UMONS
Université de Mons

Faculté Polytechnique

Vers une Navigation Accélérée dans les Grandes Bases de Données Multimédia

État d'avancement de thèse

Membres du comité d'accompagnement

Pr. O. Belhamiti	Université de Mostaganem, Président
Pr. P. Manneback	Université de Mons
Pr. X. Siebert	Université de Mons
Pr. M. Benjelloun	Université de Mons
Pr. C. De Vleeschouwer	Université Louvain-la-Neuve
Pr. K. Sehaba	Université de Mostaganem
Dr. B. Meroufel	Université de Mostaganem
Pr. S. Mahmoudi	Université de Mons, Promoteur
Pr. G. Belalem	Université Oran, Co-Promoteur

Mohammed Amin BELARBI

POLYTECH MONS

10 Mai 2022

UNIVERSITE

Université Abdelhamid Ibn Badis Mostaganem

Faculté des Sciences Exactes et Informatique

Département d'informatique

Thèse de Doctorat

Présentée par :

Mohammed Amin Belarbi

Vers une navigation accélérée dans les grandes bases de données multimédia

---

Spécialité : Informatique

Option : Apprentissage Automatique et Web Intelligence

Soutenue le 10/05/2022 devant un jury composé de :

Président	**Pr. O. Belhamiti**	(Université de Mostaganem)
Examinateurs	**Pr. P. Manneback**	(Université de Mons)
	Pr. M. Benjelloun	(Université de Mons)
	Pr. X. Siebert	(Université de Mons)
	Pr. C. Devleeshouwer	(Université de Louvain-la-Neuve)
	Pr. K. Sehaba	(Université de Mostaganem)
	Dr. B. Meroufel	(Université de Mostaganem)
Directeur de thèse	**Pr. S. Mahmoudi**	(Université de Mons)
Co-directeur de thèse	**Pr. G. Belalem**	(Université d'Oran)

Résumé

Le processus de recherche dans des bases de données multimédia représente un outil primordial dans différents domaines liés à la vision par ordinateur, tels que la reconnaissance de formes, l'imagerie médicale, la vidéosurveillance, l'analyse des mouvements, etc. Généralement, les caractéristiques visuelles de l'image, telles que la couleur, la texture et la forme sont utilisées pour identifier le contenu des images. Cependant, lors de l'utilisation de bases de données volumineuses *Big Data*, ce processus devient très consommateur en ce qui concerne le temps de recherche et l'espace de stockage. La précision risque d'être dégradée si les algorithmes utilisés ne sont pas capables d'extraire et de sélectionner la bonne information à partir des bases de données volumineuses. Pour faire face à ces contraintes, nous proposons dans ce travail une nouvelle méthode de recherche et d'indexation de gros volumes d'images et de vidéos (Big Data) offrant à la fois une précision élevée, un temps d'exécution rapide, une réduction de la dimensionnalité ainsi qu'une exploitation efficace des ressources distribuées avec les technologies du Big Data.
Nous avons proposé, dans nos travaux de thèse, deux principakes contributions, la première porte sur l'accélération de la recherche de données volumineuses avec une réduction de l'espace de stockage sans perte de précision. Cette contribution se résume en quatre points : 1. indexation des vidéos par des images clés ; 2. réduction de dimensionnalité des descripteurs de grandes bases d'images (Big Data) ; 3. réduction du temps de recherche à travers des méthodes d'indexation par approximation (VA-File) et des algorithmes de stockage de descripteurs (arbres binaires) ; 4. amélioration de la précision de la recherche en utilisant les descripteurs extraits à partir des méthodes d'apprentissage profond (Deep Learnig) avec une réduction de dimensionnalité de ces descripteurs.

La deuxième contribution consiste à exploiter de manière efficace les technologies du Big Data pour offrir une implémentation parallèle et distribuée des algorithmes décrits ci-dessus (première contribution). Les technologies de Hadoop et Map Reduce sont adaptées pour exploiter les processeurs graphiques (GPUs) de manière simultanée avec les processeurs centraux (CPUs).
Les expériences liés à l'indexation d'images ont été menées avec six bases de données de tailles différentes (petites, moyennes et volumineuses). Par ailleurs, les expériences portant sur l'indexation de vidéos ont été menés avec deux bases de données de tailles différentes (petite, grande). Ces résultats ont montré une amélioration significative du temps de recherche avec un taux d'accélération allant de 30% à 40%. L'espace de stockage de données a également été réduit avec un facteur allant de 15% à 20% tout en gardant une précision élevée. D'autre part, l'utilisation des technologies du Big Data avec des architectures Multi-CPU/Multi-GPU a permis d'accélérer le processus de recherche dans des bases de données volumineuses avec un facteur allant de 30 à 40. Grâce à ces accélérations, la consommation d'énergie a été réduite avec un facteur d'environ 20%.

Mots-clés : Indexation d'images, indexation de vidéos, extraction de caractéristiques, réduction de dimensionnalité, Big Data, Deep Learning

.

Abstract

The process of searching multimedia databases is an essential tool in various fields related to computer vision, such as pattern recognition, medical imaging, video surveillance, movement analysis, etc. Typically, visual characteristics of the image, such as colour, texture and shape are used to identify the content of images. However, when using large *Big Data* databases, this process becomes very time consuming in terms of search time and storage space. Accuracy can be degraded if the algorithms used are not capable of extracting and selecting the right information from large databases. To cope with these constraints, we propose in this work a new method for searching and indexing large volumes of images and videos (Big Data) that offers high accuracy, fast execution time, reduced dimensionality and efficient exploitation of distributed resources with Big Data technologies.

Our work is described by two main contributions, the first is about accelerating the search for large data with a reduction in storage space without loss of accuracy. This contribution can be summarized in four points : 1. indexing of videos by key frame ; 2. reduction of dimensionality of descriptors of large image databases (Big Data) ; 3. reduction of search time through approximate indexing methods (VA-File) and descriptor storage algorithms (binary trees) ; 4. improvement of search accuracy by using descriptors extracted from Deep Learnig methods with a reduction of dimensionality of these descriptors.

The second contribution is to effectively exploit Big Data technologies to provide parallel and distributed implementation of the algorithms described above (first contribution). Hadoop and Map Reduce technologies are adapted to exploit graphics processing units (GPUs) simultaneously with central processing units (CPUs).
Experiments related to image indexing were conducted with six databases of different sizes (small, medium and large). In addition, the experiments relating to video indexing were conducted with two databases of different sizes (small, large). These results showed a significant improvement in search time with an acceleration rate ranging from 30% to 40%. Data storage space was also reduced by a factor of 15-20% while maintaining high accuracy. On the other hand, the use of Big Data technologies with Multi-CPU/Multi-GPU architectures has enabled the search process in large databases to be accelerated by a factor of 30 to 40. Thanks to these accelerations, energy consumption has been reduced by a factor of about 20%.

Keywords : Image indexing, video indexing, feature extraction, dimensionality reduction, Big Data, Deep Learning

ملخص

تعد عملية البحث في قواعد بيانات الوسائط المتعددة أداة أساسية في مختلف المجالات المتعلقة برؤية الكمبيوتر ، مثل التعرف على الأنماط ، والتصوير الطبي ، والمراقبة بالفيديو ، وتحليل الحركة ، وما إلى ذلك. تستخدم لتحديد محتوى الصور. ومع ذلك ، عند استخدام قواعد بيانات كبيرة الحجم ، تصبح هذه العملية مضيعة للوقت للغاية من حيث وقت البحث ومساحة التخزين. يمكن أن تتدهور الدقة إذا كانت الخوارزميات المستخدمة غير قادرة على استخراج واختيار المعلومات الصحيحة من قواعد البيانات الكبيرة. للتعامل مع هذه القيود ، نقترح في هذا العمل طريقة جديدة للبحث وفهرسة كميات كبيرة من الصور ومقاطع الفيديو (البيانات الكبيرة) التي توفر دقة عالية ووقت تنفيذ سريع وأبعاد منخفضة واستغلال فعال للموارد الموزعة باستخدام تقنيات البيانات الضخمة.

تم وصف عملنا بمساهمتين رئيسيتين ، الأولى تتعلق بتسريع البحث عن البيانات الكبيرة مع تقليل مساحة التخزين دون فقدان الدقة. يمكن تلخيص هذه المساهمة في أربع نقاط: ١. فهرسة مقاطع الفيديو حسب الإطار الرئيسي. ٢. تقليل أبعاد واصفات قواعد بيانات الصور الكبيرة (البيانات الضخمة) ؛ ٣. تقليل وقت البحث من خلال طرق الفهرسة التقريبية (طاذل) وخوارزميات التخزين الوصفية (الأشجار الثنائية) ؛ ٤. تحسين دقة البحث باستخدام واصفات مستخرجة من أساليب ضمب بارنگ مع تقليل أبعاد هذه الواصفات.

تتمثل المساهمة الثانية في الاستغلال الفعال لتقنيات البيانات الضخمة لتوفير تنفيذ متوازي وموزع للخوارزميات الموضحة أعلاه (المساهمة الأولى). تم تكييف تقنيات حدلپ و پپ غدغ لاستغلال وحدات معالجة الرسومات (جعو) في وقت واحد مع وحدات المعالجة المركزية (ثعو).

أجريت التجارب المتعلقة بفهرسة الصور بستة قواعد بيانات مختلفة الأحجام (صغيرة ومتوسطة وكبيرة). بالإضافة إلى ذلك ، أجريت التجارب المتعلقة بفهرسة الفيديو باستخدام قاعدتي بيانات بأحجام مختلفة (صغيرة ، كبيرة). أظهرت هذه النتائج تحسنًا ملحوظًا في وقت البحث بمعدل تسريع يتراوح من ٣٠ إلى ٤٠ في المائة. كما تم تقليل مساحة تخزين البيانات بمعامل ٢٠-١٥ ي المائة مع الحفاظ على الدقة العالية. من ناحية أخرى ، فإن استخدام تقنيات البيانات الضخمة مع بنى متعددة المعالجات / متعددة وحدات معالجة الجرافيكس قد أتاح تسريع عملية البحث في قواعد البيانات الكبيرة بمعامل من ٣٠ إلى ٤٠. وبفضل هذه التسارعات ، تم تقليل استهلاك الطاقة بمقدار عامل يقارب ٢٠

الكلمات المفتاحيّة: تسريع البحث، فهرسة الفيديو ، الاشجار الثنائية ، اختيار المعلومات الصحيحة ، إدارة الطّاقة، أمثلةٌ متعدِدْة الأهداف

Remerciements

Je mesure la chance qui m'a été donnée d'être encadré par des personnes aussi impliquées, ouvertes et compétentes, qui ont cru en mes capacités pour mener à bien ce travail, et sans qui ce travail n'aurait jamais vu le jour. Qu'ils trouvent ici le témoignage de toute ma gratitude :

Saïd Mahmoudi, qui sait faire voler en éclat chaque idée reçue, qui s'est dévoué généreusement à ma thèse et qui m'a donné goût à la recherche. Il m'a fait énormément apprendre tant sur le plan scientifique qu'humain.

Ghalem Belalem, pour son encadrement minutieux et ses bonnes qualités pédagogiques et scientifiques. Ses conseils avisés sont précieux et sa gentillesse m'est inégalé.

Un grand merci à Mohammed Benjelloun, Xavier Siebert de m'avoir fait l'honneur d'être examinateurs de cette thèse, ainsi qu'à Pièrre Manneback pour avoir accepté d'être membre du jury en tant que président. Je suis très honoré de l'intérêt qu'ils ont porté à mes travaux.

Mes remerciements vont également :

À Sidi Ahmed Mahmoudi qui a été toujours disponible pour m'apporter son aide. Ses conseils m'ont été très utiles, sa rigueur au travail est un exemple pour moi.

À tous les membres d'Université de Mons en particulier : Amine Roukh, Mohamed El Adoui, Olivier Débauche, Adriano, Yassine, Rim, Fabrice, Aurélie, Edith et du L'université de Mostaganem : Walid, Jalil, Maghnia, Liela, Houria avec lesquels j'ai partagé des moments inoubliables.

À mes amis et collègues sans exception, pour leur présence, leur respect et leur gentillesse.

Je ne saurais finir sans exprimer mes remerciements aux personnes qui me sont les plus proches : Mes Parents pour avoir pu cimenter ma vie dans un douillet cocon familial, fait d'amour, de bravoure et de zèle au travail ; Mes deux frères Abdelkader et Abdelkhalek, mes soeurs Houria et son mari Laid ainsi que ces trois enfants (Douaa, Aya, Mohammed), Mansouria et son mari Hadj avec ces deux enfants (Boushra et Ilyes), Elalia et son mari Nabil avec ces enfants (Yasser et Iyad) et Fatima et son mari Hadj et ces enfants (Nada et Amanai).

Et enfin, merci à tous ceux qui ont participé de près ou de loin à l'aboutissement de ce travail.

Table des matières

Liste des figures

Liste des tableaux

Première partie

Introduction Générale

Chapitre

1

Introduction Générale

« Et au-dessus de tout homme détenant la science, il y a un savant plus docte que lui. »
— Joseph, v. 76

Sommaire

1.1 Contexte et problématique

Le processus de recherche dans des bases de données multimédia représente une étape importante pour différents domaines liés à la vision par ordinateur, tels que la reconnaissance de formes, l'imagerie médicale, la vidéosurveillance, l'analyse des mouvements, etc. Généralement, ce processus est représenté par les caractéristiques visuelles de l'image, telles que la couleur, la texture et la forme sont utilisées pour identifier le contenu pixellaire des images. Cependant, lors de l'utilisation de bases de données volumineuses (Big Data), ce processus devient très consommateur en termes de temps de recherche et espace de stockage. La précision risque d'être dégradée si les algorithmes utilisés ne sont pas capables d'extraire et sélectionner la bonne information à partir des bases de données volumineuses. Ce problème est communément appelé "Curse of dimensionality" ce qui signifie la malédiction de la dimension. La scalabilité d'un système représente sa capacité à s'étendre et à s'adapter à des modifications du volume de données. Dans le cas des grandes bases de données d'images, cela se traduit donc par la non-dégradation des performances en temps d'exécution et en espace mémoire alloué, lorsque le nombre d'images et de dimensions augmente. On parle de grandes bases de données lorsque le nombre d'images atteint approximativement le million. Ce problème apparaît dans de nombreux domaines d'applications et beaucoup de recherches sont effectuées à ce sujet.

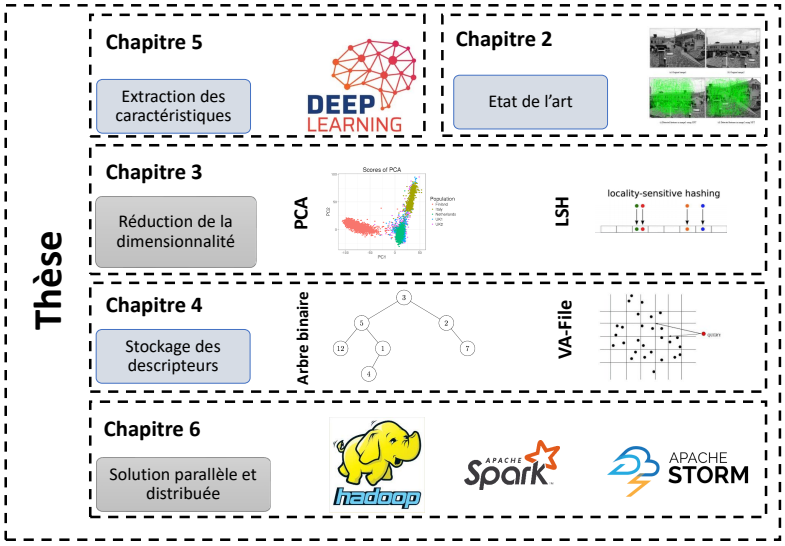
La malédiction de la dimension est le fait que le système n'est capable de traiter une ou plusieurs données avec l'infrastructure matérielle disponible. L'objectif de notre travail est le développement d'un système d'indexation multimédia (images, vidéos) adapté à ce problème. Le système effectuera d'abord une phase d'indexation à partir de l'extraction des descripteurs (couleur, forme, texture, etc.) ainsi qu'une phase de recherche. La phase d'indexation s'appuie sur trois étapes :

- **Segmentation :** pour indexer des vidéos, il faudra d'abord extraire les images clés de chaque vidéo. Cette étape n'est pas nécessaire pour l'indexation d'images ;
- **Extraction des caractéristiques :** cette étape consiste à extraire les caractéristiques à partir de chaque image, en utilisant un ou plusieurs descripteurs (index) ;
- **Compression :** afin de réduire l'espace de stockage et accélérer la recherche, les descripteurs sont compressés de telle manière que la précision ne soit pas dégradée.

Une fois l'indexation terminée, la phase de recherche à base d'une image (ou vidéo) requête peut être appliquée. Le résultat est représenté par un ensemble d'images similaires lors de l'indexation d'images ou un ensemble des vidéos lors de l'indexation de vidéos.

Les différentes contributions de cette thèse sont indiquées sur la Figure 1.1

- Une approche de réduction de la dimensionnalité est présentée dans le Chapitre 3.
- Une approche qui utilise une représentation arborescente permettant un stockage efficace des descripteurs, dans le but d'améliorer les performances de la recherche, est présentée dans le Chapitre 4.
- Une approche qui utilise les caractéristiques extraites des réseaux de neurones profonds pour indexer les images ainsi qu'une approche pour réduire ses caractéristiques est présentée dans le Chapitre 5.
- Une approche basée sur une solution parallèle et distribuée est présentée dans le Chapitre 6.



:

FIGURE 1.1 – Les différentes contributions liées à nos travaux de thèse

1.2 Organisation du document

Le présent document est organisé comme suit :

— **Chapitre 2**

Ce chapitre présente un état de l'art lié à la problématique traitée dans cette thèse, et présente : 1. Les différentes approches d'indexation d'images et de vidéos (classiques où avec Deep Learning) ; 2. Les approches de réduction de la dimensionnalité. 3. Les approches d'approximation et les structures de stockage ; 4. Les solutions utilisées actuellement dans le domaine du Big Data comme Hadoop et Map Reduce.

— **Chapitre 3**

Ce chapitre décrit notre contribution liée à l'indexation d'images qui s'appuie sur la réduction de la dimensionnalité. Dans un premier temps, nous avons testé et analysé différentes approches de réduction de dimensionnalité afin d'identifier celle qui s'adapte le mieux avec nos descripteurs (SIFT et SURF). Nous avons commencé par l'utilisation de la méthode PCA avec un taux de compression calculé en fonction des composantes de sortie. Ensuite, nous avons mis en œuvre d'autres approches de réduction de dimensionnalité telles que Locality Sensitive Hashing (LSH) et VA-File. A l'issue de cette analyse, nous présentons une solution optimale combinant d'une part les méthodes PCA et LSH pour la compression et d'autre part la méthode VA-File pour accélérer la phase de recherche, tout en préservant la précision. A la fin de ce chapitre, nous détaillerons les expérimentations et résultats obtenus lors de la phase de test.

— **Chapitre 4**

Le quatrième chapitre décrit notre contribution liée à l'indexation de vidéos, combinant les deux premières approches (Chapitre 6 et Chapitre 3) et ajoutant une nouvelle méthode de représentation des descripteurs sous forme d'arbres binaires. A la fin de ce chapitre, nous détaillerons notre méthode par un diagramme d'activités avant de présenter les résultats expérimentaux obtenus lors de phase de test.

— **Chapitre 5**

Afin d'avoir des contributions comparables avec les nouveaux travaux liés aux méthodes profondes (Deep Learning), nous avons exploité les différentes architectures neuronales (pré-entrainés ou ré-entrainés) pour l'extraction de caractéristiques. Ensuite, nous proposons deux contributions afin de réduire la dimension de ces descripteurs. Finalement, nous présentons les résultats expérimentaux obtenus lors de phase de test de cette contribution.

— **Chapitre 6**

Ce chapitre est divisé en deux parties : la première partie décrit notre contribution liée à l'indexation de vidéos. Cette approche s'appuie sur les images clés (key frame) et utilise comme descripteurs une combinaison entre l'histogramme de couleur et le descripteur SIFT ou le descripteur SURF en appliquant la contribution liée à la réduction de la dimension sur ces derniers. A la fin de cette première partie, nous détaillerons notre méthode par un diagramme d'activités avant de présenter les résultats expérimentaux obtenus à l'issue de la phase de test. La deuxième partie de ce chapitre décrit notre contribution liée à l'exploitation des technologies du Big Data (Hadoop, Map Reduce et Hadoop Image Processing Interface (HIPI)) pour offrir une solution parallèle et distribuée. Nous proposons une architecture exploitant à la fois les processeurs graphiques (GPUs) et centraux (CPUs) afin d'accélérer la recherche au sein des bases de données volumineuses. A la fin de cette section, nous présentons les résultats expérimentaux obtenus lors de la phase de test.

— **Conclusion**

La dernière partie de ce manuscrit présente une conclusion générale qui synthétise les contributions essentielles de ces travaux de thèse.

Deuxième partie

État de l’art

Chapitre

2

Les méthodes d'indexation et de recherche multimedia

« The beginning of knowledge is the discovery of something we do not understand. »

— Frank Herbert

Sommaire

2.1 Introduction

La majorité des applications multimédias [29] utilisées dans la littérature, que ce soit pour les chaînes télévisées, télésurveillances, recherches sur l'Internet ou autres, se font en général à partir d'une annotation textuelle ou des éléments de texte que nous pouvons rattacher aux données.

Cette annotation se fait manuellement par l'être humain. Par exemple, l'archivage des images et des séquences vidéo qui se fait à l'aide des mots-clefs est une tâche longue et répétitive pour l'humain, surtout avec la quantité volumineuse qui ne cesse d'augmenter avec le temps.

Dans ce chapitre, nous allons dresser un panorama des approches dédiées à la réduction de dimension, ainsi que les approches dédiées au stockage des descripteurs et finalement les approches dédiées au Big Data dans l'indexation multimédia.

2.2 Définitions

Nous allons présenter dans cette section les différents descripteurs utilisés pour l'indexation des images. Nous allons ensuite expliquer les différentes méthodes utilisées pour réduire la dimension de ces descripteurs ainsi que les différentes méthodes de stockage des descripteurs. Finalement, nous allons présenter les méthodes d'indexation dans un environnement parallèle et distribué.

2.2.1 Indexation des images avec les méthodes basées sur les réseaux de neurones profonds (Deep Learning)

Les réseaux de neurones profonds peuvent être utilisés pour l'indexation multimédia (image, vidéo). L'idée est d'utiliser les caractéristiques extraites à travers d'une couche du réseau. Ses caractéristiques seront utilisées pour indexer les images.

2.2.1.1 Les réseaux de neurones

Comme expliqué par Jean-Claude Heudin dans son livre [41], les réseaux de neurones représentent les mathodes de détection les plus récentes et qui atteignent les meilleures performances sur le marché. Pour la détection d'objets, un type d'architecture de réseaux de neurones est particulièrement efficace : les réseaux de neurones convolutionnels ou CNN.

Ce type d'architectures prend en entrée une matrice (image) et applique une suite de filtres sur cette matrice afin d'en extraire les caractéristiques pour ensuite passer à une étape de classification qui permet de déterminer la présence d'un objet particulier comme indiqué ci-dessous.

Le principe de ces réseaux est d'extraire des caractéristiques de plus en plus complexes au fur et à mesure de la progression dans les couches. Cependant, l'augmentation du nombre de couches implique un temps d'entrainement bien plus long. Certaines techniques ont donc été mises au point pour diminuer la taille des matrices en sortie de chaque couche entrainant une réduction des demandes en ressources. La Figure 2.1 montre un exemple d'opération de convolution sur une partie de la matrice.

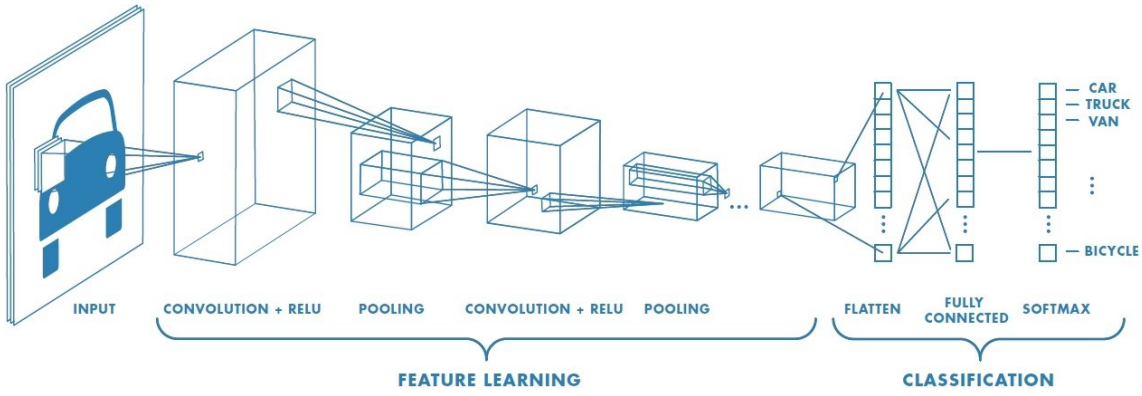


FIGURE 2.1 – Architecture d'un réseau de neurones convolutionnels CNN

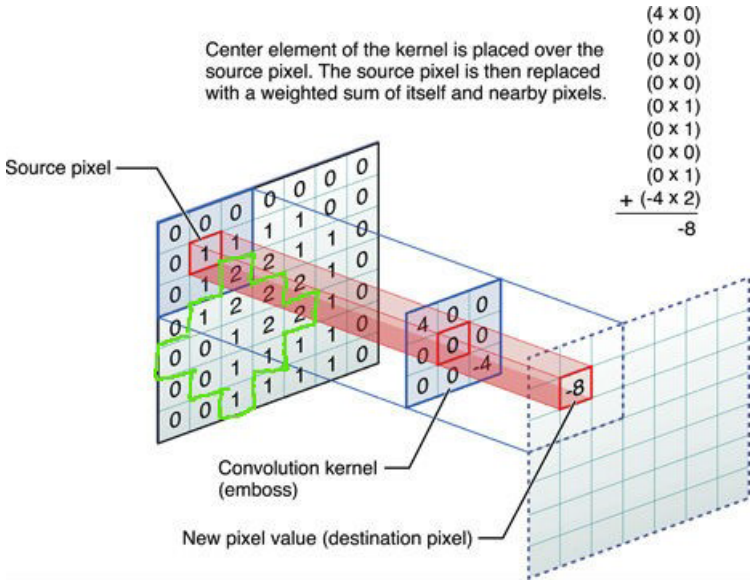


FIGURE 2.2 – Principe de la convolution

La dernière partie de ces réseaux est un algorithme de classification. Il permet de classifier toutes les caractéristiques extraites pour déterminer si un objet appartient à une catégorie ou non.

Les réseaux de neurones convolutionnels ont évolué au cours des années pour arriver en 2015 à dépasser la précision d'un humain pour la classification d'image comme nous montre la Figure 2.3.

Cependant, ces propos sont à nuancer car cette précision consiste à détecter la classe d'une image représentant un seul objet. Or la complexité du monde qui nous entoure est bien plus grande, nous sommes capables par exemple de caractériser des images composées d'une multitude d'objets, arrière-plans et d'actions différentes qui se chevauchent. Plusieurs approches ont été proposées dans la littérature scientifique pour arriver à classifier des objets dans un environnement plus complexes ainsi qu'en définir leur limite.

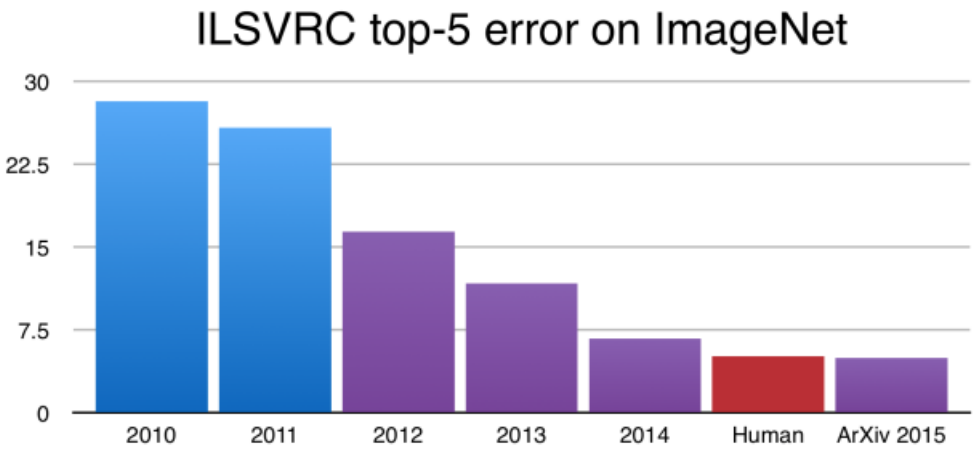


FIGURE 2.3 – Le pourcentage d'erreur au cours des années [81]

2.2.1.2 Entraînement des réseaux de neurones convolutionnels

Pour entraîner les ConvNets, l'approche classique SGD par mini-lots (descente de gradient stochastique par mini-lots) est la méthode la plus utilisée. Sachant qu'il existe d'autres approches dans la littérature (ADAM). Le nombre d'exemples que va contenir chaque mini-lot est arbitraire. En pratique, la taille des mini-lots est de 128 exemples.

L'utilisation de mini-lots permet de profiter de la puissance des cartes graphiques en faisant un grand nombre d'opérations en parallèle. Cela permet aussi de travailler avec une estimation moins bruitée du gradient (grâce à l'utilisation de la moyenne de l'erreur sur le mini-lot).

Une autre solution qui permet de stabiliser le déplacement dans la direction du gradient est l'utilisation de l'inertie. L'inertie intervient dans la formule de la mise à jour des paramètres. Cette mise à jour ne dépend plus uniquement des nouveaux gradients. Elle prend en compte aussi les gradients de l'itération précédente. En utilisant un poids α, l'importance donnée à ces derniers est contrôlée par rapport aux nouveaux. La formulation mathématique est la suivante :

$$\delta\theta^{k+1} = \alpha\delta\theta^{k} + (1+\alpha)\frac{\partial E}{\partial E} \tag{2.1}$$

Avec θ^k les paramètres à l'itération k et E est la fonction d'erreur.

2.2.1.3 Indexation avec Deep Learning

Indexer une image avec les les réseaux de neurones profonds revient a utiliser les de la dernière couche ou l'avant dernière couche par exemple. Ces caractérstiques forment ce qu'on appelle un vecteur des caractérstiques.

Dans le cas où nous avons un modèle avec des paramètres qui ont été initialisés avec des valeurs aléatoires. Par exemple, nous considérons le modèle de la Figure 2.4, une version modifiée de l'architecture AlexNet, cette architecture comprend cinq couches de convolution suivies de deux couches FC et d'une

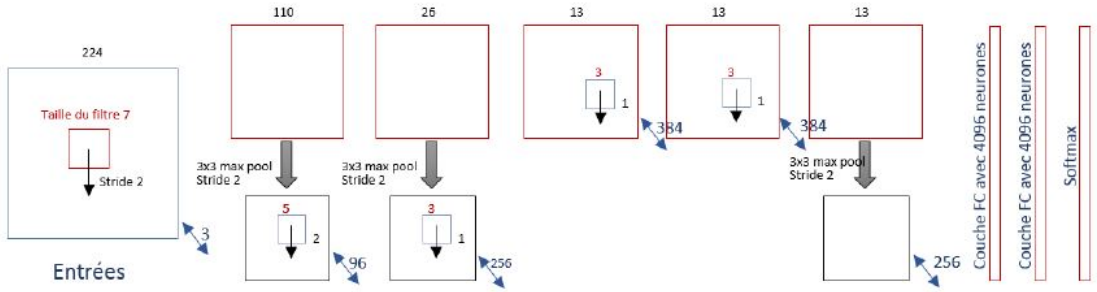


Figure 2.4 – Un exemple de ConvNets

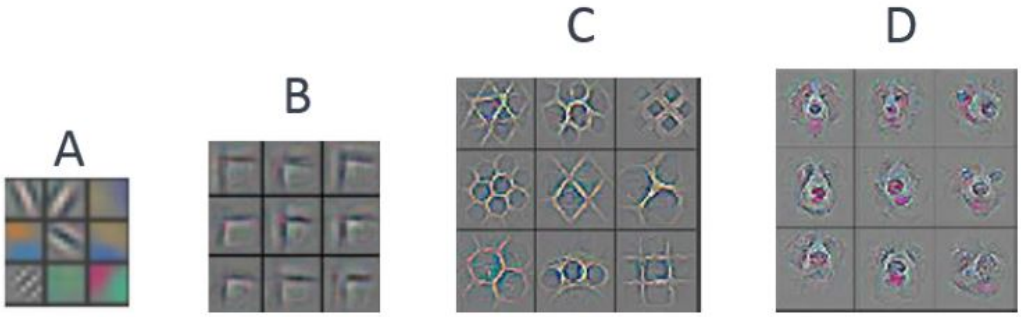


Figure 2.5 – La différence entre des filtres provenant de différentes couches

couche Softmax. Les entrées sont des images RGB avec une résolution de 3×224×224 pixels. Elles passent d'abord par la première couche de convolution qui utilise un noyau qui fait 7×7 avec un « stride » de 2×2. Les filtres de la première couche ne voient qu'une petite partie de l'image. Ils commencent par apprendre des caractéristiques comme montrées dans la Figure 2.5-A. Nous avons par exemple des détecteurs de contours qui ressemblent à des filtres de Gabor.

