avons exploité certaines propriétés de l'ACPK. En effet, nous avons remarqué que la dimension de l'espace de projection ainsi que le paramètre de l'échelle de la fonction noyau gaussienne que nous avons adopté, agissent d'une manière significative sur la distribution des données générées dans l'espace de projection. Nous avons mené des tests expérimentaux dans ce sens et nous avons proposé une stratégie de sélection des paramètres optimaux qui permettent d'atteindre un maximum de séparation des données avec le minimum de dimension possible. Les expérimentations sur la qualité de la recherche ont confirmé l'efficacité de notre stratégie de sélection.

Notre méthode d'indexation multidimensionnelle KRA+-Blocks utilise l'approximation des régions pour réduire le temps de la recherche, elle consiste d'abord en une phase hors ligne servant à quantifier l'espace de données en des cellules hyper rectangles, ensuite l'espace de données est partitionné en régions disjointes et compactes selon la stratégie de découpage du RA+-Blocks. Puis, chaque région est approximée par une chaine de bits correspondant aux deux cellules "en bas à gauche" et "en haute à droite", sachant que le processus de recherche sélectionne les régions candidates selon leurs distances minimales et maximales par rapport au vecteur requête. Enfin, les régions sélectionnées sont explorées pour calculer les $k-ppv$. Les distances utilisées dans la phase de filtrage ainsi que pour le calcul des $k-ppv$ sont toutes dérivées de l'approche à noyau. Les expérimentations menées pour évaluer le temps de réponse montrent bien que la combinaison d'une méthode de la réduction de la dimension avec l'approche approximation des régions réduit bien le temps de la recherche.

Pour améliorer la qualité de la recherche, nous avons adopté un modèle probabiliste de bouclage de pertinence qui s'appuie sur la théorie du Kernel Tricks, ce modèle définit d'une manière flexible les mesures de similarité en fonction du retour de l'utilisateur. Il exploite les propriétés des fonctions noyaux pour générer à partir d'un noyau donné, de nouvelles distances à noyau afin d'estimer la similarité entre des descripteurs hétérogènes et de s'approcher le plus possible des besoins de l'utilisateur.

Evaluation Expérimentale

Pour évaluer notre méthode, nous avons mené une étude expérimentale étendue portant à la fois sur le temps de réponse et sur la qualité du processus de la recherche. Nous avons d'abord réalisé une étude comparative en terme de précision entre notre méthode, la recherche séquentielle, RA+-Blocks et RA+-Blocks avec ACP. Cette étude a montré que notre méthode est efficace pour ce qui est de la qualité des réponses par rapport aux autres méthodes. En fait, elle présente de meilleurs taux de rappel et de précision tout en restant moins performante qu'une recherche séquentielle exhaustive. En effet, notre approche se base sur l'approximation, dont l'idée principale est de favoriser le gain en temps de réponse au détriment d'une faible dégradation de la qualité de la recherche. D'un autre côté, les mêmes comparaisons ont été reprises sur une base de plus grande taille pour étudier le comportement de notre méthode lors du passage à l'échelle. Nous avons remarqué que notre méthode garde les meilleurs taux de rappel et de précision par rapport aux autres méthodes à l'exception, encore une fois, de la recherche séquentielle exhaustive dont la qualité reste meilleure pour un temps de réponse de loin supérieur.
Dans une deuxième expérimentation, nous avons montré l'efficacité du mécanisme du bouclage de pertinence adopté dans notre méthode. Les résultats montrent qu'avec seulement une itération du bouclage de pertinence, nous améliorons significativement les taux de précision et de rappel.
Nous avons également montré dans une troisième expérimentation l'intérêt de l'utilisation d'une mesure de similarité par une fonction noyau sur la combinaison des descripteurs de nature hétérogène. Nous avons conduit des expérimentations sur des descripteurs de forme et de couleur. Les résultats montrent que la combinaison de ces descripteurs avec l'utilisation d'une distance à noyau améliore considérablement la qualité de la recherche en comparaison avec l'utilisation d'une distance Euclidienne.
Une dernière expérimentation a été menée sur une base de données synthétiques pour montrer la performance de notre méthode en terme de temps de réponse. Des comparaisons ont été effectuées entre notre méthode, KVA-File, et RA+-Blocks en fonction de la taille de la base de données et de la dimension des descripteurs. Nous avons montré à nouveau que notre méthode est plus performante que les autres, et qu'elle garde un coût de recherche assez faible aussi bien en grande dimensions qu'au passage à l'échelle.

Perspectives

Les perspectives que nous envisageons dans le prolongement de ce travail de thèse s'articulent autour des points suivants.

Coût de l'indexation

Un point que nous n'avons pas abordé dans ce manuscrit est le coût du processus d'indexation. Bien que l'indexation d'une base soit hors ligne, ce processus est généralement très couteux notamment au passage à l'échelle et en grande dimension. Il serait donc intéressant d'étudier la complexité de l'algorithme d'indexation et d'optimiser les opérations du partitionnent de l'espace de données et la création de l'index.

Le choix des paramètres de la structure d'index

La capacité des régions et le nombre de bits de codages sont deux paramètres critiques, difficiles à fixer alors que les performances des deux index RA+-Blocks et KRA+-Blocks en dépendent fortement. Il serait donc intéressant de mener une étude pour quantifier la relation, d'une part entre la capacité de région et la qualité de l'approximation, et d'autre part entre le nombre de bits de codage et le nombre de régions obtenues lors du partitionnement de l'espace de données. Quantifier la relation entre le nombre de bit de codage, la capacité des régions et la qualité de l'approximation serait un atout pour construire une structure d'index optimale en terme de qualité de l'approximation et de capacité de stockage notamment pour les grands volumes de données multidimensionnelles.

