INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Mise en base d'images indexées par des descripteurs locaux : problèmes et perspectives

Laurent Amsaleg , Patrick Gros , Rim Mezhoud

N°3903

Mars 2000

THÈME 3

ISSN 0249-6399 ISRN INRIA/RR--3903--FR

RENNES

Mise en base d'images indexées par des descripteurs locaux : problèmes et perspectives

Laurent Amsaleg* , Patrick Gros† , Rim Mezhoud‡

Thème 3 — Interaction homme-machine,
images, données, connaissances
Projets Temics et Vista

Rapport de recherche n°3903 — Mars 2000 — 17 pages

Résumé : L'interrogation de bases d'images par le contenu est notoirement onéreux. Ce type d'interrogation a été rendu possible par le mariage de techniques bases de données et traitement d'images. Les derniers développements en bases de données visent à accélérer les recherches dans des index multidimensionnels. Ceux en image visent à améliorer les capacité de reconnaissance automatique. Or, les techniques modernes de traitement d'image ont pour conséquence de changer fondamentalement la manière dont la reconnaissance s'effectue : il est maintenant nécessaire d'interroger de nombreuses fois la base puis de synthétiser les résultats avant de pouvoir retourner une réponse, et non plus de se contenter d'une seule et unique interrogation comme par le passé. Cet article montre que ce nouveau mode d'interrogation disqualifie toutes les techniques connues en indexation bases de données. Il montre en particulier que les trois algorithmes donnés comme les plus performants sont inutilisables en l'état face au besoin de multiples interrogations. Il propose plusieurs pistes de recherche pour permettre l'interrogation de base d'images avec les techniques modernes de traitement d'images.

Mots-clé : indexation multimédia, index multidimensionnels, bases d'images, recherche par le contenu, descripteurs locaux

(Abstract: pto)

Ce travail a été effectué au sein de deux équipes communes au CNRS, à l'INRIA, à l'université de Rennes 1 et à l'INSA de Rennes, dans le cadre d'un projet de collaboration entre l'INRIA et l'ENSI de Tunis.

* Équipe Temics, IRISA - CNRS
† Équipe Vista, IRISA - CNRS
‡ GRIFT - ENSI, Rue des Entrepreneurs, Charguia II, 2035 Tunis-Carthage – Tunisie

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : 02 99 84 71 00 - International : +33 2 99 84 71 00
Télécopie : 02 99 84 71 71 - International : +33 2 99 84 71 71

Indexing images using local descriptors: problems and perspective from a database point of view

Abstract: Content-based image retrieval from large databases is very expensive. Such a retrieval was made possible by using data base together with image processing techniques. The latest developments in database technology try to speed-up the retrieval process in multidimensional indexes. New image processing techniques improve the recognition capabilities. However, these techniques change fundamentally the way the retrieval process is performed: Instead of submitting a single query to get the result, multiple queries must be submitted, and their partial results processed before delivering the answer. This paper shows that the actual indexing techniques do not cope with this new querying process. In particular, the three most efficient techniques up to today are unable to deal efficiently in such a case. Several research directions are stated to build large databases used with modern image techniques.

Key-words: multimedia indexing, multidimensional indexing, image databases, content-based retrieval, local descriptors

1 Introduction

Construire un système permettant d'interroger des bases d'images par le contenu sollicite des compétences que l'on trouve dans le domaine du traitement des images et dans celui des bases de données. Du domaine de l'image sont tirées les techniques d'analyse du signal qui permettent d'extraire des images des informations caractéristiques. Ces informations doivent permettre de reconnaître le contenu de l'image même en présence de certaines variations : illuminations différentes, recadrages, points de vues différents, déplacement d'objets dans l'image, etc. Ces informations extraites se présentent généralement sous la forme d'un ou de plusieurs ensembles de valeurs numériques réelles. On peut avoir affaire à un unique ensemble caractérisant la totalité d'une image, ensemble qu'on appelle alors un *descripteur global*, ou à plusieurs d'entre eux qui caractérisent chacun une partie de l'image, et qu'on appelle donc *descripteurs locaux*. Les techniques modernes en images tendent à préférer les descripteurs locaux aux globaux car ils sont plus souples, plus puissants, permettent des recherches plus fines et absorbent mieux certaines variations.

Les compétences en bases de données (BD) sont nécessaires dès que le volume de la base d'image devient trop important pour tenir en mémoire centrale. Il devient dans ce cas obligatoire de stocker sur disque les descripteurs. Les techniques BD sont alors employées pour accroître la vitesse des recherches et minimiser les coûts liées aux entrées-sorties. Les BD savent en effet organiser spécifiquement les données sur mémoire secondaire, offrent des techniques de structuration et d'indexation accélérant les recherches, et offrent également des garanties de fiabilité et de cohérence d'accès concurrents, garanties souhaitables au sein d'un système d'interrogation réel.

La littérature propose de nombreuses structures de données destinées à indexer des images caractérisées par des descripteurs globaux. Aucune étude n'a encore exploré les conséquences de l'utilisation de descripteurs locaux sur le comportement des techniques d'indexation bases de données, et en particulier de la complexité accrue que cela implique. En effet, avant de pouvoir retourner une réponse, il faut procéder à de très nombreuses interrogations de la base et analyser l'ensemble des informations récupérées. Dans l'état actuel, *une seule* interrogation s'effectue dans un temps à la limite de l'acceptable pour une recherche interactive, lorsque le nombre de dimension des descripteurs est important et lorsque la base d'images est large. Répéter de nombreuses fois cela n'est clairement pas viable.

Aussi, le but de ce papier est d'abord d'étudier, dans le cas où l'on utilise des descripteurs locaux, le comportement des trois techniques d'indexation actuellement données comme les plus performantes. Cette étude montre que, dans ce cas, leurs performances sont bien trop médiocres pour envisager leur utilisation au sein d'un système réel. Le second but de ce papier est de proposer des pistes de recherche pour mettre au point un procédé d'interrogation par le contenu d'une large base d'images caractérisées par des descripteurs locaux.

Ce papier est organisée de la façon suivante. La section 2 présente les techniques traditionnelles de traitement d'images pour la recherche par le contenu. La section 3 présente les techniques utilisées en bases de données pour indexer des données multidimensionnelles. La section 4 évalue les performances des trois techniques considérées à l'heure actuelle comme les plus performantes lorsqu'elles indexent de nombreuses données de grandes dimension. La section 5 présente les conclusions et les perspectives qui se dégagent de cette étude.

2 Techniques de traitement d'images pour la recherche par le contenu

Cette section survole tout d'abord les techniques traditionnelles utilisées dans le domaine du traitement des images pour permettre les recherches par le contenu. Nous présentons ensuite les problèmes spécifiques qui apparaissent lorsque l'on désire faire ce type de recherche sur une base contenant des données réelles caractérisées par des descripteurs locaux.

2.1 Approches traditionnelles

L'indexation de documents multimédia par le contenu est un champ actif de recherche [FBF+94, MCL97, DRD97]. Pour constituer la base, on calcule des descripteurs du contenu des documents que l'on veut indexer. Ces descripteurs *caractérisent* le contenu et sont utilisés lors de la recherche. Cette première phase de calcul

de descripteurs sur le corpus de documents est hors ligne. Pour effectuer une recherche, on présente au système un document à partir duquel il calcule un ou plusieurs descripteur qu'il compare avec ceux stockés dans la base. La comparaison permet de retrouver les éventuels documents similaires à celui recherché. Cette phase est en ligne, et ses performances sont critiques.

Mettre au point des descripteurs est une difficulté majeure. Ils doivent être facilement comparables et la distance entre deux descripteurs doit être reliée à la ressemblance entre les documents qu'ils décrivent. De plus, pour que la recherche de documents *similaires* soit possible, ils doivent prendre en compte les transformations de l'image qui ne changent pas la nature de ce qui est représenté : changement de l'illumination, changement du point de prise de vue, occultations partielles, présence d'autres objets, ...