Après cette première couche, nous retrouvons une couche de « max-pooling » qui va réduire les dimensions des cartes de caractéristiques. Ceci permettra à la couche de convolution qui suit d'avoir un champ réceptif virtuel plus large. Ainsi les filtres pourront apprendre à détecter des combinaisons des caractéristiques détectées par les filtres précédents. Nous avons par exemple des détecteurs de coins dans la Figure 2.5-B.

Ensuite, les couches suivantes vont donner des filtres qui détectent des textures, des parties d'objets, etc. Plus nous allons en profondeur plus les caractéristiques deviennent de haut niveau comme montré dans la Figure 2.5.

Pour indexer une base d'images, nous utilisons les caractéristiques extraites depuis la dernière couche du réseau de neurones profond. Il existe d'autres approches qui utilisent l'avant-dernière couche, comme d'autres utilisent une combinaison des caractéristiques des différentes couches. Dans la suite, nous utilisons des méthodes de comparaison afin de comparer les caractéristiques d'images requêtent avec l'ensemble des caractéristiques. Les vecteurs générés peuvent être de grandes tailles ce qui rend la phase de recherche lente. Il existe cependant des méthodes qui permettent de réduire la taille de ces derniers. Ces méthodes peuvent être appliquées directement au niveau du réseau de neurones comme

ils existent d'autres qui sont appliqués sur le vecteur généré. Nous allons présenter maintenant les méthodes permettant de réduire la dimension des caractéristiques extraites à par les réseaux de neurones profonds.

2.2.1.4 RMAC

Tolias et al dans [94] ont rapporté pour la première fois une méthode d'indexation d'images basée sur les caractéristiques de ConvNet qui donnent des résultats qui surpassent les résultats des méthodes traditionnelles sur des benchmarks de CBIR. Afin d'extraire des caractéristiques, ils ont éliminé les couches de classification (les couches FC) d'un ConvNet pré-entraîné (VGG16). Ils ont utilisé un ConvNet entièrement convolutionnel résultant pour l'extraction de caractéristiques.

Supposons que nous avons une image en entrée I de taille ($W_I \times H_I$), les cartes de caractéristiques en sortie formeront un tenseur 3D sous la forme C × W × H (où « C » est le nombre de canaux, « W » et « H » la largeur et la hauteur des cartes de caractéristiques). Si nous représentons ce tenseur 3D comme un ensemble de cartes de caractéristiques 2D : X = X_c, c=1...C, nous pouvons calculer le descripteur MAC (Maximum Activations of Convolutions) en utilisant la formule suivante :

$$f = [f_1 ... f_c ... f_c], \quad \text{avec} \quad f_c = \max_{x \in X_c} x \tag{2.2}$$

Afin de calculer le descripteur RMAC (Activation Maximale Régionale des Convolutions), Tolias et al ont proposé une approche simple pour l'échantillonnage de R = R_i, un ensemble de régions carrées dans X. L'échantillonnage proposé est effectué à L = 3 échelles différentes. A chaque échelle, un noyau carré est utilisé pour balayer les différentes régions et effectuer sous-échantillonnage : "max-pooling". La largeur du noyau est :

$$k_w = 2 \quad \times \quad min(W, H)/(l+1), l = 1...L \tag{2.3}$$

et le chevauchement permis est égal à 40% de k_w. Une fois les régions sélectionnées, le descripteur RMAC pour une région Ri peut être calculé en utilisant la formule suivante :

$$f_{R_i} = [f_{R_{i,1}} \cdots f_{R_{i,c}} \cdots f_{R_{i,c}}] \quad \text{avec} \quad f_{R_{i,c}} = \Big(\sum_{x \in R_{i,c}} x^{\alpha} \Big)^{\frac{1}{\alpha}} \approx \max_{x \in X_c} x \quad \text{et} \quad \alpha = 10 \tag{2.4}$$

Ensuite, pour chaque région, ils normalisent (normalisation $L2$) le vecteur résultant. Ensuite ils appliquent une PCA, et normalisent à nouveau avant de combiner tous ces vecteurs et finalement les normalisent une dernière fois pour obtenir le descripteur RMAC final. L'approche RMAC présente plusieurs avantages, en commençant par l'utilisation de ConvNets entièrement convolutionnels, ce qui permet de conserver les proportions des entrées sans utiliser de techniques telles que le zéro-padding (ce qui peut nuire à la précision de la recherche). En plus de cela, le descripteur RMAC encode efficacement les informations spatiales tout en gardant la taille du descripteur indépendante de la résolution de

l'entrée. Cette taille ne dépend que du nombre de canaux de la couche sélectionnée pour l'extraction des caractéristiques.

2.2.1.5 Les CNN dans l'indexation multimédia

Parmi les travaux dans le domaine d'indexation et de recherche multimédias basé sur les réseaux de neurones profondes, nous pouvons citer les suivants :

- Les auteurs dans [82] proposent un système de recherche et d'indexation des images par le contenu CBIR qui uilise les caractéristiques extraites à partir de la dernière couche des réseaux de neurones CNN. Ils utilisent l'algorithme Support Vector Machine (SVM) afin de séparer entre les images similaires et non similaires. L'entrée du SVM est une paire de caractéristiques qui sont assemblées par paire d'images : l'image de requête et chaque image test dans l'ensemble de données d'image. Les images de tests sont ensuite classées en fonction de la distance entre les deux éléments et l'hyperplan formé. Les expérimentations ont été menées avec la base d'images Caltech256 [2]. L'application des CNN au CBIR est un bon moyen de résoudre le problème de représentation des caractéristiques fondamentales qui se pose depuis longtemps, sachant que l'utilisation des SVM pendant l'entrainement a permis de choisir les meilleurs paramètres. Finalement les résultats obtenus montrent que l'approche proposée est meilleure comparant les méthodes classiques.
- D'autres auteurs dans [57] proposent un système d'indexation et de recherche des images par le contenu CBIR en utilisant les réseaux de neurones 3D ainsi que les auto-encodeurs 3D pré-entraînés pour la détection précoce de la maladie d'Alzheimer. Les 3D-Capsule Networks (CapsNets) sont généralement des groupes de neurones qui gissent de telle manière que différentes poses paramétriques sont représentées par des vecteurs d'activité de ces neurones, tandis que les longueurs des vecteurs représentent la probabilité respective d'existence d'une caractéristique spatiale. En général, les défauts d'un CNN sont principalement liés à la présence de couches de regroupement. Cependant, les couches dans CapsNet sont largement remplacées par un critère plus approprié appelé "routage par accord". Sur la base de ce critère, les sorties résultantes générées dans la première couche sont transmises aux capsules mères de la couche suivante. Les 3D-CapsNets ont la capacité d'apprendre rapidement même quand il s'agit d'une petite base. Aussi ils sont robustes face aux rotations et transitions des images. Les résultats obtenus montrent que l'utilisation des 3D-CapsNets et les CNN avec les auto-encodeurs 3D augmente les performances en comparaison avec les CNN classiques. Ue précision de 98.42% est obtenue pour la classification de la maladie d'Alzheimer.
- Dans [5], les auteurs proposent une combinaison des méthodes classiques (sac de mots et motif binaire local qui est une méthode utilisée pour reconnaître des textures ou pour la détection des objet dans les images) pour l'extraction des caractéristiques des images avec ceux qui sont extraits de la dernière couche des CNN. Cette approche est appelée WNAHVF, qui est une combinaison des poids normalisés de l'architecture AlexNet avec les caractéristiques visuelles. Les expérimentations ont été testées sur la base d'images Corel-1k et Corel-10k et aussi sur une base d'images médicales. Les résultats obtenus montrent l'efficacité de combiner les deux méthodes. Une précision de 99,667% est obtenue pour le moteur de recherche des images CBIR.
- Les auteurs dans [71] ont présenté un système d'indexation pour les images de bandes dessinées

2. Caltech256 : https ://www.kaggle.com/jessicali9530/caltech256

numériques qui peuvent améliorer les performances de la recherche de ces images dans le Web en extrayant et en indexant automatiquement le contenu textuel et visuel. Ils ont également comparé les approches basées sur l'apprentissage profond et les approches traditionnelles pour avoir une idée complète sur les performances et les limites de ces techniques. Pour évaluer leurs contributions, ils ont utilisé une nouvelle base d'images qui est disponible en ligne sous le nom DCM772 [3]. Cette dernière est conçue spécialement pour la détection du corps et du visage des personnages des bandes dessinées. Les auteurs ont trouvé qu'ils peuvent combiner à la fois les méthodes basées sur les réseaux de neurones profonds ainsi que les approches classiques pour améliorer l'analyse des images des bandes dessinées. Dans le cas de la détection des visages ou de la reconnaissance de texte, Ils utilisent l'approche utilisé dans le système YOLOV2 qui est une architecture de détection et localisation des objets basés sur les réseaux de neurones profonds, parce qu'ils ont trouvé que ça permet d'améliorer les performances par rapport aux méthodes traditionnelles. Par contre, dans le cas de la détection des panneaux, les méthodes basées sur les réseaux de neurones profonds ne sont meilleures que sur un ensemble de données particulières, mais les méthodes traditionnelles se généralisent mieux pour n'importe quel ensemble de données.

- D'autres auteurs dans [84] proposent un système d'indexation et de recherche des images avec un temps de recherche rapide. L'objectif de ce travail était de surmonter les limites des méthodes de recherche basées sur les réseaux conventionnelles dans le domaine de la biologie végétale pour augmenter les performances des moteurs de recherche en termes de précision. L'approche proposée utilise pour l'indexation une représentation sous forme d'une structure hiérarchique car cette dernière permet de gérer les bases d'images d'une manière dynamique. La structure hiérarchique est générée à travers les caractéristiques extraites à partir des CNN. En effet, les images similaires sont regroupées dans un seul nœud en utilisant la méthode k-means. Pour la phase de recherche la méthode Brute force est utilisée d'une manière récursive car cette dernière est trés utilisée pour les bases d'images à grande échelle. Les résultats obtenus montrent que le temps de recherche est 16 fois plus rapide que la recherche séquentielle. Sachant que l'approche proposée ne nécessite aucune connaissance préalable des référentiels des images. Pour comparer les images, les auteurs utilisent les mesures de similarité d'une manière récursive, ce qui réduit considérablement le temps de calcul dans les systèmes CBIR multimédia à grande échelle.
- Dans l'article [1], les auteurs utilisent une combinaison des caractéristiques extraites des couches entièrement connectées dans un modèle AlexNet pré-entraîné. Une seule caractéristique ne garantie pas généralement de donner des résultats précis et des performances optimales, c'est pourquoi, pour un moteur de recherche de haute précision, les auteurs ont combiné les vecteurs de caractéristiques extraits de deux couches entièrement connectées en une seule. Les résultats expérimentaux ont démontré l'efficacité de l'approche proposée en augmentant le nombre d'images pertinentes extraites ainsi que la précision moyenne.
- Les auteurs dans [66] présentent d'abord une approche basée sur les méthodes classiques pour extraire les descripteurs des images, les résultats obtenus sont acceptable, par contre, ils proposent une amélioration basé sur le retour (revelance feedback) des utilisateurs afin d'améliorer la précision du moteur de rechercher. Par la suite, ils proposent une autre approche basée sur les réseaux de neurones, en utilisant les caractéristiques issue de la dernière couche. Les résultats obtenus sont meilleurs en terme de précision comparés aux méthodes classiques, mais cette

3. DCM772 : https ://fgc.univ-lr.fr/task/

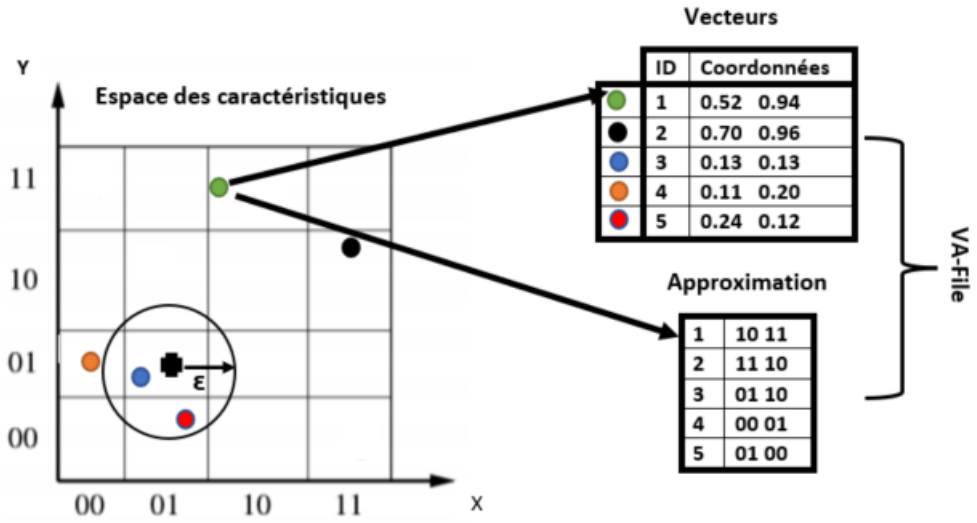


FIGURE 2.6 – Représentation géométrique du codage de VA-File

méthode n'est pas robuste face à la rotation des images ce qui n'est pas le cas des méthodes classiques.

Dans le chapitre Chapitre 5, nous proposons deux approches qui utilisent les CNN ainsi qu'une comparaison avec les travaux éxistant.

2.2.2 Les méthodes d'indexations par approximation

Les méthodes d'indexations par approximation sont généralement utilisées lors du passage à des dimensions élevées (Big Data). Dans ce cas, nous pouvons utiliser des techniques d'indexation multidimensionnelles répondant à ce problème en proposant une méthode d'indexation par approximation de l'espace. Parmi les approches d'indexation par approximation nous pouvons citer :

2.2.2.1 VA-File

VA-File qui présente de bonnes performances quand la dimension augmente, elle se base sur la compression des données. Elle peut être considérée comme une recherche séquentielle améliorée. Son principe est le suivant :

- Elle utilise deux ensembles de données : un fichier qui contient tous les vecteurs de la base et un deuxième fichier qui est de taille petite qui contient des approximations géométriques de ces vecteurs comme nous montre la Figure 2.6.
- Lors de la première requête, le fichier d'approximation est parcouru séquentiellement dans le but de sélectionner les vecteurs qui ont le plus de chance d'appartenir à l'ensemble des résultats. Ensuite, l'accès au fichier de données est effectué sur la base des résultats issus de la première phase.

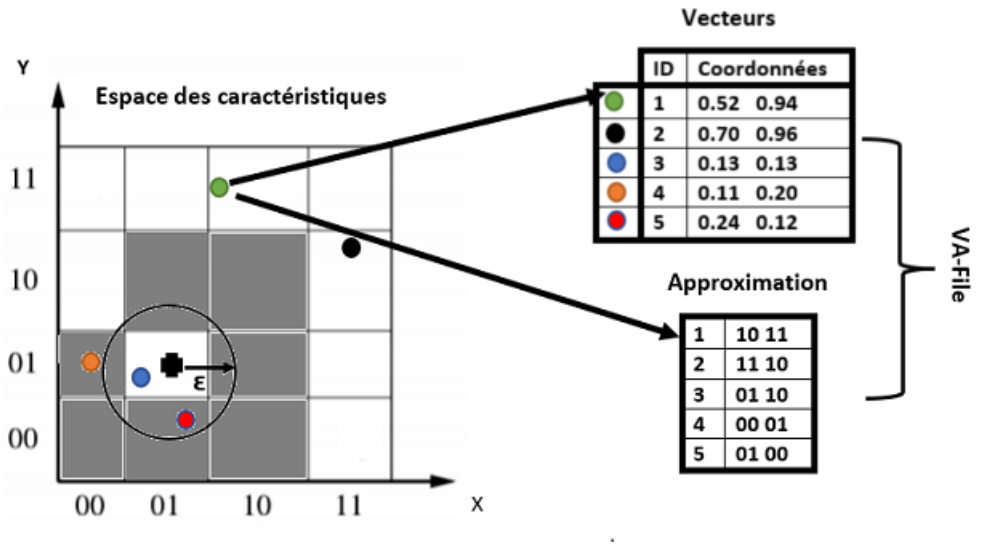


FIGURE 2.7 – La phase de recherche avec VA-File

— Par la suite, la recherche séquentielle est effectuée sur le fichier d'approximation (petite taille), ce qui rend la recherche très rapide parce qu'elle est à base d'un sous ensemble réduit de vecteurs, comme nous montre la Figure 2.7 [97] [17].
Pour effectuer une recherche K-Nearest Neighbors (KNN) à l'aide de la méthode VA-File, il existe deux méthodes : l'algorithme VA-File Simple Scan Algorithm (VA-SSA) ou bien l'algorithme VA-File Near Optimal Algorithm (VA-NOA) [17]. La méthode VA-SSA est résumée dans Algorithme 2.1.

L'avantage de cette méthode est qu'elle utilise une signature des images qui possède une faible taille, ainsi qu'un nombre restreint d'opérations d'E/S. Le coût de calcul Central Processing Unit (CPU) est diminué par rapport à la recherche séquentielle qui analyse l'ensemble de la base.

2.2.2.2 RA-blocks

Le principe de la méthode RA-blocks (RA-blocks)[22] est de diviser l'espace de données en des cellules hyper-rectangulaires qui sont codées sur une chaîne de bits. De même, l'espace des données est divisé sous forme des régions de même capacité et qui sont disjointes et contiguës. Par la suite, chaque région est codée par deux chaines de bits correspondants aux deux cellules en bas à gauche et en haut à droite comme nous montre la Figure 2.8. Cette méthode se base sur une structure Kdb-tree dont le but est d'avoir un arbre binaire multi-dimensionnel où chaque niveau est divisé en deux sous-espaces.
La méthode RA-blocks peut présenter des problèmes dans la partie relative au partitionnement de l'espace où plusieurs régions peuvent être vides et d'autres peuvent avoir des vecteurs moins par rapport à leur capacité. Ainsi, nous remarquons que le temps de calcul augmente à cause d'un nombre importants d'opérations de calculs de distance qui sont effectuées [22].

Algorithme 2.1 Algorithme VA-NOA

Entrée : γ distance ;

Sortie : **vi** vecteur d'approximation, **vd** fichier de distance, **req1** la première requête, **req** une requête, **dist1** fonction de distance entre **req1** et **vi, dist2** fonction de distance entre la **req** et **req1** ;

```
pour tout tous les points de vecteur vi faire
    γ = max(dist1(req1,vi))
    vd += dist1(req1,vi)
fin pour
pour tout i ∈ les points de vecteur vi faire
    si dist2(req1,req) < γ alors
        si dist2(req,req1)<γ alors
            ajouter le point i aux points similaires
            mettre à jour vd
            γ=max(vd)
        fin si
    fin si
fin pour
retourner les points similaires ;
```

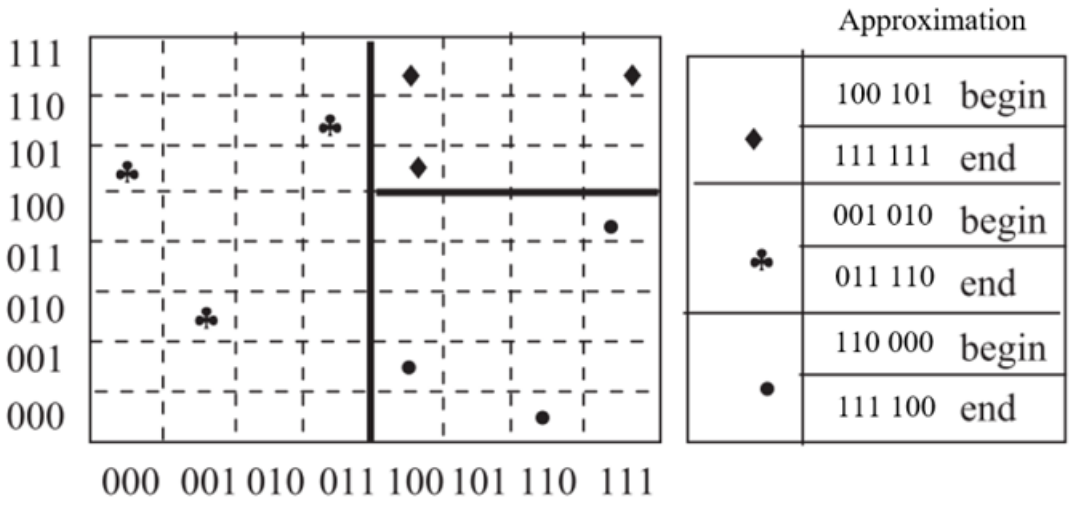


Figure 2.8 – Exemple d'un partitionnement de l'espace 2D pour une capacité des régions [27]

2.2.2.3 RA+-Blocks

La méthode RA+-Blocks (RA+-Blocks) utilise un principe différent pour partitionner l'espace. Seules les régions surchargées sont divisées et sans utiliser une structure en arbre, ce qui permet de diminuer le nombre de régions sélectionnées ce qui explique que le temps de calcul est amélioré. Un des inconvénients de cette méthode est que les données doivent être homogènes [28].

2.2.2.4 IQ-Tree

IQ-Tree [14] est une structure basée sur une approche VA-File en utilisant le principe des deux fichiers. Les données sont subdivisées sous forme de Minimum Bounding Rectangles (MBRs). Le nombre de bits utilisés pour la quantification dépend des MBRs. En effet, les régions ayant un grand nombre de vecteurs sont approximés par un nombre de bits assez élevé, mais les régions peu denses sont approximées par un nombre de bit assez petit. C'est une méthode d'accès très performante puisqu'elle utilise des approximations relatives de la position des vecteurs dans l'espace de données, elle présente une sélectivité meilleure que le méthode VA-File. Cependant, elle possède une structure d'index assez complexe.

2.2.2.5 A-Tree

A-Tree [85] est une méthode d'indexation basée sur l'approximation similaire à R*-tree (R*-tree). Contrairement à IQ-Tree, A-Tree utilise l'approximation des données dans tous les niveaux de l'arbre, elle introduit la notion Virtuel Bounding Rectangles (VBRs) dont l'idée principale est d'approximer chaque MBRs ou vecteur relativement par rapport à son MBRs père. Cette approximation adaptative est beaucoup plus performante et permet d'avoir un arbre avec une grande capacité. Elle a montré de bonnes performances par rapport à VA-File à la fois pour les données uniformes et non uniformes et pour une dimension <64.

Les méthodes d'indexation basées sur l'approximation reposent sur l'idée d'approximer les données (vecteurs) pour réduire leur taille dans le fichier de données et par la suite, réduire le nombre d'entrée/-sortie au fichier de données réelles. Ces méthodes sont très adaptées aux applications réelles puisqu'elles présentent un temps de réponse assez réduit par rapport aux méthodes d'indexation conventionnelles, pour un volume important de données et pour des dimension d>10. Le Tableau 2.1 présente une synthèse des méthodes d'indexation basées sur l'approche approximation citée dans les paragraphes précédents.

Le Tableau 2.1 nous montre une comparaison entre les différentes méthodes par approximations, la méthode VA-File utilise une approximation des vecteurs et comme résultat elle donne un temps de recherche rapide. La méthode IQ-Tree qui est une amélioration de la méthode VA-File où les données sont subdivisées sous forme des rectangles minimums englobant, est utilisée pour les grandes dimensions. La méthode A-Tree est une combinaison entre la méthode VA-File et la structure R*-tree. Elle est utilisée à la fois pour les données uniformes et non uniformes et pour une dimension < 64.

Tableau 2.1 – Les méthodes d'indexation basées sur l'approximation

Méthodes	Structure	Points faibles	Points forts
VA-File	Approximation des vecteurs	Compromis précision d'approximation et taille du fichier d'approximation	Temps de recherche rapide
IQ-Tree	Approximation des vecteurs MBRs	Structure d'index complexe	Adapté aux grandes dimensions, bonne sélectivité des régions
A-Tree	R*-Tree + approximation des vecteurs / MBRs	Taille importante des nœuds	Adapté aux données non uniforme, grande capacité des régions

2.2.2.6 Les méthodes d'indexation par approximation dans les CBIR

Parmi les travaux qui utilisent les méthodes d'approximations nous pouvons citer les suivants :

- Les auteurs dans [58] proposent une méthode d'indexation basée sur la quantification optimisée dans la méthode VA-File. Ils conservent à la fois le pré calcul hors ligne dans une table de recherche et l'approximation conservatrice en ligne du Minimum Distance (MinDist). Donc ils ont obtenu une taille efficace. Les auteurs montrent l'efficacité de l'approche proposée avec différentes bases d'images. L'algorithme donne des résultats sensiblement meilleurs par ordre de grandeur et il s'adapte extrêmement bien en termes de dimensionnalité et en termes de taille de base de données.
- Dans [45] les auteurs proposent le premier framework pour la Co-Reduction of Data (CRD) pour les données et leurs caractéristiques. Avec l'utilisation des méthodes de co-clustering récemment développées qui est une technique de classification consistant à réaliser une partition simultanée des lignes et des colonnes d'une matrice de données. la méthode CRD réduit simultanément la taille et la dimension des données originales en un sous-espace compact, où les bornes inférieures des distances dans l'espace d'origine peuvent être efficacement utilisées pour obtenir une similarité rapide et sans perte. Cette méthode est basée sur la méthode VA-File. Les expérimentations ont été menées sur des bases de données multimédia réelles avec une dimension allant de 432 à 1936. Les résultats obtenus montrent que la méthode CRD est meilleure par rapport aux méthodes existantes dans la littérature.
- Les auteurs dans [90] présentent d'abord une brève description des systèmes CBIR et de ses phases ainsi qu'un moteur de recherche des images basé sur le vecteur KAZE. Il est utilisé pour la phase d'indexation (extraction des caractéristiques). C'est un descripteur qui est calculé à partir des points d'intérêt d'une image comme le vecteur SIFT. Une représentation sous forme d'arbre A-tree est utilisée pour représenter les descripteurs et aussi pour accélérer la phase de recherche. Les images similaires sont calculées à l'aide des différentes mesures de similarité telles que la distance euclidienne, Distance cosinus ainsi que la distance Mahalanobis. L'approche est appliquée sur différentes base d'images comme Corel 10k, GHIM 10k, UK-Bench et MSRC v2. Pour la phase d'indexation, ils utilisent un simple fichier où ils stockent le vecteur KAZE. Bien que le descripteur de KAZE ait ses avantages, sa plus grande limitation est qu'il est très

coûteux en temps de calcul. Ceci est dû au calcul de l'espace d'échelle non linéaire. Il en résulte des inconvénients pour les méthodes de mesures de similitude utilisées.

— Dans cet article [62], les auteurs ont présenté une nouvelle méthode pour traiter le vecteur BOF de haute dimension. Ils ont utilisé la méthode VA-File afin de réduire le vecteur BOF ce qui a permis de traiter des bases de données à grande échelle et ainsi réduire le nombre d'E/S. La combinaison vectrice BOF ainsi que la méthode VA-File ont montré l'efficacité de l'approche proposée. Par la suite les expérimentations ont prouvé que la méthode K-NN basée sur le BOF est meilleur en terme temps de recherche.

Dans le Chapitre 4, nous présenterons une méthode d'approximation basée sur la méthode VA-File. Cette méthode a été combinée avec d'autres méthodes afin d'accélérer la phase de recherche.

2.2.3 Les méthodes de réduction de dimensionnalité

La réduction de la dimension se pose comme une étape primordiale dans le processus de pré-traitement des données (filtrage, nettoyage, élimination des points aberrants, etc.). En effet, pour des données appartenant à un espace de grande dimension, certains attributs n'apportent aucune information ou encore expriment que du bruit, d'autres sont redondants ou corrélés. Ceci rend les algorithmes de décision complexes, inefficaces, moins généralisables et d'interprétation délicate. Les méthodes de réduction de la dimension de l'espace de représentation peuvent être divisées en méthodes d'extraction d'attributs et méthodes de sélection d'attributs. L'extraction d'attributs transforme l'espace d'attributs de départ en un nouvel espace formé de la combinaison linéaire ou non linéaire des attributs initiaux. La sélection d'attributs choisit les attributs les plus pertinents selon un critère donné. Les données sont alors analysées après projection dans un espace de représentation composé des attributs les plus pertinents. Toutefois, l'interprétation des attributs extraits est plus délicate que l'interprétation des attributs sélectionnés. Le point clé de la sélection d'attributs est la définition d'un score mesurant la pertinence de chacun des attributs. Cette sélection s'appuie sur la connaissance explicite et implicite sur les données. Quand nous ne disposons d'aucune information à priori sur le regroupement des données en classes, le contexte d'apprentissage est dit non supervisé. La pertinence d'un attribut est alors mesurée en évaluant ses capacités à préserver la structure des données. Pour de nombreuses applications, nous disposons des informations à priori sur la répartition des données en classes. Ainsi, pour ces données, les labels des classes ont été fournis. Dans ce cas, la sélection supervisée consiste à mesurer la corrélation entre l'attribut et les labels des classes des données.

Nous citons à titre d'exemple les méthodes de réduction de dimension suivantes :

2.2.3.1 Principal Component Analysis (PCA)

PCA, aussi connue sous le nom de transformée de Karhunen–Loève [30], est une méthode très utilisée en statistique. Dans le cas d'indexation des images basée sur une approche PCA, l'objectif est de trouver un nouvel espace qui est une combinaison (linéaire ou non) des dimensions initiales, de dimension significativement inférieure et qui contient une grande part de l'information totale. le détail de la méthode PCA est dans Annexe A

2.2.3.2 t-distributed stochastic neighbor embedding (t-SNE)

L'approche t-distributed stochastic neighbor embedding (t-SNE) [65] est une méthode utilisée pour réduire la dimension ainsi pour visualiser les données avec un nuage de points. C'est une technique qui permet la représentaion d'un ensemble de points de grande diemnsion vers un espace de deux ou trois dimensions. Le critère de la théorie de l'information est utilisé par l'algorithme t-SNE afin de trouver une configuration optimale dont le but est de respecter le voisinage entre les points.
L'interprétation probabiliste des proximités est la base de l'algorithme t-SNE. Sur toutes les paires de points de l'espace d'origine, une distribution de probabilité est définie d'une manière que les points qui sont proches l'un de l'autre ont une forte probabilité d'être séléctionnés. Par contre, les poins qui sont éloignés ont une faible probabilité d'être choisis. Pour l'espace de visualisation, la même distribution est également définie.

2.2.3.3 Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) est née des travaux de Belhumeur et al. en 1997 [11]. Elle effectue une véritable séparation de classe et cela en minimisant les variations entre les images d'un même individu tout en maximisant les variations entre les images d'individus différents. Néanmoins, lorsque le nombre d'individus est inférieur à la résolution de l'image, il est difficile d'appliquer l'approche LDA qui peut faire apparaître des matrices de dispersions singulières. Afin de contourner ce problème, certains algorithmes basés sur la LDA ont été proposés, le plus connu est Regularized LDA (RLDA).

2.2.3.4 Karhunen Loeve Transform (KLT)

Le principe de l'approche Karhunen Loeve Transform (KLT)[30] est de réduire le nombre de dimensions des descripteurs les moins dominants, cela veut dire de concentrer la recherche sur l'information la plus dominante, et donc on élimine le bruit qui est un des avantages de cette méthode. Le principe général de cette méthode est d'abord la décomposition de la matrice covariance des caractéristiques, à travers les vecteurs propres nous trouvons les dimensions dominantes, et par la suite les données projetées vers un nouvel espace au nombre de dimensions diminué.

2.2.3.5 Fisher Linear Discriminant Analysis (FLDA)

Fisher Linear Discriminant Analysis (FLDA) est une méthode supervisée linéaire de réduction de la dimension. La projection de la matrice W_{flda} est obtenue en maximisant la dispersion inter-classe de la matrice S_{inter} comme indiqué dans l'équation Équation (2.5).

$$W_{flda} = arg_v max|\frac{V^T S_{inter} V}{V^T S_{intra} V}| \tag{2.5}$$

Où V^T est la transformation linéaire de la matrice.

$$S_{inter} = \frac{1}{2}\sum_{i,j=1}^{N} Inter_{i,j}(x_i - x_j)(x_i - x_j)^T \tag{2.6}$$

$$S_{intra} = \frac{1}{2}\sum_{i,j=1}^{N} Intra_{i,j}(x_i - x_j)(x_i - x_j)^T \tag{2.7}$$

$$Inter_{i,j} = \begin{cases} \frac{1}{n} i, j \in C \\ 0 \quad quand \quad i \in C, j \notin C \end{cases} \tag{2.8}$$

$$Intra_{i,j} = \begin{cases} (\frac{1}{n} - \frac{1}{n_c}) i, j \in C \\ \frac{1}{N} \quad quand \quad i \in C, j \notin C \end{cases} \tag{2.9}$$

En général, le nombre des caractéristiques est supérieur au nombre des images, pour cela, le résultat de la dispersion inter-classe de la matrice S_{intra} est non carrée, ce qui ne permet pas de résoudre le vecteur propre de la matrice. Dans ce cas PCA est utilisée afin de réduire la dimension des caractéristiques d'entrées ce qui rend la valeur de S_{intra} carrée.