La mise à jour de la structure d'index

Dans les deux méthodes d'indexation basées sur l'approximation RA+-Block et KRA+-Blocks, nous avons travaillé avec des bases d'images de taille fixes. L'extension des deux méthodes à la problématique de l'ajout de nouvelles images dans une base reste à traiter. Les nouvelles images rajoutées à la base sont susceptibles de modifier la distribution des données dans l'espace des caractéristiques, et un travail de construction des fonctions noyaux reste nécessaire pour tout changement de données. Il serait intéressant là encore, d'une part de mettre en place une procédure de mise à jour afin de maintenir dynamiquement la structure d'index, et d'autre part, d'étudier l'impact de ces modifications sur le choix de la fonction noyau et de ses paramètres car, vu que la distribution des données risque d'‘être modifiée, cela peut entraîner des changements nécessaires dans le choix de la fonction noyau et de ses paramètres.

Apprentissage de la fonction noyau

Bien que l'approche noyau nous permette d'améliorer la qualité de la recherche, elle reste néanmoins basée sur le critère de maximisation du taux de séparation des classes (indicateur de performance dans notre manuscrit). Or dans notre contexte, c'est le taux de précision qui est déterminant. Il serait donc un atout de construire les fonctions noyau les plus adaptées à la recherche d'images par le contenu, en se basant directement sur un paramètre d'évaluation de la qualité de la recherche (précision moyenne par exemple).

Annexe

1 Construction de l'index KDB-Tree

Algorithme Construction_index_KDB_Tree (P : *ensemble de vecteurs*, *Racine* : la région correspondant à l'espace de données, RR, ID : les couples régions et le pointeur correspondant sur la feuille, feuille : une page point, x_i : hyperplan de subdivision)

1: **Si** Racine est nulle,

2: **alors** créer une page point contenant p est une région (RR, ID) telle que ID est un pointeur sur la page point

3: Sinon{

4: chercher la page point devant contenir p soit $feuille_1$

5: Inserer p dans $feuille_1$}

6: **Si** $feuille_1$ est non saturée

7: **alors** terminer

8: Sinon {

9: déterminer un hyperplan de subdivision x_i

10: Subdiviser la page suivant x_i}

11: **Si** $feuille_1 = Racine$

12: **alors** aller à

13: Sinon

14: remplacer (RR, ID) par (RR_gauche, ID_gauche) et (RR_droite, ID_droite)

15: **Si** $feuille_1$ est saturée

16: **alors** répéter à partir de 8

17: Insérer (RR_gauche, ID_gauche) et (RR_droite, ID_droite) respectivement à gauche et à droite de la page région

2 Construction de l'index KD-Tree

Algorithme construction_KD_index recConst(sous espace V , objets O)

---

1 : Tester s'il faut diviser le sous espace courant V :

2 : Si non : Rendre la Feuille(O) contenant la liste des objets O

3 : Si oui : Continuer

4 : Trouver le bon plan séparateur p

5 : Couper le voxel V avec p pour obtenir VG et VD

6 : Repartir les objets O dans VG et VD) OG et OD

7 : Rendre le Nœud (p, recConst(VG ,OG), recConst(VD,OD))

3 Algorithme de recherche VA-NOA

Proc VA-NOA (Vect : vector) ;

Var i ,Cand , Li , Hi : int
Heap : HEAP ; Init (Heap) ;

//La phase filtrage

1. Cand : = Initcandidate () ;
2. **For** i : = 1 TO Vect **Do**
3. Li , Ui : = Getbounds (Ai , Vect) ;
4. **If** Li <= Cand THEN
5. Insertheap (Li , i) ;
End ;
End ;

//La phase accès au vecteurs
6. Cand : Initcandidate () ;
7. Li , i : = PopHeap (Heap)
8. **While** Li < Cand **Do**
9. D : = Candidate (Lp (Vect(i) , Vect(q)) , i) ;
10. Li , I : = PopHeap (Heap)
End ;
End VA – NOA ;

4 centrage des données dans l'espace à noyau

Etant donnée un ensemble de vecteurs $S = \{p_1, p_2, ...p_N\}$ appartenant à un espace d'entrée χ et ϕ, une application non linéaire associée à un noyau k, telle que $k(p_1, p_2) = <\Phi(p_1), \Phi(p_2)>$.
Les vecteurs : $\widetilde{\Phi}(p_i) = \Phi(p_i) - \frac{1}{N}\sum_{j=1}^{N}\Phi(p_i)$ sont centrés.
Donc la matrice $K = k(p,q)_{p,q \in S}$ dans la reformulation de l'ACP du paragraphe devient :
$K_{i,j} = \widetilde{k}(p_i, p_j) = (\widetilde{\Phi}(p_i), \widetilde{\Phi}(p_j))$ dans l'espace H.
Le problème des valeurs propres devient : $\widetilde{K}\widetilde{Z}_i = \widetilde{\lambda}_i \widetilde{Z}_i$ tel que $\widetilde{Z}_i = (\widetilde{z}_{i,1}, ..., \widetilde{z}_{i,N})$ est un vecteur dont les composantes vérifient:

$$\widetilde{e}_i = \sum_{p \in S} \widetilde{Z}_{i,rank(p)} \widetilde{\Phi}(p)$$

Puisque les données ne sont pas centrés, le calcul de $\widetilde{K}$ n'est pas direct, il peut etre exprimé en fonction de $K_{i,j} = (\Phi(p_i), \Phi(p_j))$. Soient $1_{i,j} = 1$ pour tout i et j, $1_N(i,j) = 1/N$

$$\begin{aligned}
\widetilde{K}_{i,j} &= ((\Phi(p_i) - \frac{1}{N}\sum_{t=1}^{N}\Phi(p_t)).(\Phi(p_j) - \frac{1}{N}\sum_{t=1}^{N}\Phi(p_t))) \\
&= K_{i,j} - \frac{1}{N}\sum_{t=1}^{N} 1_{it} K_{tj} - \frac{1}{N}\sum_{l=1}^{N} K_{il} 1_{lj} + \frac{1}{N^2}\sum_{t,l=1}^{N} 1_{it} K_{tl} 1_{lj} \\
&= (K - 1_N K - K 1_M + 1_M K 1_M)_{ij}
\end{aligned}$$

Donc $\widetilde{K}$ peut etre calculé à partir de K, pour résoudre l problème de l'ACP à noyau il suffit de diagonaliser la matrice $\widetilde{K}$.