On utilise en général des descripteurs numériques, qui se présentent sous forme de vecteurs réels dont la dimension varie typiquement entre 1 et 900. De tels descripteurs peuvent être comparés à l'aide de distances comme L_1, L_2 ou L_∞. Pour absorber l'effet des transformations, on calcule des descripteurs invariants [Rot95] ou quasi-invariants [BL93] à ces transformations, c'est-à-dire tels que deux images ne se différenciant que par une de ces transformations aient le même descripteur : ils peuvent ainsi être invariants aux changements d'illumination [FDF94, SH96], à certaines variations géométriques liées au changement de point de vue [MZ92, MZF93], ou aux deux à la fois [vGMU96].

Suivant les transformations considérées, on peut aussi choisir entre descripteurs globaux ou locaux. Dans le premier cas, un seul descripteur décrit l'image dans son ensemble. Les histogrammes de couleur ou de niveaux de gris en sont des exemples classiques [SS94], mais d'autres descripteurs existent comme les corrélogrammes [HKM+97] ou les angles de couleur [FCF96]. Ces descripteurs ont pour avantage d'intégrer de l'information sur toute l'image, ce qui est gage de robustesse contre le bruit affectant le signal. Par contre, ils ne permettent pas de distinguer des parties et donc des objets dans cette image, sauf cas particulier où l'objet apparaît seul sur un fond noir dans l'image.

Dans le cas de descripteurs locaux, chaque descripteur ne décrit qu'une partie de l'image et caractérise par exemple un seul objet. Chaque descripteur est en fait associé à une zone de l'image, un point, un segment ou une région que l'on commence par détecter ou sélectionner avant de calculer le descripteur lui-même. Cela permet donc de reconnaître un objet indépendamment de sa position dans l'image (propriété d'invariance aux translations dans l'image) ou des autres objets visibles dans l'image, même si ces derniers cachent partiellement l'objet recherché. Cette occultation ne modifiera en moyenne que certains descripteurs et la reconnaissance sera basée sur les autres.

Chaque image est ainsi associée à plusieurs descripteurs. Lors d'une requête, il faut interroger plusieurs fois la base consécutivement et synthétiser les résultats avant de retourner une réponse. Cette synthèse peut être faite suivant des techniques de vote de complexité variable, ou par une technique bayésienne [LG96, MPS97]. Cette complexité supplémentaire par rapport à l'utilisation de descripteurs globaux est amplement justifiée par les possibilité de reconnaissance d'objets offertes.

Les interrogations par le contenu procèdent le plus souvent par l'exemple : un utilisateur présente au système une image et attend que lui soient retournés les documents similaires. Cette notion de similarité est fondamentale : l'obligation de rechercher des documents similaires et non exactement identiques est due au *bruit* qui caractérise les images et tout ce qui en est dérivé : bruit thermique de la caméra, sensibilité des capteurs, discrétisation, ... ainsi qu'aux transformations qui changent l'image sans changer ce qui est représenté (rotations, ...). Les descripteurs sont eux aussi atteints par ce bruit et ces transformations et présentent donc une variabilité, même pour des images extrêmement proches. L'invariance des descripteurs est une manière d'absorber une partie de cette variabilité, mais cela ne suffit pas. De plus, toutes les transformations que l'on aimerait absorber par invariance ne peuvent être modélisée. On en peut donc se contenter de requêtes exactes.

Cela implique l'existence de deux types d'interrogations : une recherche des k-plus proches voisins de chaque descripteur de l'image requête [NN97] ou une recherche de tous les descripteurs de la base qui sont distants de moins de ε du descripteur requête [LG96]. Les recherches de k-plus proches voisins ramènent un nombre fixe de descripteurs. Ce sont les plus proches, ce qui ne signifie pas qu'ils soient proches. Dans un espace de grande dimension, cette recherche peut devenir très coûteuse car il est peut être nécessaire d'explorer une large portion de l'espace pour localiser les plus proches. Choisir k est généralement difficile. Les recherches par intervalle garantissent un retour de réponses pertinentes, mais peut être très nombreuses.

La difficulté ici est de fixer ε, bien que celui-ci ait une signification physique claire lorsque l'on sait caractériser la variabilité des descripteurs.

2.2 Obstacles à l'utilisation d'un SGBD

Dès que la quantité de documents à indexer est importante, il devient nécessaire de stocker les descripteurs sur disque. Ceci est particulièrement vrai lorsque l'on utilise des descripteurs locaux. Par exemple, une agence de photo possède entre 500 000 et 12 000 000 d'images. Si chacune est caractérisée par 100 descripteurs locaux de dimension 24, cela fait 9 Go de données à stocker par million d'images. Stocker ces descripteurs sur disques et structurer les données pour y accéder de manière efficace suggère d'utiliser une base de données (SGBD). La mise sur disque de ces données et leur organisation se heurte à quelques difficultés, qui font que l'emploi direct d'un SGBD existant n'est pas possible.

Premier problème: des vecteurs réels imprécis de grande dimension. Les descripteurs sont des vecteurs de réels, dont la dimension peut varier de 1 à 900 dans le cas général. La difficulté est double. Tout d'abord, la distance de comparaison des descripteurs fait intervenir toutes les dimensions à la fois (par ex. les distances L_1,L_2,L_∞, la distance de Mahalanobis, ...). On se retrouve donc à chercher au sein d'espaces réels de grande dimension, sans pouvoir séparer les dimensions pour les traiter les unes après les autres.

D'autre part, la variabilité des descripteurs fait que l'on doit *explorer* l'espace et non pas seulement chercher un point particulier. Ce type d'exploration a une complexité le plus souvent exponentielle avec la dimension de l'espace, problème connu sous le nom de *dimensionality curse* [WW96, LG97, BBK98].

Ces difficultés disqualifient la plupart des techniques d'indexation usuelle basées sur les R-Tree [Gut84], qui ont été mises au point pour des données exactes où la recherche n'est faite que sur peu de dimensions à la fois.

Deuxième problème: 600 interrogations pour une requête. Lorsque l'on utilise des descripteurs locaux, chaque image possède généralement entre 50 à 600 descripteurs. Cela impose une forte contrainte sur les performances du système, car le temps de réponse ressenti par l'utilisateur est celui des 50 à 600 interrogations successives et non celui dû à une seule. De plus, le temps de réponse doit intégrer le processus de décision. Dans l'état actuel, *une seule* interrogation s'effectue dans un temps à la limite de l'acceptable pour une recherche interactive, lorsque le nombre de dimension des descripteurs est important et lorsque la base d'images est large. Répéter de nombreuses fois cela n'est clairement pas viable.

Troisième problème: des distributions non uniformes. Divers systèmes d'indexation de la littérature sont testés en utilisant des jeux de données générés aléatoirement, le plus souvent selon une loi uniforme. L'indexation de données réelles viole l'hypothèse d'uniformité faite par beaucoup de méthodes.[1] On risque donc d'obtenir des performances dégradées par rapport à ce qu'annoncent les concepteurs. On peut tenter d'uniformiser les données en inversant par exemple la fonction de répartition pour chaque composante des vecteurs que forment les descripteurs. Cette transformation ne modifie pas le résultat des requêtes par intervalles mais change en revanche celui des requêtes de plus proches voisins. Uniformiser toutes les dimensions à la fois est beaucoup plus difficile [PTVF92] et modifierait les résultats des deux types de requêtes.

3 Techniques BD pour indexer de nombreuses images

Cette section présente un panorama des techniques employées en bases de données pour indexer des données multimédias (généralement restreintes à des images fixes). Ces techniques sont nécessaires dès que la base d'image devient conséquente et qu'il est obligatoire de stocker les descripteurs sur disque. Elles *structurent* les données en index pour accélerer les recherches de similarités en limitant autant que possible les entrées-sorties. Nous présentons tout d'abord les approches traditionnelles avant de détailler les deux approches les plus performantes à l'heure actuelle.