W_{FLDA} est calculé comme montré dans les équations Équation (2.10) Équation (2.11)

$$W_{FLDA} = W_{PCA} W_{flda} \tag{2.10}$$

$$W_{flda} = arg_v max |\frac{V^T W_{PCA}^T S_{inter} W_{PCA} V}{V^T W_{PCA}^T S_{intra} W_{PCA} V}| \tag{2.11}$$

FLDA préserve la discrimination des caractéristiques des images en réduisant la dimension lors de passage à l'échelle.

2.2.3.6 Les méthodes de réduction de la dimensionnalité dans les CBIR

Parmi les travaux dans le domaine de la réduction de dimensionnalité, nous pouvons citer :

- Dans les travaux présentés dans [20], les auteurs ont proposé un modèle robuste et compact utilisé pour améliorer la méthode Vector Of Locally Aggregated Descriptors (VLAD) [52]. Ils redessinent un nouveau cluster en fonction d'une méthode LDA [43] pour réduire la dimensionnalité du descripteur VLAD, en optimisant la dimensionnalité dans son ensemble et rendant la description de l'image plus efficace pour l'indexation.
- Roli en 2002 [67], a remarqué que les deux méthodes LDA et PCA ne sont pas corrélées car la LDA génère un espace propre significativement différent de PCA. Les expériences effectuées dans [67] montrent que la fusion de la LDA et de la PCA ont donné de bons résultats.
- Les auteurs dans [95] proposent une technique biaisant PCA par la prise en compte de la distance entre les classes de couleur de leur signature. Ils ont ainsi montré que le passage d'une quantification de l'espace couleur CIELab de 512 à 25 ou 50 classes augmentent l'efficacité de la recherche, pour un histogramme calculé sur l'image entière ou segmentée en régions.
- Les auteurs dans [33] présentent l'implémentation de l'algorithme PCA dans une carte reconfigurable Field Programmable Gate Array (FPGA), pour réaliser la réduction de dimensionnalité dans les images hyperspectrales. Les résultats expérimentaux obtenus démontrent que la version FPGA de l'algorithme PCA surpasse la version logicielle, ce qui rend le système attrayant pour le traitement de données hyperspectrales embarqué. De plus, l'approche proposée présente des

performances en temps réel en ce qui concerne le temps que prend la carte FPGA pour collecter les données d'image.

— Les auteurs de [15] ont présenté des résultats expérimentaux portant sur la projection aléatoire dans la réduction de la dimensionnalité des ensembles de données du monde réel à haute dimension. Ils ont comparé différentes méthodes de réduction de la dimensionnalité. Les critères utilisés dans ce travail sont la quantité de distorsion causée par la méthode et sa complexité de calcul. Leurs résultats indiquent que la projection aléatoire préserve les similitudes des vecteurs de données, et même les données sont projetées pour modérer les dimensions des nombres. Avec les résultats obtenus la projection est devenue rapide à calculer.

— Dans [24], les auteurs proposent une approche basée sur la combinaison des deux descripteurs SIFT et ORB. Comme les deux descripteurs sont connu par leurs grandes tailles. Afin de corriger le problème de l'espace de stockage, les auteurs appliquent les deux algorithmes K-mean et Locality Preserving Projections (LPP) sur les deux descripteurs, l'application de l'approche k-means permet de réduire la taille des vecteurs à 32, et la méthode LPP à 4 ou 8. La précision de l'approche proposé était à 86.20% and 99.53% en utilisant les deux vecteurs de taille 4 et 8. Les expérimentations ont été testé sur la base d'image WANG. Finalement, les auteurs trouvent que la combinaison des deux vecteurs SIFT et Oriented FAST and Rotated BRIEF (ORB) améliore la précision du moteur de recherche

Nous avons testé les différentes méthodes de réduction de dimension avec le descripteur SIFT et SURF, les détails sont montrés dans le chapitre Chapitre 3.

2.2.4 Les fonctions de hachage

Nous utilisons les fonctions de hachage pour calculer une empreinte courte relative à une donnée et ainsi pouvoir identifier plus rapidement celle-ci. Le résultat de ces fonctions est appelé hash, empreinte ou signature. Il est important qu'une fonction de hachage évite les collisions, c'est-à-dire les états dans lesquels des données différentes ont une même empreinte. En effet, les collisions n'effectuent pas de différenciation entre les données. Nous associons à une fonction de hachage un ensemble de définitions (par exemple un chaîne d'entiers) et un ensemble d'arrivées (par exemple une chaîne de bits de taille fixe). Les empreintes des données sont structurées dans des tables dites de hachage, les empreintes étant les index des cases de la table.

2.2.4.1 La fonction Locality Sensitive Hashing (LSH)

LSH [36] est utilisée pour réduire la dimension des données. Son principe général est particulièrement intéressant dans le cadre des systèmes CBIR. Son principe est le suivant :

— Pré-traitement (Indexation) : l'algorithme utilise l différentes fonctions de hachage pour un point. En d'autres termes, une fonction de hachage f est obtenue par une concaténation de l fonctions de hachage choisies aléatoirement. Par la suite, l'algorithme construit l tables de hachage, où chaque table correspond à une fonction de hachage. Chaque table i contient le résultat de la fonction de hachage f_i. Nous conservons uniquement les tables non vides ce qui permet de réduire la mémoire interne.

— La phase de recherche : lors d'une requête q, l'algorithme utilise les mêmes fonctions de hachage

définit dans la phase de pré-traitement pour hacher la requête. En comparant le résultat par rapport à l tables de hachage, nous trouvons les points hachés à la même position que le point requête q. Le point d'arrêt de l'algorithme est lorsque un point r est trouvé.

2.2.4.2 La fonction LSH multi-probe

LSH multi-probe est une fonction dont l'objectif est d'avoir une meilleure utilisation des tables de hachage, donc réduire le nombre de tables. Son principe est d'appliquer une exploration systématique des tables en parcourant les tables qui sont proches par rapport à la requête, où ces derniers sont les plus probables de contenir des voisins, qui est fait à partir d'un ordre de sondage. Les résultats sont des tables proches qui contiennent les valeurs hachées de la requête avec une certaine tolérance appelée score [64].

Parmi les fonctions de hachage utilisées nous pouvons citer :

— La fonction Min-hash proposée par [19] qui se base sur l'estimation de la similarité entre deux ensembles. Son principe consiste à sélectionner aléatoirement une permutation π de l'espace U. Pour un ensemble de donné A nous définissons la fonction de hachage par :

$$h(x) = min\pi(a)|a \in A \tag{2.12}$$

Cette technique est très utilisée par les moteurs de recherche lors de la détection des pages web redondants.

— La méthode Hachage pour la distance de Hamming [49] qui est une fonction de hachage adaptée à la distance de Hamming, dans un espace binaire de dimension d. Son principe est que pour un vecteur binaire qui appartient à $0, 1^d$, la fonction est l'ensemble des projections sur une des d coordonnées définie comme suit :

$$F = h : 0, 1^d \longleftrightarrow 0, 1|h(x) = x_i, i = 1, ..., d \tag{2.13}$$

x_i c'est la i^e coordonnée de x, la fonction h permet de sélectionner un bit au hasard à partir des coordonnées du vecteur x.

2.2.4.3 Les fonctions de hachage dans les CBIR

Parmi les travaux qui utilisent les fonctions de hachage nous pouvons citer :

— Les auteurs dans [76] proposent une nouvelle méthode à base du vecteur Fisher. Cette approche utilise différentes fonctions de compression comme LSH et Sensitive Hashing (SH) afin de compresser le vecteur Fisher. D'abord, le vecteur Fisher est extrait, où sa taille dépend de la base d'images. Dans ce cas la fonction LSH est appliquée afin de réduire la taille de ce vecteur. Les expérimentations ont été appliquées sur deux bases d'images : la base d'image Holiday [51] et le benchmark de l'université de Kentucky [72]. Les résultats obtenus montrent que le vecteur Fisher est meilleur pour la recherche des images appliquée sur une petite bases d'images, par contre pour les grandes bases d'images, la compression de vecteur avec la combinaison des deux

méthodes LSH et SH donne un temps de recherche rapide.

— D'autres auteurs dans [99] présentent une nouvelle technique de hachage supervisée pour la recherche d'images. Cette approche adapte la représentation d'image par rapport aux fonctions de hachage. La méthode proposée factorise d'abord la matrice de similarité sémantique par deux valeurs en codes de hachage approximatifs pour les images d'apprentissage. Par la suite, ils appliquent un réseau de neurone convolutif pour l'apprentissage pour les valeurs de hachage approximatives. L'approche a été appliquée sur la base d'images MNIST [4]. Les résultats obtenus montrent l'intérêt de l'utilisation des fonctions de hachage quand les dimensions des vecteurs de caractéristiques sont très importantes.

— Dans [96], les auteurs Proposent un système d'indexation et de recherche pour les téléphones portable. L'approche utilise le descripteur SIFT qui est un vecteur de 128 dimensions. Pour rechercher une image, les processus extraction des caractéristiques, calcul d'index de l'image requête sont transférer vers un serveur dans le cloud. Grace aux fonctions de hachages. Cette fois la méthode LSH est utilisée afin de sécuriser l'envoie de l'index de la requête en mode crypté ainsi avec une taille petite afin d'accélérer la recherche. La méthode LSH et appliqué directement sur le vecteur SIFT. De l'autre côté du serveur cloud, la requête est récupérée, puis un calcul de similarité est effectué afin d'envoyer les résultats aux utilisateurs. Cette approche nécessite une connexion internet afin de lancer une recherche ce qui n'est pas le cas pour tout le monde.

— Les auteurs dans [23] ont proposé une méthode de hachage adaptative et asymétrique est proposée pour une recherche rapide des images. Cet algorithme s'appelle Adaptive and Asymmetric Residual Hash (AASH). L'approche proposée consiste à d'abord proposer un algorithme de hachage adaptatif et asymétrique qui permet d'apprendre et reconnaitre les codes de hachage pour une recherche rapide, De plus, ces codes de hachage sont appris rapidement et efficacement d'une manière asymétrique. L'algorithme apprend uniquement le code de hachage de l"image requête. Le modèle d'apprentissage utilisé est basé sur les réseaux de neurones profonds en utilisant les architectures VGG16, VGG19 et FSDH ce qui permet d'apprendre le code de hachage l'ensemble de la base d'images. La base de données utilisée a été collectée par leurs équipe de travail. Les auteurs ont d'abord testé l'efficacité du code de hachage de la base de données d'apprentissage avec l'un des modèles cités ci-dessus. Ensuite, ils optimisent les paramètres du hachage en fonction du code de hachage de la base de données appris. A la fin, ils comparent l'algorithme AASH avec l'algorithme de hachage traditionnel. Les résultats obtenus montrent que l'approche proposée sur la base d'images Cifar10 est optimale avec un hyper paramètre de 100 et un vecteur de hachage de taille 32.

— Dans [61] les auteurs proposent un moteur de recherche des images basé sur les réseaux de neurones profonds ainsi que les fonctions de hachage. Ils proposent un nouveau modèle basé sur une nouvelle architecture CNN qui apprend d'une manière simultané les représentations des images ainsi qu'un ensemble des fonctions de hachage. Les résultats obtenus montrent qu'avec un seul changement des CNN la méthode proposée donne un résultat meilleur allant de 1% à 30% en terme de précision. Les expérimentations ont été menées sur la base d'image CIFAR-10 et MNIST.

Dans le Chapitre 4, nous présentons une méthode basée sur les fonctions de hachage. Cette méthode a été combinée avec d'autres méthodes afin d'accélérer la phase de recherche.

4. MNIST : http ://yann.lecun.com/exdb/mnist/

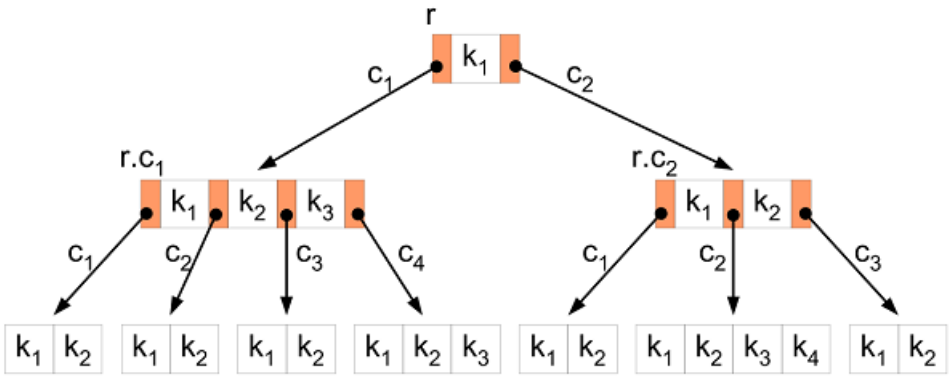


FIGURE 2.9 – Exemple d'un arbre B-Tree

2.2.5 Les méthodes de stockage des descripteurs

Une des façons d'accélérer la recherche dans les systèmes d'indexation est de trouver une représentation des descripteurs utilisés. La démarche consiste à identifier une meilleure structuration des descripteurs.

Parmi les approches de structuration des descripteurs qui utilisent une représentation arborescente nous pouvons citer les suivantes :

2.2.5.1 B-Tree

La représentation B-Tree comme nous le montre la Figure 2.9, est une structure de données en arbre équilibré. Son principe est de permettre aux nœuds parents de posséder plus de deux nœuds enfants, c'est une généralisation de l'arbre binaire de recherche. Les B-Tree sont principalement mis en œuvre pour les mécanismes de gestion des bases de données et de systèmes de fichiers. Ils stockent les données sous une forme triée et permettent une exécution des opérations d'insertion et de suppression en un temps toujours logarithmique [26].

2.2.5.2 Kd-Tree

Kd-Tree est une méthode de partitionnement de l'espace de caractéristiques. Kd-Tree est représenté par un arbre binaire multidimensionnel consistant pour chaque niveau de l'arbre à partitionner l'espace en deux sous-espaces successivement selon chaque dimension [13]. Cette structure arborescente fait partie des structures Binary Space Partionning Trees (BSP-trees) [34]. La recherche dans cette structure est basée sur une recherche de plus proche voisin KNN.

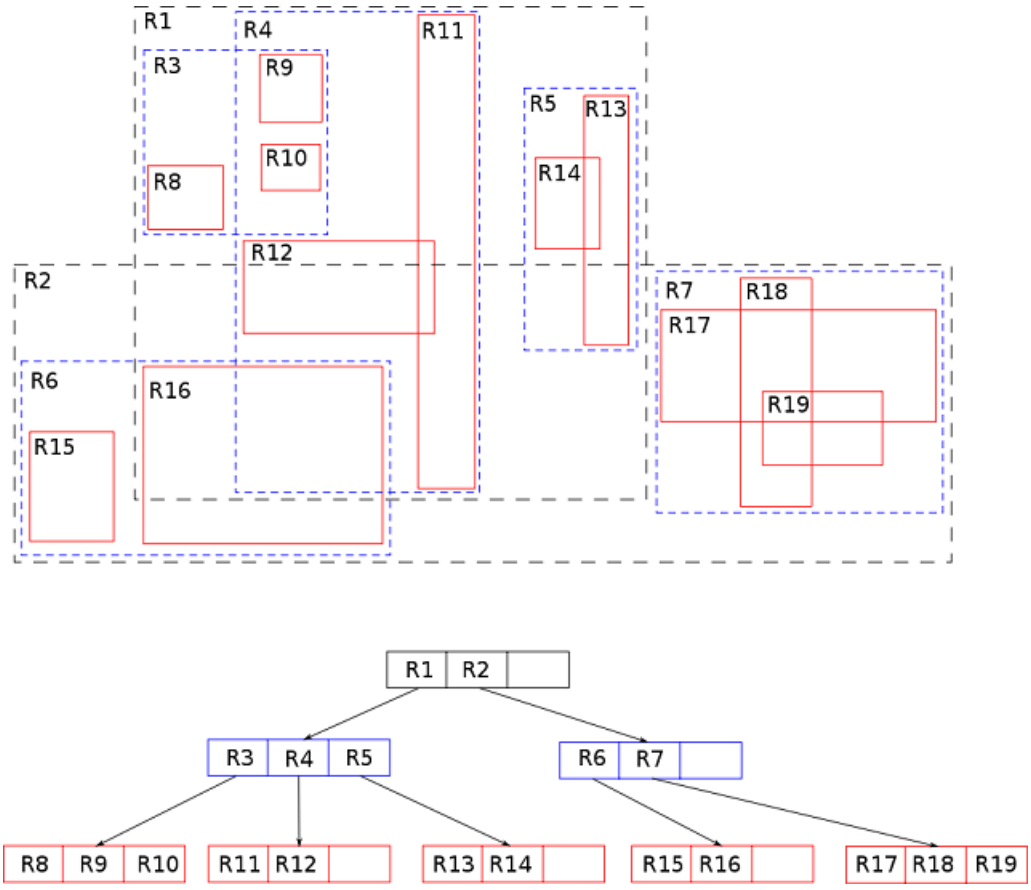


FIGURE 2.10 – Exemple d'un arbre R*-tree [53]

2.2.5.3 R*-tree

La structure arborescente R*-tree proposé par Antonin Guttman [38] est une méthode de partitionnement des données. Cette méthode se base sur la combinaison de volumes hyper-rectangulaires et hyper-sphériques. Il s'agit d'un arbre équilibré qui correspond à une hiérarchie de rectangles imbriqués, ou les feuilles sont des pointeurs vers les données. Cependant, la structure de l'arbre et son évolution sont conçues pour que la recherche spatiale soit la plus efficace possible, c'est-à-dire qu'elle nécessite le parcours d'un petit nombre de nœuds de l'index. Le temps de recherche dans cette structure dépend de la quantité de chevauchement, qui ne peut pas être déterminée par sa hauteur. La Figure 2.10 montre un exemple d'un arbre R*-tree.

2.2.5.4 SS-tree

La méthode SS-tree est plus performante que la structure Kd-Tree et glsR*-tree [53]. Cette méthode utilise les boites sphériques. Parmi les avantages d'utilisation de la représentation SS-tree on peut citer l'utilisation de moins d'espace mémoire que la représentation R*-tree (rectangles), la réduction de la hauteur de l'arbre, le doublement du nombre de feuilles des nœuds, l'accélération de l'opération d'insertion en temps linéaire en utilisant un algorithme de choix des sous-arbres ainsi que les points qui

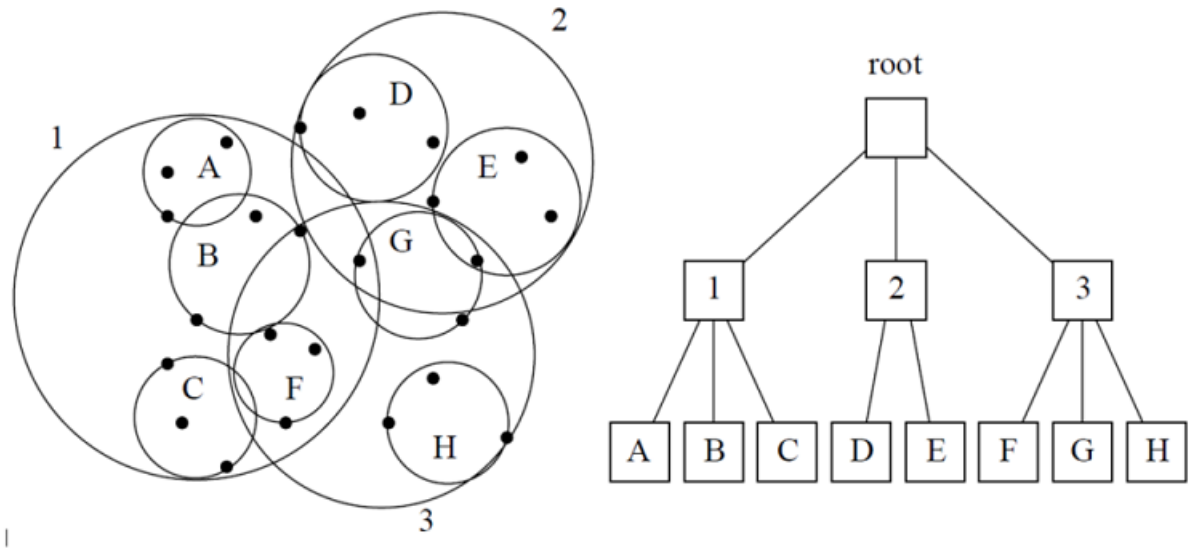


FIGURE 2.11 – Exemple d'un arbre SS-tree [53]

sont divisés en voisins isotropiques ce qui permet l'amélioration des performances de la méthode de recherche KNN. La Figure 2.11 montre un exemple d'un arbre SS-tree.

2.2.5.5 SR-tree

La structure arborescente SR-tree (SR-tree) est une extension des deux méthodes R*-tree et SS-tree. Elle se base aussi sur la combinaison des deux volumes rectangulaires et sphériques. Cette structure, dans le cas des données multimédia, donne de bons résultats parce que les données présentent de grandes dimensions et ils sont uniformes, mais elle possède des inconvénients. En effet, quand la dimension augmente le nombre maximal de branches d'un nœud intermédiaire diminue et la dimension du nœud dépend de la dimension des jeux de données.

Les volumes rectangles améliorent la séparation entre les régions parce que les points sont divisés en petites régions. En revanche, les volumes sphériques permettent de diviser les points en régions mais avec un diamètre court contrairement au volume rectangulaire qui présente des diamètres de grandes tailles. Mais les volumes sphériques ont tendance à avoir des volumes plus grands. L'efficacité de la recherche KNN dépend de la longueur du diamètre. L'insertion entre une boite sphérique et rectangulaire forme une région SR-tree. Pour insérer un nouveau nœud on utilise l'algorithme d'insertion SS-tree, et la suppression se base aussi sur l'algorithme de suppression du SR-tree. La Figure 2.12 montre un exemple d'un arbre SR-tree.

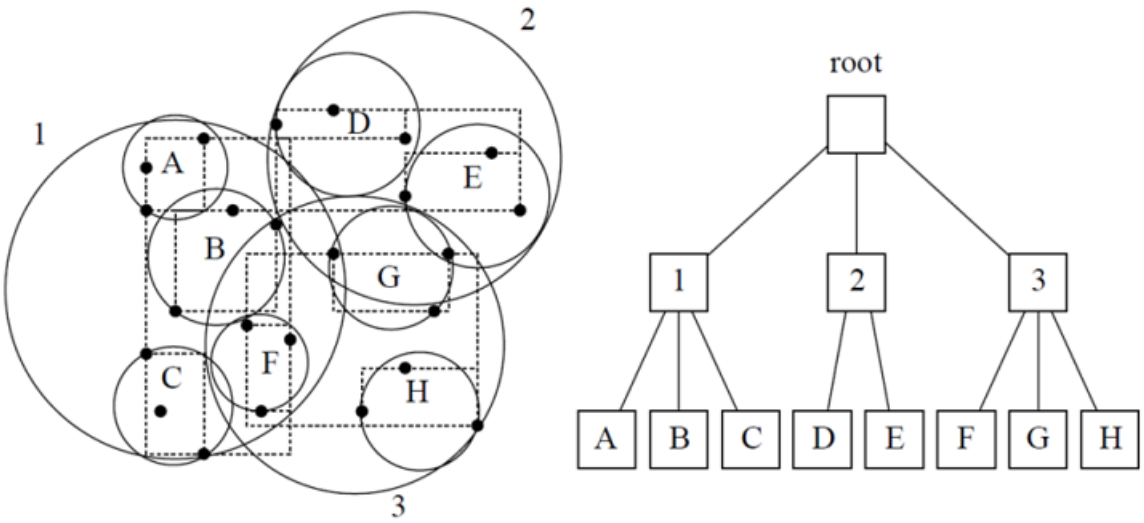


FIGURE 2.12 – Exemple d'un arbre SR-tree [53]

2.2.5.6 LSD-tree

LSD-tree (LSD-tree) [40] est une méthode à deux niveaux, le premier niveau se situe dans la mémoire centrale, le deuxième niveau réside au niveau disque. C'est une structure sous forme d'un arbre binaire où chaque nœud est un partitionnement de l'espace en deux. Chaque nœud de l'arbre est représenté par le numéro de la dimension selon laquelle le partitionnement est effectué, et la position sur l'axe associé à cette dimension. Cependant, les feuilles contiennent les pages de données. Ces pages sont de taille limitée et fixée à priori. La construction de l'arbre se fait par insertion successive des vecteurs. Lors d'un dépassement de la capacité des régions, un partitionnement du nœud est effectué, le choix de la position et la dimension du partitionnement se fait aléatoirement.

Afin de comparer entre les différentes méthodes de structure de stockage nous avons tracé le Tableau 2.2 qui montre un récapitulatif de ces différentes méthodes.

TABLEAU 2.2 – Récapitulatif des méthodes de structure de stockage

Méthodes	**Formes englobantes**	**Équilibre**	**Chevauchement**	**Complexité**
R*-tree	Hyper rectangle	Oui	Oui	Calculable
SS-tree	Hyper sphère	Oui	Oui	Calculable
SR-tree	Rectangle et sphère	Oui	Oui	Calculable
LSD-tree	Hyper rectangle	Non	Non	Non calculable

2.2.5.7 Les méthodes de stockage des descripteurs dans les CBIR

Parmi les travaux dans le domaine de stockage des descripteurs sous forme arborescence, nous pouvons citer les suivants :

- Les auteurs dans [50] présentent une technique d'indexation utilisée dans un système d'identification dans des bases de données biométriques. Cette approche s'appuie sur une méthode de réduction de dimensionalité, qui indexe les données (réduites) sous forme d'arbres Kd-Tree. L'approche proposée permet de réduire le temps de recherche de données avec un faible taux d'erreur. Le système proposé a été testé sur une base de données multimodales (5400 images avec 150 objets)
- Les auteurs dans [98] proposent une nouvelle méthode de structuration appelée SS-tree. Les tests ont été effectués sur des données de grande dimension. Les auteurs ont constaté que cette structure était plus performante que l'arbre R-tree (R-tree). Les résultats obtenus montrent également que cette méthode est mieux adaptée aux requêtes approximatives par rapport à la méthode R-tree.
- Dans [78], les auteurs proposent un nouveau modèle basé sur la combinaison entre une méthode de triangulation spatiale modifiée Triangular Spatial Relationship (TSR) avec la structure d'arbre B-Tree. Le modèle préserve TSR parmi les composants dans une image symbolique en utilisant des quadruples. Ce modèle de recherche à une complexité logarithmique. Le système proposé a été testé sur plusieurs images symboliques, ainsi que sur les bases de données de détection de visages ORL [86] et YALE [12].
- Les auteurs dans [89], proposent une méthode appelée projection et hachage multi-échelle qui utilise une projection aléatoire et des structures d'index basées sur le hachage. Les auteurs considèrent que tous les objets sont bien annotés. Cette méthode s'appuie sur une architecture qui permet la recherche à travers les mots clés sur des grandes bases d'images. Cette méthode utilise une représentation R-tree qui facilite le calcul de la distance entre la requête et les indexes.

Dans ce travail, nous avons proposé l'utilisation d'une méthode à base d'arbres binaires, que nous détaillerons dans le chapitre Chapitre 4. Nous y comparons aussi la méthode proposée à d'autres méthodes de structure de stockage.

2.3 Les solutions basées sur les technologies du Big Data pour le CBIR

Dans cette section, nous allons présenter quelques notions liées au traitement de grandes masses de données et les technologies Big Data appliquées au domaine de l'indexation multimédia. Nous allons présenter les principales technologies du Big Data, Hadoop ainsi que Mapreduce.

2.3.1 Introduction

Chaque jour 2,5 trillions d'octets de données sont générés. Ces données proviennent de partout : de capteurs utilisés pour collecter les informations climatiques, des messages sur les sites de médias sociaux, des images numériques et des vidéos publiées en ligne, des enregistrements transactionnels d'achats en ligne et de signaux Global Positioning System (GPS) de téléphones mobiles etc. Ces données

sont appelées Big Data ou volumes massifs de données.

Big Data est un paradigme émergeant appliqué aux ensembles de données dont la taille est au-delà de la capacité des outils logiciels couramment utilisés (outils classiques de gestion de base de données ou de gestion de l'information) pour capturer, gérer et traiter les données dans un temps écoulé tolérable.

Big Data vise à proposer une alternative aux solutions traditionnelles de bases de données et d'analyse (serveur Structured Query Language (SQL), plate-forme de business intelligente...). Confrontés très tôt à des problématiques de très grands volumes, les géants du Web dont Yahoo, Google et Facebook, ont été les premiers à déployer ce type de technologies.

Les techniques appliquées dans le domaine du Big Data s'approchent beaucoup du Data Mining dans sa transformation de l'information stockée en information clé pour une utilisation future. Là où le Big Data marque une grande différence, c'est dans le besoin émanant de ces données clés. Souvent, les entreprises ont une idée de ce qu'elles peuvent tirer de leurs informations, mais ne savent pas les rendre utiles. Dans d'autres cas, la question est de savoir si on ne révèle pas de nouvelles informations en établissant des corrélations entre ces ensembles de données.

2.3.2 Les caractéristiques du Big Data

Selon Gartner le Big Data regroupe une famille d'outils qui répondent à trois problématiques :

- **Un volume de données important à traiter :** Facebook génère environ 500 téraoctets des données chaque jour ; Boeing 737 génère 240 téraoctets des données pour chaque vol.
- **Une grande variété d'informations :** en provenance de plusieurs sources non structurées telles que les médias sociaux, les capteurs, les applications scientifiques, vidéo surveillance, textes et documents Internet, recherche sur Internet par contenu, les dossiers médicaux, les transactions commerciales et les journaux électroniques.
- **Un certain niveau de vélocité (vitesse) à atteindre :** c'est-à-dire de fréquence de création, collecte et partage de ces données.

2.3.3 Les bases de données du Big Data

Elles sont nombreuses. Pour optimiser les temps de traitement sur des bases de données géantes, plusieurs solutions peuvent être utilisées :

- Des bases de données Not Only SQL (NoSQL) (comme MongoDB, Cassandra ou Redis) qui implémentent des systèmes de stockage considérés comme plus performant que le traditionnel SQL pour l'analyse de données en masse (orienté clé/valeur, document, colonne ou graphe).
- Des infrastructures de serveurs pour distribuer les traitements sur des dizaines, centaines, voire milliers de nœuds. C'est ce qu'on appelle le traitement massivement parallèle. Le Framework Hadoop est sans doute le plus connu d'entre eux. Il combine le système de fichiers distribué HDFS, la base NoSQL HBase et l'algorithme MapReduce.
- Le stockage des données en mémoire (Memtables) permet d'accélérer les temps de traitement des requêtes.

2.3.4 Les solutions Big Data

En 2004, Google a publié un article présentant son algorithme basé sur des opérations analytiques à grande échelle sur un grand cluster de serveurs, le MapReduce, ainsi que son système de fichiers en clusters, le GoogleFS. Doug Cutting travaillait à cette époque sur le développement d'Apache Lucene et a été confrontés à des problèmes similaires à ceux rencontrés par Google [83]. Il décida alors de reprendre les concepts décrits dans l'article pour développer sa propre version des outils en version Open Source, c'est ce qui donnera le projet Hadoop.

2.3.4.1 La technologie Hadoop

Hadoop est l'environnement le plus populaire pour la gestion des bases de données Big Data. Son code source est librement accessible au public. Il met en oeuvre un système de gestion des fichiers distribués tolérant aux pannes, appelé HDFS et un modèle de traitement distribué MapReduce [83].

Apache Hadoop est un Framework Java libre open source réputé pour sa puissance d'indexation, de transformation, de recherche ou d'élaboration de modèles sur de très gros volumes de données, destinés à faciliter la création d'applications distribuées et échelonnables (scalables) qui permettent le traitement distribué de grands ensembles de données à travers des grappes d'ordinateurs utilisant des modèles de programmation simples. Il est conçu pour évoluer à partir de serveurs individuels à des milliers de machines, chaque machine offre un calcul et un stockage local.

2.3.4.1.1 Caractéristiques

- Évolutif : il utilise plus de ressources physiques, selon les besoins, et d'une manière transparente.
- Rentable : il optimise les coûts via une meilleure utilisation des ressources présentes.
- Souple : il répond à la caractéristique de variété des données en étant capable de traiter différents types de données.
- Résilient : pas de perte d'information et capable de poursuivre le traitement si un nœud du système tombe en panne.