Référence de l'auteur

Revues:

[1] : I. Daoudi, K. Idrissi , S.E. Ouatik, A. Baskurt, and D. Aboutajdine, "An efficient High-Dimensionnal Indexing Method for Content-Based Image Retrieval in Large Image Databases," EURASIP Journal, Signal processing : Image communication, 2009, (A apparaître)

[2] : I. Daoudi, S.E. Ouatik, A. El Kharraz, K. Idrissi, and D. Aboutajdine, "Vector Approximation Based Indexing for High-Dimensional Multimedia Databases". Engineering Letters, Volume 16 Issue 2, 2008, Pages 210-218.

Conferences:

2009

[3] : I. Daoudi, K. Idrissi, S. E. Ouatik, "A multi-class metric learning for content-based image retrieval, " IEEE international Conference on Image processing, ICIP 2009. (papier soumis)

[4] : I. Daoudi, K. Idrissi, S. E. Ouatik, "Une mesure de similarité par une approche noyau pour l'indexation et la recherché par le contenu dans les bases de données hétérogènes, " COmpression et Representation des Signaux Audiovisuels, CORESA 2009.

2008

[5] : I. Daoudi, K. Idrissi, S. E. Ouatik, "Kernel Based Approach for High Dimensional Heterogeneous Image Features Management in CBIR Context," Advanced Concepts for intelligent Vision Systems, ACIVS 08, Juan-les-Pins, France, October 20-24,2008

[6] : I. Daoudi, K. Idrissi, S. E. Ouatik, "Kernel Region Approximation Blocks For Indexing Heterogonous Databases," IEEE International Conference on Multimedia & Expo, ICME, Hannover, Germany June23,26, 2008.

2006

[7] : S.E. Ouatik, A. El Kharraz., I. Daoudi, D. Aboutajdine, "Une Structure d'Index Multidimensionnelle pour la Recherche dans les Grandes Bases de Données Multimédia, " The 3rd International Symposium on Image/Video Communications over fixed and mobile networks, ISIVC, September 13-15, Hammamet, Tunisia, 2006.

[8] : I. Daoudi, S.E. Ouatik, El Kharraz., "An Indexing Method for Similarity Search in High-Dimensional Image Databases," Second International Symposium on Communication, Control and Signal Processing, ISCCSP , 13-15 March, Marrakech, Morocco, 2006.

[9] : I. Daoudi, S.E. Ouatik, El Kharraz, "Une Méthode d'Indexation des Grandes Bases de Données Multidimensionnelles par une Approche de Filtrage, " Workshop on Telecom and Information Systems, E-Week´06 , Morocco Student Technical Paper Competition, First Best Paper Contest Award , February 21, fez, Morocco, 2006.

2005

[10] : I. Daoudi., S.E. Ouatik, El Kharraz., D. Aboutajdine, "Recherche par approximation dans les grandes bases de données multimédias, " Workshop sur les Technologies de l'Information et de la communication, WOTIC'05, June 24-25 , Kenitra, Morocco 2005.

Références Bibliographiques

[Aloi] http://staff.science.uva.nl/~aloi/

[Ams 01] Amsaleg, L. & Gros P.. Content-base retrieval using local descriptors: problems and issues from database perspective. Pattern Analysis and Applications, Special Issue on Image Indexation, 4: 108-124, 2001

[Aro 50] Theory of reproducing kernels. Transactions of the American Mathematical Society, 68 :337–404.

[Baj 73] Bajscy, R. Computer Identification of Visual Surfaces. Computer Graphics and Image Processing, 2, 118-130,1973

[Bal 04] Balko, S., Schmitt, I., et Saake, G.The active Vertice Methode :a performant Filtring Approach to High-Dimensional Indexing. Data and Knowledge. Engineering, 51(3°, 369-397,2004

[Bay 72] R. Bayer, E. McCreight, Organization and maintenance of large ordered indexes, Acta Informatica 1 (3) 173–189,1972

[Bay 71] R. Bayer, Binary B-Trees for Virtual Memory, Proceedings of 1971 ACM-SIGFIDET Workshop on Data Description, Access and Control, San Diego, California, November 11-12, 1971.

[Bbj 00] S. Berchtold, C. Böhm, H. V. Jagadish, H. P. Kriegel, and J. Sander. independent quantization : An index compression technique for high-dimensional data spaces. In Proceedings of 16-th Int. Conf. on Data Engineering, IEEE ICDE, pages 577–588, San diego, California, USA, 2000.

[Bbk 98] S. Berchtold, C. Bohm and H.-P. Kriegel, “The Pyramid Technique: Towards Breaking the Curse of Dimensionality”, in Proc. ACM SIGMOD Int. Conf. on Management of Data,p. 142-153,1998

[Bec 90] 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 ACM SIGMOD International conference on Management of Data, pages 322–331, Atlantic City, NJ, USA, 1990. ACM Press.

[Bel 61] R. E. Bellman.Adaptive Control Processes. Princeton University Press, Princeton, NJ. [points: high dimensional data; curse of dimensionality] cited in F.P (p.xvi), F.4 (p.486), F.4.6 (p.663) , 1961

[Bel 02] Belongie S., Malik J., Puzicha J., “shape matchnig and object recognition

using shape contextd", IEEE transactiond on Pattern Analysis and Machine intelligence, vol. 24, n°4, p.509-522, 2002

[Ben 75] J. L. Bently. Multidimensional binary search tree used for associative searching. Communications of ACM, 18(9):509-517, 1975.