1. Nos évaluations, reportées dans la section 4.1.2, exhibent des distributions loin d'être uniformes. Par exemple, la deuxième composante de nos descripteurs locaux varie entre -20 et 36, et 99.14% des 350 000 valeurs obtenues sont comprises entre -1 et +1.

3.1 Approches traditionnelles

Les algorithmes d'indexation multimédia se classent en deux grandes catégories : ceux qui regroupent les données en fonction de leur proximité *(data-partitionning index methods)*, et ceux qui découpent *à priori* l'espace multidimensionnel et qui ensuite stockent les données selon ce découpage *(space-partitionning index methods)*. Toutes ces approches peuplent l'espace avec les descripteurs des images ou avec des approximations de ces descripteurs.

Les algorithmes de la première catégorie sont tous dérivés du R-Tree [Gut84], qui a été conçu à l'origine pour indexer des données bidimensionnelles. La notion centrale est celle de rectangle englobant : les feuilles de l'arbre référencent chaque objet au travers de son rectangle englobant, et les niveaux internes de l'arbre stockent les rectangles englobants ceux du niveau inférieur. Plusieurs critères peuvent être employés pour décider d'englober ou de conserver distincts deux rectangles de niveau inférieur [BKK96]. Le principe du R-Tree a ensuite été généralisé pour s'appliquer à des données multidimensionnelles. Le SS-Tree [WJ96] est une première variante basée sur les sphères qui apportent un gain net en performances. Par contre, il a été montré que le taux de chevauchement des hyper-sphères croît avec la dimension, augmentant le nombre de nœuds à analyser. Pour pallier à cela, le SR-Tree se base sur un découpage définit par l'intersection de sphères et de rectangles [KS97].

Lin, Jagadish et Faloutsos présentent avec les TV-Tree une technique demandant de classer les dimensions des descripteurs en trois classes [LJF94]. L'idée sous-jacente vient de la constatation que toutes les dimensions ne sont pas équivalentes et n'ont pas la même importance dans la recherche. Or, les performances des index étant fortement liées au nombre de dimensions des données, leur technique distingue les dimensions que le processus de recherche peut systématiquement ignorer, celles toujours utilisées et celles utilisées éventuellement pour discriminer plus efficacement les données. Les recherches utilisent ainsi un nombre *variable* de dimensions. Le défaut majeur de cette technique est qu'elle nécessite la connaissance *à priori* de la distribution précise des données selon chaque dimension.

Ces approches utilisent toutes un facteur de remplissage égal à 50% lors de l'éclatement des nœuds de l'arbre, ce qui garantit l'obtention d'arbres équilibrés et permet également de maximiser le taux d'utilisation des pages sur le disque. Or, [BBK98] démontre que, dans le cas général, ce facteur induit une probabilité d'accès à chacune des pages de l'index proche de 100% lors de chaque recherche. Il y a donc de fortes chances pour que la totalité de l'index soit parcourue, et ce au moyen de nombreux accès aléatoires au disque.

Les techniques de la seconde catégorie divisent l'espace multidimensionnel suivant une grille plus ou moins complexe et régulière, comme le font les grid-file [NHS84], K-D-B-Tree [Rob81], LSD$^{\mathrm{h}}$-Tree [Hen98]. Les données sont ensuite stockées dans les cases appropriées. Ces techniques deviennent rapidement inefficaces lorsque croît le nombre de dimensions des données. Le découpage rigide de l'espace, induisant l'existence de larges régions vides qui sont indexées, en est la principale raison. Par exemple, diviser chacune des 30 dimensions d'un espace en 2 donne naissance à 2^{30} cases (soit plus de 1 milliard), nombre typiquement bien plus grand que le nombre de descripteurs d'une base. De plus, lorsque le point recherché se situe près des frontières de découpage de l'espace, la recherche peut alors devoir explorer les cases voisines, en très grand nombre (généralement pour se rendre compte qu'il n'y a rien dans chacune). Le coût d'une recherche est alors prohibitif. Les techniques les plus modernes comme [AGGR98] ou [HK99] sont évaluées comme efficaces pour un nombre de dimensions faible, et pour un bruit (ε) peu important.

3.2 VA-File et Pyramid-Tree

De façon générale, les techniques présentées ci-dessus ne sont d'aucun secours lorsque la dimension des données indexée est importante. [WSB98] montre que le nombre de dimensions au dessus duquel une recherche séquentielle devient plus performante que toute navigation dans un index se situe autour de 10. Aussi, deux approches, le Pyramid-Tree [BBK98] et le VA-File [WSB98] ont été récemment proposées et conçues spécifiquement pour contourner l'obstacle de la dimensionalité des données. Elles ont, en effet, été conçues dès le départ de façon à ce que leur comportement ne dégénère pas trop rapidement lorsque sont indexées des données ayant de nombreuses dimensions. Le VA-File utilise une première phase d'approximation permettant de diminuer drastiquement le nombre de descripteurs à comparer et le Pyramid-Tree utilise une méthode

dont la complexité croît linéairement (et non exponentiellement) avec la dimension. Elles sont actuellement les plus performantes et nous les avons donc utilisés pour mener à bien nos évaluations.[2]

Avec le VA-File [WSB98], Weber, Schek et Blott développent une méthode qui vise à améliorer les performances de la recherche séquentielle, puisque celle-ci s'avère la plus efficace en grandes dimensions. Cette technique gère deux ensembles de données : un fichier contenant les descripteurs et un autre contenant une approximation géométrique de ces descripteurs. Ce dernier fichier doit tenir en mémoire centrale pour garantir de bonnes performances.

Lors de la création de l'index, pour réaliser l'approximation géométrique, le VA-File découpe chaque dimension d_i en 2^{b_i} cases (ce découpage est codé sur b_i bits), de telle sorte que les cases contiennent approximativement le même nombre de points. On découpe de même chacune des d dimensions de l'espace. Il existe ainsi 2^b cases, où $b = \sum_i b_i$, qui sont numérotées de 0 à $2^b - 1$. Les descripteurs de la base sont alors lus les uns après les autres[3], et l'approximation d'un descripteur est déterminée par la case entre les limites de laquelle il tombe. L'approximation est égale au numéro de la case identifiée. Le fichier d'approximation est ainsi composé de paires (identifiant de descripteur, numéro de case). Seules les informations liées aux cases contenant au moins un descripteur sont stockées, évitant ainsi le problème d'avoir à gérer un très grand nombre de cases vides, comme évoqué ci-dessus.

Lors de la recherche, le descripteur requête subit le même processus d'approximation. L'algorithme détermine alors les cases les plus proches de celle où se trouve la requête et les parcourt par distance croissante. Tant que les n plus proches voisins ne sont pas trouvés, les descripteurs associés à chaque case sont lus séquentiellement et le calcul de distance est effectué. Cette étape agit donc comme un filtre qui élimine de la recherche les cases (et donc les descripteurs) n'ayant aucune chance de faire partie de la réponse, limitant le nombre de comparaisons à faire par rapport à la recherche séquentielle.

Berchtold, Böhm et Kriegel proposent, avec le Pyramid-Tree [BBK98], une méthode qui divise l'espace $[0,1]^d$ en $2 \times d$ pyramides. Chaque pyramide a sa pointe placée au centre de l'espace (0.5, 0.5, etc.), possède une base ayant une surface de $d-1$ dimensions et est numérotée. Chaque pyramide est ensuite découpée en tranches, parallèlement à sa base. Les tranches près du sommet sont donc plus petites que celles qui sont proches de la base. Ce découpage de l'espace à la propriété de créer un nombre de cases croissant linéairement, et non exponentiellement, avec la dimension.