2.3.4.1.2 Architecture

Le système HDFS est conçu pour stocker de très grands ensembles de données de manière fiable, et de les diffuser à haute bande passante pour les applications utilisateurs. Dans un grand groupe, des milliers de serveurs sont directement connectés pour stocker et exécuter des tâches d'application de l'utilisateur.

En distribuant le stockage et le calcul sur plusieurs serveurs, la ressource peut se développer à la demande tout en restant économique à chaque taille [87].

HDFS comme nous montre la Figure 2.13 est un système de fichiers virtuel qui se décompose en un *namenode*, le maître, et plusieurs *datanodes*, les nœuds de données.

Les nœuds de données regroupent les blocs de données en les répliquant. Les blocs sont tous répliqués trois fois. Le maître, quant à lui, va orchestrer les données, et contient les informations concernant

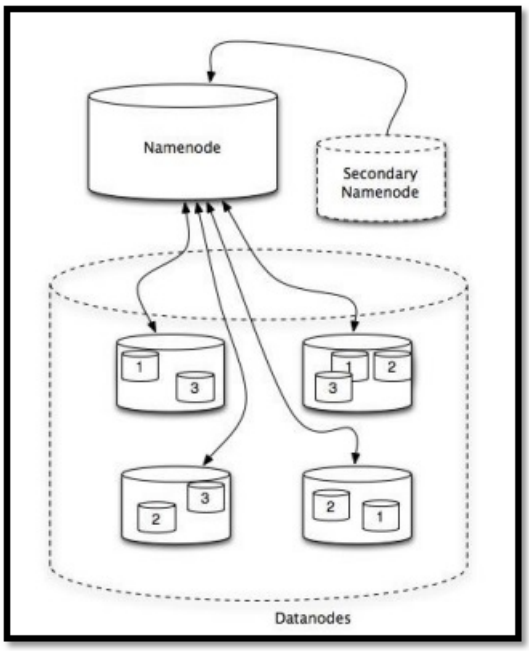


FIGURE 2.13 – Les différents nœuds du HDFS

l'emplacement des différentes répliques. Le deuxième *namenode* sert à effectuer les points de contrôle réguliers du *namenode*, afin de les réutiliser en cas de problème.

Hadoop fournit un système de fichiers distribués et un framework pour l'analyse et la transformation de très grands ensembles de données (Big Data) en utilisant le MapReduce paradigme. Une caractéristique importante de Hadoop est le partitionnement des données ainsi que le calcul à travers de nombreux (en milliers) hôtes. Nous pouvons aussi ajouter l'exécution des applications de calcul en parallèle à proximité de leurs données.

Un cluster Hadoop permet l'exploitation des capacités de calcul, les capacités de stockage et la bande passante par le simple ajout des serveurs des produits de base.

Le cluster Hadoop de Yahoo comporte environ 25 000 serveurs et stocke 25 pétaoctets de données [87].

2.3.4.2 Map Reduce

MapReduce est un framework de traitement de données en clusters. Composé des fonctions Map et Reduce, il permet de répartir les tâches de traitement de données entre différents ordinateurs, pour ensuite réduire les résultats en une seule synthèse.

MapReduce est en train de devenir un modèle de programmation important pour les applications parallèles à grande échelle, alors que Hadoop représente une implémentation open source de MapReduce jouissant d'une grande popularité pour développer des applications à forte intensité de données dans le Cloud.

Comme, dans le Cloud, l'unité de calcul de base est la machine virtuelle. Il est possible de démontrer

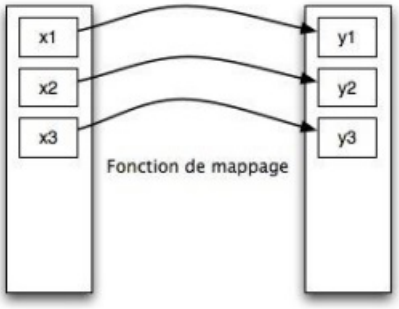


Figure 2.14 – La fonction de mappage

l'applicabilité de MapReduce dans les centres de données virtualisées.

Les machines virtuelles peuvent être utilisées pour ne pas exploiter les ressources du système, faciliter la gestion de ces systèmes, améliorer la fiabilité et économiser l'énergie [47].

MapReduce est un patron d'architecture (modèle de référence) mis en place par Google pour soutenir le traitement distribué de grands ensembles de données typiquement supérieurs en taille à 1 téraoctet sur de grandes grappes d'ordinateurs.

Le modèle de logiciel prend comme entrée un ensemble de couples (clé, valeur) et produit comme sortie un ensemble de couples (clé, valeur).

L'utilisateur doit fournir deux fonctions :

— Mapping : comme nous montre la Figure 2.14 c'est une étape qui accomplit une opération spécifique sur chaque élément de la liste en entrée. Le principe est simple, à partir d'une liste sous la forme (clé, valeur), il génère une liste en sortie sous la même forme.
— Reduce : L'opération qui se situe entre le Mapping et le Reducing est appelée le Shuffling, et réarrange les éléments de la liste afin de préparer le Reducing. Le traitement voulu est alors effectué, donnant la sortie finale, comme nous montre la Figure 2.15 :

En outre, l'utilisateur fournit les spécifications pour le travail de MapReduce à exécuter, par exemple en spécifiant la mémoire requise, le disque, le nombre de machines, etc. L'exemple canonique pour MapReduce est de compter combien de fois chaque mot apparaît dans un document [6].

La fonction Map est appelée pour chaque ligne du document d'entrée, où la clé est le numéro de ligne et la valeur est une chaîne de caractères contenant cette ligne de texte.

Pour chaque mot dans la ligne, la fonction Map envoie une paire intermédiaire avec le mot comme la clé et une valeur de "1". Le MapReduce prend soin de regrouper toutes les paires intermédiaires avec la même clé (le mot), et les présente à la fonction Reduce, qui résume simplement les valeurs pour chaque clé (mot) et émet une paire avec le mot comme la clé et le nombre de fois à ce mot est apparu comme la valeur. Le résumé des deux opérations Map et Reduce est montré dans la Figure 2.16.

Le framework MapReduce est utilisé car cette solution a largement fait ses preuves. De même, il est utilisé par un grand nombre d'entreprises, notamment Google, Yahoo, Amazon et Facebook.

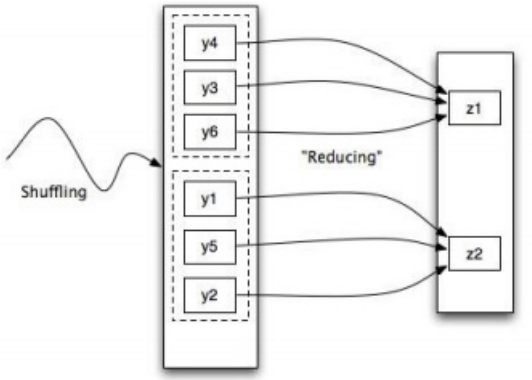


FIGURE 2.15 – La fonction de Reduce

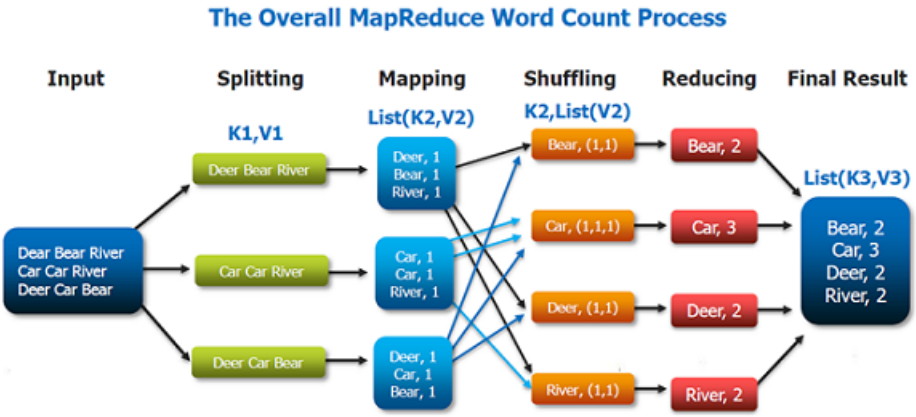


FIGURE 2.16 – Exécution de l'exemple compter le nombre de mots

MapReduce est simple, fiable et évolutive. L'utilisateur dispose de deux fonctions, et l'infrastructure prend soin de tout traitement intermédiaire : tri de la clé, les communications inter-machines, les défaillances de la machine, redémarrage d'emploi, etc.

Il peut facilement évoluer jusqu'à des dizaines de milliers de machines. En outre, il existe déjà une forte mise en œuvre open source, appelée Hadoop, qui est largement utilisée.[6].

2.3.4.3 Indexation des grandes bases de données dans les CBIR

Parmi les travaux qui existent dans la littérature d'indexation des grandes bases de données nous pouvons citer :

- OSIRIM qui est un projet fédératif de l'Institut de recherche en informatique de Toulouse (IRIT) conduit par les équipes de recherche SAMOVA, SIG, et IRIS et principalement soutenu par le Fonds Européen de Développement Régional (FEDER), le Centre national de la recherche scientifique (CNRS), le gouvernement français, et la région Midi-Pyrénées. Le but est de proposer un environnement homogène pour la recherche sur l'indexation et la recherche d'information dans des contenus multimédias. Cela correspond pour l'essentiel à la mise en place d'une architecture matérielle et logicielle permettant de construire, tester et évaluer facilement des chaînes plus ou moins complexes d'outils visant à étiqueter et à retrouver des contenus textuels, audio ou vidéo. Le projet est adapté aux traitements distribués sur de gros volumes de données (Hadoop, Spark, MongoDB, ...).
- Les auteurs dans [7] proposent une nouvelle technique de détection de texte en utilisant de nombreux blocs provenant de la décomposition de vidéos en images. Chaque bloc est analysé et classifié ce qui a permis l'extraction des coordonnées de texte en utilisant la fonction MapReduce. Le système a été testé sur un ensemble de données provenant depuis YouTube Video Text (YVT). Finalement, ils ont constaté que la vitesse de cette approche peut être plus de deux fois plus rapide qu'une approche classique.
- D'autres auteurs dans [100] présentent un nouveau système pour l'extraction et la recherche des concepts sémantiques. Le système est conçu pour qu'il s'adapte avec l'infrastructure Hadoop MapReduce, puis il a été étendu pour qu'il s'adapte avec Spark. Les résultats obtenus montrent l'intérêt d'utilisation des technologies Big Data dans ce domaine.
- Les auteurs dans [80] proposent une approche qui permet d'indexer les images en utilisant le descripteur SIFT ainsi que Hadoop et MapReduce. Cette approche divise l'image en plusieurs blocs avec la fonction Map, chaque machine calcule le descripteur SIFT de chaque bloc attribué. Ensuite la fonction Reduce regroupe les blocs des descripteurs en un seul fichier pour chaque image.
- Dans [46] les auteurs proposent un moteur de recherche basé sur l'architecture Hadoop ainsi que MapReduce. L'approche utilise ces deux Framework pour l'indexation des images. La texture est la caractéristique de l'image qui est utilisée. Ces caractéristiques sont stockées en parallèles avec MapReduce. Pour la recherche l'algorithme KNN est utilisée pour rechercher les images les plus proches de l'image requête en utilisant toujours MapReduce afin de paralléliser aussi la recherche. Les auteurs ont remarqué que l'utilisation de la mémoire cache permet d'accélérer encore la phase de recherche sachant que Hadoop est un Framework qui repose sur la forte utilisation de

la mémoire RAM

- Dans ce travail [35], les auteurs proposent un moteur de recherche des images en utilisant le descripteur SIFT pour l'extraction des caractéristiques des images en s'appuiant sur Hadoop. L'approche utilise MapReduce comme une plateforme pour les données de type BigData pour le traitement et le stockage des descripteurs des images. Ils ont intégré HIPI dans le système pour charger et filtrer les données en un format HDFS. L'approche utilise deux phases : l'extraction des caractéristiques et la recherche. Ces deux phases utilisent MapReduce afin d'accélérer les opérations E/S. Les expérimentations ont été mené sur la base d'images TumIndoor. Les résultats obtenus montrent l'intérêt d'utiliser Hadoop et MapReduce afin d'accélérer le processus de la recherche, mais ce processus nécessite une machine équipée d'une mémoire RAM qui dépasse les 8GB.
- Les auteurs dans [25], proposent une nouvelle architecture appelée HIPI. HIPI est un Framework écrit en java basée sur Hadoop qui est utilisé pour le traitement des images. Les auteurs proposent un nouveau format de stockage des images appelé HIPI bundle images. Par la suite la fonction MapReduce lit ce bloc et applique un algorithme de traitement d'images comme SIFT. Dans cette phases les images sont dévissées à plusieurs groupe grâce à la fonction Map. Une fois que le traitement est effectué, La fonction Reduce regroupes l'ensemble des images avec leurs descripteurs et stocke les résultats dans une base de données.

Dans ce travail, nous avons proposé l'utilisation d'une solution parallèle et distribuée à base de Hadoop, que nous détaillerons dans le chapitre Chapitre 6. Nous y comparons aussi la méthode proposée à d'autres méthodes existent dans la litérature.

2.4 Conclusion

De très nombreux travaux ont été menés dans le domaine d'indexation multimédia, plusieurs aspects, méthodologies et techniques sont employés pour améliorer l'efficacité de la recherche, pour réduire le temps, et avoir des bons résultats. Nous proposons dans ce document deux contributions : la première permet de réduire les dimensions des descripteurs en utilisant les méthodes PCA et VA-File, ainsi que l'identification d'une meilleure structure de stockage qui permet d'accélérer le temps de recherche, réduire l'accès au disque et réduire l'espace de stockage. La deuxième contribution est l'étude d'une implémentation parallèle et distribuée des algorithmes d'indexation en utilisant les solutions Big Data comme Hadoop, HIPI, Spark, etc.

Troisième partie

Contributions

Chapitre

3

Réduction de la dimensionnalité

« La simplicité a une dimension morale qui inclue le désintéressement et le détachement des biens matériels. »

— John Pawson

Sommaire

3.1 Introduction

Dans ce chapitre, nous allons présenter une nouvelle méthode de réduction de dimensionnalité des descripteurs sur base de l'analyse en composante principale PCA, ainsi que d'autres méthodes qui utilisent les fonctions de hachage (LSH) et des méthodes d'indexation par approximation (VA-FILE). Ces différentes méthodes seront comparées afin de valider notre choix. Ces expérimentations ont conduit à une amélioration de la méthode proposée à base de PCA par le biais d'une combinaison avec les méthodes LSH et VA-FILE. L'application de ces méthodes a permis la réduction de la dimension des descripteurs des images, en gardant la précision du moteur de recherche. Ces approches ont été testées sur plusieurs bases d'images, ainsi que sur plusieurs machines (physiques et virtuelles).

Les résultats de ce chapitre font l'objet d'une publication dans un article de journal dont le titre est : **PCA as Dimensionality Reduction for Large-Scale Image Retrieval Systems** dans la revue : **International Journal of Ambient Computing and Intelligence (IJACI)**.

De même, la deuxième partie des résultats de ce chapitre font l'objet d'une publication qui est en cours de soumission.

3.2 Réduction de dimension des descripteurs

Comme décrit dans la section "état de l'art", il existe deux types de méthodes pour la réduction de la dimension : les méthodes supervisées et les méthodes non supervisées. La Figure 3.1 nous montre ces différentes méthodes.

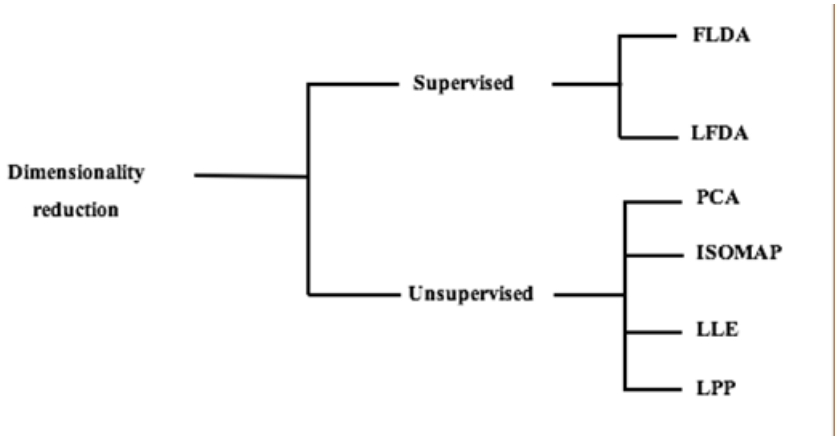


FIGURE 3.1 – La classification des méthodes de réduction de dimension

D'abord, le choix des deux deux descripteurs SIFT et SURF est basé sur le fait que ces derniers génèrent un vecteur des caratérstiques adapaté à la problématique de Big Data.

Nous avons testé les différentes méthodes de réduction de dimension appliqués aux descripteurs SIFT et SURF afin de les comparer et trouver celle qui s'adapte le mieux à nos descripteurs. Les résultats de ces expérimentations sont présentés sur le Tableau 3.1 :

Tableau 3.1 – Comparaison entre les différentes méthodes de réduction de la dimension

Technique	Complexité	T. Compression (ms)	T. Calcul (ms)
PCA	$O(DN) + O(D^3)$	19000	56700
FLDA	$O(DN) + O(D^3)$	36500	64410
LFDA	$O(D^2N) + O(D^3)$	36481	69786
ISOMAP	$O(kN^2 \log N) + O(N^3)$	99761	110563
LLE	$O(DN^2) + O((D+k)k^2N) + O(dD^2)$	56501	69210
LPP	$O(DN^2) + O(kDN) + O(dN^2)$	90142	94253

Où :

— **k** : présente le nombre des voisins les plus proches : (k-nearest neighbor).
— **D** : présente le nombre total des caractéristiques d'images de la base de données.

Les résultats obtenus montrent qu'en terme de complexité, les méthodes PCA, FLDA et LFDA sont similaires, mais celles-ci sont moins simples par rapport aux autres méthodes. En termes de temps de compression et de temps de calcul, la méthode PCA est la mieux adaptée avec les descripteurs SIFT et SURF, sachant que les expérimentations ont été appliquées sur plusieurs bases d'images.

Nous proposons donc d'utiliser la méthode PCA pour la réduction de dimension en ajoutant un taux de compression à l'algorithme de base de PCA comme paramètre d'entrée. L'algorithme PCA que nous proposons prend comme paramètre d'entrée le descripteur de l'image et non pas l'image. De plus, il permet d'appliquer différents taux de compression. En effet, dans notre cas, la compression est appliquée aux descripteurs et non aux images.

PCA est une procédure mathématique qui transforme un ensemble de variables, qui peuvent être corrélées en un ensemble de variables non corrélées, représentant la plus grande partie de la variabilité des données possibles. C'est l'une des méthodes largement utilisées pour l'analyse statistique des données dans la reconnaissance et la compression d'image.

Cependant, PCA n'est pas toujours le choix optimal pour la réduction de la dimensionnalité. En effet la dimension des fonctionnalités, qui a été réduite, ne convient pas à la classification interclasse et affectera de manière négative les performances de la recherche d'image dans une certaine mesure. Pour cela, nous appliquons l'algorithme PCA seulement pour les descripteurs d'image. Nous avons proposé d'appliquer l'algorithme PCA avec une réduction de dimensionnalité de 10% à 90%. La Figure 3.2 montre l'architecture générale de notre approche.

Comme tout système d'indexation, notre approche est composée de deux phases, une hors ligne et une autre en ligne. La phase hors ligne est la phase d'indexation. Tout d'abord, il faut extraire les descripteurs de l'image. Dans notre cas nous avons utilisé les deux descripteurs SIFT et SURF. Après l'extraction des descripteurs, nous stockons les indexes dans une base de données. Le résultat de l'extraction du descripteur SIFT est une matrice de taille n×128 par contre pour SURF, le vecteur résultant est de n×64 (n étant le nombre de ligne de la matrice). Après l'extraction des descripteurs, nous appliquons la technique PCA comme méthode de réduction de dimension en ajoutant un taux de compression variable de 10% à 90% comme paramètre. L'application de la commpresions par PCA fournit comme résultat des matrices

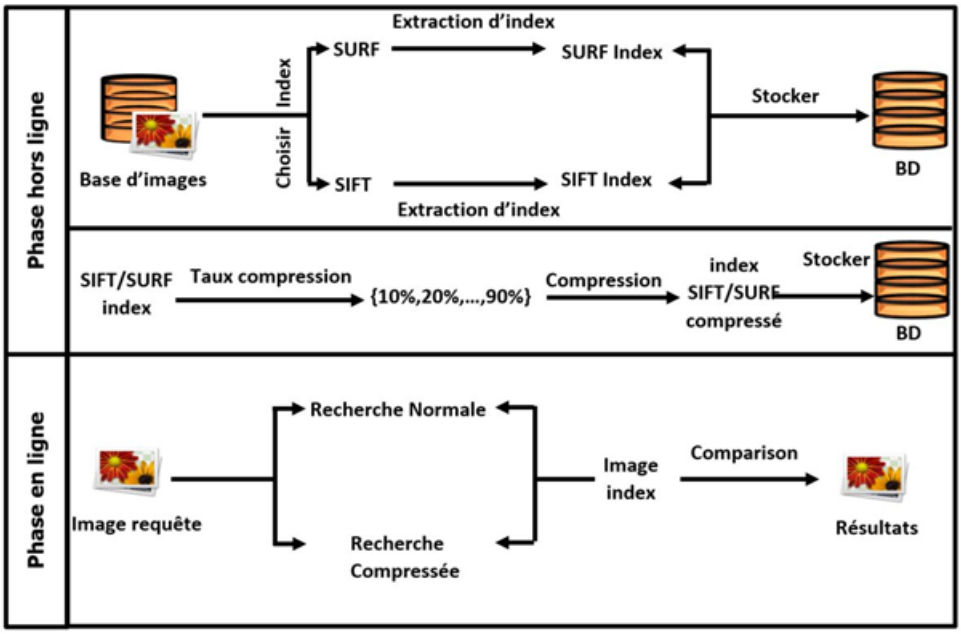


FIGURE 3.2 – Architecture générale de l'approche proposée

qui dépendent du taux de compression, comme par exemple :

$$SIFT : 10\% compression = 128 - (10 \times 128)/100 = 115 \tag{3.1}$$

$$SURF : 10\% compression = 64 - (10 \times 64)/100 = 57 \tag{3.2}$$

Dans la Figure 3.3, nous montrons le diagramme d'activité de notre approche. La phase d'indexation dure plusieurs heures. Pour cette raison, elle devrait être faite avant. Cette étape comprend l'extraction des caractéristiques ainsi que l'application de l'algorithme PCA. Par la suite, l'utilisateur choisit une image requête, puis lance le processus de recherche. A ce moment-là, le système calcul le descripteur de cette image, puis il appliquera une compression avec le taux choisi lors du processus d'indexation. L'utilisateur recevra une liste des images similaires.

3.2.1 Résultats expérimentaux

3.2.1.1 Préparation des données

Nous avons proposé l'algorithme PCA comme méthode de réduction de dimension des descripteurs en spécifiant comme paramètre le nombre des valeurs retenues qui s'adaptent avec le descripteur SIFT de manière à ne pas perdre l'information. Cette approche a été testée sur plusieurs machines et sur différents systèmes d'exploitation. Nous avons utilisé les courbes Rappel/Précision afin d'évaluer la pertinence des résultats renvoyés par les descripteurs utilisés pour les différents taux de compression. Nous avons tracé les différentes courbes Rappel/Précision pour les taux de compression de 10% à 90%. De même nous avons tracé les graphes relatifs au temps de recherche en fonction du taux de compression.

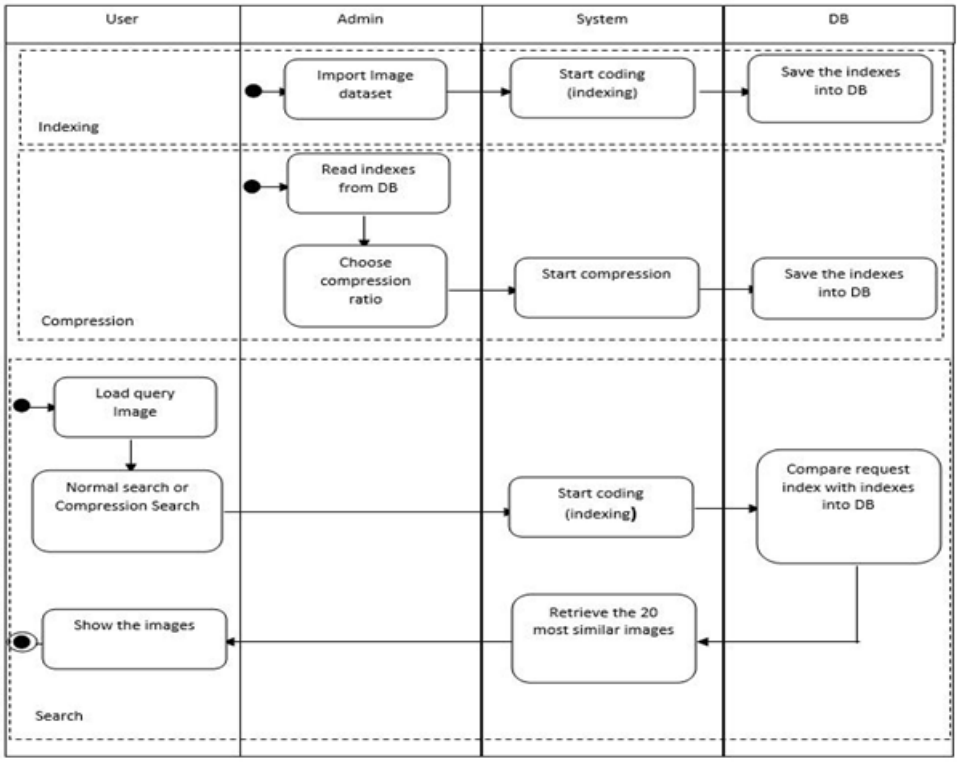


FIGURE 3.3 – Diagramme d'activité de notre approche

Algorithme 3.1 $\mathcal{PCAM}$ SIFT/SURF

Entrée : $S\_matrixi$: des matrices résultantes des algorithmes SIFT/SURF.
Sortie : $S\_matrixf$: des matrices SIFT/SURF compressées.

1 : $S\_matrixf \leftarrow \emptyset$;
2 : **pour tout** $i \in 10, ..., 90$ **faire**
3 : Lire $taux = 128 - (i \times 128)/100$;
4 : $S\_matrixf = PCA(S\_matrixi, taux)$; ▷ Application de la méthode $\mathcal{PCA}$ avec un taux de compression
5 : **fin pour**
6 : **retourner** $S\_matrixf$;

Nous avons effectué nos expériences en utilisant plusieurs bases d'images : la base d'images Wang. [5] qui contient 7000 images. [6] Ces images sont groupées en plus de 50 catégories. Une base d'images de 7200 images depuis Coil [7], 1 million d'images depuis Mirflickr [8] et 16 millions d'images depuis ImageNet. [9]

5. Wang : http ://wang.ist.psu.edu/docs/related/
6. Wang : hhttp ://wang.ist.psu.edu/docs/related/
7. Coil : http ://www.cs.columbia.edu/CAVE/software/softlib/coil-100.php
8. Mirflickr : http ://press.liacs.nl/mirflickr/
9. ImageNet : http ://image-net.org/about-stats

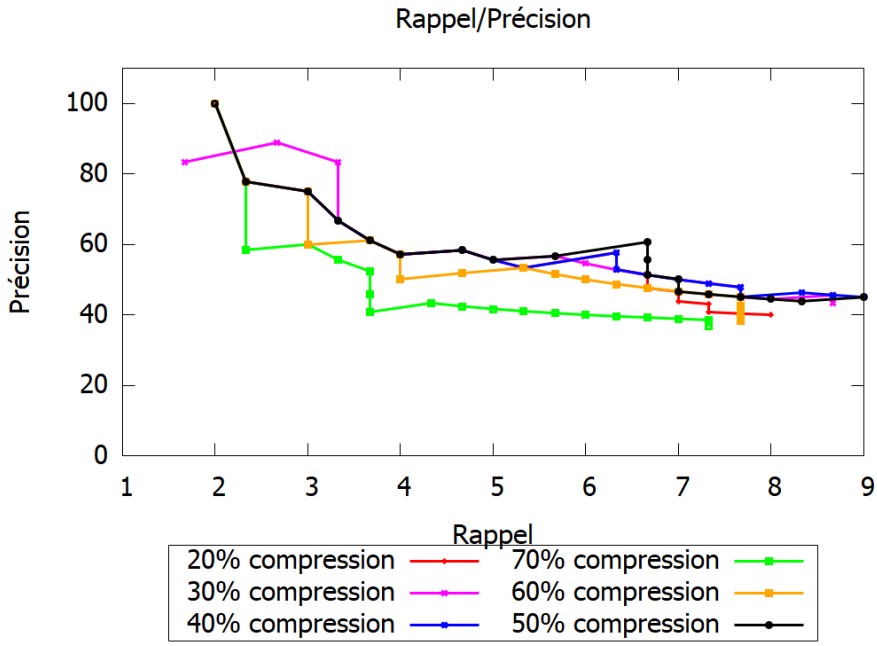


FIGURE 3.4 – Rappel/Précision pour la base d'image Wang

3.2.1.2 Résultats

La Figure 3.4 nous montre les courbes rappel/ précision pour la base d'image Wang.

La Figure 3.5 nous montre les courbes rappel/ précision pour la base d'image ImageNet.

La Figure 3.6 nous montre le temps recherche par rapport le taux de compression.

Dans la Figure 3.4 et la Figure 3.5 la courbe Rappel/Précision pour le taux 20% est quasiment similaire à celle du taux 70%, alors que dans la Figure 3.6 nous remarquons que le temps de recherche pour le taux de compression de 70% qui est de 19881 ms est plus rapide par rapport à celui de 20% qui est de 31796 ms. Ce qui veut dire que la méthode améliore le temps de recherche. En effet, pour la même précision, le temps de recherche est meilleur. De même nous gagnons en espace de stockage des descripteurs SIFT ou SURF pour la même précision. En effet, l'espace de stockage pour le taux de compression de 70% est la moitié par rapport à celui de 20%.

3.2.1.3 Discussion

L'approche proposée dans cette section est basée sur la méthode PCA. Nous avons introduit un taux de compression, et en variant ce taux de 10% à 90% nous avons calculé à chaque fois les courbes Rappel/Précision. Après plusieurs tests, nous avons constaté que le meilleur taux permettant de garder la même précision tout en accélérant la recherche correspond à 70%. De plus, le résultat avec un taux de compression de 70% est similaire à une compression correspondant à 20%. Par contre au delà de 80%, la précision se dégrade. Par ailleurs, notre méthode est moins performante avec les images qui contiennent plusieurs objets dans une même image, cela est dû à l'utilisation des descripteurs classiques qui sont

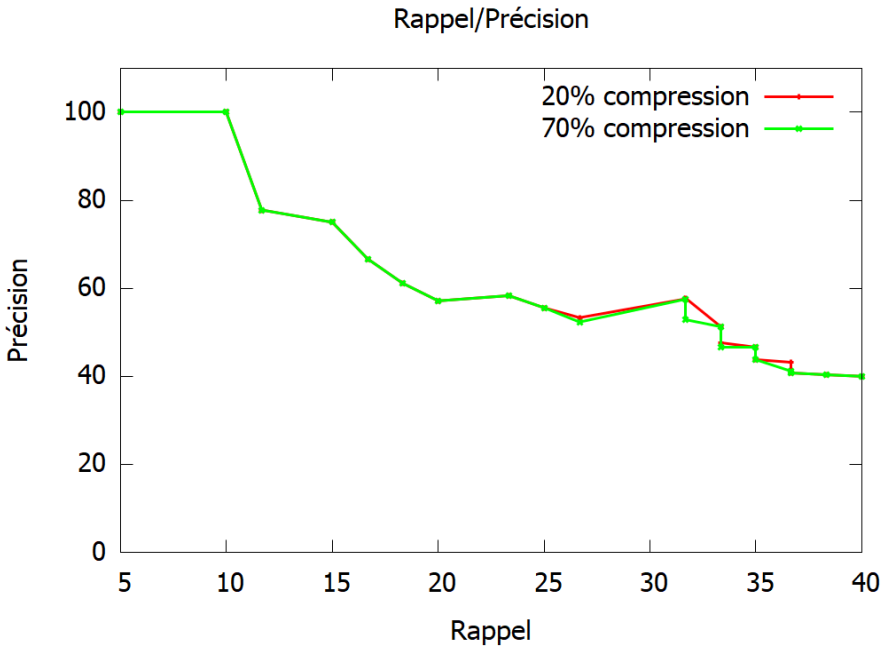


FIGURE 3.5 – Rappel/Précision pour la base d'image ImageNet

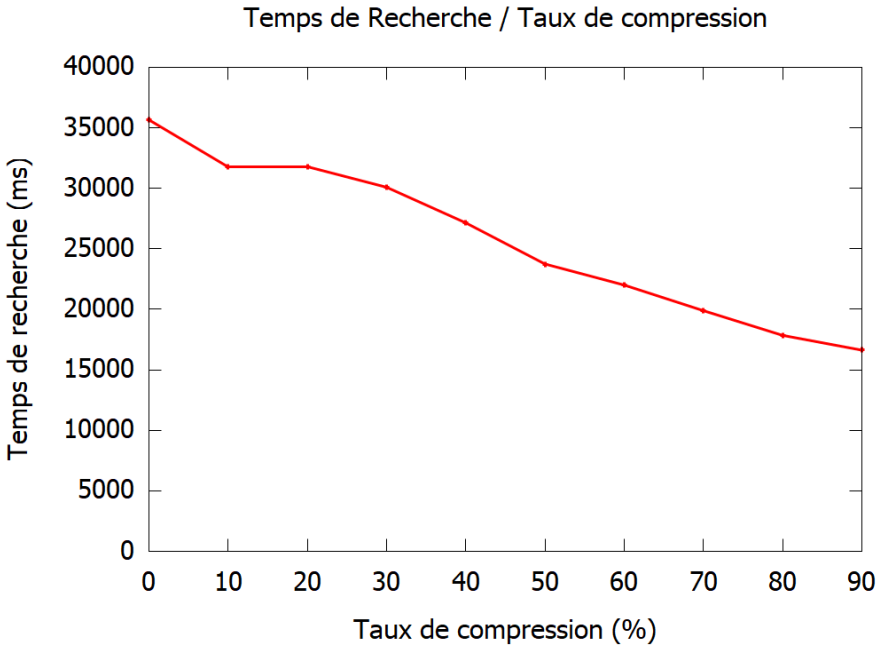


FIGURE 3.6 – Temps de recherche par rapport taux de compression

moins performants quand il s'agit de ce type d'image.