[Ben 90] J.L Bentley. Kd-trees for semi-dynamic point set. In 6 th Ann. ACM, pages 187–197, Sympos. Comput. Geom,1990

[Ben 79] J. L. Bentley and J. H. Friedman, \Data structures for range searching," ACM Computing Surveys, vol. 11, pp. 397{409, 1979.

[Ber 04] Sid-Ahmed Berrani Recherche approximative de plus proches voisins avec contrôle probabiliste de la précision ; application à la recherche d'images par le contenu., 2004

[Ber96] S. Berchtold, D.A. Keim, H.-P. Kriegel, The X-tree: an index structure for high-dimensional data, in: VLDB, 1996, pp. 28–39.

[Ber 84] Berg, C., Christensen, J. P. R., & Ressel, P. Hamonic analysis on semigroups. Springer-Verlag,1984

[Bey 99] Beuer, K. S., Goldstein, J., Ramakrishnan, R., et Shaft, U. (1999). When Is "Nearest Neighbor" Meaningful? In Proceeding of the 7th International Conference on Database Theory (ICDT'99), pages 217-235, London, UK, Springer-Verlag

[Bfe 79] J. L. Bently and J. H. Friedman. Data structures for range searching. ACM Computing Surveys, 11(4):397-409, 1979.

[Bff 77] J.L Bentley, J.H Friedman, and R.A Finkel. An algorithm for finding best matches in logarithmic expected time. ACM Transaction on Mathematic Software, 3(3) :pages 209–226, 1977.

[Bha 06] Bharat, R. " A comparative analysis of dimensionality reduction techniques"

[BIM 99] Del Bimbo A., Visual information retrieval, Morgan Kaufmann, 1999

[Bos 92] Boser, B. E., Guyon, I., & Vapnik, V. N. (1992). A training algorithm for optimal margin classi_ers. Proceedings of the Fifth Workshop on Computational Learning Theory (pp. 144-152).

[Cha 01] Charu, C. Aggarwal. „On the effects of dimensionality reduction on High Dimensional Similarity search", In Procceding of ACM Symposium on Principles of database systems (PODS), 2001

[Cha 98] Cha G.-H., CHUNG C.-W., "Object-Oriented Retrieval Mechanism for Semi-structured Image Collections", 6th ACM International Conference on Multimedia Bristol, Royaum-Uni, 14-16 september 1998

[Cha 01] E. Chavez, G. Navarro, R. Baeza-Yates, and J. Marroquın. Searching in Metric Spaces. ACM Computing Surveys, 33(3):273–321, 2001.

[Cha 02] Cha, G.-H., Zhu, X, Petkovic, D., & Chang, C.-WAn efficient indexing method for nearest neighbor searches in high-dimensional image databases. IEEE Transactions on Multimedia, 4(1) :76-87, 2002

[Cia 97] P. Ciaccia, M. Patella, P. Zezula, M-tree: an efficient access method for

similarity search in metric spaces, in: M. Jarke, M.J. Carey, K.R. Dittrich, F.H. Lochovsky, P. Loucopoulos, M.A. Jeusfeld (Eds.), Proceedings of the 23rd International Conference on Very Large Data Bases, VLDB'97, Morgan Kaufmann, Greece, pp. 426–439, 1997

[Cou] http://www1.cs.columbia.edu/CAVE/software/softlib/coil-100.php

[Cri 00] Cristianini, N., & Shawe-Taylor, J. (2000). An introduction to support vector machines. Cambridge University Press.

[Cva 02] H. Chen, J. An, K. Furuse, and N. Ohbo, "C2VA: Trim High Dimensional Indexes," The Third Inter. Conf. on Web-Age Information Management (WAIM), Beijing, China, 2002, pp.303-315.

[Cui 07] J. Cui., S. Zhou, s. Zhao, 'PCR-Tree: a compression-based Index Structure for similarity searching in High-dimensional Image Databases. In Proceeding of the Fourth International Conference on Fuzzy Systems and Knwledge Discovery (FSKD 07), 2007

[Dao 08] I. Daoudi, S.E. Ouatik, A. El Kharraz, K. Idrissi, and D. Aboutajdine, "Vector Approximation Based Indexing for High-Dimensional Multimedia Databases". Engineering Letters, Volume 16 Issue 2Pages 210-218, 2008

[Dao 00] M. Daoudi and S. Matusiak. Visual image retrieval by multiscale description of user sketches. Journal of Visual Computing and languages, 11 :pages 287–301, 2000. special issue on image database visual querying and retrieval.

[Dau 85] J.G. Daugman: Uncertainty relations for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters, Journal of the Optical Society of America A, vol. 2, pp. 1160-1169, 1985

[Dou 05] Douglas R. Heisterkamp and Jing peng. "Kernel Vector approximation Files for Relevance Feedback Retrieval in Large Image Databases" In Computer science: Multimedia tools and applications, volume 26, pages 175-189, 2005

[EAK 99] John P Eakins and Margaret E Graham , " Content-based Image Retrieval", A report to the JISC Technology Applications Programme. Institute for Image Data Research, University of Northumbria at Newcastke. January 1999

[Ebe 94] D.H. Eberly. Geometric Methods for analysis of Ridges in N-dimensional Images. Phd thesis, University of North Carolina at Chapell Hill, 1994.

[Fal 94] Faloutsos C., Barber R., Flickner M., Hafner J., Niblack W., Petkovoc D., "efficient and effective querying by Image Content", Journal of Intelligent Information Systems, vol. 3, n°3-4, p.231-262, 1994.

[Fer 05] Ferecatu M., Image retrieval with active relevance feedback using both visual and keyword-based descriptors, these de doctorat, université de versailles Saint-Quentinen-Yvelines, Juillet 2005.

[Fli 95] M. Flickher, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J.Hafner, D. Lee, D. Petkovic, D.Steele, and P. Yanker, "Query by Image and Video Content: The QBIC System," IEEE Computer, vol. 28, no. 9, pp. 23-32, Sept. 1995.