Ce double découpage permet à tout point de l'espace multidimensionnel d'être exprimé comme une paire (numéro de pyramide, hauteur dans la pyramide). Cette contraction de l'espace multidimensionnel permet l'emploi d'un index notoirement efficace pour ce type de données et pour des requêtes de type intervalle : le B^+-Tree. Une page du B^+-Tree correspond en fait à une tranche spécifique d'une pyramide particulière. Outre l'efficacité de cette structure, le B^+-Tree gère bien les accès concurrents et est résistant aux pannes, ces deux caractéristiques, fondamentales, étant souvent absentes des autres solutions proposées.

4 Évaluations de performance

Cette section décrit les évaluations de performance des trois techniques retenues (VA-File, Pyramid-Tree et recherche séquentielle) auxquelles nous avons procédé. La première évaluation montre l'évolution des performances des trois techniques lorsque les données sont décrites par des descripteurs de dimension croissante. La seconde évaluation montre l'influence de la taille de la base sur les performances. La troisième évaluation montre l'influence qu'a le nombre de descripteurs composant la requête sur les performances de la recherche. Nous présentons tout d'abord le contexte dans lequel ces expérimentations ont été réalisées.

4.1 Contexte des expérimentations

Pour procéder aux évaluations de performance présentées dans les sections suivantes, nous avons récupéré auprès des auteurs l'implémentation du VA-File et du Pyramid-Tree[4]. Nous avons également implémenté un algorithme de recherche séquentielle. Ces algorithmes ont été exécutés sur une station de travail SUN Ultra 5

2. Les performances de ces deux algorithmes n'ont pas encore été mis en rapport, à notre connaissance.

3. Nous en sommes toujours à la phase de création de l'index.

4. Roger Weber nous a gracieusement communiqué l'implémentation du VA-File. L'implémentation du Pyramid-Tree est disponible sur la page Web de Stefan Berchtold (`http://www.research.att.com/~berchtol`)

sous SunOS 5.7, disposant d'un processeur UltraSPARC-IIi cadencé à 333 MHz, de 384Mo de mémoire centrale et d'un disque local de 8Go. Les temps d'exécution des algorithmes qui sont reportés dans la suite sont obtenus grâce à `getrusage()` en additionnant le temps qu'a passé le processus en mode système et en mode utilisateur.

Nous avons analysé le code du VA-File et du Pyramid-Tree pour déterminer où placer les instructions de début et de fin de mesure du temps d'exécution. Nous avons procédé de même pour la recherche séquentielle. Nous avons également modifié les codes pour les rendre capables d'afficher, au besoin, les identifiants (et les distances) des descripteurs appartenants aux résultats. Ces identifiants permettent ensuite de retrouver les images associées, et de vérifier manuellement la similarité des images retournées avec celle composant la requête. Nous avons également modifié le code du Pyramid-Tree pour que celui-ci calcule une distance L_2 entre les descripteurs et non une distance L_∞. Sans cela, les plus proches voisins retournés par le Pyramid-Tree n'auraient pas été les mêmes que ceux retournés par le VA-File et la méthode séquentielle. Cette modification ne semble pas avoir eu d'impact sur les performances, seule la métrique employée ayant changé.

4.1.1 Implémentation de la recherche séquentielle

Les performances des techniques d'indexation multimédia sont telles qu'une approche séquentielle, sans index, passant en revue l'intégralité de la base reste compétitive. Nous avons ainsi implémenté cette stratégie pour la comparer aux deux autres techniques considérées comme les plus performantes. L'algorithme est implémenté en C++ et fonctionne de la manière suivante : l'algorithme suppose que lui sont donnés en entrée deux fichiers contenant une suite de flottants codés en binaire. Le premier contient tous les descripteurs de la requête, l'autre tous les descripteurs de la base. Le fichier requête est supposé pouvoir tenir entièrement en mémoire centrale (il y a en moyenne 150 descripteurs par requêtes, et, pour 24 dimensions, ce fichier occupe seulement 14400 octets en mémoire centrale). Celui-ci est chargé d'un coup. Puis, chaque descripteur du fichier de la base est lu, et comparé à tous les descripteurs requête. Les plus proches voisins de ceux-ci sont maintenus dynamiquement (il y a typiquement 10 plus proches voisins à maintenir pour chaque descripteur requête. Dans ce cas, pour 150 descripteurs requête, l'implémentation occasionne alors une occupation mémoire maximale inférieure à 100Ko). Ainsi, cet algorithme adopte, d'une certaine manière, le comportement typique des jointures où la plus petite relation est chargée en mémoire et la plus grande lue tuple à tuple. Cette stratégie est incomparablement plus efficace que de relire naïvement toute la base pour chaque descripteur requête. L'algorithme complet comporte moins de 300 lignes.

4.1.2 Nature des descripteurs

Notre étude utilise plus particulièrement un descripteur qui a été développé pour reconnaître des objets dans des images [SM97, MGS98]. Ce descripteur est de type descripteur local. Il tolère des occultation partielles et des changements de composition de la scène observée (déplacements des objets dans l'image, absence ou présence d'autres objets à proximité, ...). Il est également invariant aux translations et aux rotations 2D de l'image, ainsi qu'aux variations affines de l'illumination. L'évaluation de ce descripteur montre qu'il permet de retrouver dans la base près de 100% des objets recherchés si une image de ceux-ci a été préalablement stockée. [5]

Son mode de calcul est le suivant : on commence par détecter dans l'image des points particuliers, appelés points d'intérêt. Ces points doivent pouvoir être retrouvés de manière précise dans diverses images représentant un même objet dans des conditions différentes. Cette propriété de répétabilité est tout à fait importante : si on ne retrouve pas les mêmes points entre deux images, aucun descripteur ne peut être utile [SMB98]. Pour notre part, nous utilisons une version modifiée du détecteur de Harris, qui cherche dans l'image les minima de la fonction d'auto-corrélation bidimensionnelle du signal [HS88].

La figure 1 montre deux images et les points qui y ont été détectés : les croix claires correspondent aux points qui ont été trouvés dans les deux images, les autres sont des points qui n'ont été trouvés que dans une seule image.

On décrit alors le signal autour de chacun de ces points en convoluant le signal image avec les dérivées d'une gaussienne jusqu'à l'ordre 3 [FtHRKV94]. Cela donne 10 dérivées sur chacun des canaux rouge, vert

5. Une démonstration en ligne est accessible en :
`http://www.inrialpes.fr/movi/pub/Demos/Reconnaissance_Cordelia/fr/reconnaitre_objets.html`

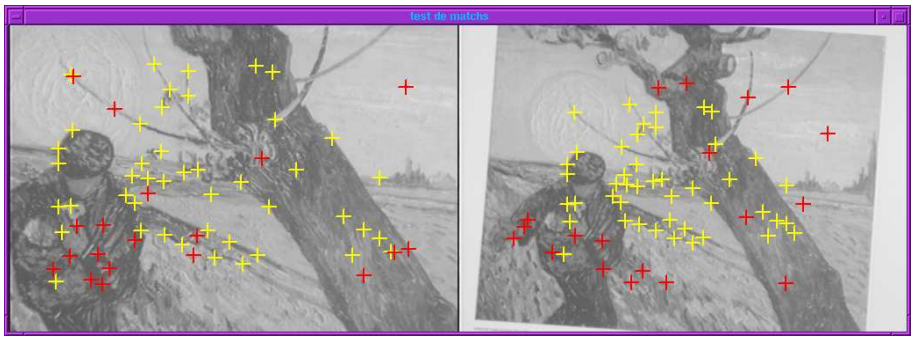

FIG. 1 – *Deux images et leurs points d'intérêt. En clair, les points détectés dans les deux images, en foncé, les autres.*

et bleu. On mélange alors ces dérivées pour obtenir les propriétés d'invariance souhaitées. Cela correspond à éliminer un paramètre pour la rotation et 6 pour les changements d'illumination.On obtient donc $3 \times 10 - 7 = 23$ invariants. Pour des raisons numériques [Mor93], on utilise des descripteurs de dimension 24. Suivant le nombre de points d'intérêt détectés, chaque image est donc décrite par des descripteurs qui sont des vecteurs de $\mathbb{R}^{24}$ en nombre variant de 50 à 600. Comparer ces descripteurs demande de tenir compte de la différence de dynamique entre les différentes composantes des vecteurs. Pour cela, on calcule leur matrice de variance–covariance, puis en l'utilisant, on peut se ramener à une distance euclidienne.