3.3 Méthodes d'indexation par approximation et les fonctions de hachage

Comme nous l'avons décrit dans l'état de l'art, il existe plusieurs méthodes d'indexation par approximation, telle que la méthode VA-FILE. D'après les travaux présentés dans la section état de l'art, nous avons choisi cette méthode parmi les autres, vu son point fort qui est le temps de recherche.

La méthode VA-FILE est composée de deux parties : 1. Construction d'un fichier qui contient tous les vecteurs de la base. 2. Application de la méthode de recherche séquentielle améliorée. Dans notre cas, nous avons utilisé la méthode de recherche séquentielle améliorée sans l'utilisation de la première partie de l'algorithme de VA-FILE. Ce choix est fait vu que dans notre cas, les points d'entrés de la première partie de l'algorithme sont des matrices. Ces matrices sont les descripteurs SIFT ou SURF, alors que l'algorithme prend comme entrée des points représentés sous forme cartésienne. Nous avons essayé d'adapter l'algorithme afin qu'il prenne en charge ces matrices, mais nous avons constaté une grande perte d'informations.

D'autre part, il existe d'autres méthodes permettant de compresser le vecteur des caractéristiques des images, comme les fonctions de hachage. Parmi les méthodes qui existent nous avons choisi la méthode la plus utilisée qui est la méthode LSH [2]. Cette méthode utilise plusieurs fonctions de hachage afin de compresser le vecteur des caractéristiques.

Par la suite, nous avons testé plusieurs combinaisons des méthodes décrites ci-dessous, afin de trouver la meilleure combinaison qui s'adapte avec les deux descripteurs SIFT et SURF, en gardant la précision du moteur de recherche. La Figure 3.7 montre l'architecture générale de l'approche proposée.

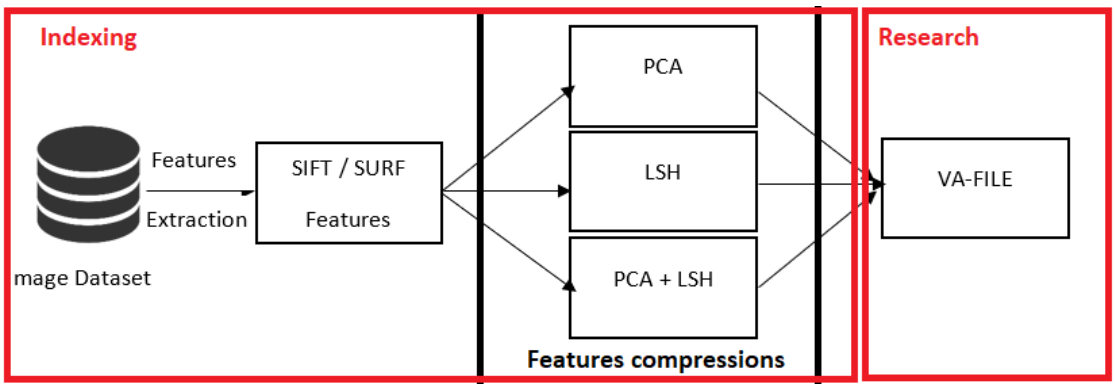


Figure 3.7 – Architecture générale de l'approche proposée

Comme nous le montre la Figure 3.7, l'extraction des caractéristiques des images est faite en utilisant les deux algorithmes SIFT et SURF. Par la suite, nous reprenons les résultats obtenus par l'approche de

Tableau 3.2 – Le temps de recherche et le temps d'indexation des différentes combinaisons appliquées sur la base d'image WANG

Méthode	T. Recherche (ms)	T. Indexation (ms)
PCA (70%)	168	460
PCA (70%) / VA-FILE	146	460
LSH	164	570
PCA (70%) / LSH	159	511
PCA (70%) / LSH / VA-FILE	132	511

réduction de dimension à base de l'algorithme de PCA, ainsi que la méthode LSH et une combinaison de ces deux méthodes. Dans notre cas, la méthode VA-FILE est utilisée comme méthode de recherche. Nous allons détailler par la suite les combinaisons possibles.

3.3.1 Résultats expérimentaux

3.3.1.1 Préparation des données

Nous avons proposé deux algorithmes PCA et LSH comme des méthodes de réduction de dimension des descripteurs. La méthode PCA est appliquée directement avec le taux de 70%. Par contre pour l'algorithme LSH nous avons utilisé 9 fonctions de hachage. Ces fonctions sont implémentées et utilisables avec la bibliothèque OpenCV [10]. Ces méthodes ont été testées sur plusieurs machines ainsi que différents systèmes d'exploitation. Afin d'évaluer la pertinence des résultats renvoyés par les descripteurs utilisés pour les différentes combinaisons des méthodes, nous avons utilisé les courbes Rappel/Précision comme métrique d'évaluation. De même, nous avons tracé les graphes relatifs au temps de recherche en fonction des combinaisons possibles.

Nos expériences ont été effectuées en utilisant plusieurs bases d'images : la base d'images Wang qui contient 7000 images. Une base d'images de 7200 images depuis Coil, 1 million d'images depuis Mirflickr et 16 millions d'images depuis ImageNet.

3.3.1.2 Résultats

Afin de montrer les différentes combinaisons testées ainsi que les résultats de chacune, nous avons tracé un tableau comparatif avec deux critères de comparaison : le temps de recherche et le temps de calcul des descripteurs comme indiqué dans le Tableau 3.2.

Dans un premier temps, nous avons réalisé les expérimentations en utilisant la base d'images WANG comme nous montre le Tableau 3.2. Nous avons trouvé que l'utilisation de la méthode VA-FILE apporte une amélioration au niveau du temps de recherche. En effet, nous remarquons un gain de 22 ms si nous utilisons la méthode PCA avec un taux de 70% avec VA-FILE par rapport la méthode PCA avec le même taux de 70%. Cependant, le temps d'indexation est le même, parce que comme nous avons expliqué

10. OpenCV : https ://opencv.org/

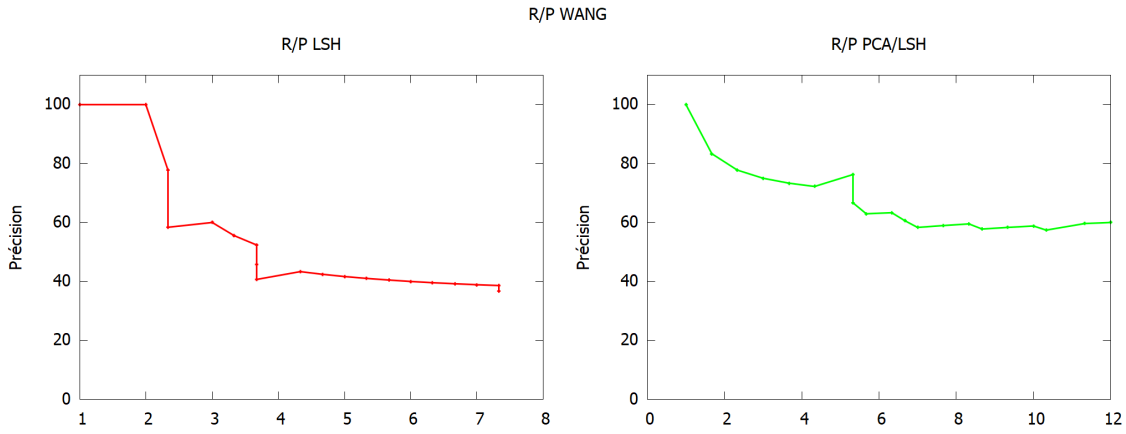


FIGURE 3.8 – Les courbes Rappel/Précision LSH (à gauche) PCA/LSH (à droite) de la base d'images WANG.

TABLEAU 3.3 – Le temps de recherche et le temps d'indexation des différentes combinaisons appliquées sur la base d'image ImageNet

Méthode	**T. Recherche (ms)**	**T. indexaton (ms)**
PCA (70%)	19561	56700
PCA (70%) / VA-FILE	19010	56700
LSH	19421	59812
PCA (70%) / LSH	19171	59004
PCA (70%) / LSH / VA-FILE	12423	59004

dans l'introduction de la section Section 3.3, VA-FILE est utilisée comme une recherche séquentielle améliorée.
La Figure 3.8 nous montre les courbes Rappel/Précision avec la base d'images WANG. Nous avons choisi aléatoirement trois images requêtes de la base, et nous avons lancé notre moteur de recherche. Le résultat obtenu nous a permis de tracer les courbes Rappel/Précision.

Comme nous montre la Figure 3.8 , l'utilisation de la méthode LSH dégrade l'information des descripteurs, ce qui affecte la précision du moteur de recherche. Par contre, la combinaison des deux méthodes PCA et LSH par ordre d'écriture ne dégrade pas l'information, parce que la sortie de l'algorithme PCA avec le taux de 70% est une matrice avec moins de valeurs comparé à la matrice de départ, donc la sortie de la méthode LSH est une matrice proche de la matrice d'entrée. Cette combinaison a permis de garder la même précision, mais avec un temps de recherche plus rapide.

Afin de valider notre approche, nous avons lancé les expérimentations avec la base d'images ImageNet, et nous avons essayé de tracer le même tableau ainsi que les mêmes courbes de Rappel/Précision avec la même philosophie.

Les résultats obtenus avec la base d'images ImageNet montrent toujours que la meilleure combinaison est l'utilisation de PCA et LSH simultanément comme méthodes de réduction de dimension. Ces techniques démontrent qu'au niveau de l'indexation, le temps de calcul devient important. En revanche,

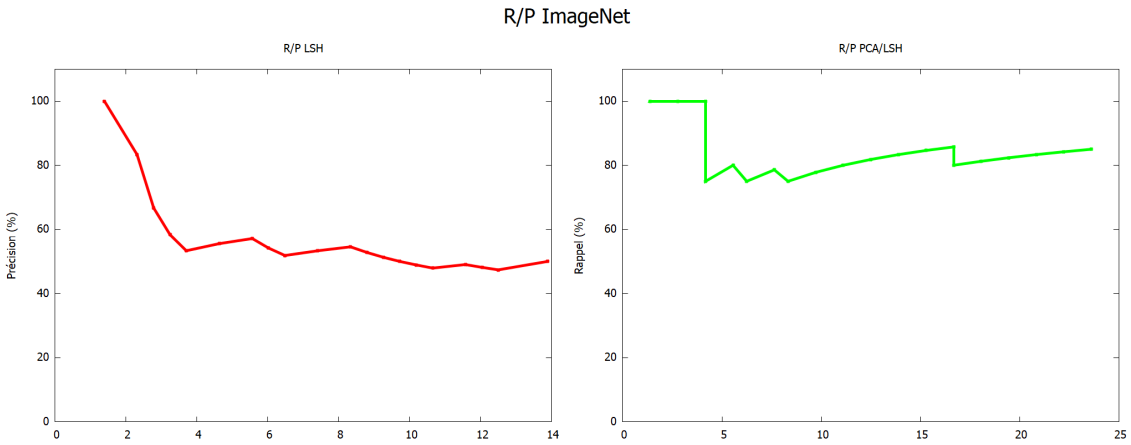


FIGURE 3.9 – Les courbes Rappel/Précision LSH (à gauche) PCA/LSH (à droite) de la base d'images ImageNet.

le temps de recherche est très rapide avec l'application de la méthode VA-FILE comme une technique de recherche avancée.

De même, les courbes Rappel/Précision de la base d'images ImageNet montrent et valident notre approche. Afin de montrer l'efficacité de notre approche, nous avons tracé un histogramme relatif au temps de recherche comme le montre la Figure 3.10.

3.3.1.3 Discussion

L'approche proposée dans cette section est basée sur une combinaison de deux méthodes de réduction de la dimension ainsi qu'une méthode de recherche séquentielle améliorée. Lors de l'utilisation de l'algorithme LSH, nous avons remarqué que cette méthode est rapide dans le cas où l'élément que nous voulons chercher apparait dans la première ou la deuxième fonction de hachage. Au-delà le temps de recherche augmente. Pour cette raison nous avons proposé une combinaison PCA/LSH afin de réduire la dimension des descripteurs ainsi que l'application de VA-FILE comme technique de recherche séquentielle améliorée. Le processus de réduction de dimension est lourd par rapport à la consommation des ressources matérielles. De même, il prend un temps important, ce qui sera ajouté au temps total de la phase d'indexation. Nous avons remarqué parfois que la sortie de l'algorithme de PCA avec le taux de 70% est une matrice de taille < 10. Par conséquent, si nous appliquons par la suite l'algorithme LSH, la sortie sera une matrice vide, ce qui pose un problème pour le moteur de recherche.

3.4 Conclusion

Dans ce chapitre, nous avons présenté un système d'indexation et de recherche des images. Notre système est composé de trois phases : une phase d'indexation, dans laquelle les descripteurs (SIFT ou SURF) sont calculés. La deuxième phase est la phase de compression basée sur une combinaison de deux méthodes, où nous appliquons l'algorithme PCA, qui prend un taux de compression des descripteurs

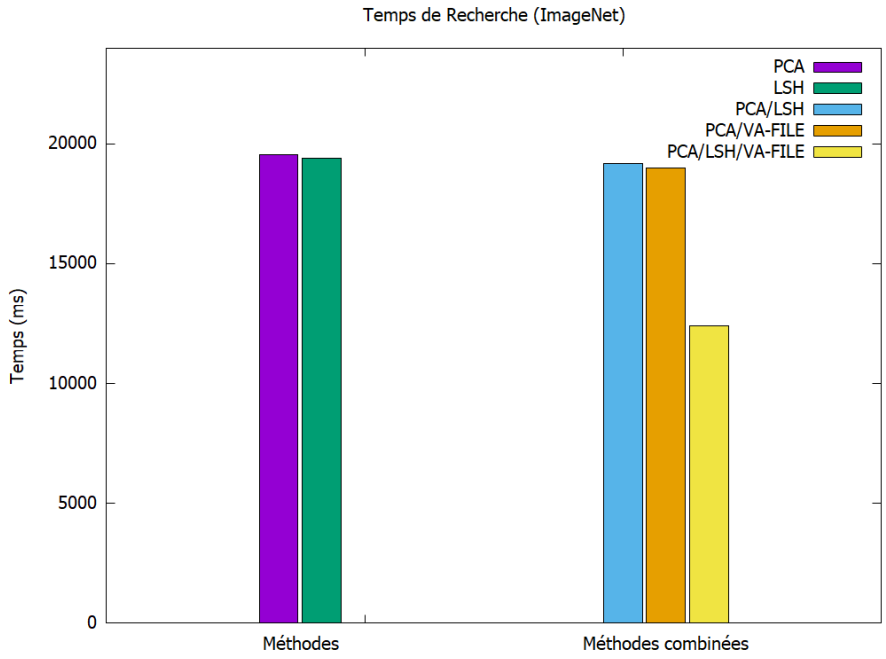


FIGURE 3.10 – Le temps de recherche des différentes méthodes appliquées sur la base d'images ImageNet.

comme un paramètre, ainsi que la méthode LSH. La troisième étape est la phase de recherche, où le système prend comme paramètre d'entrée une image requête. Le processus se poursuit par le calcul du descripteur (SIFT ou SURF) de cette image. Ensuite, nous appliquons la combinaison des deux algorithmes PCA et LSH à cette image. Finalement, le système lance la recherche à base de la méthode VA-FILE et affiche les images similaires à la requête.

Notre système d'indexation nous a permis de gagner du temps au niveau de la recherche d'un facteur allant de 30% à 50% ainsi que de l'espace de stockage avec un facteur allant de 20% à 30%.

Chapitre

4 Accélération de la recherche avec les techniques de stockage des descripteurs

« Tout le monde est un génie. Mais si vous jugez un poisson à sa capacité de grimper à un arbre, il vivra toute sa vie en croyant qu'il est stupide. »
— Albert Einstein

Sommaire

4.1 Introduction

Dans ce chapitre, nous allons proposer une méthode de recherche des images accélérée. Cette approche repose sur l'utilisation des techniques de stockage des descripteurs sous forme arborescence. Cette approche combine deux techniques proposées dans les chapitres Chapitre 6 qui utilise le principe des images clés et le Chapitre 3 avec la compression des caractérstiques, en y ajoutant une étape de prétraitement qui utilise le filtre Gaussien, ainsi qu'une représentation des descripteurs sous forme d'arbre binaire.

Les résultats de chapitre font l'objet d'une publication dans les proceeding de la conférence **Cloud-Tech 2017 : Cloud Computing and Big Data : Technologies, Applications and Security** dans : **Springer Link** [10].

4.2 Représentation des descripteurs

La méthode que nous proposons dans cette section pour accélérer le temps de recherche a été appliquée sur les vidéos. Cette méthode regroupe les deux méthodes précédentes plus un stockage des descripteurs sous forme d'un arbre binaire. Cette approche améliore à la fois le temps de recherche ainsi que l'espace de stockage. Les étapes principales de notre approche sont montrées dans la Figure 4.1. Cette approche est basée sur 4 sous-étapes : prétraitement, extraction des caractéristiques, compression

FIGURE 4.1 – Les pricipales étape de la méthodes proposé

et finalement la représentation sous forme d'arbre binaire.

4.2.1 Prétraitement

L'entrée de cette étape est un ensemble des vidéos. D'abord, il faut découper la vidéo en un ensemble des images individuelles. Ensuite nous appliquons le filtre Gaussien à toutes ces images, afin de réduire le nombre des points d'intérêts qui sont fournis par l'algorithme SIFT et SURF. Nous avons appliqué différents filtres à ces images. Après plusieurs expérimentations, nous avons trouvé que le meilleur filtre qui s'adapte avec l'algorithme SIFT et SURF est le filtre gaussien. Ce filtre utilise une fonction gaussienne pour calculer la transformation à appliquer sur chaque pixel de l'image. Le processus de prétraitement est montré dans Algorithme 4.1.

Algorithme 4.1 Prétreatment

Entrée : S_videos : un ensemble des vidéos ;
Sortie : $jpeg\_file$: un ensemble des images ;

```
jpeg_file ← ∅ ;
pour tout video_i ∈ S_videos faire
    Lire video_i ;
    Nb_Images ← Frame_NB_Video(video_i) ;        ▷ Le nombre des images de la vidéo i
    pour j := 1 to Nb_Images faire
        Image ← Get_Current_Image(video_i, j) ;   ▷ récupurer une nouvelle image de la vidéo
        Image_GB ← GaussianBlur(Image) ;
        jpeg_file ← jpeg_file ∪ Image_GB ;
    fin pour
fin pour
retourner jpeg_file ;
```

4.2.2 Extraction des caractéristiques

Il s'agit de l'extraction des caractéristiques de l'image, dans cette étape nous avons utilisé les deux algorithmes SIFT et SURF comme des descripteurs. Nous avons aussi utilisé la première approche proposée dans le Chapitre 3.

Le résultat de cette étape est un résumé de vidéo (des images clés). Ces images seront utilisées comme paramètres d'entrée à l'étape suivante.

4.2.3 Compression

Dans cette étape nous avons utilisé la deuxième approche décrite dans le Chapitre 3, basée sur une PCA modifiée. Cette méthode permet de réduire la dimension des descripteurs des images. Nous avons appliqué le taux de compression qui correspond à 70% en se basant sur les résultats obtenus dans le Chapitre 3.

4.2.4 Représentation des images clés

Une fois les images clés ainsi que leurs descripteurs extraits et leurs tailles réduites, nous proposons de les stocker dans une structure de stockage sous la forme d'un arbre binaire. Cette structure permet d'accélérer la phase de recherche car elle est très simple à gérer comparée à d'autres structures. Aussi, au niveau implémentation, les arbres binaires sont souvent représentés par les tableaux qui représente l'avantage de fournir des accès rapides. La Figure 4.2 nous montre un exemple d'un arbre binaire. Tous les nœuds qui sont à gauche sont inférieur à la racine, et ceux qui sont à droite sont supérieur à la racine. Les arbres binaires contiennent généralement des valeurs comparables. Cependant, les descripteurs SIFT et SURF que nous proposons de stocker dans les arbres binaires ont des valeurs sous forme des matrices qui sont difficiles à comparer.
Pour comparer deux matrices SIFT ou SURF nous proposons d'utiliser une mesure de similarité où le

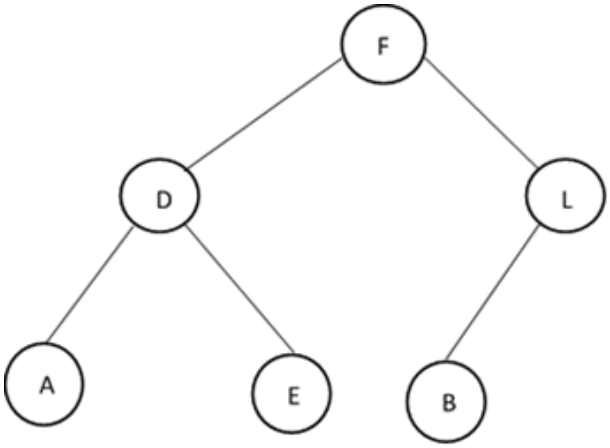


FIGURE 4.2 – Un exemple d'un arbre binaire

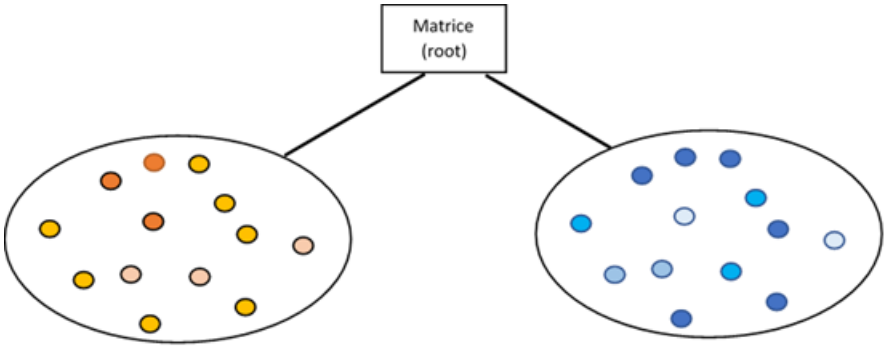


FIGURE 4.3 – Résultat de la première itération de Algorithme 4.2

résultat est une valeur normalisée entre 0 et 1. En effet, nous avons défini deux intervalles : [0 0.5] pour chaque nœud inférieur à la racine, et [0.6 1] pour chaque nœud supérieur à la racine. Algorithme 4.2 explique cette démarche.

Dans Algorithme 4.2, d'abord on choisit la racine avec la fonction random, ensuite nous calculons la distance entre la racine et le descripteur i en utilisant une des mesures de similarité. Après on décide en fonction de la distance si on place le nœud à gauche ou à droite de la racine. L'application de cet algorithme est montrée dans la Figure 4.3. Nous appliquons Algorithme 4.2 jusqu'à l'obtention d'un arbre binaire complet. L'un des problèmes des arbres binaires sont les situations où tous les nœuds sont à gauche ou à droite. Dans ce cas il est nécessaire d'appliquer un équilibrage d'arbre binaire comme nous le montre la Figure 4.4. Nous proposons d'appliquer le système d'équilibrage jusqu'à l'obtention d'un arbre binaire complet quand l'arbre n'est pas équilibré.

Algorithme 4.2 *Binary_Tree_Generation*(*S_Features*)

Entrée : *S_Features* : les caractéristiques des images
Sortie : *Table_Root* : un tableau qui contient la position des noeuds.

```
root ← choix aléatoire d'une racine
Table_Root ← root ;
Features_Left ← ∅ ;
Features_Right ← ∅ ;
Nb_features ← length(S_Features) ;
si (Nb_features ≠ 1) alors
    pour i := 1 to Nb_features faire
        si (feature_i ≠ root) alors
            Dist ← Distance(feature_i, root) ;
            si (Dist ⩽ 0.5) alors
                Features_Left ← Features_Left ∪ feature_i ;
            sinon
                Features_Right ← Features_Right ∪ feature_i ;
            fin si
        fin si
    fin pour
    Binary_Tree_Generation(Features_Left) ;
    Binary_Tree_Generation(Features_Right) ;
sinon
    Break ;
fin si
retourner Table_Root ;
```

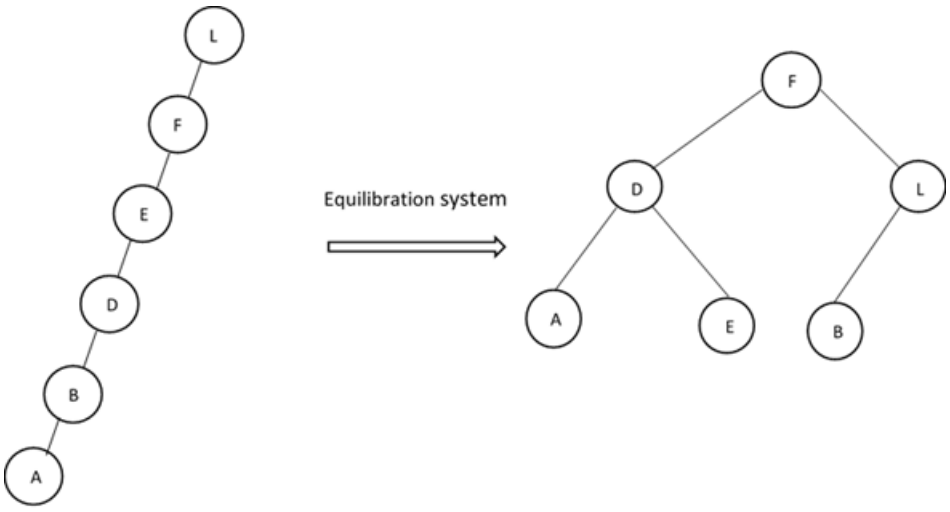


FIGURE 4.4 – Système d'équilibrage d'un arbre binaire

4.3 Résultats expérimentaux

4.3.1 Données

Nous avons utilisé la même base vidéos utilisée pour la première méthode décrite au Chapitre 6, c.a.d.

- Une base de vidéos qui contient 11 catégories, chaque catégorie contenant de 4 à 6 séquences de vidéos provenant de ChangeDetection [11].
- Une base de vidéos contient 7 million de vidéos obtenues depuis Youtube-8M [12], cette base contient environ 45000 d'heures de vidéos ainsi que 4716 classes.

4.3.2 Matériel utilisé

Cette approche a été testée sur le Cluster de la Faculté Polytechnique de l'Université de Mons [13] ainsi que sur des machine Cloud de Microsoft Azure [14], et Amazon Web service [15], avec déférents systèmes d'exploitation et déférentes configurations.

4.3.3 Résultats

La Figure 4.5 compare le temps de recherche de l'approche basée sur l'arbre binaire et l'approche classique qui utilise un stockage classique des descripteurs dans de simples fichiers textes. Nous remarquons que le meilleur temps de recherche a été obtenu avec les arbres binaires. En effet, l'utilisation des arbres binaires comme structure de stockage a permis une accélération de 30% à 38%.

Nous avons comparé notre approche avec la méthode R-Tree, on constate que le meilleur temps de recherche a été obtenu avec les arbres binaires. En effet, l'utilisation des arbres binaires comme structure de stockage a permis une accélération de 30% à 38% (Figure 4.6). De même nous avons comparé notre approche avec la méthode B-Tree, en Effet B-tree se base sur les arbres binaires, mais l'avantage dans notre approche est la manière de représenter l'arbre binaire au niveau implémentation, ou nous avons utilisé la représentation la plus basique qui est un tableau. Les résultats montrent l'efficacité de notre approche en terme temps de recherche comme nous montres la Figure 4.7.

4.3.4 Discussion

La nouvelle représentation des descripteurs sous forme d'arbres binaires nous a permis d'améliorer les performances des deux premières méthodes d'accélération de la recherche proposées au chapitre Chapitre 3. Notre système prend comme paramètre d'entrée une image requête. Il calcule le descripteur de cette image, puis il applique la méthode de PCA pour compresser ce descripteur. Finalement, le système compare l'image requête avec l'ensemble des images clefs, mais en effectuant un parcours

11. ChangeDetection : http ://www.changedetection.net/
12. Youtube8M : https ://research.google.com/youtube8m/
13. Cluster : https ://www.ig.fpms.ac.be/fr/content/cluster-de-calcul-ig
14. Azure : https ://portal.azure.com
15. Amazon : https ://console.aws.amazon.com

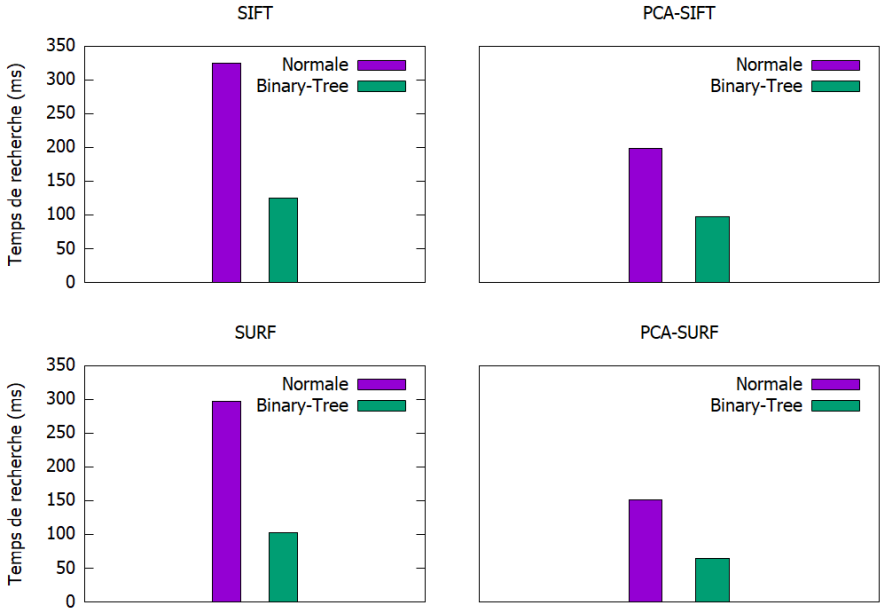


FIGURE 4.5 – Les temps de recherche avec arbre binaire et avec une recherche normale

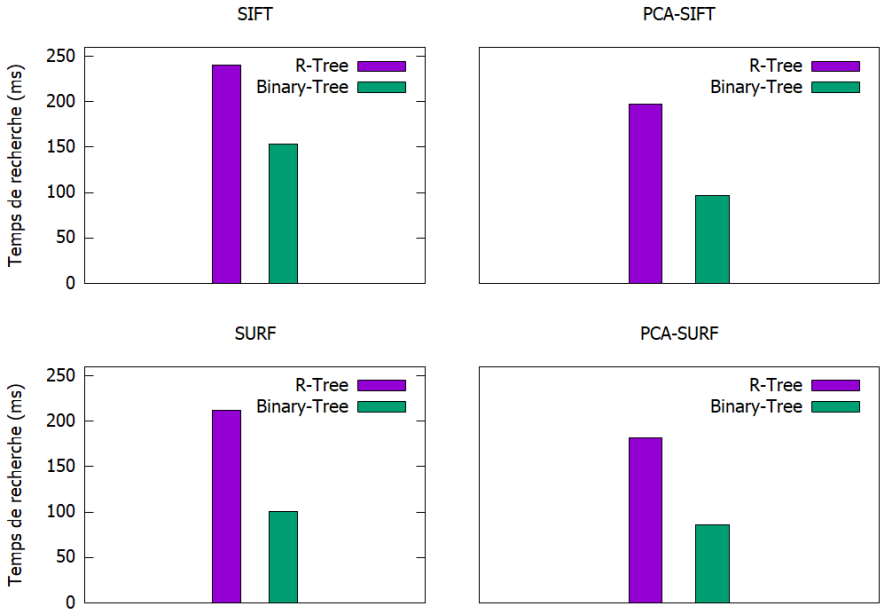


FIGURE 4.6 – Le temps de recherche avec arbre binaire et R-tree

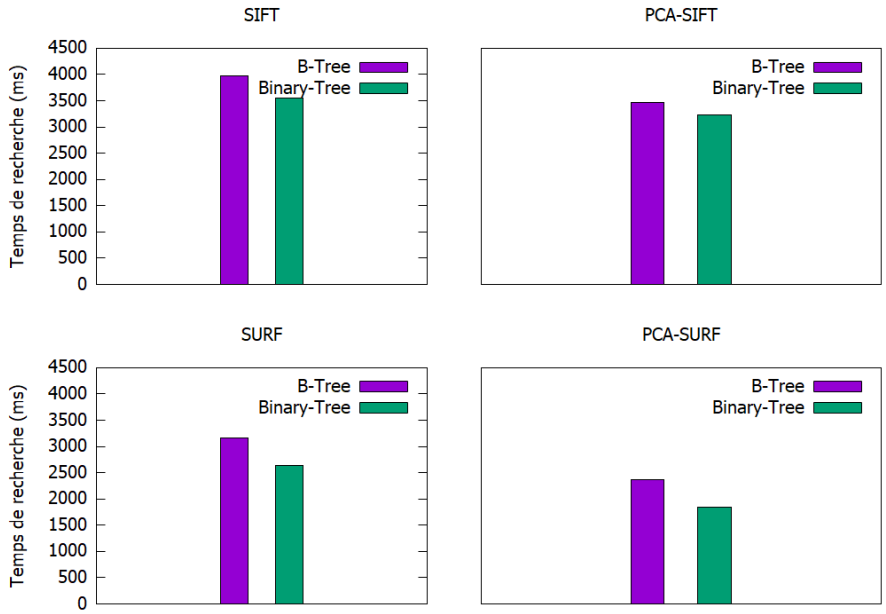


FIGURE 4.7 – Le temps de recherche avec arbre binaire et B-Tree

TABLEAU 4.1 – Comparaison entre la méthode VA-FILE et les arbres binaires

Technique	**Complexité**	**Temps de recherche (ms)**
VA-FILE	$O(N)$	151
Arbre binaire	$O(Log(N))$	77

arborescent pour la recherche au lieu de réaliser une recherche séquentielle.