[GEV 00] Gevers T., Smeulders A., "Pictoseek: combining shape and color invariant features for image retrieval.", IEEE transactions on Image Processing, vol. 9,

n°1, p.102-119, 2000.

[GEV 04] Gevers T., Smeulders A. W., “Content-based image retrieval: An overview”,Medioni G., Kang S., Eds., Emerging topics in Computer Vision, Prentice Hall, 2004.

[Gae 98] V. Gaede and O. G unther. Multidimensional access methods. ACM Computing Surveys, 30(2):170{231, 1998.

[Gua 02a] Guang- Ho Cha, and al. An Efficient Indexing Method for Nearest Neighbour Searchs in high-Dimensional Image Databases. IEEE Transactions On Multimedia, Vol4, N°2 June 2002.

[Gua 02b] 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.

[Guo 07] Guoren W., Zhou X., Bin W., Qiao B., Han D., “A hyperplane based indexing technique for high dimensional data,” In Proceeding of Information Sciences 177 (2007) 2255-2268

[Gut 84] A. Guttman. R-trees: A dynamic index structure for spatial searching. In Proc. of the ACM SIGMOD Int. Conf. on Management of Data, pages 47-57, Boston, MA, June 1984.

[Had 93] Haddon, J. F. et Boyce, J.FCo-occurrence Matrices for Image Analysis. IEE Electronics and communication Engineering Journal, 5(2), 71-83,1993.

[Haf 95] J. Hafner et al, "Efficient color histogram indexing for quadratic form distance functions", IEEE trans. Pattern Analysis and Machine Intelligence, Vol. 17, pp 729-736, 1995.

[Hak 00] H. Ferhatosmanoglu, E. Tuncel, D. Agrawal, “Vector approximation based indexing for non-uniform high dimensional data sets”, Proc. of the ACM Int’1 Conf. on Information and Knowledge Management(CIKM2000).New York, 2000, pp. 202-209

[Har 73] Haralik, R. M., Shanmugam, K., et Dinstein, I.Textural features for Images Classification. IEEE Transaction on System, Man, Cybernetics, 3, 610-621 ,1973.

[Has 01] [T. Hastie, R. Tibshirani, and J. Friedman. The elements of statistical learning. Springer, New York, 2001.

[Has 95] : Hasting S., « Query categories in a study of intellectual access to digitized art images”, 58th Annual meeting of the American Society for information Sciences, 1995

[Hei 01] Heisterkamp, D., Peng J., Dai, H.: An Adaptive Quasi-conformal Kernel Metric for Image Retrieval. In Proceedings of IEEE CVPR, Kauai Marriott, Hawaii. pp. 236-243,(2001).

[Hen 89] A. Henrich, H.S. Six, and P. Widmayer, The LSD tree: spatial access to multidimensional point and non-point objects, Proc. 15th VLDB Conference, pp. 45-53,1989

[Hja 95] [G. R. Hjaltason and H. Samet. Ranking in Spatial Databases. In Proceedings of the 4th Symposium on Spatial Databases, pp. 83–95,

Portland, Maine, August 1995.

[Hot 33] Hotelling, H. (1933). Analysis of complex statistical variables into principal components. Journal of Educational Psychology, 24:417-441, 498-520

[Hsh 89] A. Henrich, H.-W. Six, and P. Widmayer. The LSD-tree: Spatial access to multidimensional point and non-point objects. In Proceedings of the 15th International Conference on Very Large Data Bases (VLDB), pages 45-53, Amsterdam, The Netherlands, 1989.

[Hua97] Huang, J., Kumar, S., Mitra, M., Zhu, W.-J., and Zabih, R. Image indexing using color correlogram. In Proc. of IEEE Conf. on Computer Vision and Pattern Recognition. , pages 762-768. 1997.

[Hu 62] Hu M., "Visual Pattern recognition by moment invariants", IEEE Transactions on Information Theory, vol. 8, p. 179-187, 1962

[Idr 03] Khalid Idrissi. Segmentation et description de la couleur pour l'indexation, la navigation visuelle et la recherche dans les bases d'images fixes., 2003

[Idr 01] K.Idrissi, J. Ricard, A. Anwander, A. Baskurt, « An image retrieval system based on local and global color descriptors », In Proc.of the 2nd IEEE Pacific PIM conference on Multimedia, pp.55-62, Beijing, China, October, 2001.

[Ing00] Ingemar J. Cox, Matthew L. Miller, Thomas P. Minka, Thomas Papathomas, and Peter N. Yianilos. The bayesian image retrieval system, PicHunter: Theory, implementation and psychophysical experiments. IEEE Transactions on Image Processing, pages 20-37. 2000.

[Iqb02] Q. Iqbal and J. K. Aggarwal, CIRES: A System for Content-based Retrieval in Digital Image Libraries , Invited session on Content Based Image Retrieval: Techniques and Applications International Conference on Control, Automation, Robotics and Vision (ICARCV), Singapore, pp. 205-210, December 2-5, 2002.

[Iso 00] J.B. Tenenbaum, V. de Silva, and J.C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. Science, 290 :2319–2323, December 2000

[Jai 98] Jain A., Vailaya A., « image retrieval using color and shape », Pattern recognition, vol.29, n°8, p.1233-1244, 1998

[Jea 01] Jeannim, S. « MPEG-7 visual pary of experimentation model, Version 9.0 » dans ISO/IEC JTC1/SC29/WG11/N3914,55th Mpeg Meeting. Pisa, Italy.2001.pages 27, 39, 41

[Kar 47] Karhumen, K. (1947). Uber linear Methode n wahrsche in linlichtstsrechnug. Ann. Acad. Sci. Fenn.37.

[kat97] N. Katayama, S. Satoh, The SR-tree: an index structure for high-dimensional nearest neighbor queries, in: Proc. Int. Conf. on Management of Data, vol. 26, no. 2, 1997, pp. 369–380.