4.1.3 Nature des données et caractéristiques des requêtes

Nos tests utilisent soit des données réelles soit des données tirées aléatoirement selon une loi uniforme dans l'intervalle [0,1]. Ayant utilisé le générateur `drand48()` et vu le nombre de données générées, on ne peut tout à fait exclure des phénomènes de corrélation entre les différentes dimensions des données [PTVF92, Chap. 7]. Les données réelles consistent en 610 images en couleur issues de la base MOVI[6], et en 1206 images constituant une cinquantaine de secondes consécutives d'un épisode de la série *Chapeau melon et bottes de cuir*.[7] La base MOVI comprend des séquences d'images où un seul des paramètres de prise de vue varie. Cela permet de caractériser les limites de reconnaissance de différents types de descripteurs. Les différentes séquences sont indépendantes entre elles. Les images utilisées, et par suite les descripteurs calculés, présentent donc de fortes corrélations dues aux conditions de prise de vue. Cela n'est pas sans conséquence sur les performances évaluées pour les systèmes d'indexation. Cela rend aussi l'évaluation en terme de reconnaissance plus exigeante. Au total, cela donne 413 412 descripteurs correspondant à 1 816 images. Tous ces descripteurs sont utilisés dans les expériences, sauf celles où l'on fait varier explicitement ce nombre. Ils ont tous été calculés avec 24 dimensions. On a simplement tronqué les descripteurs en ne gardant que les premières composantes pour obtenir des descripteurs de dimension inférieure.

Les descripteurs utilisés comme requêtes sont issus de nouveaux tirages aléatoires ou d'images non présentes dans la base. Lorsqu'une même expérience comprend plusieurs requêtes successives (chaque requête comprenant un nombre croissant de descripteurs par exemple), ces descripteurs sont tous différents, afin ne pas biaiser les résultats en permettant au système de retrouver en mémoire des pages déjà lues pour une requête précédente. Un tel phénomène peut toutefois se produire de manière occasionnel.

Toutes les requêtes recherchent les 10 plus proches voisins de chaque descripteur requête. Les expériences suivantes utilisent des requêtes composées de 150 descripteurs, sauf dans le cas où varie explicitement ce nombre. Au total, 1 500 descripteurs sont retournés par le résultat.

4.2 Expérience 1 : influence de la dimension

La première expérience montre l'influence qu'à la dimension des données sur les performances des trois techniques étudiées. Pour cette expérience, nous avons d'abord calculé les 413 412 descripteurs de 24 di-

6. Voir `http://www.inrialpes.fr/movi/pub/Images/index.html`.
7. Ces données sont issues du corpus AIM fourni par l'INA aux membres du GT-10 du GDR ISIS.

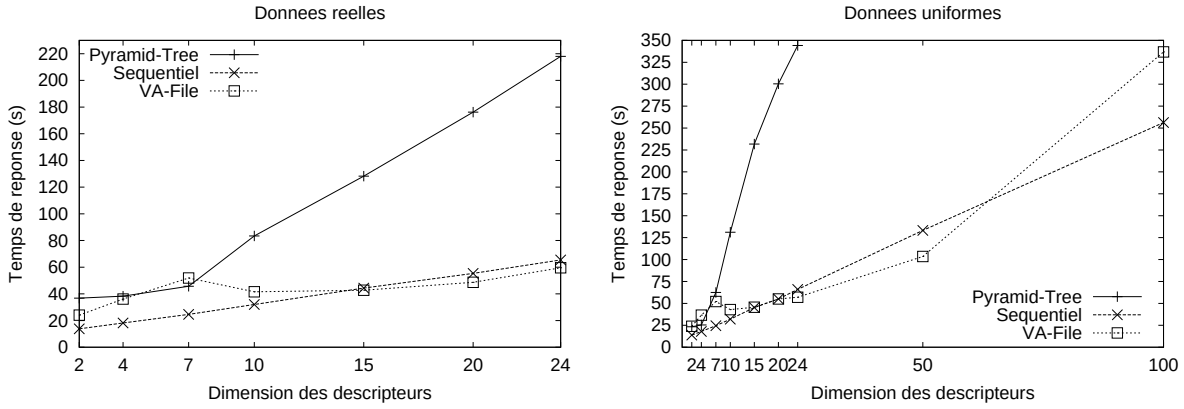

FIG. 2 – *Base de 413 412 descripteurs, 150 descripteurs requête, distributions réelle et uniforme, dimension des descripteurs croissante*

mensions à partir des données réelles. Ces descripteurs ont ensuite été tronqués à 2, 4, 7, 10, 15, 20 et 24 dimensions. Les dimensions 2, 7 et 24 sont exemplaires : 2 représente la dimension des descripteurs utilisés pour caractériser des configurations de segments, 7 est la dimension des descripteurs pour les images en niveaux de gris, et 24 pour les images en couleur. Les autres dimensions sont utilisées comme données intermédiaires. 413 412 descripteurs tirés suivant une distribution uniforme cette fois, ont été fabriqués, pour les mêmes dimensions, mais également pour des dimensions plus importantes (jusqu'à 1000 dimensions). Les tailles des bases résultantes sont données par la table 1.

dimension	taille	dimension	taille	dimension	taille
2	3 307 296	15	24 804 720	100	165 364 800
4	6 614 592	20	33 072 960	250	413 412 000
7	11 575 536	24	39 687 552	500	826 824 000
10	16 536 480	50	82 682 400	1000	1 653 648 000

TAB. 1 – *Tailles en octets des bases en fonction de la dimension retenue*

Une fois les bases créées, nous avons calculé 150 descripteurs issus de données réelles. Ceux-ci ont été ensuite tronqués aux bonnes dimensions pour former les requêtes d'interrogation de la base réelle. Nous avons procédé de même pour fabriquer les requêtes uniformes composées elles aussi de 150 descripteurs. Le temps de réponse indiqué pas la figure 2 correspond au temps nécessaire pour effectuer la recherche complète, c'est-à-dire pour effectuer 150 recherches consécutives sur la base.

Les performances correspondantes à l'utilisation de données réelles sont données par la partie gauche de la figure 2. Dans ce cas, on constate que les performances du Pyramid-Tree se dégradent notablement à partir de 7 dimensions. Au delà, cette technique ne s'avère plus du tout compétitive. Les performances du VA-File et de la recherche séquentielle sont nettement meilleures, et dégénèrent moins rapidement lorsque la dimension des descripteurs croît. La recherche séquentielle à des performances linéaires en fonction de la dimension. Ce comportement est normal et sans surprise. Rechercher séquentiellement 413 412 descripteurs ayant 2 dimensions demande toutefois près de 14 secondes, et près de 66 secondes pour 24 dimensions (ceci correspond à *seulement* 1 816 images).

Les performances du VA-File sont très proches de celle de la recherche séquentielle, sauf lorsque le nombre de dimension est faible. Dans ce cas, pour 2 dimensions, rechercher à l'aide du VA-File demande plus de 24 secondes, et près de 52 pour 7 dimensions (contre près de 25 secondes pour le séquentiel et 7 dimensions). Lorsque le nombre de dimensions croît, le VA-File découpe son espace en « cases » plus petites, et limite ainsi le nombre de comparaisons entre descripteurs. À l'inverse, peu de dimensions tend à impliquer une visite exhaustive des « cases » pré-calculées, et un coût supérieur au séquentiel causé par les calculs et l'exploitation des approximations.

Les performances correspondantes à l'utilisation de données uniformes sont données par la partie droite de la figure 2. Cette figure ne montre les résultats que pour des dimensions inférieures ou égales à 100. Au dessus, les recherches *via* le VA-File ne se sont pas terminées, cela probablement en raison d'un bug. Par ailleurs, nous n'avons pas effectué de recherches au travers du Pyramid-Tree pour des dimensions supérieures à 24, le temps de réponse devenant trop important.