Dans le Chapitre 3, nous avons présenté une méthode qui a permis d'accélérer la phase de recherche. C'est une technique de réduction de dimension de base, par contre dans notre cas, nous utilisons la méthode VA-File comme méthode de recherche séquentielle améliorée. Afin de trouver la meilleure approche à appliquer, nous avons tracé un tableau pour comparer les arbres binaires et **VA-FILE**. Cette comparaison est présentée sur le Tableau 4.1.

En termes de complexité, la méthode de recherche avec les arbres binaires est meilleure. De même, pour le temps de recherche, l'approche à base des arbres binaires est rapide par rapport à la méthode VA-File. Par contre, la complexité de la méthode VA-FILE est définie en fonction de la première requête, parce qu'à travers cette requête le vecteur qui contient les images les plus similaires est construit. Ce qui rend la complexité dépendante d'un nombre N, qui est le nombre des images que le moteur de recherche doit afficher.
Le choix final dépend du nombre des images similaires à l'image requête. D'après la littérature, le moteur de recherche doit afficher les Top 20. Dans ce cas, l'utilisation des arbres binaires est plus intéressante. Par contre, au-delà de 50 images en sortie, l'utilisation de la méthode de recherche VA-File devient indispensable à condition de lancer au moins une requête.

4.4 Conclusion

Nous avons présenté dans ce chapitre un système d'indexation des vidéos amélioré. Ce système est composé de quatre étapes : on commence par une phase de prétraitement, où les vidéos sont segmentées en images individuelles. Après, nous appliquons le filtre gaussien sur l'ensemble des images afin de réduire les points d'intérêts. La deuxième étape consiste à extraire les descripteurs des images, par la suite on génère les images clefs ou le résumé de vidéo pour chaque vidéo. La troisième étape est l'étape de compression des descripteurs en utilisant la méthode PCA. La dernière étape consiste à stocker les descripteurs sous une forme arborescente en utilisant les arbres binaires.

Les résultats obtenus montrent que notre système d'indexation des vidéos amélioré permet d'accélérer la phase de recherche d'un facteur allant de 30% à 40%.

Chapitre

5

Indexation par les réseaux de neurones profonds

« Real stupidity beats artificial intelligence every time. »
— Terry Pratchett

Sommaire

5.1 Introduction

Jusqu'à maintenant nous avons proposé différentes contributions permettant de réduire la dimension des descripteurs afin d'accélérer la phase de recherche mais en préservant la précision. Dans ce chapitre, nous proposons l'utilisation des réseaux de neurones profonds comme moyen d'extraction des descripteurs des images ainsi que les méthodes de réduction de la dimmensionalité appliquées sur ces dernières. Nous allons présenter trois composantes de notre système d'indexation et de recherche qui sont : l'entraînement, l'indexation et la recherche. D'abord, nous allons utiliser les différentes architectures connues sans entraînement. L'idée est de prendre les poids de la plus grande base d'images ImageNet, puis nous calculons les descripteurs des images pour chaque architecture, afin de trouver la meilleure. Le choix se fera à base des courbes Rappel/Précision. Les résultats obtenus dans cette première partie montrent l'importance de faire un entraînement. Un entraînement dans notre cas consiste à prendre une architecture entrainée sur la base d'image ImageNet, récupérer ces poids puis changer la dernière couche en fonction de notre base d'images et entraîner à nouveau. Afin d'améliorer les résultats en termes de précision, l'entraînement a été effectué sur plusieurs bases d'images avec neuf architectures, pour comparer ces dernières, dans le but de trouver la meilleure en terme de précision. Les résultats obtenus montrent une amélioration de la précision jusqu'à 99,5%. Par la suite, nous proposons une amélioration de la précision en combinant les meilleures architectures retenues. La combinaison de plusieurs architectures a ainsi permis d'obtenir une précision de 100% mais le temps de recherche a été multiplié par trois. À la fin de ce chapitre, nous présentons deux contributions : $\mathcal{RFB}$ (RMAC Features Extraction Before) et $\mathcal{RFA}$ (RMAC Features Extraction After) liées à la réduction de la dimensionalité des descripteurs, et nous comparons notre méthode par rapport d'autres travaux. Les différentes étapes de ce chapitre sont montrées dans la Figure 5.1

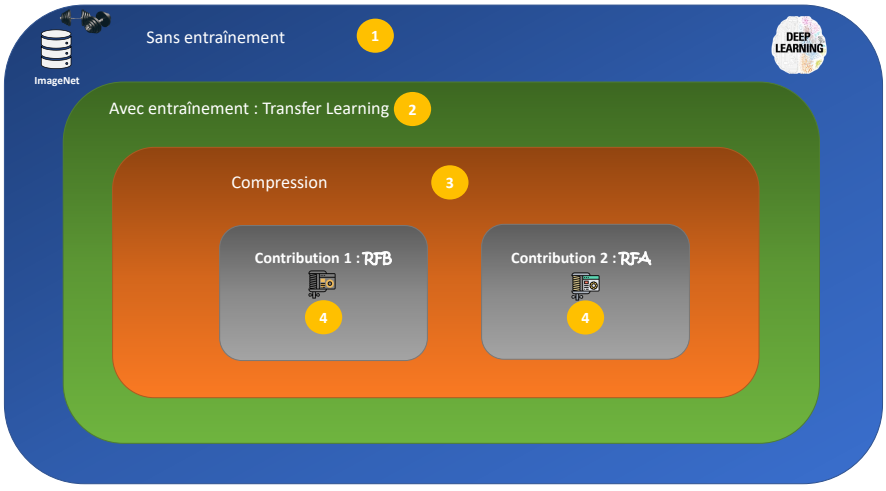


Figure 5.1 – Le résumé des différentes étapes d'extraction des caractéristiques avec les réseaux de neurones profonds

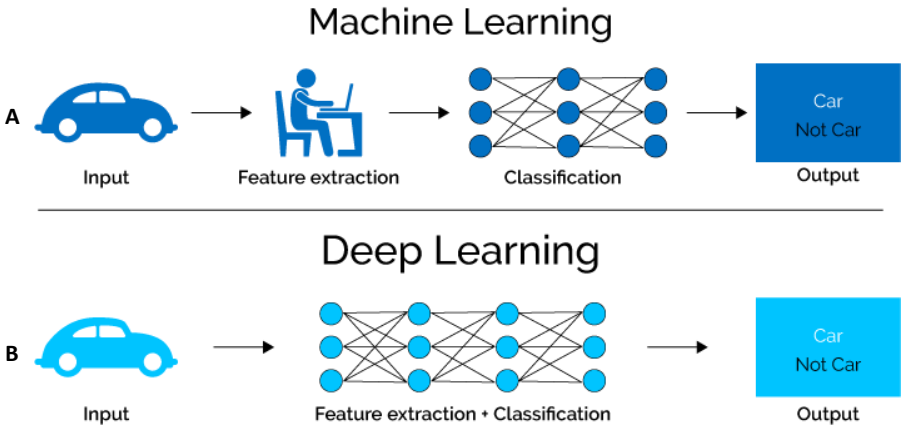


FIGURE 5.2 – La différence entre le Machine Learning et le Deep Learning [102]

5.2 Pourquoi les réseaux de neurones profonds CNN ?

L'utilisation des réseaux de neurones profonds vient du fait que ces derniers ont montré leur efficacité dans plusieurs domaines (Santé, Agriculture, Véhicules autonomes, etc.). Les auteurs dans [32] utilisent les CNN qui représentent une architecture basée sur les réseaux de neurones profonds afin de prédire la réponse à la tumeur liée au cancer des seins. Les résultats obtenus montrent l'efficacité d'utiliser le Deep Learning dans un domaine sensible. D'autre part, les auteurs dans [4] comparent les arbres de décision au réseaux de neurones profonds pour les voitures autonomes. Les résultats obtenus dans cet article montrent que les arbres de décision sont meilleurs pour la classification si le temps de réponse n'est pas nécessaire ce qui n'est pas le cas pour les voitures où le temps de réagir est très importants. D'autres part, l'avantage du Deep Learning est que la partie extraction des caractéristiques est implémenté dans l'architecture du réseau de neurones comparant aux de méthodes classiques Machine Learning, où cette partie est faite en utilisant d'autres algorithmes tel que SIFT, SURF, etc. La Figure 5.2 nous montres la différence entre le Machine Learning et le Deep Learning.

5.2.1 Motivations

Notre problématique liée aux moteurs de recherche rentre dans la thématique de la classification des images ou vidéos. Une classification consiste à entrainer un système avec une base d'images ou de vidéos contenant au moins deux classes. Le but est de rendre le système capable de prédire la classe d'un fichier d'entré (image ou vidéo). Une fois que le système est entraîné, nous passons à l'étape d'extraction des caractéristiques. Dans notre cas, c'est au niveau de la dernière couche que ça se fait. Le résultats est stocké sous forme d'un fichier pour chaque images.

5.2.2 Données utilisées

Nous avons effectué nos expériences en utilisant plusieurs bases d'images comme nous montre le Tableau 5.1.

- Le benchmark GHIM-10K [16] est une base d'images typique pour les systèmes d'indexation et de recherche multimédia. Cette base contient 10000 images réparties en 100 catégories comme le coucher de soleil, les voitures, etc.
- De même, le benchmark Corel-10K [17] est une base d'images contenant 10000 images. Ces images sont regroupées sous 20 catégories.
- Paris-6K [18], une base d'images pour la validation, qui contient 60000 images réparties sur 15 catégories.

Tableau 5.1 – Les différentes bases d'images utilisées pour l'extraction des caractéristiques avec le Deep Learning

Base d'images	Nombre d'images	Nombre catégories	Résolution
GHIM-10K	10000	100	192 x 128
Corel-10K	10000	20	300 x 400
Paris-6K	60000	15	-

5.3 Extraction des descripteurs sans entraînement

Dans cette partie, nous avons utilisé les architectures les plus connues, afin de lancer le processus d'extraction des caractéristiques des images (indexation). Le Tableau 5.2, décrit les différentes architectures entrainées sur la base d'images ImageNet et leurs classements.

Les différentes architectures utilisées dans ce chapitre sont : **MobileNet, VGG16,VGG19, ReNet50, InceptionV3**. Le choix de ces architectures est basé sur plusieurs expérimentations avec les bases d'images citées dans la section Section 5.2.2. Sachant que chaque architecture est adaptée à un problème connu. Selon les auteurs dans [55], les architectures choisies sont les meilleures qui s'adaptent à notre problématique.

5.3.1 Environnement de travail

Pour lancer le processus d'extraction des caractéristiques, nous avons utilisé plusieurs machines. Parmi ces machines :

1. Le cluster de la Faculté Polytechnique de l'Université de Mons [20] avec la configuration suivante :
   - un processeur de Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz avec 16 cœurs physiques et 32 cœurs logiques ;

16. GHIM-10K : http ://www.ci.gxnu.edu.cn/cbir/Dataset.aspx
17. Corel-10K : http ://www.ci.gxnu.edu.cn/cbir/Dataset.aspx
18. Paris-6K : http ://classif.ai/dataset/paris-retrieval/
19. https ://keras.io/applications
20. Cluster : https ://www.ig.umons.ac.be/fr/content/cluster-de-calcul-ig

Tableau 5.2 – Classement des différentes architectures selon la précision (Top 1, Top 5). [19]

Model	La taille	Top-1 précision	Top-5 précision	Paramètres	Profondeur
Xception	88 MB	0.79	0.945	22 910 480	126
VGG16	528 MB	0.713	0.901	138 357 544	23
VGG19	549 MB	0.713	0.900	143 667 240	26
ResNet50	98 MB	0.749	0.921	25 636 712	-
ResNet101	171 MB	0.764	0.928	44 707 176	-
ResNet152	232 MB	0.766	0.931	60 419 944	-
ResNet50V2	98 MB	0.760	0.930	25 613 800	-
ResNet101V2	171 MB	0.772	0.938	44 675 560	-
ResNet152V2	232 MB	0.780	0.942	60 380 648	-
InceptionV3	92 MB	0.779	0.937	23 851 784	159
InceptionResNetV2	215 MB	0.803	0.953	55 873 736	572
MobileNet	16 MB	0.704	0.895	4 253 864	88
MobileNetV2	14 MB	0.713	0.901	3 538 984	88
DenseNet121	33 MB	0.750	0.923	8 062 504	121
DenseNet169	57 MB	0.762	0.932	14 307 880	169
DenseNet201	80 MB	0.773	0.936	20 242 984	201
NASNetMobile	23 MB	0.744	0.919	5 326 716	-
NASNetLarge	343 MB	0.825	0.960	88 949 818	-

   - 64 Gb de Ram ;
   - quatre cartes graphiques GeForce GTX 1080 avec 12 Gb de Ram ;

2. Une machine réservée sur Goolge cloud [21] avec la configuration suivante :
   - un processeur de Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz avec 6 cœurs physiques et 12 cœurs logiques ;
   - 32 Gb de Ram ;
   - une carte graphique K80 12 Gb de Ram ;

Les librairies utilisées sont les suivantes :

- Tensorflow 1.9 compilé en Graphic processing Unit (GPU) ;
- Keras version 2.2.0 ;
- Theano version 0.9.0 ;
- Python version 3.5.

Python est principalement utilisé par la communauté de la data science, **Tensorflow** est une librairie open source proposée par Google, **Keras** pour la simplification de la syntaxe de **Tensorflow**, **Theano** est utilisée pour les deux méthodes $\mathcal{RFB}$ et $\mathcal{RFA}$.

5.3.2 Résultats expérimentaux

Comme tout système d'indexation et de recherche CBIR, la validation est réalisée en utilisant les deux métriques Rappel et Précision. Dans notre cas, nous avons choisi 5 classes aléatoirement. Pour chaque classe, nous avons trois images requêtes.

21. Google Cloud : https ://cloud.google.com/

Dans cette partie, nous avons lancé le processus d'extraction des descripteurs sans la phase d'entraînement. Afin de comparer les différentes architectures, nous avons calculé le temps de recherche ainsi que la précision moyenne.

La précision moyenne MaP est calculée comme indiqué dans l'équation Équation (5.1) :

$$\sum_{i=0}^{n} \frac{P_i}{n} \tag{5.1}$$

— P_i : la précision de l'image i

— n : le nombre des images en sorties

Le Tableau 5.3 nous montre la précision moyenne ainsi que le temps de recherche des 500 images similaires à nos images requêtes.

Tableau 5.3 – Le temps de recherche et le MaP de notre moteur de recherche avec l'architecture **VGG16** sans entraînement

Classe	N° image	T. R (s)	MaP (%)
0	367	0,036	**86,26**
	400	0,029	81,47
	168	0,028	44,68
2	1057	0,027	30,08
	1320	0,026	47,28
	1250	0,026	48,79
3	1742	0,027	74,08
	1820	0,026	**94,45**
	1703	0,026	72,21
5	2917	0,027	51,95
	2914	0,026	49,58
	2650	0,026	**25,52**
7	3527	0,026	51,61
	3493	0,028	49,09
	3608	0,026	60,01

5.3.3 Résultats

Les résultats obtenus dans cette section montrent que notre moteur de recherche est moins précis, vu que la précision varie entre 25% et 86%. Cette précision est moins bonne parce que dans cette partie nous n'avons pas procédé à la phase d'entraînement. De même, pour les autres architectures, la précision de notre moteur est comprise entre 25% (minimum) et 94% (maximum).

5.3.4 Discussion

Les résultats des expérimentations montrent l'importance de la phase d'entraînement. Dans notre première approche, nous avons utilisé les poids egénérés par un apprentissage réalisé par la base

Tableau 5.4 – Comparaison entre les différentes architectures pendant la phase d'entraînement

Modèle	Loss	Acc	Val Loss	Val Acc
MobileNet	0, 6	99, 96	0, 0258	98, 83
VGG16	10	97, 05	0, 1077	96, 88
VGG19	15	96, 42	0, 2722	91, 41
ResNet50	**0,4**	**99,97**	**0,0538**	**97,66**
InceptionV3	1, 4	99, 66	0, 0267	99, 22
Xception	0, 024	99, 56	0, 0304	99, 61
DenseNet21	0, 005	99, 96	0, 0023	100
DenseNet169	0, 010	99, 81	0, 0524	99, 22
DenseNet201	0, 007	99, 93	0, 0372	98, 44

ImageNet. Cette dégradation de la précision est due au faite que nos bases d'images contiennent des classes différentes de ImageNet. Pour cette raison, le réseau de neurones n'est pas capable de classifier les images. Une phase d'entraînement est importante afin de définir et préciser les classes de nos bases d'images. Dans la Section 5.4, nous allons présenter une approche avec la phase d'entraînement.

5.4 Extraction des descripteurs avec entraînement

Dans cette partie, nous avons utilisé les mêmes architectures mais avec entraînement. La technique utilisée dans ce cas est le Fine Tuning, qui consiste à récupérer les poids entrainés par la base d'images ImageNet, mais en modifiant la dernière couche de notre architecture. Cette étape permet de changer le nombre de classes de notre réseau de neurones.

5.4.1 Choix des paramètres

Pour lancer l'entraînement, nous avons divisé notre base d'images en deux sous-parties : une partie pour l'entraînement avec un taux de 70%, et 30% pour la partie test. Nous avons utilisé les paramètres suivants :

- EarlyStopping avec Learning rate=0.00001, patiente=3, ces paramètres permettent d'arrêter l'entraînement si le réseau de neurones trouve que la précision (Accuracy) se dégrade après trois pas (patiente) ;
- le nombre d'époques pour l'entraînement est 20, avec un batch size de 16.

Le choix des paramètres était basé sur plusieurs expérimentations, nous avons remarqué que la précision ne s'améliore pas au bout de 20 époques. De même, le batch size est un paramètre qui dépend de la puissance de la carte graphique utilisée. Ce paramètre permet de lire plusieurs images en parallèle dans la limite possible de la taille de la RAM du GPU. Le EarlyStopping est utilisé pour éviter le problème de sur apprentissage.

Le processus d'entraînement est lent et nécessite plusieurs heures. Cette opération dépend aussi du matériel utilisé comme expliqué dans la Section 5.7. Le résultat de l'entraînement est montré sur le Tableau 5.4.

TABLEAU 5.5 – Le temps de recherche (secondes) de notre moteur de recherche avec les différentes architectures

Classe	N° image	VGG16	VGG19	ResNet50	InceptionV3	MobileNet
0	367	0,029	0,046	0,031	0,032	0,030
	400	0,028	0,031	0,032	0,032	2,71
	168	0,027	0,032	0,032	0,032	0,029
2	1057	0,305	0,032	0,030	0,030	0,031
	1320	0,027	0,029	0,033	0,033	0,034
	1250	0,027	0,032	0,032	0,033	0,033
3	1742	0,030	0,031	0,030	0,028	0,032
	1820	0,029	0,033	0,031	0,031	0,033
	1703	0,028	0,033	0,031	0,032	0,031
5	2917	0,028	0,033	0,029	0,030	0,033
	2914	0,028	0,032	0,033	0,032	0,034
	2650	0,027	0,032	0,033	0,034	0,030
7	3527	0,026	0,033	0,031	0,030	0,030
	3493	0,026	0,030	0,032	0,032	0,033
	3608	0,026	0,030	0,032	0,034	0,033

Les résultats obtenus durant la phase d'entraînement montrent l'importance de lancer ce processus. Les neufs architectures donnent un taux de précision entre 96% et 99%. Nous avons trouvé que la meilleure architecture en terme de la précision est **ResNet50**. Par contre, ceci n'est pas un critère de choix pour notre moteur de recherche, donc nous avons utilisé la précision moyenne comme critère de sélection.

5.4.2 Résultats expérimentaux

Comme tout système d'indexation et de recherche CBIR, il faut le valider en utilisant les deux métriques Rappel et Précision. Dans notre cas, nous avons choisi 5 classes aléatoirement, où pour chaque classe nous avons trois images requêtes.

Selon l'état de l'art[73, 104], pour un système d'indexation et de recherche par le contenu, nous calculons les deux métriques à base de 20 images de sortie. Dans notre cas, après avoir effectué un entraînement, la moyenne de la précision de ces 20 images était à 100% pour n'importe quelle image. Donc, nous avons relancé notre moteur de recherche avec 50 images comme résultat. Le MaP est toujours à 100%, de même pour le **Top 50**, **Top 100**, **Top 200**, **Top 300**, **Top 400**.

Comme notre base d'images contient 500 images par classe, nous avons décidé de lancer notre de moteur recherche avec le Top 500.

Le Tableau 5.5 montre le temps de recherche (T.R) en secondes avec les différentes architectures que nous avons implémenté. De même, le Tableau 5.6 montre le MaP de ces architectures.

Tableau 5.6 – La précision moyenne de notre moteur de recherche avec avec les différentes architectures

Classe	N° image	VGG16	VGG19	ResNet50	InceptionV3	MobileNet
0	367	99,995	99,999	**100**	**100**	**100**
	400	99,995	99,999	**100**	**100**	**100**
	168	99,995	99,999	**100**	**100**	**100**
2	1057	99,943	99,749	99,999	99,992	99,996
	1320	99,924	97,020	99,999	99,992	99,996
	1250	99,939	99,741	99,99	99,992	99,996
3	1742	100	100	100	100	100
	1820	100	100	100	100	100
	1703	100	100	100	100	100
5	2917	99,964	99,766	99,974	99,965	99,986
	2914	99,964	99,757	99,975	99,966	99,987
	2650	99,965	99,738	99,974	99,956	99,986
7	3527	100	99,991	100	99,999	99,999
	3493	99,999	99,988	99,999	99,999	99,997
	3608	100	99,992	100	99,999	99,999

Les résultats obtenus pour les différentes architectures nous montrent l'avantage de lancer un entraînement. Nous avons trouvé que les meilleures architectures au niveau de la précision sont : **ResNet50**, **InceptionV3** and **MobileNet**.

Afin de comparer au mieux les différentes architectures, nous avons tracé les graphes Rapell/Précision mais cette fois par classe (la moyenne de trois images). Les résultats seront présentés dans la Section 5.4.3.

5.4.3 Résultats

Dans cette partie, nous allons présenter les différents graphes obtenus, dans le but de comparer les trois architectures : ResNet50, InceptionV3 et MobileNet. La valeur minimale de la précision est 99, 9 %.

La Figure 5.3 nous montre la précision moyenne de trois requêtes. La précision pour cette classe N° 0 est toujours à 100%.

Comme nous montre la Figure 5.4, la précision est dégradée à 99,4% pour l'architecture InceptionV3, et 99,6% pour les deux autres architectures.

Pour la classe N° 3, la précision moyenne est à 100% comme nous montre la Figure 5.5 pour les trois architectures.

La Figure 5.6 nous montre la précision moyenne de la classe N° 5. Cette précision baisse à 99% sur les 500 images retournées par le moteur de recherche. De même, pour la classe N° 7, la précision descend à un taux de 99,93% pour l'architecture ResNet50 et à 99,8% pour les deux autres architectures. Ces résultats sont montrés sur la Figure 5.7.

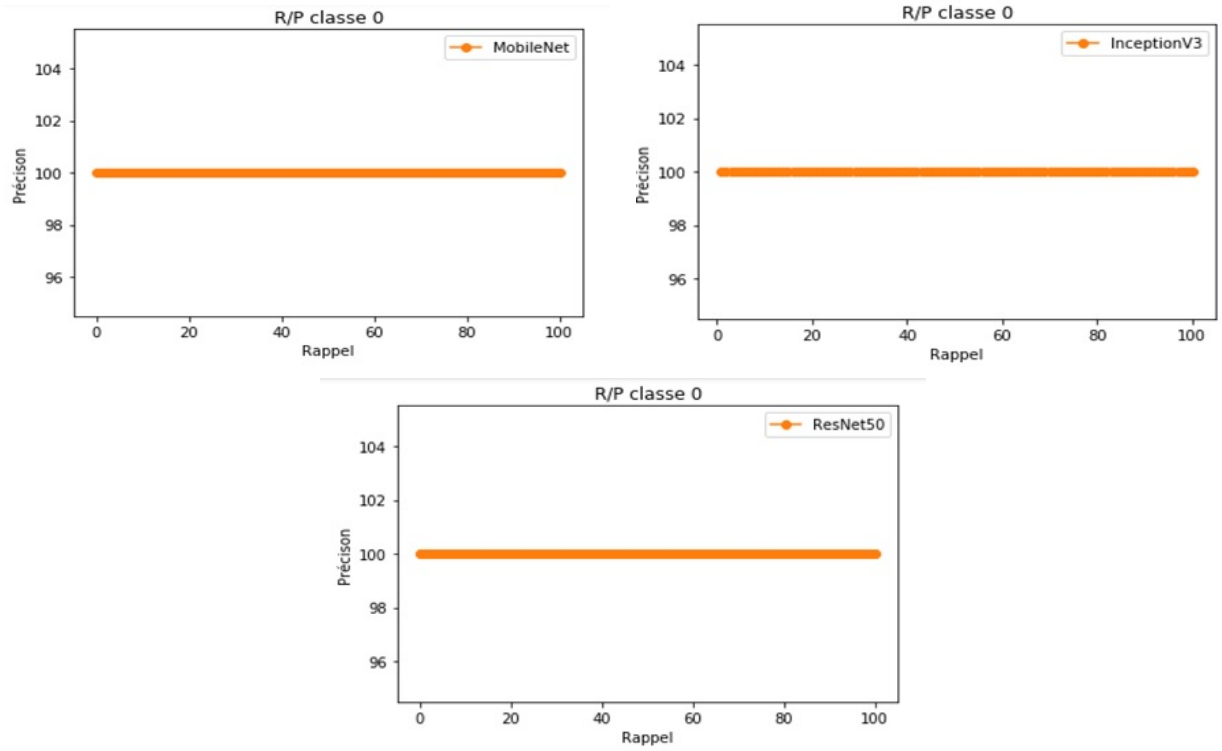


FIGURE 5.3 – Les courbes Rappel/Précision de la classe N° 0

5.4.4 Discussion

Les résultats obtenus durant cette partie montrent l'importance d'entrainer un réseau neurone. Les différentes architectures utilisées donnent un très bon résultat. Nous avons commencé d'abord par afficher les 20 meilleures images (Top 20), alors le résultat de la précision moyenne était toujours à 100%. Par la suite, nous avons essayé avec les Top 50, 100, 200, 300, 400. Le résultat de la précision est toujours à 100% pour les différentes architectures. Comme notre base d'images contient 500 images par classe, nous avons calculé la précision moyenne pour les Top 500.

Les résultats obtenus ont permis de choisir les meilleures architectures en termes de précision, sachant qu'au niveau temps de recherche la différence est vraiment très légère (en milliseconde), alors que le temps de recherche dépend aussi de la machine utilisée. Les meilleures architectures pour notre moteur de recherche sont : **Mobilenet, InceptionV3, ResNet50**.

Dans la Section 5.4.3, nous avons vu que la précision moyenne baisse à un taux jusqu'au 99%. Afin d'améliorer cette précision, nous avons proposé une amélioration qui est basée sur une combinaison des différentes architectures. Les résultats obtenus sont montrés dans la section Section 5.5.

5.5 Combinaison des différentes architectures

Dans cette section, nous allons présenter une amélioration de la précision moyenne en se basant sur une combinaison des différentes architectures. Cette combinaison est détaillée ci-dessus.