[Kel 95] Kelly, P. M., Cannon, T. M., et Hush, D. R.Query by Image Example: The CANDID Approach. SPIE Storage and retrieval for Image and Video Databases III, 2420, 238-248,1995

[Kim 99] W .Y. KIM, Y.S.Kim: 'A new region-based shape descriptor', TR# 15-01, December 1999

[Kim 00] Kim H., Kim J., Sim D., “Zernike moment shape descriptor invariant to translation, rotation and scale for similarity based image retrieval”, IEEE International Conference on Multimedia expo, p. 307-310, juillet 2000.

[Kul 59] Kullback, S.Information theory and statistics. Wiley (New York),1959

[LEV 76] Levy-Schoen, A. « Exploration et connaissance de l’espace visuel sans vision pèriphérique ; quelques données sur le comportement oculomoteur de l’adulte normal ». Journal Psychologique, 39(1) :77–91. pages 37,1976.

[LLE00] : Sam T. Roweis and Lawrence K. Saul. Nonlinear Dimensionality Reduction by Local Linear Embedding. Science, 290 :2323–2326, December 2000.

[Loe 48] Loève, M. (1948); fonnctions aléatoires de second ordre. Hermann, Paris.

[Lon02] Fuhui Long, Hongjiang Zhang, David D. Feng: Fundamentals of Content-based Image retrieval, in Multimedia Information Retrieval and Management – Technological Fundamentals and Applications, D. Feng, W.C. Siu, and H.J.Zhang. (ed.), Springer, 2002.

[LS90] D. Lomet and B. Salzberg. The hB-tree: A multiattribute indexing method with good guaranteed performance. ACM Transactions on Database Systems, 15(4):625-658, 1990.

[Ma99] Wei-Ying Ma and B. S. Manjunath. Netra: A toolbox for navigating large image databases. Multimedia Systems, 7(3):184-198, 1999.

[Mac 96] MacQueen J., « Some methods for classification and analysis of multivariate observations »,in Proc. Fifth Berkeley Symp, University of California Press 1, p. 281-297,1996

[Man 02] B. S. Manjunath, P. Salembier, T. Sikora, "Introduction to MPEG-7", Willey, 2002.

[Maj 96] F. Mokhtarian, S. Abbasi, , and J.Kittler. Robust and efficient shape indexing through curvature scale space. Proceedings of the sixth British Machine Vision Conference, BMVC 96, pages 53–62, 10-12 September 1996.

[Man 96a] Manjunath, B. S. et Ma, W. Y., , « Texture features for Browsing and Retrieval of Image Data », IEEE Transaction on Pattern Analysis and Machine Itelligence, 18(8), 837-842, 1996

[Man 96b] Manjunath, B. S., Salembier, P.; et Sikora, T. Introduction to MPEG-7. John Wiley & Sons,2002

[Mar 80] Marcelja, S. Mathematical description of the response of simple cortical cells Journal of Optical Society of America, A 70(11) :1297–1300.pages 18,1980.

[Man 96] Manjunath, B.S., S., Salembier, P., et Sikora, T. 2002. Introduction to MPEG-7. John Wiley &Sons

[MB+] [7] M. Ishikawa, H. Chen, K. Furuse, J.X. Yu, N. Ohbo, MB+tree: a dynamically updatable metric index for similarity search, in: H. Lu, A. Zhou (Eds.), Proceedings of the 1st International Conference on Web-Age

Information Management, WAIM'00, Springer,Shanghai, China, pp. 356–366,2000

[Mer 09]. Functions of positive and negative type and their connection with the theory of integral equations. Philosophical Transactions Royal Society London, A209 :415–446.

[Min 95] E. Horowitz, S. Sahni, and D. Mehta, Fundamentals of Data Structures in C++. Rockville, MD: Computer Science, 1995.

[Mic 94] L. Mico, J. Oncina, and E. Vidal. A new version of the nearest-neighbor approximating and eliminating search (aesa) with linear preprocessing-time and memory requirements. Pattern Recognition Letters, 15:9–17, 1994.

[Mok 92] Mokhtarian F., Mackworth A., " A theory of multiscale, curvature-based shape representation for planar curves", IEEE Transactions on Pattern Analysis and Machine Intelligences, Vol. 14, n°8, p. 789-805, 1992

[MTR97] P. Ciaccia, M. Patella, P. Zezula, M-tree: an efficient access method for similarity search in metric spaces, in: M. Jarke, M.J. Carey, K.R. Dittrich, F.H. Lochovsky, P. Loucopoulos, M.A. Jeusfeld (Eds.), Proceedings of the 23rd International Conference on Very Large Data Bases, VLDB'97, Morgan Kaufmann, Greece, pp. 426–439, 1997

[Nas 98] Nastar, C., Mitschke, M., Meilhac, C., etBoujemaa, N. Surfimage : a flexible Content-based Image Retrieval System. In proceeding of the 6th ACM International Conference on multimedia (MULTIMEDIS 98), pages 339-344, Bristol, UK,1998

[Nhs 84] J. Nievergelt, H. Hinterberger, and K. Sevcik. The grid _le: An adaptable symmetric multikey _le structure. ACM Transactions on Database Systems, 9(1):38-71, 1984.

[Nib 93] Niblack, W., Barber, R.,Equitz, W., Flickner, M., Glasman, E., Petkovic, D., Yanker, P., Faloutsos, "The QBIC project: Quering images by content unsing color, texture and shape." In proceeding of SPIE Conference on Storage and Retrieval for image and Video Databases, 2-3 February, San Jose, CA, pp. 173-187, 1993

[Lu 06] H.Lu, B.C.Ooi, H.T.Shen, X.Xue « Hierarchical Indexing Structure for Efficient Similarity Search in Video Retrieval ». IEEE Transactions On Knowledge And Data Engineering, Vol 18, No. 11, November 2006.

[Ogl95] Virginia E. Ogle and Michael Stonebraker. Chabot: Retrieval from a relational database of images. IEEE Computer, 28(9):40-48, September 1995.