Dans cette figure, le Pyramid-Tree est de nouveau la technique la moins performante, quelle que soit la dimension. En dessous de 15 dimensions, le séquentiel est meilleur que le VA-File, pour les mêmes raisons que celles évoquées ci-dessus. Avec des données ayant 50 dimensions, le VA-File retourne une réponse au bout d'un peu moins de 104 secondes alors qu'il faut près de 134 secondes au séquentiel pour répondre. Dans ce cas, le VA-File tire pleinement profit des approximations et limite le nombre de comparaisons. Au dessus, le séquentiel devient plus performant. Pour 100 dimensions, la recherche séquentielle demande 256 secondes et celle au travers du VA-File 336 secondes. Ces résultats sont en accord avec ceux présentés dans l'article définissant le VA-File, et en particulier avec la figure 12 montrant que le séquentiel bat le VA-File pour des dimensions supérieures à 45 [WSB98]. L'explication données pas les auteurs est qu'à partir de ce moment, le fichier des approximations ne tient plus en mémoire et entraîne de nombreuses entrées-sorties additionnelles.

Quelles que soient les données gérées, uniformes ou réelles, les temps de réponse pour rechercher 150 descripteurs dans une base somme toute petite sont importants : autour d'une minute pour le VA-File avec des données réelles, et autant pour le séquentiel avec des données uniforme. Ces temps sont déjà supérieurs au temps d'attente admissible pour un système interactif. L'expérience suivante explore plus en détail les conséquences de l'augmentation de la taille de la base sur le temps de réponse.

4.3 Expérience 2 : influence de la taille de la base

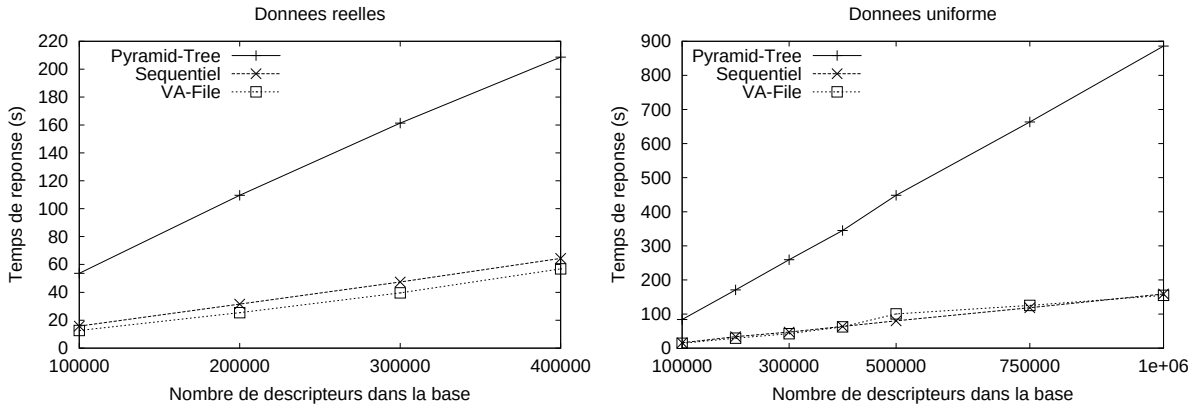

FIG. 3 – *Descripteurs de 24 dimensions, 150 descripteurs requête, distributions réelle et uniforme, nombre de descripteurs dans la base croissant*

La figure 3 montre l'influence de la taille de la base sur les performances des trois techniques étudiées. Pour cette expérience, nous avons repris les 413 412 descripteurs en 24 dimensions calculés précédemment, puis nous avons généré des bases en ne conservant que 100 000, 200 000, 300 000 et 400 000 de ces descripteurs, en gardant les 24 dimensions. Les requêtes sont effectuées avec les mêmes 150 descripteurs de 24 dimensions que ceux utilisés précédemment. Nous ne pouvions pas générer facilement des bases plus importantes de descripteurs de données réelles en raison de la taille limitée de notre fond d'images. Cette limitation ne vaut bien évidemment pas pour des données uniformes. Dans ce cas, nous avons pu générer des bases contenant jusqu'à 1 000 000 de descripteurs (96Mo), ce qui pourrait correspondre à plus de 6 500 images en supposant que 150 descripteurs sont en moyenne générés pour chaque image.

La partie gauche de la figure isole le cas où les données sont réelles. On retrouve dans cette partie les temps constatés dans l'expérience précédente pour 24 dimensions. Les constatations sont du même ordre, à savoir que le Pyramid-Tree à un coût nettement supérieur aux autres techniques, et que le VA-File est légèrement

plus performant que la recherche séquentielle. Il faut tout de même près de 15 secondes pour effectuer une recherche (séquentielle ou *via* le VA-File) dans une base composée seulement de 100 000 descripteurs.

La partie droite de la figure montre les temps de réponse des techniques face à des données artificielles et qui suivent un tirage uniforme. Pour une base composée d'1 000 000 de descripteurs, le temps de recherche pour le séquentiel ou le VA-File est de l'ordre de 160 secondes (près de 3 minutes !). Il est clair que ce résultat disqualifie l'emploie de ces techniques au sein d'un système réel où sont stockées plusieurs millions d'images (et donc bien plus de descripteurs). De plus, les requêtes utilisées pour le moment sont constituées de 150 descripteurs. La présentation de la nature des descripteurs locaux, section 2.2, montre que leur nombre peut être bien plus important dans le cas général. L'expérience suivante explore les conséquences de l'augmentation du nombre de descripteurs dans chaque requête.

4.4 Expérience 3 : influence du nombre de descripteurs dans la requête

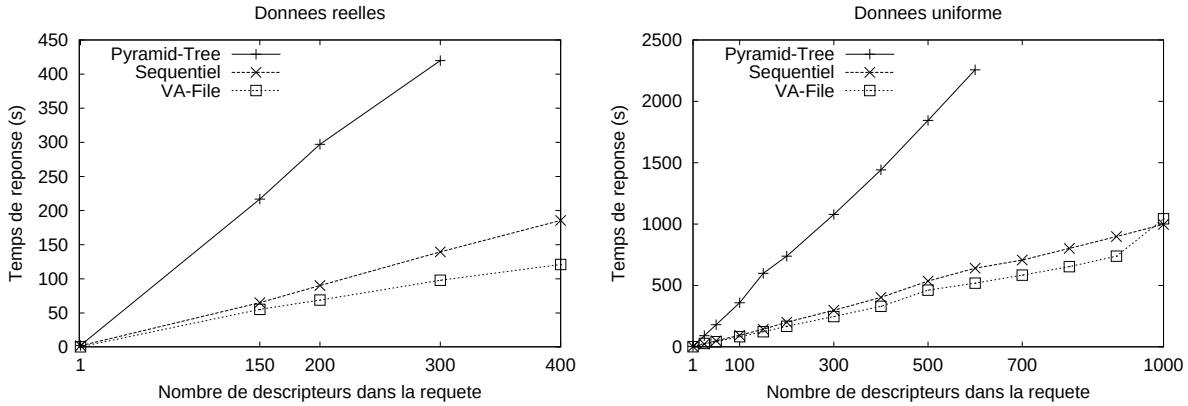

FIG. 4 – *Base de 413 412 descripteurs de 24 dimensions, distributions réelle et uniforme, nombre de descripteurs dans la requête croissant*

La figure 4 montre l'influence du nombre de descripteurs dans la requête sur les temps de réponse. Tous les descripteurs utilisés ici ont 24 dimensions. Nous avons, pour former les requêtes de cette expérience, recherché des images dans notre base pour lesquelles 150, 200, 300 et 400 descripteurs étaient calculés. Nous avons également forgé une requête artificielle n'ayant qu'un descripteur afin de se mettre dans le cadre traditionnel de l'indexation d'images. La fabrication de requêtes uniforme ne requiert pas ce type de processus, et nous avons pu facilement construire des requêtes contenant jusqu'à 1000 descripteurs.