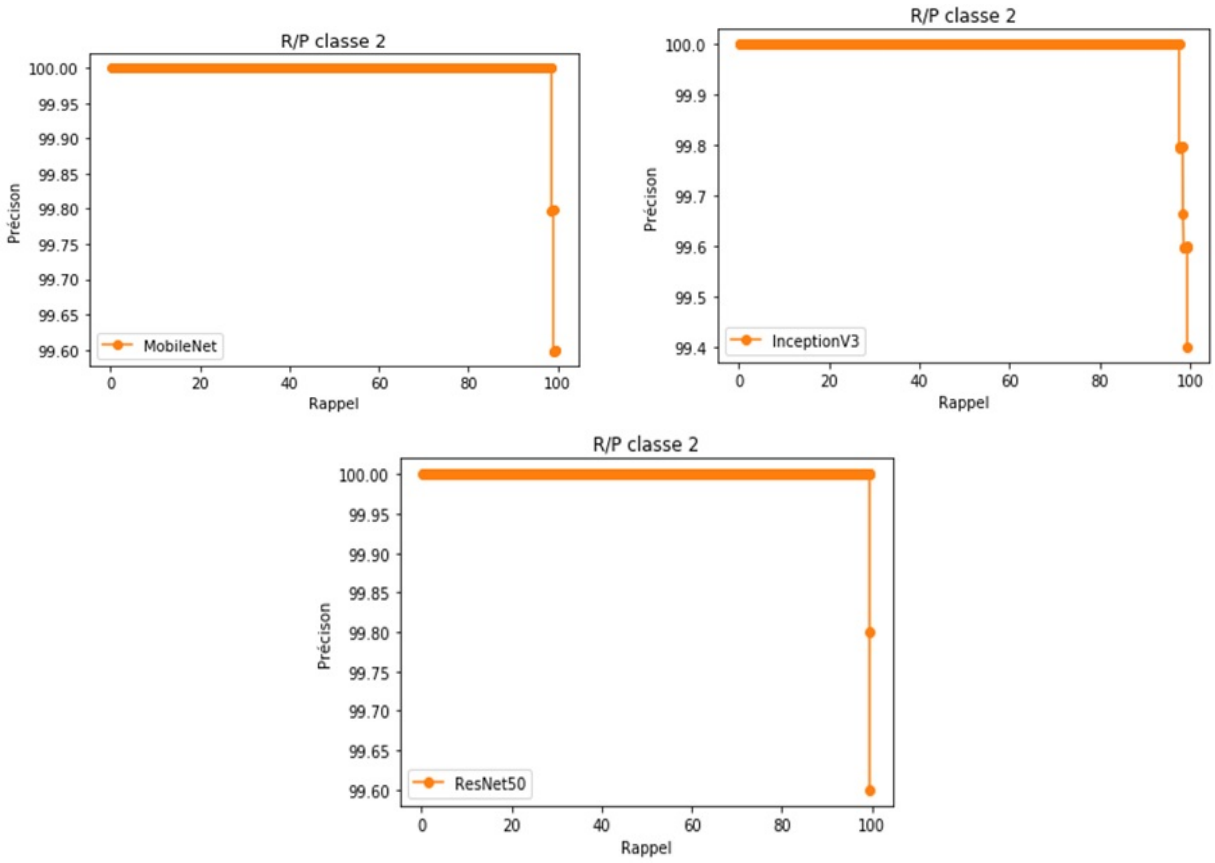


FIGURE 5.4 – Les courbes Rappel/Précision de la classe N° 2

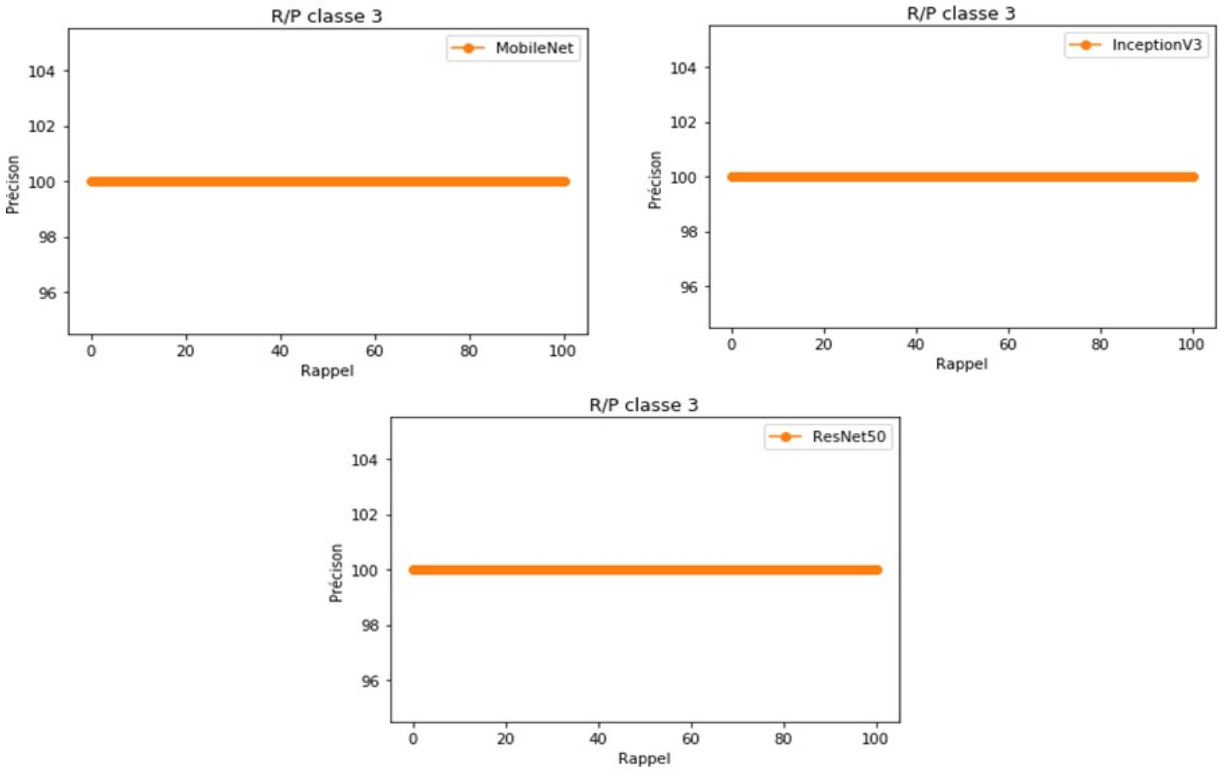


FIGURE 5.5 – Les courbes Rappel/Précision de la classe N° 3

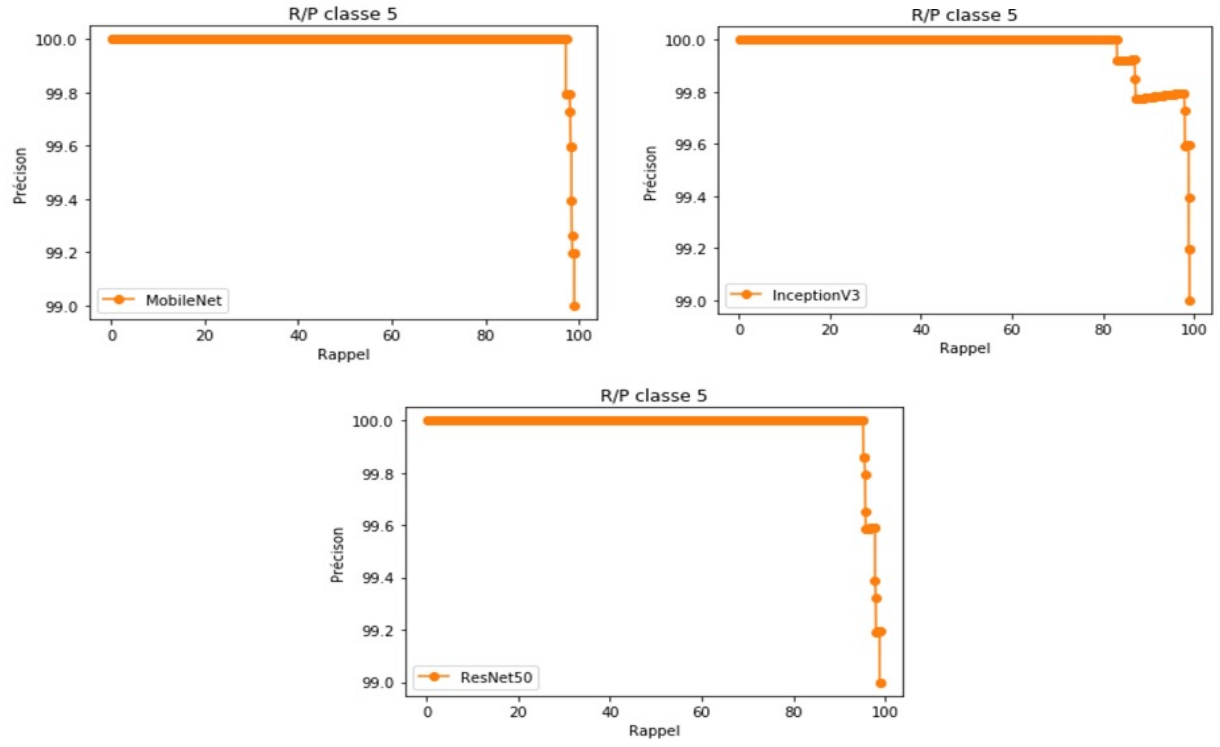


FIGURE 5.6 – Les courbes Rappel/Précision de la classe N° 5

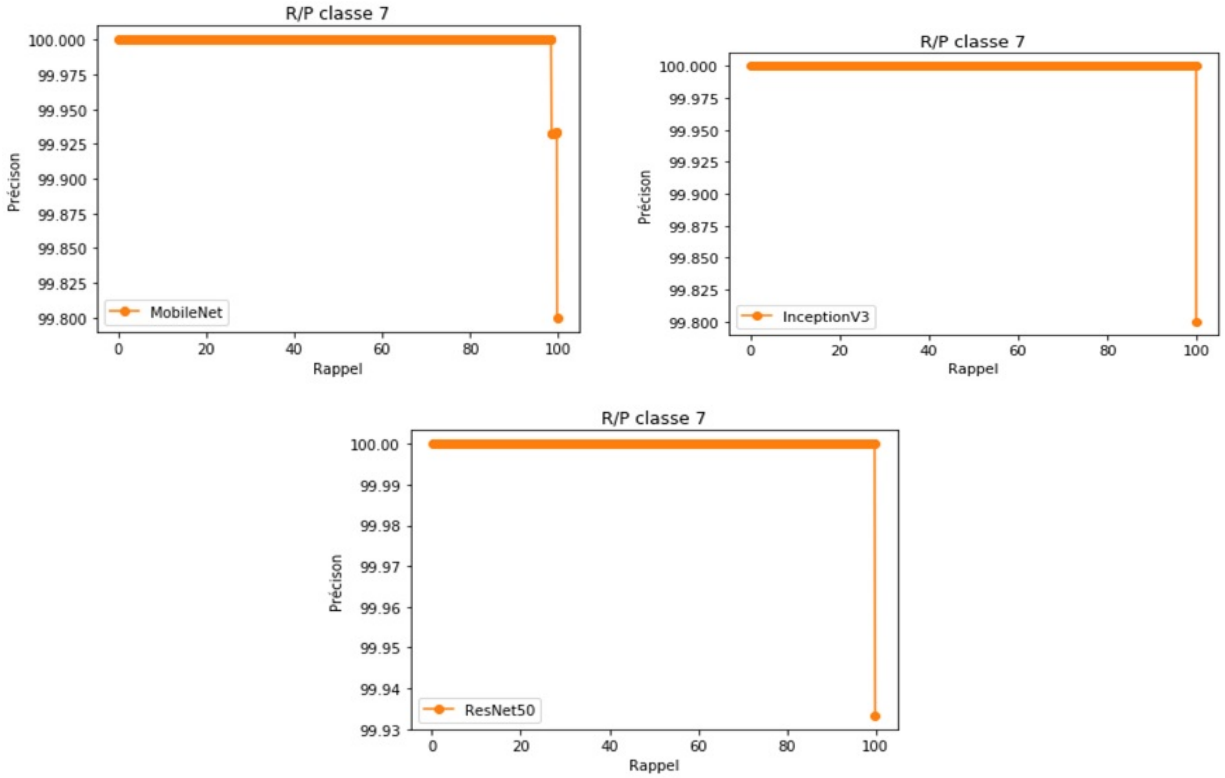


FIGURE 5.7 – Les courbes Rappel/Précision de la classe N° 7

- Suite aux résultats obtenus dans Section 5.4, les architectures retenues sont : **MobileNt, InceptionV3, ResNet50**.
- Lancer l'entraînement avec ces architectures.
- Pour chaque architecture, nous calculons le descripteur associé à chaque image.
- Combiner le résultat des trois descripteurs associés à chaque image en un seul descripteur. La combinaison est séquentielle sous forme d'une seule matrice.

5.5.1 Résultats

Nous avons calculé la précision moyenne ainsi que le temps de recherche pour chaque classe. Les résultats de cette expérimentation sont montrés dans le Tableau 5.7.

Comme nous le montre le Tableau 5.7, la précision moyenne est améliorée à un taux de 100%. Par contre, le temps de recherche a augmenté de 28%. De même, pour la taille des descripteurs qui a augmenté avec un facteur allant entre 10% et 12%. Cette taille dépend aussi du type d'image ainsi que de sa résolution.

5.5.2 Discussion

Les résultats de cette amélioration à base d'une combinaison à amélioré notre moteur de recherche pour une précision de 100%. Cette précision est calculée pour les meilleures 500 images affichées (Top

TABLEAU 5.7 – Le temps de recherche et la précision moyenne avec une combinaison de trois architectures.

Classe	N° image	T. R (s)	MaP (%)
	367	0,075	100
0	400	0,075	100
	168	0,086	100
	1057	0,085	100
2	1320	0,085	100
	1250	0,086	100
	1742	0,087	100
3	1820	0,086	100
	1703	0,084	100
	2917	0,086	100
5	2914	0,088	100
	2650	0,084	100
	3527	0,086	100
7	3493	0,084	100
	3608	0,085	100

500). Nous avons essayé de donner à notre moteur de recherche une image que notre réseau de neurones n'a jamais vue. Cette image provient du moteur de recherche Google. Le résultat était toujours à 100%, sachant que le type de cette image (classe) doit être parmi les classes de notre base d'images. L'inconvénient de cette approche est le temps de recherche qui a augmenté ainsi que l'espace de stockage. Afin de faire face à ces problèmes, nous allons proposer une nouvelle approche pour réduire le temps de recherche ainsi que l'espace de stockage dans la Section 5.6.

5.6 Réduction de la dimension des descripteurs

Le but de notre thèse est d'améliorer le temps de recherche des moteurs de recherche multimédia en gardant la précision. Comme nous avons montré dans la Section 5.5, la précision a atteint un taux de 100% même avec des images qui n'appartiennent pas à la base d'images, mais le temps de recherche a augmenté de 28%. Dans cette section, nous allons présenter la méthode RMAC [94] (Regional Maximum Activations of Convolutions) ainsi que son amélioration.

5.6.1 La méthode RMAC

Les auteurs dans [94] ont proposé une méthode d'extraction des caractéristiques des images par région. Elle utilise la méthode PCA afin de compresser le descripteur final. Dans notre cas le descripteur produit comme résultat est un vecteur de 512 valeurs.

En se basant sur les résultats obtenus dans les sections précédentes, nous avons calculé le descripteur de cette méthode avec les trois architectures retenues.

Tableau 5.8 – Le temps de recherche (m.s) la méthode RMAC avec les trois architectures retenues.

Classe	N° image	ResNet50	InceptionV3	MobileNet
0	367	0,021	0,022	0,020
	400	0,019	0,020	0,016
	168	0,018	0,018	0,015
2	1057	0,012	0,012	0,013
	1320	0,016	0,016	0,016
	1250	0,014	0,015	0,016
3	1742	0,013	0,011	0,015
	1820	0,015	0,015	0,017
	1703	0,015	0,016	0,015
5	2917	0,015	0,016	0,020
	2914	0,019	0,019	0,020
	2650	0,019	0,020	0,017
7	3527	0,016	0,015	0,014
	3493	0,017	0,017	0,018
	3608	0,016	0,018	0,018

Par la suite, nous avons calculé la précision moyenne de notre moteur de recherche ainsi que le temps de recherche pour chaque architecture. Le Tableau 5.8, nous montre le temps de recherche obtenu par le moteur de recherche.

Afin de comparer mieux les différentes architectures, nous avons calculé ainsi la précision moyenne des 500 images retenues pour chacune de ces architectures, comme nous montre le Tableau 5.9.

Les résultats obtenus nous montrent que la méthode RMAC a permis d'améliorer le temps de recherche (accélération) en gardant la même précision, ainsi de réduire l'espace de stockage sachant que la sortie de la méthode RMAC pour notre cas est un vecteur de 512 valeurs. Par la suite, nous avons appliqué cette méthode sur la combinaison des trois descripteurs avec les trois architectures **ResNet50**, **MobileNet** et **InceptionV3**.

5.6.2 Notre contribution

Dans cette section, nous allons proposer deux approches $\mathcal{RFA}$ et $\mathcal{RFB}$, dont le but est d'améliorer la précision. Cette contribution se base sur l'approche où nous avons amélioré la précision à 100%. Le détail de ces deux approches est le suivant :

- $\mathcal{RFA}$ calcule le descripteur RMAC sur l'ensemble des trois descripteurs des différentes architectures, c-à-d nous allons calculer le descripteur qui est à base d'une combinaison des trois descripteurs. Par la suite, nous appliquons la méthode RMAC sur ce descripteur. Le résultat sera un vecteur de 512 valeurs.
- $\mathcal{RFB}$ calcule pour chaque architecture le descripteur associé. Par la suite, nous calculons le vecteur RMAC pour chaque descripteur et à la fin nous combinons le résultat des trois vecteurs.

L'architecture de l'approche $\mathcal{RFB}$ est montrée sur la Figure 5.8. Le résultat de cette méthode est un

Tableau 5.9 – La précision moyenne de la méthode RMAC avec les trois architectures retenues.

Classe	N° image	ResNet50	InceptionV3	MobileNet
	367	100	100	100
0	400	100	100	100
	168	100	100	100
	1057	99,998	99,992	99,995
2	1320	99,998	99,992	99,995
	1250	99,998	99,992	99,995
	1742	100	100	100
3	1820	100	100	100
	1703	100	100	100
	2917	99,992	99,965	99,986
5	2914	99,992	99,966	99,987
	2650	99,992	99,956	99,986
	3527	100	99,996	99,997
7	3493	99,991	99,996	99,997
	3608	100	99,996	99,999

vecteur de taille de 3 x 512 = 1536 valeurs.

Le Tableau 5.10, présente les résultats associés à la première approche. Nous avons calculé la précision moyenne ainsi que le temps de recherche pour chaque classe. Le moteur de recherche affiche les 500 meilleures images (Top 500) similaires à l'image requête.

Les résultats obtenus de la première approche montrent l'intérêt de combiner plusieurs architectures. La précision moyenne est toujours à 100%. Par contre, le temps de recherche augmente d'un facteur allant de 20% à 24 %, ainsi que l'espace de stockage qui a été multiplié par 3 par rapport à la méthode RMAC appliquée sur une seule architecture.
Le Tableau 5.11 nous montre les résultats associés à notre deuxième approche. Afin de montrer l'efficacité de la méthode, nous avons calculé la précision moyenne ainsi que le temps de recherche. Notre moteur de recherche permet d'afficher les 500 meilleures images (Top 500) similaires à chaque image requête.

La deuxième approche proposée montre que le temps de recherche est très rapide par rapport à la première. De même, la précision est toujours à 100% sauf pour la classe 5 qui est à 99,999% de précision sur les 500 images affichées. Donc, si nous affichons les 498 images la précision est toujours à 100% avec un temps de recherche rapide. De même, nous avons gagné ainsi au niveau de l'espace de stockage, vu que le résultat de cette approche est un vecteur de 512 valeurs.

5.7 Matériels utilisés

Les méthodes proposées dans ce chapitre ont été testées sur plusieurs machines :

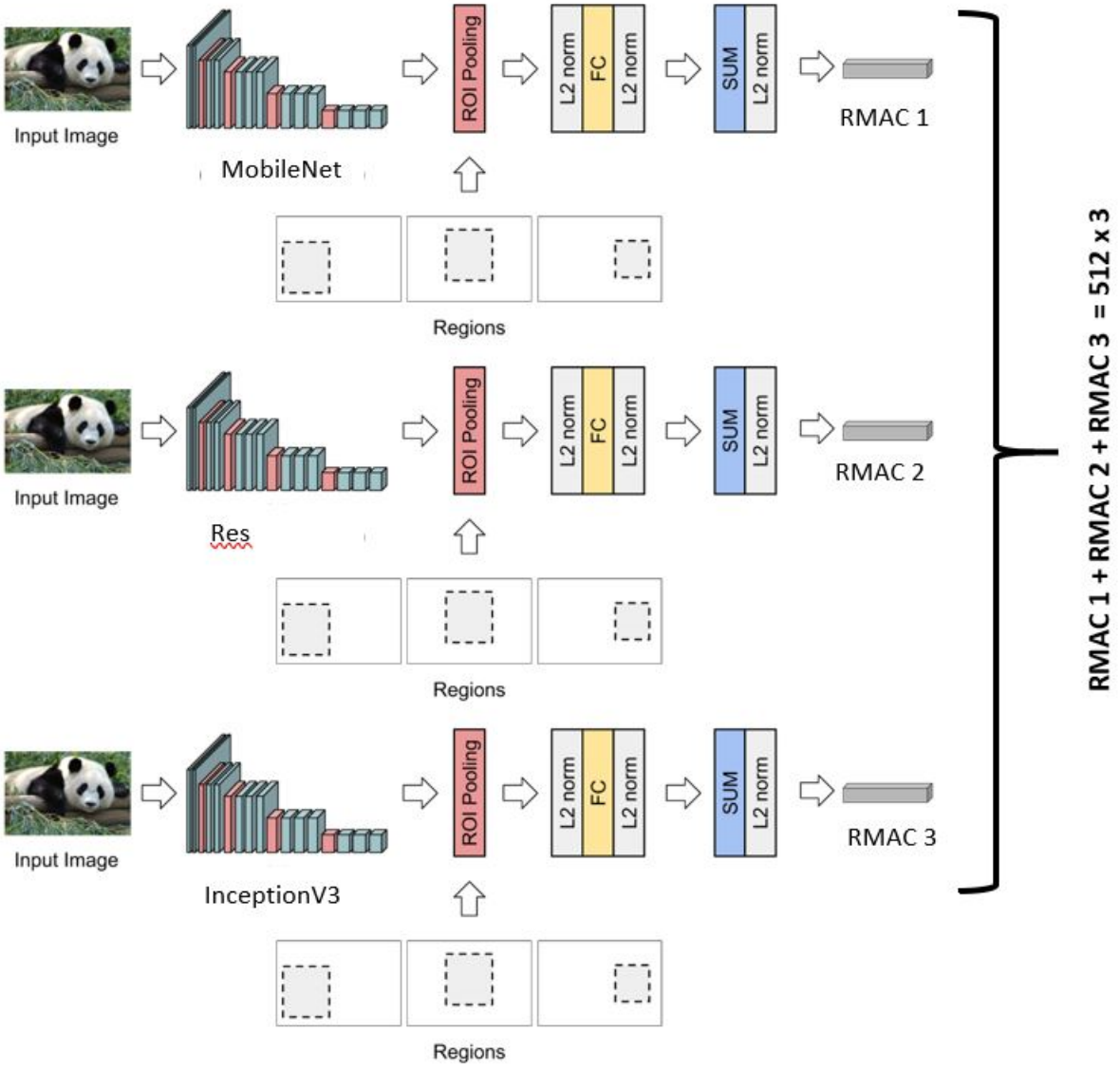


FIGURE 5.8 – Architecture de la méthode $\mathcal{RFB}$

TABLEAU 5.10 – Le temps de recherche (m.s) des deux approches $\mathcal{RFA}$ et $\mathcal{RFB}$.

Classe	Indice image	RFA	RFB
0	367	0,058	0,020
	400	0,058	0,019
	168	0,062	0,018
2	1057	0,039	0,012
	1320	0,052	0,015
	1250	0,048	0,014
3	1742	0,044	0,013
	1820	0,059	0,015
	1703	0,049	0,015
5	2917	0,058	0,015
	2914	0,063	0,019
	2650	0,058	0,019
7	3527	0,045	0,015
	3493	0,057	0,017
	3608	0,059	0,016

1. Le cluster de la Faculté Polytechnique de l'Université de Mons [22] avec la configuration suivante :
   - un processeur de Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz avec 16 cœurs physiques et 32 cœurs logiques ;
   - 64 Gb de Ram ;
   - quatre cartes graphiques GeForce GTX 1080 avec 12 Gb de Ram ;
   - Tensorflow 1.9 compilé en GPU ;
   - Keras version 2.2.0 ;
   - Theano version 0.9.0 est utilisé pour la méthode RMAC ;
   - Python version 3.5.

2. Une machine réservée sur Goolge cloud [23] avec la configuration suivante :
   - un processeur de Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz avec 6 cœurs physiques et 12 cœurs logiques ;
   - 32 Gb de Ram ;
   - une carte graphique K80 12 Gb de Ram ;
   - Tensorflow 2.1 compilé en GPU ;
   - Keras version 2.2.0 ;
   - Theano version 0.9.0 est utilisé pour la méthode RMAC ;
   - Python version 3.5.

22. Cluster : https ://www.ig.umons.ac.be/fr/content/cluster-de-calcul-ig
23. Google Cloud : https ://cloud.google.com/

Tableau 5.11 – La précision moyenne des deux approches $\mathcal{RFA}$ et $\mathcal{RFB}$.

Classe	Indice image	RFA	RFB
	367	100	100
0	400	100	100
	168	100	100
	1057	100	100
2	1320	100	100
	1250	100	100
	1742	100	100
3	1820	100	100
	1703	100	100
	2917	100	99,99
5	2914	100	99,99
	2650	100	99,99
	3527	100	100
7	3493	100	100
	3608	100	100

5.8 Conclusion

Nous avons présenté dans ce chapitre un système d'indexation et de recherche des images par le contenu. Ce système utilise les descripteurs extraits à travers des méthodes de l'apprentissage profond (Deep Learning). D'abord nous avons présenté une approche qui utilise comme entrée les poids des architectures de classification entrainées sur la base d'images ImageNet afin de calculer les descripteurs associés à nos bases d'images. Les résultats obtenus au niveau de la précision insistent sur le fait de lancer un entraînement sur nos bases d'images. La deuxième partie consiste à entrainer nos bases d'images, afin de trouver les meilleures architectures en termes de précision ainsi qu'en termes du temps de recherche. Les résultats obtenus ont permis la sélection de trois architectures : ResNet50, MobileNet, InceptionV3. De même, ces résultats ont permis de proposer une nouvelle approche qui se base sur la combinaison de ces trois architectures dont le but est d'améliorer la précision. Comme résultat, nous avons obtenu une précision de 100%, mais avec un temps de recherche plus lent. À la fin de la dernière partie de ce chapitre, nous avons proposé la méthode RMAC appliquée sur les trois architectures retenues. Cette approche a permis d'accélérer le temps de recherche en gardant la même précision. Finalement, nous avons proposé une amélioration de la méthode RMAC ainsi que les résultats obtenus.

Chapitre

6

Accélération de l'indexation dans un environnement parallèle et distribué

« Je pense que la caméra vous aime si elle peut vous voir penser et encore plus important, vous voir écouter. »

— Alan Rickman

Sommaire

6.1 Introduction

Dans ce chapitre, nous allons expliquer une approche que nous proposons pour accélérer l'indexation des vidéos. Cette approche est basée sur les images clés. En effet, les images clés forment ce qu'on appelle un résumé de vidéo. Le principe de base de notre méthode est de décrire une série de frames par une image clé. Cette méthode nous a permis de réduire l'espace de stockage ainsi que le temps de recherche.

Par la suite, nous proposons dans ce chapitre une nouvelle approche qui permet d'accélérer l'indexation multimédia. Cette méthode est basée sur la technologie Hadoop. Elle se repose sur le framework HIPI et qui est utilisée pour les deux phases : indexation et recherche. Ce framework est une librairie utilisée dans le domaine de traitement d'images destiné à utiliser les technologies Hadoop et MapReduce ainsi que la programmation parallèle. Nous avons proposé une architecture basée sur ce framework et qui utilise la programmation parallèle par l'utilisation du processeur graphique GPU dans un environnement parallèle et distribué.

Les premiers résultats de ce chapitre font l'objet d'une publication dans la conférence **Information Systems Design and Intelligent Applications** dans : **Springer Link** [9].

Les deuxièmes résultats de ce chapitre font l'objet d'une publication dans la conférence **CloudTech 2017 : Cloud Computing and Big Data : Technologies, Applications and Security** dans : **Springer Link** [10].

6.2 Etape 01 : Indexation séquentielle des vidéos

6.2.1 Pourquoi l'indexation des vidéos ?

D'abord, comme une vidéo est une suite d'images, tout ce que nous pouvons appliquer sur une image et appliquable sur les vidéos. Certe, ils existent des algorithmes qui sont spécifiques que pour les vidéos, mais dans le domaine de l'indexation, nous pouvons appliquer les mêmes descripteurs et distances.
De plus, le domaine de vidéosurveillance [37] a connu une croissance rapide durant les récentes années, avec l'arrivée des réseaux neurones profonds, ce domaine est de plus en plus importants dans la vie quotidienne (aéroport, centres commerciaux, parkings), d'où la grande croissance des bases de vidéos, ce qui a rendu la recherche dans ces bases rapidement difficile.

6.2.2 Document vidéo

Un document vidéo est défini comme une combinaison de différents flux de données. Les deux principaux flux de données sont les flux audio et visuels. Le flux visuel se compose d'une séquence d'images qui sont animées, en général par 24 à 30 images par seconde. Le flux audio est un mélange de différents sons (par exemple : mono, stéréo), qui sont, en général, un échantillonnage de 1600 à 48000 k-hertz. Un troisième flux qui peut être associé est le texte, il peut apparaître dans des documents vidéo pour aider les utilisateurs à comprendre la présentation audiovisuelle.

L'objectif d'un système d'indexation de vidéo est de permettre à un utilisateur d'effectuer une

recherche dans un ensemble de vidéos. La recherche est conduite sur des critères faciles à obtenir comme le réalisateur, le type d'émission, les acteurs ou le thème, mais il est extrêmement plus compliqué ou long d'obtenir une description précise du contenu des vidéos. Une indexation vidéo par le contenu est schématisée dans la Figure 6.1 [9].

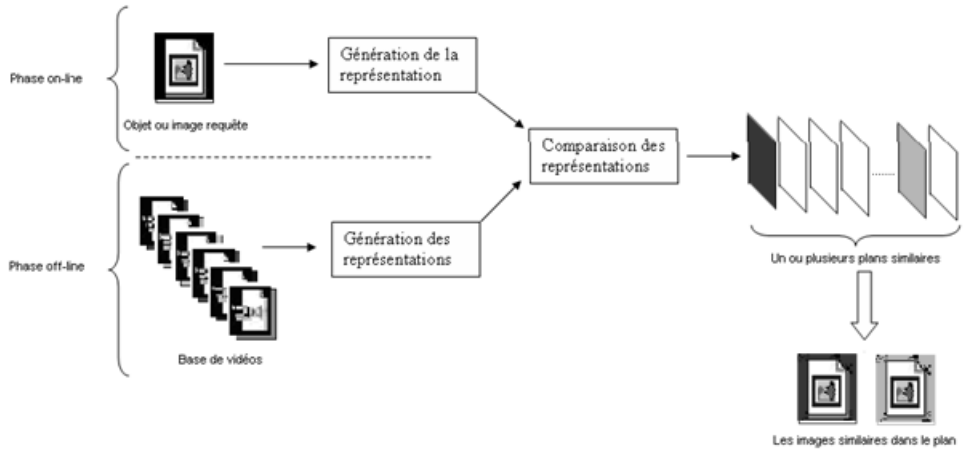


FIGURE 6.1 – Architecture d'un système d'indexation et de recherche de vidéos par le contenu [9]

6.2.3 Indexation bas niveau

Des descripteurs de bas niveau, tels que la couleur, la texture, la forme et le mouvement, peuvent être associés aux plans ou aux objets. La couleur des images choisies peut être décrite par l'histogramme de couleur ou par les couleurs dominantes [74]. Les paramètres de mouvement de caméra et le taux d'activité décrivent le mouvement au niveau du plan [74]. Le mouvement des objets peut être décrit par des trajectoires. Les sommaires d'images clés et les sommaires de segments importants sont généralement employés dans des applications commerciales. Les images clés, qui se rapportent à une ou plusieurs images représentatives dans un plan, fournissent une représentation intéressante et compacte.

Plusieurs méthodes existent pour choisir automatiquement les images clés par l'analyse de caractéristiques de bas niveau [74].

6.2.4 Processus

Le but de la caractérisation par image clés est de reprendre aux mieux les caractéristiques essentielles de ces vidéos dans un volume minimal. L'extraction des images clés passe par les étapes suivantes :

- Découpage en images : il s'agit dans cette étape de découper une vidéo en une série d'images individuelles. Ces images nous permettront de définir par la suite des plans d'images [9].
- Application d'un descripteur visuel des images, dans notre cas nous avons utilisé le descripteur histogramme couleur, SIFT et SURF.
- Calculer une mesure de similarité correspondante au descripteur appliqué dans l'étape précédente.

FIGURE 6.2 – Découpage en série d'images

— Segmentation temporelle : plusieurs techniques ont été proposées [74] pour segmenter une vidéo en plusieurs unités de base appelées "plans", nous pouvons citer :
  — Différence d'histogrammes comme nous montre la Figure 6.3.
  — Différence de blocs [74].

  Cette segmentation est dite temporelle parce que c'est une étape intermédiaire qui permet de générer par la suite les images clés, mais c'est la partie la plus importante de notre processus. Elle se base sur l'algorithme de clustering k-means [60].
— Sélection de l'image clé : nous devons extraire dans cette étape les caractéristiques visuelles de chaque plan, ces caractéristiques sont définies dans une ou plusieurs images appelées "images clés". L'image clé que nous proposons dans notre approche est la première image de chaque plan, car ce dernier contient les images similaires à cette image. Les images clés sont plus riches en informations par rapport aux autres images, l'ensemble de ces images forme ce que l'on appelle "résumé vidéo". Le résultat de cette partie est montré dans la Figure 6.4.