[Ooa 00] B.C. Ooi., K.L. Tan, C. Yu, S. Bressan, "Indexing the Edges - A Simple and Yet Efficient Approach to High-Dimensional", in 19th ACM SIGMOD SIGACT SIGART Symposium on Principles of Database Systems, Dallas, USA, May, p.166-174, 2000

[Orl 02] R. Orlandic, and B. Yu, A retrieval technique for high-dimensional data and partially specified queries, Data and Knowledge Engineering 42(1), 2002,

pp.1-22

[Ort97] Michael Ortega, Yong Rui, Kaushik Chakrabarti, Sharad Mehrotra, and Thomas S. Huang. Supporting similarity queries in MARS. In Proceedings of the 5th ACM International Multimedia Conference, Seattle, Washington, 8-14 Nov. '97, pages 403- 413, 1997.

[OTYB00] B. C. Ooi, K. L. Tan, C. Yu, and S. Bressan. Indexing the edges - a simple and yet efficient approach to high-dimensional. In Proceedings of 19-th ACM SIGMOD
SIGACT SIGART Symposium on Principles of Database Systems, pages 166–174, Dallas, Texas, USA, 2000.

[Pas96 a] Pass G., Zabih R., « histogram refinement for content based image retrieval », 3rd IEEE workshop on Applications of computer Vision, IEEE Computer Society, page 96, 1996

[Pen 96] Pentland A., Picard R, Sclaroff S., “Photobook: Content-base Manipulation for Image Databases” International Journal of Computer Vision, vole.18, n°3, p. 233-254, 1996.

[Peng 03] Peng. J.; Heisterkamp, D.R. “Kernel indexing for relevance feedback image retrieval” Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on Volume 1, Issue , 14-17 Sept. 2003 Page(s): I - 733-6 vol.1

[Per 77] Persson E., Fu K., “Shape discrimination using Fourier Descriptors”, IEEE Transactions on Systemes, Man and Cybernetics, vol.7, n)3, p.170-102, 1996

[Pet 00] Petrakis E., Milios E., “shape retrievzl based on dynamique programming”, IEEE Transactionon image Processing, vol. 9, n°1, p. 141-147, 2000

[Pic 96b] Picard R., Minka T., Szummer M., “Modeling subjectivity in Image Libraries”, IEEE international Conference on Image, Lausanne, september 1996.

[Phi 08] Philip, H., Matthieu, C., Sylvie, P., « Combining visual dictionary, kernel based similarity and learning strategy for image category retrieval”, Computer Vision and Image Understanding110 403-417,2008)

[Sko 04] T. Skopal, J. Pokorny, V. Snasel, PM-tree: pivotin metric tree for similarity search in multimedia databases, in: G. Gottlob, A.A. Benczur, J. Demetrovics (Eds.), Proceedings of the 8th East-European Conference Advances in Databases and Information Systems, ADBIS’04, Springer, Budapest, Hungary, pp. 99–114, 2004

[Pre 92] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. Numerical Recipes in C – The Art of Scientific Computing. Cambrigge University Press, 2nd edition, 1992.

[Puz 97] Puzicha, J., Hofmann, T., & Buhmann, JNon-parametric similarity measures for unsupervised texture segmentation and image retrieval. IEEE Conference on Computer Vision and Pattern Recognition (pp. 267{272) ,1997.

[Rob 81] 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.

[Row 00] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. Science, 290(5500) :2323–2326, 2000.

[Rui99] Yong Rui, Thomas S. Huang, and Shih-Fu Chang, Image retrieval: current techniques, promising directions and open issues, Journal of Visual Communication and Image Representation, Vol. 10, no. 4, pp. 39-62, April 1999.

[Rub 99] Rubner, YPerceptual metrics for image database navigation. Doctoral dissertation, Stanford University,1999

[Sah 03] Sahbi, H., "Machnies a vecteur de support pour une détection hiérarchiques des visages »,2003

[Sam 84] Samet, H.The quadtree and Related Hiearchical Data structures. ACM Computing Surveys, 16(2), 187-260,1984

[San 95] Santini, S. et Jain, R. « Similarity Matchnig », In Proceeding of the 2nd Asian Conference on Computer Vision (ACCV'95), pages 571-580, Singapore, 1995

[Sbb 02] S. M. Savaresi, D. Boley, S. Bittanti, and G. Gazzaniga. Cluster selection in divisive clustering algorithms. In Proceedings of the Second SIAM International Conference on Data Mining, Arlington, VA, USA. SIAM, 2002

[See 90] B. Seeger, and H.P. Kriegel, The Buddy-tree: An Efficient and Robust Access Method for Spatial Data Base Systems, Proc. 16th VLDB Conference, pp. 590-601, 1990

[Sch 98] B. Schölkopf, A. Smola, and K. Müller. Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation, 10 :1299–1319, 1998.

[Sch 01] Scholkopf, B., Platt, J. C., Shawe-Taylor, J. C., Smola, A. J., and Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. Neural Computation, 13(7) :1443‑1471.