Les résultats de cette expérience montrent de nouveau que seuls le VA-File et la méthode séquentielle reste compétitifs. Malgré cela, les temps de réponse augmentent fortement à mesure que le nombre de descripteurs dans une requête croît. Par exemple, 400 descripteurs réels entraînent une recherche durant 121 secondes pour le VA-File et 185 pour le séquentiel. Pour ce qui est des données uniforme, 1000 descripteurs portent les temps de réponse à près de 997 secondes et 1042 respectivement pour le séquentiel et le VA-File.

Le nombre de descripteurs formant une requête est la conséquence du nombre de points d'intérêts détectés dans une image. Ce nombre peut à l'évidence être très grand selon la stratégie de détection employée et les caractéristiques des images. Il est fondamental que le coût d'une requête comportant de nombreux descripteurs ne s'accroisse pas comme les courbes de cette expérience l'indique. Il est ainsi important d'arriver à mettre au point des processus de recherche qui soient le plus indépendants possible du nombre de descripteurs requête.

5 Conclusion et perspectives

Cette étude a montré que trois facteurs rendent inefficaces les techniques actuelles d'indexation bases de données appliquées à des bases d'images caractérisées par des descripteurs locaux. Le premier facteur est lié

à la dimension des descripteurs, le deuxième à la taille de la base et le troisième au nombre de descripteurs composant chaque requête. Nos évaluations ont montré que l'accroissement de l'un de ces facteurs allonge fortement le temps de réponse. Notons, toutefois, que les paramètres que nous avons utilisé et fait varier ont des valeurs moyennes et non pas extrêmes, suggérant une dégradation de performance encore plus importante pour de très grandes bases (plusieurs Go) stockant des descripteurs ayant de très nombreuses dimensions (plusieurs centaines) et interrogées par des requêtes très longues (millier de descripteurs).

Il est ainsi fondamental d'essayer de dégager des pistes de recherches pour que les techniques d'indexation multimédia employant des descripteurs locaux puissent être utilisées en pratique à l'avenir. Les pistes que nous proposons ici visent à contourner les problèmes posés par chacun des facteur évoqué ci-dessus.

Éviter l'exploration des cases voisines. Découper un espace multidimensionnel pose le problème de devoir systématiquement explorer de nombreuses cases voisines lorsque le point recherché est proche d'une frontière. On peut envisager un système dans lequel il n'y ait jamais besoin de devoir analyser le contenu des cases voisines d'une case donnée. Pour cela, on peut imaginer la création de deux index identiques, mais découpés différemment, c'est-à-dire n'ayant pas leurs frontières placés aux mêmes endroits. On peut alors déterminer, par calcul, l'index à utiliser pour que le descripteur recherché soit le plus éloigné possible d'une frontière. Il est ainsi facile de ventiler à priori tous les descripteurs requête vers l'index adéquat et bénéficier de parallélisme gros grain dans la recherche si ces deux index sont placés sur deux machines différentes. Cette stratégie bénéficie évidemment de celle présentée ci-dessous visant à effectuer une recherche approximative (on se satisfait de ce qui se trouve dans la case, sans aller voir ailleurs).

Utiliser la redondance constituée par les nombreux descripteurs d'une requête. L'utilisation de descripteurs locaux fait que la reconnaissance d'un objet ne se base pas sur une correspondance stricte entre un descripteur de la requête et un de la base. La reconnaissance procède plutôt par accumulation d'évidences, c'est-à-dire que certaines images de la base appartiendront au résultat parce que plusieurs descripteurs de la requête ont pu être mis en correspondance avec plusieurs descripteurs de ces images. Il y a ainsi une certaine redondance dans les informations utilisées pour bâtir le résultat. Cette redondance peut notamment être exploitée de deux manières.

Premièrement, la recherche peut se limiter exclusivement à une case au lieu d'explorer celles environnantes. Cette recherche est approximative, mais l'existence de nombreux descripteurs dans la requête permet de profiter de la redondance des informations pour consolider le résultat final. Ce type de recherche est avantageux car une fois la case déterminée, on se contente d'accéder séquentiellement aux descripteurs qu'elle englobe pour calculer les distances. Chaque recherche va au moins aussi vite que la recherche séquentielle, mais le résultat risque d'être dégradé, car le point cherché peut se trouver dans une autre case.

Deuxièmement, on peut essayer d'arrêter la recherche avant d'avoir utilisé tous les descripteurs de la requête si la probabilité que le résultat courant soit le résultat définitif est forte. L'utilisation de modèles statistiques et de seuils de décision est possible.

Exploiter la distribution des valeurs dans la base pour accélérer les requêtes. Tous les descripteurs ne sont pas porteurs de la même information. Certains sont rares, d'autres présents dans une grande majorité d'images. Mettre en correspondance un descripteur requête avec une descripteur de la base n'apporte donc pas toujours autant d'information sur l'image associée. On peut mathématiser cela à l'aide d'un formalisme bayésien, qui va donner pour chaque appariement de descripteurs la probabilité de reconnaissance de telle ou telle image. On peut utiliser cela pour accélérer la recherche : on peut ordonner les descripteurs de la requête en fonction de l'information qu'ils sont susceptibles d'apporter, et commencer par utiliser les plus informatifs. On peut aussi arrêter la recherche quand la probabilité associée à une image candidate est suffisamment importante, ou suffisamment supérieure aux probabilités associées aux autres images candidates. Une telle technique offre l'avantage de travailler par raffinement, ce qui permet de choisir le degré de finesse en fonction du temps maximal que l'utilisateur accepte attendre.

Changer la gestion mémoire pour exploiter la succession d'interrogations intra-requête. Les techniques traditionnelles supposent qu'une seule interrogation suffit pour retourner la réponse. Aussi, le contenu de la mémoire n'est pas conservé après une interrogation et ne bénéficie en rien à l'interrogation

suivante. Or, dans le cas où l'on utilise des descripteurs locaux, on sait que plusieurs interrogations vont être enchaînées pour une même requête. Il devient alors envisageable d'exploiter le contenu de la mémoire chargé par une interrogation pour accélérer l'interrogation suivante. Cela signifie par exemple que le prochain descripteur que l'on va utiliser va être celui ayant des chances de voir ses plus proches voisins en mémoire parce qu'ils ont été amenés par l'interrogation précédente. Les descripteurs de la requête ne sont alors plus utilisés simplement en séquence, mais choisis en fonction de ce qui se trouve en mémoire.

Gérer plusieurs index de petite dimension plutôt qu'un unique index de grande dimension. Explorer des espaces de grande dimension est très coûteux. Il peut être intéressant d'évaluer plutôt les performances de l'interrogation en parallèle de multiples index ayant chacun un petit nombre de dimension. Ces index doivent être bâtis, et les traitements additionnels doivent être tels que les réponses retournées soient identiques à celles que retournerait l'interrogation d'un index unique.

Un petit index contiendrait alors les mêmes descripteurs que ceux que stockerait un index unique, mais ces descripteurs seraient tronqués et ne conserveraient que certaines dimensions choisies avec soin. Il est possible qu'une même composante doive alors se trouver dans plusieurs petits index. L'interrogation devrait alors être transformée en un certain nombre de sous-requêtes, chacune interrogeant un petit index, éventuellement en parallèle sur des ordinateurs différents (parallélisme gros grain ici). Une étape de synthèse des résultats partiels retournés est nécessaire : il faut identifier les descripteurs qui seraient retournés si l'on avait interrogé un index unique. Cette piste limite l'impact de l'accroissement des dimensions des descripteurs en préférant de multiples interrogations de petits index pour lesquels on sait développer des schémas d'indexation performants. Il y a en contrepartie des traitements additionnels à faire et une multiplication de l'espace nécessaire au stockage des descripteurs (mais pas des images). L'accroissement de l'espace disque utilisé est à mettre en rapport avec le parallélisme potentiel et l'accélération de chaque sous-interrogation.

Interroger des bases d'images dont les index contiennent des descripteurs locaux est à l'évidence complexe. La mise au point de stratégies performantes est indispensable si l'on veut bâtir des systèmes fondés sur cette technologie. Les pistes que nous proposons ne sont qu'un premier pas sur une voie prometteuse à explorer.

Références

[AGGR98] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, June 2–4, Seattle, Washington, USA*, pages 94–105, 1998.

[BBK98] Stefan Berchtold, Christian Böhm, and Hans-Peter Kriegel. The Pyramid-Tree: Breaking the curse of dimensionality. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, June 2–4, Seattle, Washington, USA*, pages 142–153, 1998.

[BKK96] Stefan Berchtold, Daniel A. Keim, and Hans-Peter Kriegel. The X-tree : An index structure for high-dimensional data. In *Proceedings of the 22nd International Conference on Very Large Data Bases, September 3–6, Mumbai (Bombay), India*, pages 28–39, 1996.

[BL93] T.O. Binford and T.S. Levitt. Quasi-invariants: Theory and exploitation. In *Proceedings of* DARPA *Image Understanding Workshop*, pages 819–829, 1993.

[DRD97] M. Das, E.M. Riseman, and B.A. Draper. Focus : Searching for multi-colored objects in a diverse image database. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Puerto Rico, USA*, pages 756–761, 1997.

[FBF+94] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, 3:231–262, 1994.

[FCF96] G. Finlayson, S. Chatterjee, and B. Funt. Color angular indexing. In *Proceedings of the 4th European Conference on Computer Vision, Cambridge, England*, pages 16–27, 1996.

[FDF94] G.D. Finlayson, M.S. Drew, and B. Funt. Color constancy: Generalized diagonal transforms suffice. *Journal of the Optical Society of America A*, 11(11):3011–3019, November 1994.

[FtHRKV94] L.M.T. Florack, B. ter Haar Romeny, J.J Koenderink, and M.A. Viergever. General intensity transformation and differential invariants. *Journal of Mathematical Imaging and Vision*, 4(2):171–187, 1994.

[Gut84] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Boston, Massachusetts, June 18–21*, pages 47–57, 1984.

[Hen98] Andreas Henrich. The LSD^h-tree: An access structure for feature vectors. In *Proceedings of the Fourteenth ICDE International Conference on Data Engineering, February 23–27, Orlando, Florida, USA*, pages 362–369, 1998.

[HK99] Alexander Hinneburg and Daniel A. Keim. Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering. In *Proceedings of the 25th International Conference on Very Large Data Bases, September 7–10, Edinburgh, Scotland, UK*, pages 506–517, 1999.

[HKM+97] J. Huang, S. Ravi Kumar, M. Mitra, W.J. Zhu, and R. Zabih. Image indexing using color correlograms. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Puerto Rico, USA*, pages 762–768, June 1997.

[HS88] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, pages 147–151, 1988.

[KS97] Norio Katayama and Shin'ichi Satoh. The SR-tree: An index structure for high-dimensional nearest neighbor queries. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, May 13–15, Tucson, Arizona, USA*, pages 369–380, 1997.

[LG96] B. Lamiroy and P. Gros. Rapid object indexing and recognition using enhanced geometric hashing. In *Proceedings of the 4th European Conference on Computer Vision, Cambridge, England*, volume 1, pages 59–70, April 1996. Postscript version available by `ftp`[8].

[LG97] B. Lamiroy and P. Gros. Object indexing is a complex matter. In *Proceedings of the 10th Scandinavian Conference on Image Analysis, Lappeenranta, Finland*, volume I, pages 277–283, June 1997. Postscript version available by `ftp`[9].

[LJF94] King-Ip Lin, H. V. Jagadish, and Christos Faloutsos. The TV-tree: An index structure for high-dimensional data. *VLDB Journal*, 3(4):517–542, 1994.

[MCL97] M. De Marsicoi, L. Cinque, and S. Levialdi. Indexing pictorial documents by their content: a survey current techniques. *Image and Vision Computing*, pages 119–141, 1997.

[MGS98] R. Mohr, P. Gros, and C. Schmid. Efficient matching with invariant local descriptors. In *Proceedings of the Joint IAPR International Workshops SSPR98 and SPR98: Advances in Pattern Recognition, Sydney, Australia*, volume 1451 of *Lecture Notes in Computer Science*, pages 54–71. Springer-Verlag, August 1998.

[Mor93] L. Morin. *Quelques contributions des invariants projectifs à la vision par ordinateur.* Thèse de doctorat, Institut National Polytechnique de Grenoble, January 1993.

[MPS97] R. Mohr, S. Picard, and C. Schmid. Bayesian decision versus voting for image retrieval. In *Proceedings of the 7th International Conference on Computer Analysis of Images and Patterns, Kiel, Germany*, pages 376–383, 1997.

[MZ92] J.L. Mundy and A. Zisserman, editors. *Geometric Invariance in Computer Vision.* The MIT Press, Cambridge, MA, USA, 1992.

[MZF93] J.L. Mundy, A. Zisserman, and D. Forsyth. Application invariance in computer vision. *Lecture Notes in Computer Science*, 825, 1993.

[NHS84] Jürg Nievergelt, Hans Hinterberger, and Kenneth C. Sevcik. The grid file: An adaptable, symmetric multikey file structure. *ACM Transactions On Database Systems*, 9(1):38–71, 1984.

[NN97] S.A. Nene and S.K. Nayar. A simple algorithm for nearest neighbor search in high dimensions. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 19(9):989–1003, 1997.

8. ftp://ftp.imag.fr/pub/MOVI/publications/Lamiroy_eccv96.ps.gz
9. ftp://ftp.imag.fr/pub/MOVI/publications/Lamiroy_scia97.ps.gz

[PTVF92] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C - The Art of Scientific Computing*. Cambridge University Press, 2nd edition, 1992.

[Rob81] John T. Robinson. The K-D-B-Tree: A search structure for large multidimensional dynamic indexes. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Ann Arbor, Michigan, April 29–May 1*, pages 10–18, 1981.

[Rot95] C.A. Rothwell. *Object Recognition Through Invariant Indexing*. Oxford Science Publication, 1995.

[SH96] D. Slater and G. Healey. The illumination-invariant recognition of 3D objects using color invariants. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 18(2):206–210, 1996.

[SM97] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 19(5):530–534, May 1997.

[SMB98] C. Schmid, R. Mohr, and Ch. Bauckhage. Comparing and evaluating interest points. In *Proceedings of the 6th International Conference on Computer Vision, Bombay, India*, pages 230–235. IEEE Computer Society Press, January 1998.

[SS94] M. Stricker and M. Swain. The capacity of color histogram indexing. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Seattle, Washington, USA*, 1994.

[vGMU96] L. van Gool, T. Moons, and D. Ungureanu. Affine / photometric invariants for planar intensity patterns. In *Proceedings of the 4th European Conference on Computer Vision, Cambridge, England*, pages 642–651, 1996.

[WJ96] David A. White and Ramesh Jain. Similarity indexing with the SS-tree. In Stanley Y. W. Su, editor, *Proceedings of the Twelfth ICDE International Conference on Data Engineering, February 26–March 1, New Orleans, Louisiana*, pages 516–523, 1996.

[WSB98] Roger Weber, Hans-Jörg Schek, and Stephen Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of the 24th VLDB International Conference on Very Large Data Bases, August 24–27, New York City, New York, USA*, pages 194–205, 1998.

[WW96] M. Werman and D. Weinshall. Complexity of indexing: Efficient and learnable large database indexing. In *Proceedings of the 4th European Conference on Computer Vision, Cambridge, England*, volume 1, pages 660–670, 1996.

Table des matières

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399