[Sch 02] B. Schölkopf, A. Smola. Learning with kernels. MIT Press, Cambridge, MA,2002

[Scl 95] Sclaroff S., Pentland A., "Model Matching for correspondance and recognition", IEEE Transactions on Pattern Analysis and Machine intelligence, vol. 17, n°6, p.545-56, 1995

[Sco 83] Scott, D.& Thompson, J. R. Probability density estimation in higher dimensions. In the 15th Symposium on the Interface, North Holland-Elsevier, Amsterdam, New York, Oxford., pages 173-179.1983

[Set 95] Sethi, I., & Patel, N. Statistical approach to scene change detection. Storage and Retrieval for Image and Video Databases (pp. 329-338).1995

[Slim00] C. Traina Jr., A. Traina, B. Seeger, C. Faloutsos, Slim-trees: high performance metric trees minimizing overlap between nodes, in: C.Zaniolo, P.C. Lockemann, M.H. Scholl, T. Grust (Eds.), Proceedings of the 7th International Conference on Extending Database Technology, EDBT'00, Springer, Konstanz, Germany, pp. 51–65, 2000

[Smi 96] Smith J., Chang S.-F., “VisualSEEK: A Fully automated Content-based Image retrieval System”, 4th ACM International Conference on Multimedia, Boston, Massachussets, Etats-Unis, p. 87-98, November 1996

[Soz 05] H. T. Shen, B. C. Ooi, and X. Zhou. Towards effective indexing for very large video sequence database. In Proceedings of the ACM SIGMOD International Conference on Management of Data, pages 730–741, Baltimore, Maryland, USA. ACM, 2005

[Str 95] M. Stricker, A Orengo, "Similarity of color images", In Proc. SPIE Storage and Retrieval for Image and Video Databases III, Vol 2420, pp. 381-392, February 1995.

[Str 96] M. Stricker, A. Dimai, "Color indexing with weak spatial constraints", In Proc. SPIE Storage and Retrieval for Image, Vol 2670, pp. 29-40, 1996

[Str 95] Stricker, M., & Orengo, MSimilarity of color images. SPIE, Storage and Retrieval for Image Video Databases III (pp. 381-392).1995

[suy 02] Suykens, J. A. K., Van Gestel, T., De Brabanter, J., & De Moor, B. Least square support vector machines. World Scientifc Publishing Co., Pte, Ltd., Singapore. 2002

[Swa 91] Swain M., Ballard D., “Color Indexing”, International Journal of Computer Vision, Vol. 7, n°1, p. 1-13, 1991.

[Syu 00] Y. Sakurai, M. Yoshikawa, S. Uemura, and H. Kojima. The a-tree : An index structure for high-dimensional spaces using relative approximation. In VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt, pages 516–526.Morgan Kaufmann, 2000.

[Sye 00] T. Syeda-Mahmood, D. Petkovic, "On describing color and shape information in images", Signal Processing: Image Communication, Vol 16, pp. 15-31, 2000.

[Tea 80] Teague M., “image analysis via the general theory of moment”, Journal of optical society of America, vol. 20, n°8, p.920-930, 1980

[Thi 06] Thierry Urruty, Fatima Belkouch and Chabane Djeraba. Indexation Multidimensionnelle : KpyrRec, une amélioration de Kpyr. 22ème journées Informatique des Organisation et Systèmes d’Information et de Décision (INFORSID-2006), p831-846, Hermès Ed., ISBN 2-906855-22-7, Hammamet, Tunisie, 1er au 3 Juin 2006.

[Thi 05] Urruty, T.; Belkouch, F.; Djeraba, C. “KPYR: An Efficient Indexing Method”, Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on volume , Issue , 6-6 July 2005 Page(s):1448 – 1451

[Ten 00] J. Tenenbaum, V. De Silva, and J. Langford. A global geometric framework for non linear dimensionality reduction. Science, 290(5500) :2319–2323, 2000.

[Ter 98] Terrillon C., David M., Akamatsu S., « Automatic detection of human faces

in natural scene images by use of a skin color model and of invariant moments », IEEE transaction in natural scene images by use of a skin color model and of invariant moments », IEEE International Conference on Automatic Face and Gesture Recognition, p 112-117, avril 1998

[Ter 02] T. Chen, M. Nakazato, S. Huang. Speeding up the similarity search in multimedia database.C2002 IEEE.

[Tor 65] Torgeson, W. S. Multidimensional Scaling of Similarity. Psychometrika, 30, 379-393, 1965

[Tra 00] C. Traina Jr., A. Traina, B. Seeger, C. Faloutsos, Slim-trees: high performance metric trees minimizing overlap between nodes, in: C. Zaniolo, P.C. Lockemann, M.H. Scholl, T. Grust (Eds.), Proceedings of the 7th International Conference on Extending Database Technology, EDBT'00, Springer, Konstanz, Germany, 2000, pp. 51–65.

[Tve 77] Tversky, A. Feature of Similarity. Psychological Review, 84-(4), 327-352, 1977

[Vap 99] Vapnik, , V. N. (1999). Statistical learning theory. Wiley-Interscience, New York.

[Wal 95] Walin A., Kubller O., "compete sets of complex Zernik moments and the role of pseudoinvariancets", IEEE Transactions on Pattern Analysis and Machine intelligence, vol. 17, n°22, p. 1106-1110, 1995

[Web 98] 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.

[Wu 00] Wu, P., Manjunath, B.S., & Shin, H. D. (2000). Dimensionality reduction for image search. In Proceedings of the 7th International conference on Image Processing, Vancouver, Canada

[Yi 05] Yi Lin, h., Huang P., "Perfect KDB-Tree; A compact KDB-Tree Structure for indexing Multidimensional Data. In Proceeding of the Third International Conference on Information Technology and Applications (ICITA'05), 2005.

[YOT 01] C. Yu, B.C. Ooi, K.-L. Tan and H.V. Jagadish, "Indexing the distance: An efficient method to Knn processing", in VLDB, September 2001, p. 421-430.

[Yu 03] B. Yu, T. Bailey, R. Orlandic, and J. Somavaram, KDBKD-Tree: a compact KDB-tree structure for indexing multidimensional data, Proc. International Conférence on Information Technologie, 2003.

[Zha 01] Zhang D., Lu G., "Content-based shape retrieval using different shape descriptors: a comparative study", IEEE International conference on multimedia and Expo, p. 137-320, août 2001

[Zhang 04] Zhang D. and G. Lu. Review of shape representation and description techniques. Pattern Recognition, 37:1-19, 2004.

[Zot 04] R. Zhang, B.C. Ooi and K.L. Tan, "Making the Pyramid Technique Robust to Query Types and Workloads", in IEEE ICDE, 20th International Conference on Data Engineering, Boston, USA, April p. 313-324,2004