Comme nous montre la Figure 6.5, notre approche passe par deux phases :
— L'indexation où toutes les vidéos sont indexées en se basant sur l'approche des images clés. Cette phase dure longtemps, pour cette raison il est préférable de lancer ce processus la nuit. le résultat de cette première partie sont des sous dossiers, où chaque dossier contient les images similaires.
— Recherche : dans cette partie l'utilisateur fournit une image requête, par exemple un utilisateur souhaite chercher les films de son acteur préféré, il donne son image requête, le système traduit cette image sous forme d'un descripteur. Par la suite, nous allons comparés ce descripteur avec l'ensemble des descripteurs de toutes les images clés. A la fin, le système affiche l'ensemble des

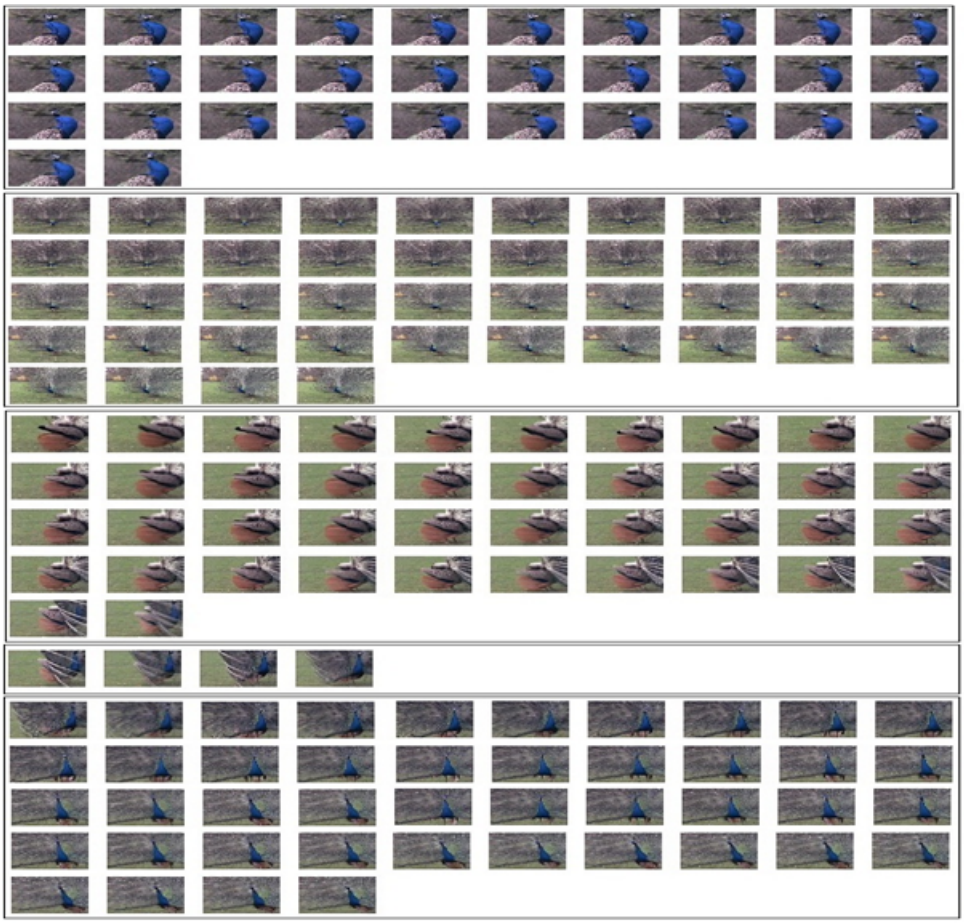


FIGURE 6.3 – Segmentation temporelle (en plan)

vidéos dont l'image requête apparaît.

6.2.5 Résultats expérimentaux

Afin de mieux évaluer notre approche, nous allons calculer le temps de recherche ainsi que la précision de notre moteur de recherche. La précision de notre moteur est calculée en utilisant les deux métriques rappel et précision comme expliqué dans la Section A.1.4

6.2.5.1 Données

Nos expérimentations ont été menées par l'utilisation de deux bases de vidéos :

1. Une base de vidéos qui contient 11 catégories, chaque catégorie contenant de 4 à 6 séquences de vidéos provenant de ChangeDetection [24].
2. Une base de vidéos contient 7 million de vidéos obtenues depuis Youtube-8M [25], cette base contient environ 45000 d'heures de vidéos ainsi que 4716 classes.

24. ChangeDetection : http ://www.changedetection.net/
25. Youtube8M : https ://research.google.com/youtube8m/

FIGURE 6.4 – Résumé vidéo (l'image clé de chaque plan)

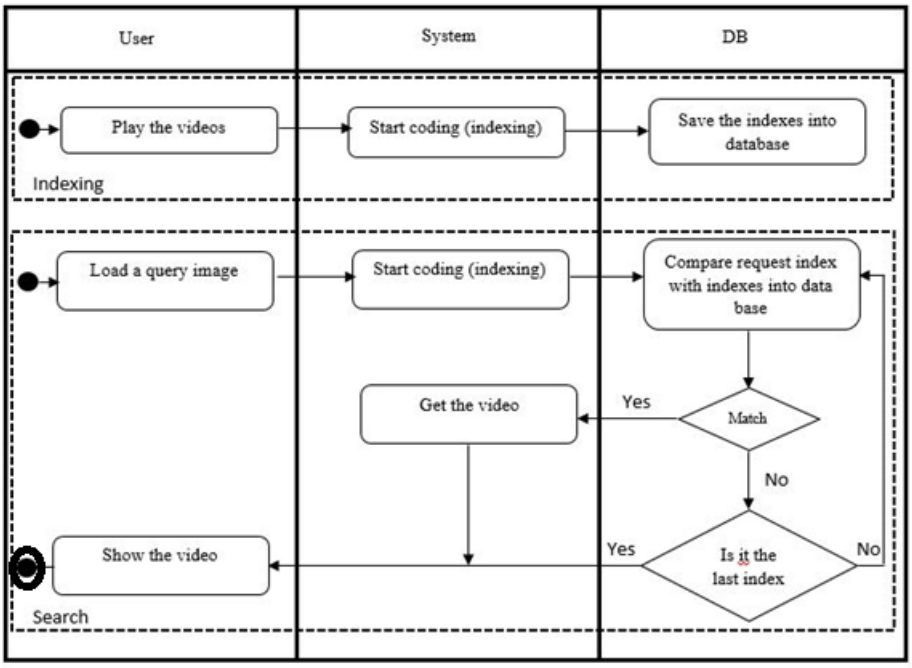


FIGURE 6.5 – Diagramme d'activité de notre approche

3. Afin de comparer mieux notre travail à d'autres travaux, nous avons utilisé la même base vidéos utilisées par Lakshmi Priya et al. [77]. Comme indiqué dans l'article cette base de vidéos a étée collécté en utilisant le site https ://www.nationalgeographic.fr/video. L'ensemble des catégories choisies sont indiquées dans le Tableau 6.1.

6.2.5.2 Résultats

Afin de comparer mieux notre approche, nous l'avons comparer d'abord par une recherche séquentielle afin de voir la différence que notre approche nous apporte. Le plus important dans le domaine d'indexation est le temps de recherche. Nous avons calculé le temps de recherche avec plusieurs exemples. La Figure 6.6 nous montre l'importance d'utiliser notre approche qui est basée sur des images clés. Nous constatons une accélération de la phase de recherche (en ligne). L'utilisation des images clés nous a permis d'avoir un taux d'accélération de 50% à 78% avec la base de vidéos ChangeDetection.

De même, afin de valider notre approche, nous avons utilisé cette fois la base de vidéo Youtube8M. En

Tableau 6.1 – Les différentes séquences vidéos utilisées par les auteurs dans [77]

Sequence video	Durée (sec)	Nombre d'images
Beetle	171	4959
Nature-1	540	15660
Nature-2	1476	42804
Cemsbok	170	4930
Relationships	380	11020
Nature-3	720	20880
Deadliest animals-1	1380	40020
Animal hunts-1	395	11455
Migrations	273	7917
Killer plants	236	6844
Animal battle	418	12122
Life of elephant	326	9454
Deadliest animals-2	1321	38309
Animal hunts-2	538	15602
Parasitic mind control	207	6003
Loudest animals	355	10295
Africa's predator	158	4582
Tiger queen	234	6786
White lion cub	123	3567
Eye of leopard	477	13833
Jaguar	222	6438
Funny animals	369	10701
Cape buffalo	183	5307
Zebra	98	2842
Wild rhinos	262	7598
Total	11032	319928

gardant le même environnement avec les mêmes exemples, nous avons calculé le temps de recherche. La Figure 6.7 nous montre le temps de recherche de notre moteur d'indexation. Nous avons lancé plusieurs tests afin d'avoir plusieurs résultats pour pouvoir valider notre approche. L'utilisation des images clés nous a permis d'avoir un taux d'accélération de 50% à 78% avec la base de vidéos ChangeDetection.

Nous avons deux critères à préserver, le temps de recherche et la précision, après avoir validé et confirmé la rapidité de notre approche, nous allons comparé la précision de notre moteur de recherche avec le travail présenté dans [77]. Le tableau Tableau 6.2 indique la précision moyenne de toutes les catégories de la base de vidéos.

Tableau 6.2 – Comparaison entre la précision moyenne de notre approche avec d'autres travaux

Base de vidéos	Notre approche	Priya et al. 2013 [77]	Lian. 2011 [59]	Yoo et al. 2006 [101]
Moyenne	85.1	95.8	92.3	89.9

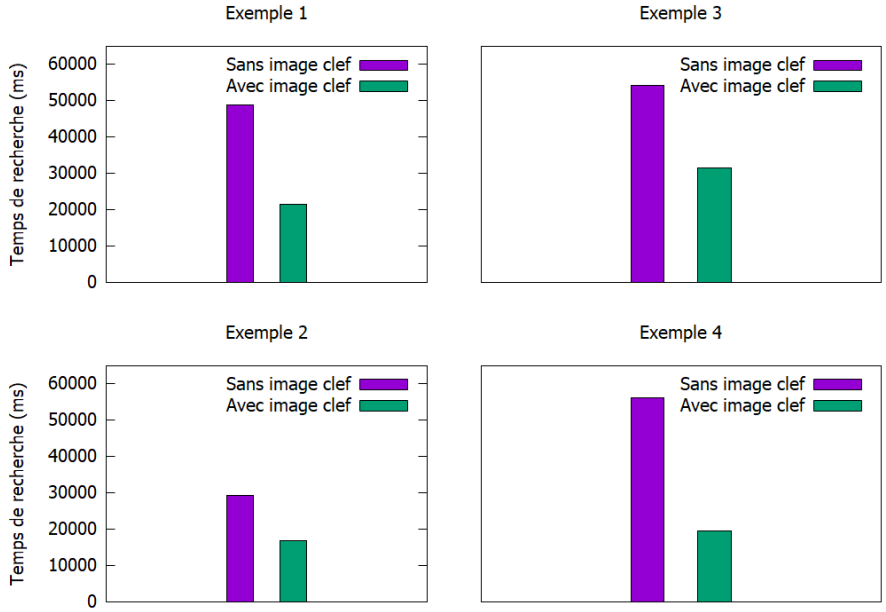


FIGURE 6.6 – Le temps de recherche avec et sans les images clés réalisé sur la base ChangeDetection.

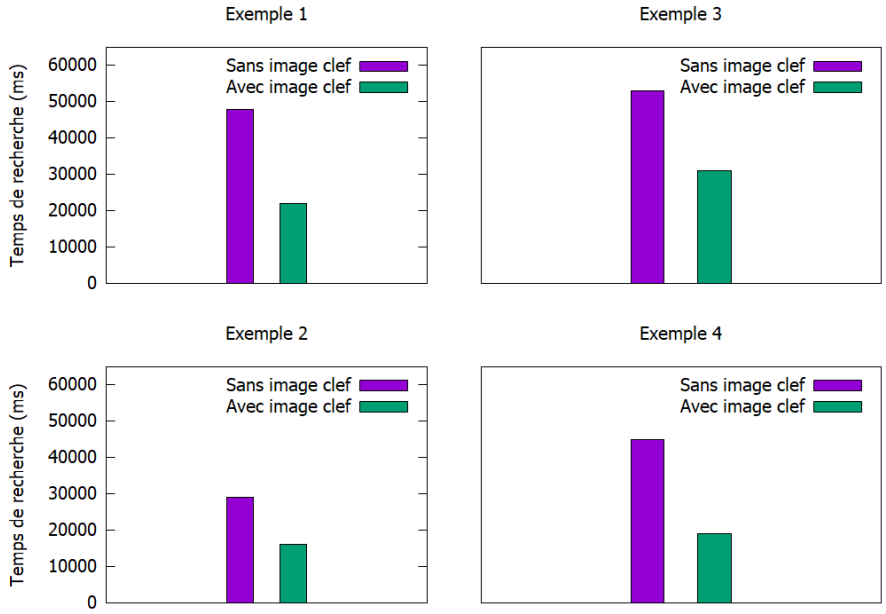


FIGURE 6.7 – Le temps de recherche avec et sans les images clés réalisé sur la base Youtube8M.

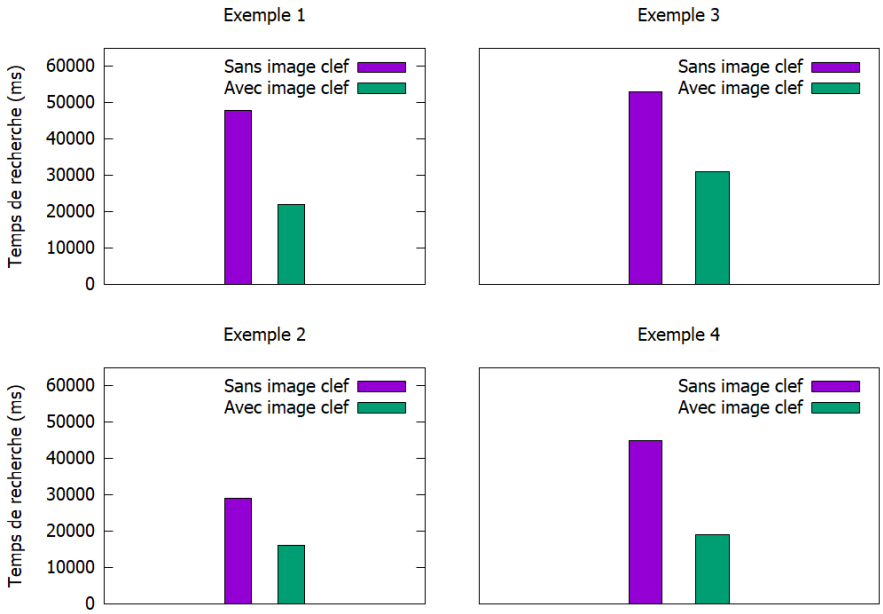


FIGURE 6.8 – Le temps de recherche avec et sans images clés réalisé sur la base Youtube8M.

6.2.5.3 Performances de l'algorithme

Notons aussi que notre méthode de description des vidéos par image clés a été testée sur le cluster de la faculté polytechnique de l'université de Mons [26].

De même, les tests ont été réalisés sur des machine Cloud provant de Microsoft Azure [27] et Amazon Web service [28], avec différents systèmes d'exploitation et différents configurations.

La Figure 6.8 nous montre le temps de recherche d'une image avec la base de vidéos Youtube8M. Ce test a été réalisé sur une machine Cloud (Machine virtuelle). Nous avons lancé plusieurs tests afin de comparer le temps de recherche avec l'utilisation des image clés ou sans.

6.2.5.4 Discussion

Notre approche prend comme paramètre d'entrée une image requête. Le système va ensuite calculer le descripteur de cette image. Après, le système lance une méthode de comparaison entre le descripteur de l'image requête et les descripteurs des images clés. Finalement, le système retourne les vidéos similaires. Nous avons testé notre approche avec plusieurs images requêtes ainsi que plusieurs machines. Comme résultat, le temps de recherche est toujours plus rapide lors de l'utilisation des images clés.

Par contre, notre méthode est moins précise comparée à la méthode proposée par Priya et al. [77],

26. Cluster : https ://www.ig.umons.ac.be/fr/content/cluster-de-calcul-ig
27. Azure : https ://portal.azure.com
28. Amazon : https ://console.aws.amazon.com

parce que nous utilisons l'histogramme couleur comme un descripteur des images, ce qui dégrade largement la précision de notre moteur de recherche. Après vérification, nous avons remarqué que la combinaison image clés et histogramme couleur a causé cette dégradation, parce que lors de la segmentation temporelle nous avons remarqué qu'il y'a plusieurs images qui appartiennent à un sous ensemble alors que visuellement ces images sont complètement différentes par rapport à l'image clés, ce qui a permis la perturbation de notre système.
Afin de corriger ce problème, la combinaison de plusieurs descripteurs peut corriger ce problème, mais il ne faut pas oublié le temps de recherche, comme nous avons constaté dans le Chapitre 5 Section 5.5.

La phase d'indexation reste lente, pour indexer une vidéo 4K d'une heure avec les images clés, le temps d'indexation dépasse les trois heures mais ça reste une phase hors ligne.

6.3 Etape 02 : Indexation dans un environnement parallèle et distribué

Après avoir réalisé différents systèmes d'indexation permettant d'accélérer la phase de recherche. Nous allons maintenant proposer une approche d'indexation et de recherche dans un environnement parallèle et distribué. La méthode que nous proposons dans cette section permet d'indexer et de rechercher les images en parallèle sur la même machine en utilisant un processeur graphique GPU ainsi qu'une parallélisation par rapport aux plusieurs machines (nœuds). Nous rappelons que cette approche se base sur l'architecture HIPI [25].

6.3.1 Motivations

Sweeney et al. dans [25] ont proposé une approche appelée HIPI qui utilise la solution Hadoop Mapreduce pour la programation parallèle. Cette approche est efficace pour touts les algorithmes de traitement d'images. Avec l'utilisation de MapReduce les algorithmes sont exécutés dans mes machines parallèles. De même, cette approche propose une solution de stockage pour les grandes bases d'images en utilisant le système de fichier HDFS. En plus, HIPI intègre la bibiothèque OpenCV [29], qui est une bibiothèque opensource.

Avec l'arrivée du langage CUDA et les cartes graphiques, plusieurs algorithme dans le domaine du traitement d'images on été améliorés afin d'utiliser le processeur graphique GPU pour faire les calculs. Pour ces deux raisons, nous proposons une nouvelle approche qui se base sur l'architecture HIPI en expoiltant les ressources des processeurs graphiques.

L'architecture de l'approche proposée par Sweeney et al. dans [25] est montrée sur la Figure 6.9.

29. OpenCV : https ://opencv.org/

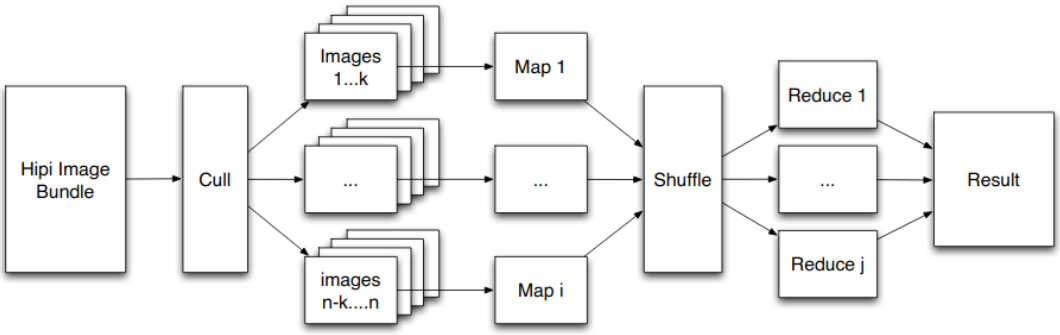


FIGURE 6.9 – Architecture générale du Hadoop Image Processing Interface [25]

6.3.2 Environnement

Notre approche a été développée dans l'environnement suivant :

- Une machine hôte (ou appelé Master), le rôle de cette machine est d'exécuter nos algorithmes. Par la suite, grâce à l'architecture Hadoop, les algorithmes seront dispatchés sur les différents noeuds.
- Neufs machines équipées des processeurs graphiques GPU, avec la bibiothèque OpenCV compilée avec GPU mais avec le langage JAVA [30] et aussi la bibiothèque JCUDA [31].
- Une machine utilisée pour la sauvegarde (backup) de la machine hôte.

La provenance de ces machines est :

1. Le cluster de la faculté polytechnique de l'université de Mons [32] avec la configuration suivante :
   - Un processeur de Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz avec 16 cœurs physiques et 32 cœurs logiques.
   - 64 Gb de Ram.
   - Quatre cartes graphiques GeForce GTX 980 avec 12 Gb de Ram.
   - Cuda compilé en GPU avec C++ et Java (JCuda).
   - Une image Docker avec le framework Hadoop pré-installé.
2. Un cluster Hadoop (machines) réservé sur Goolge cloud [33] avec la configuration suivante :
   - Un processeur de Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz avec 6 cœurs physiques et 12 cœurs logiques.
   - 32 Gb de Ram.
   - une carte graphique K80 12 Gb de Ram.
   - Cuda compilé en GPU avec C++ et Java (JCuda).
   - le framework Hadoop Hadoop pré-installé.

30. JAVA : https ://www.java.com/
31. JCUDA : https ://opencv.org/
32. Cluster : https ://www.ig.umons.ac.be/fr/content/cluster-de-calcul-ig
33. Google Cloud : https ://cloud.google.com/

6.3.3 Résultats expérimentaux

6.3.3.1 Processus

Dans cette section, nous allons détailler notre approche comme indiqué dans l'architecture présentée dans la Figure 6.9.

1. **Hipi Image Bundle** : ici nous allons convertir toutes nos images en cet format. C'est une étape nécéssaire afin que notre approche fonctionne. Cette étape dépend du nombre des images de notre base.
2. **Cull** : cette opération divise l'ensemble des images sur les neufs machines utilisées, la division dans ce cas est homogène sur l'ensemble de ces machines.
3. **Map** : dans notre approche le calcul des descripteurs des images se fait à ce niveau. Chaque machine calcule le descripteur SIFT ou SURF mais en utilisant le processeur graphique GPU avec l'aide de la bibiothèque JCUDA. L'image est divisée en quatre parties, pour chaque partie nous allons calculer son descripteur associé.
4. **Shuffle** : dans cette étapes, nous allons regrouper les quatres partie divisée en une seule partie pour avoir un seul descripteur pour chaque image.
5. **reduce** : c'est la dernière étape de notre approche, où les descripteurs seront regroupés par machine, par la suite l'ensemble des descripteurs seront répartis sur l'ensemble des machines.

6.3.3.2 Données

Nous avons effectué nos expériences en utilisant plusieurs bases d'images :

1. La base d'images Wang. [34] qui contient 7000 images regroupées en 50 catégories.
2. La base d'images Coil [35] qui contient 7200 images.
3. La base d'images Mirflickr [36] qui contient 1 millions d'images.
4. La base d'images ImageNet [37] qui contient 16 millions d'images.

Les deux premières bases d'images ont été utilisées pour faire les premiers tests, les deux autres pour valider notre approche

6.3.3.3 Résultats

Afin de comparer mieux nos résultats, nous avons calculé le temps d'indexation ainsi que le temps de recherche de notre système d'indexation. Par la suite, nous avons comparé les résultats obtenus avec des travaux existant dans la litérature. Le travail en question est celui de Sweeney et al. dans [25].

L'étape d'indexation de notre cas est très importante, parce que c'est elle qui décide la précision de notre moteur de recherche. Nous avons eu des résultats de calcul des descripteurs qui sont indiqués dans la Figure 6.10.

34. Wang : hhttp ://wang.ist.psu.edu/docs/related/
35. Coil : http ://www.cs.columbia.edu/CAVE/software/softlib/coil-100.php
36. Mirflickr : http ://press.liacs.nl/mirflickr/
37. ImageNet : http ://image-net.org/about-stats

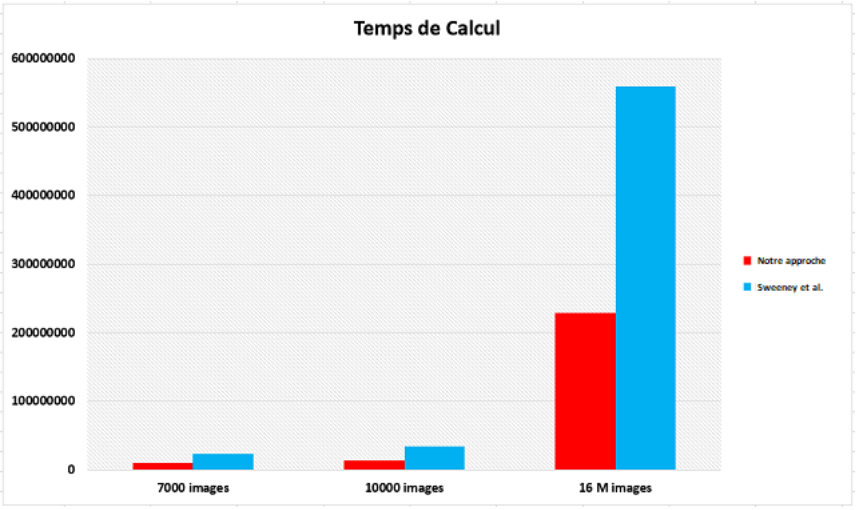


FIGURE 6.10 – Le temps de calcul (ms) des descripteurs de notre approche compré à l'approche de Sweeney et al. dans [25]

Après avoir indexé toute notre base d'images, nous avons lancé le processus de la recherche et nous avons calculé le temps nécessaire pour effectuer une recherche comme nous montre le Tableau 6.3 ainsi que la Figure 6.11.

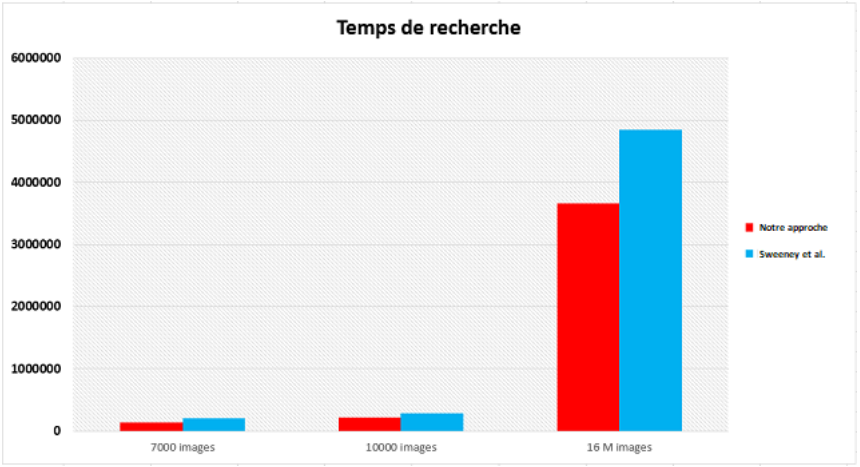


FIGURE 6.11 – Le temps de recherche (ms) de notre approche compré à l'approche de Sweeney et al. dans [25]

TABLEAU 6.3 – Le temps de recherche (ms) de notre approche comprée a l'approche de Sweeney et al. dans [25]

Approche (ms)	7000 images	10000 images	16M images
Notre approche	140071	221943	**3662059,5**
Sweeney et al.	207249	293470	4842255

6.3.3.4 Discussion

Nous avons remarqué une accélération au niveau des deux phases : indexation et recherche. Plus particulièrement, nous remarquons une grande amélioration de la phase d'indexation par rapport la phase de recherche. Ce résultat est dû parce que, dans la phase d'indexation toute la base d'images sera indexée, et donc le processeur graphique GPU est mieux exploité ce qui donne une large différence quand il s'agit des grandes bases d'images comme ImageNet. Dans le cas de la recherche, le processeur graphique GPU est moins exploité, parce que nous allons calculer le descripteur associé à l'image requête, puis un calcul de similarité entre le descripteur de cette image et tous les autres descripteurs.
A ce stade là, nous avons remarqué une légère amélioration du temps de recherche. Pour cette raison, nous avons implémenté la méthode KNN [38] sur GPU afin d'accélérer la phase de recherche.
L'inconvénient est dans son architecture. Il faut avoir des machines équipées des processeurs graphiques, ce qui reste chère en ce moment même pour une location sur le Cloud. Nous avons eu de la chance d'avoir des coupons de Google Cloud ainsi que grâce à l'utilisation du cluster de l'université.

6.4 Conclusion

Dans ce chapitre, nous avons présenté d'abord, notre système d'indexation de vidéo. Ce système est composé de deux phases : une phase hors ligne (Offline) où les vidéos sont segmentées en un ensemble d'images individuelles. Ensuite, nous calculons les descripteurs de chaque image pour pouvoir par la suite générer le résumé de vidéo à travers les images clés. La phase en ligne qui est la phase de recherche (Online), ou dans laquelle le système prend une image requête comme un paramètre d'entrée. Le résultat de la recherche est un ensemble des vidéos où figure l'image requête. Cette approche nous a permis de gagner au niveau du temps de recherche d'un facteur allant de 20 % à 30 % ainsi que l'espace de stockage avec un facteur allant de 30 % à 40 %. Par contre, l'utilisation des histogrammes de couleur nous a montré l'utilité d'utiliser d'autres descripteurs afin d'améliorer la précision de notre approche.

Par la suite, nous avons présenté dans ce chapitre un système d'indexation et de recherche des images parallèle et distribué. Ce système se base sur la solution Hadoop avec la librairie HIPI ainsi que la programmation sur GPU. L'idée de cette approche est le parallèle au niveau des machines en utilisant l'architecture Hadoop ainsi que le parallèle dans la machine elle-même lorsque cette machine est équipée d'un processeur graphique. Cette approche est très efficace pour la phase d'indexation, par contre pour la phase de recherche, une amélioration de la manière dont le processeur GPU est utilisé est à revoir, sachant qu'il existe plusieurs techniques pour le calcul d'un descripteur sur GPU par exemple diviser l'image en fonction de la capacité de ce dernier.

Finalement, les résultats obtenus montrent que notre système d'indexation et de recherche permet d'accélérer ces deux phases.

38. KNN GPU : https ://github.com/vincentfpgarcia/kNN-CUDA

Quatrième partie

Conclusion et perspectives

Chapitre

7

Conclusion générale et Perspectives

« Research is to see what everybody else has seen, and to think what nobody else has thought. »
— Albert Szent-Gyorgyi

Sommaire

7.1 Conclusion

Le travail présenté dans cette thèse est lié aux domaines d'extraction de caractéristiques et de recherche de données multimédia présentant une taille volumineuse (Big Data). Plus particulièrement, nous nous adressons à la mise en œuvre d'une méthode recherche et indexation de gros volumes d'images et de vidéos (Big Data) offrant à la fois une précision élevée, un temps d'exécution rapide, une réduction de la dimensionnalité ainsi qu'une exploitation efficace des ressources distribuées avec les technologies du Big Data.

Dans cette thèse, nous avons présenté les principales approches et algorithmes utilisés au sein de notre moteur de recherche. Les principales contributions apportées sont :

1. **Réduction de la dimensionnalité :** cette méthode s'appuie sur la méthode d'analyse en composantes principales PCA où nous avons optimisé le choix d'un paramètre d'entrée, représenté par un taux de compression allant de 10% et 90% de telle sorte que la précision ne soit pas dégradée. Par la suite, nous avons proposé une amélioration de notre système de réduction de la dimensionnalité en utilisant une combinaison de la méthode PCA avec VA-FILE et LSH. Cette combinaison a permis d'accélérer le temps de recherche.
2. **Accélération de la recherche avec les techniques de stockage des descripteurs :** nous avons proposé une nouvelle méthode de représentation des descripteurs sous la forme d'un arbre binaire, et cela afin de remplacer la recherche séquentielle par une recherche arborescente plus rapide.
3. **Les méthodes de Deep Learning :** dans cette partie, nous avons amélioré la précision de notre moteur de recherche par l'exploitation des descripteurs extraits par les méthodes de Deep Learning. De même, la dimension de ces descripteurs a été réduite en utilisant une approche adaptée RMAC.
4. **Accélération de l'indexation dans un environnement parallèle et distribué :** Nous avons proposé une version parallélisée et distribuée de notre méthode exploitant à la fois les processeurs centraux (CPUs) et graphiques (GPUs) afin de gérer des données volumineuses. Les technologies du Big Data (Hadoop et HIPI) ont été utilisées pour cette contribution.

7.2 Perspectives

Comme perspectives de notre travail nous proposons les suivantes :

— Nous avons choisit d'utiliser les architectures les plus connues pour ensuite se limiter à cinq architectures. Un travail futur serait de continuer les recherches sur la réduction avec les cinq architectures non retenues et essayées de nouvelles architectures.
— Explorer d'autres méthodes de réduction de la dimmensionalités appliquées sur les réseaux de neurones profonds et les comparer aux méthodes classiques.
— Utiliser d'autres solutions basées sur les environnements parralèles et distribués autres que Hadoop et voir la possibilité d'utiliser les caractérstiques de Deep Learning.

— Pour l'indexation des vidéos, remplacer les deux descripteurs SIFT et SURF par les caractérstiques de Deep Learning et générer les images clés.
— Utiliser d'autres bases d'images et de ne pas se limiter aux benchmarks les plus connus.
— Aborder l'explicabilité des réseaux de neurones convolutionnels afin de comprendre mieux ce qui se passe lors de la phase de recherche.

Cinquième partie

Annexes

Annexe

A Définitions liées à l'état de l'art

Dans cette annexe, nous décrirons les différentes méthodes d'indexation et de recherche par le contenu classiques ainsi que des définitions des méthodes utilisées pour réduire la dimension des descripteurs comme la méthode PCA comme indiqué dans le Chapitre 2

A.1 Les méthodes d'indexation et de recherche par le contenu classiques

Le but de l'indexation des données en général, qu'elles soient des données textuelles ou audiovisuelles (image, sons et vidéos) est de faciliter l'interrogation qui se fait à travers des requêtes de plusieurs types. Les types de requêtes les plus connus sont :

- **Parcours au hasard :** la base est parcourue aléatoirement jusqu'à ce que l'utilisateur trouve la donnée qui l'intéresse.
- **Navigation par catégorie :** Les données de la base sont classées par catégories. L'utilisateur peut donc directement choisir la catégorie dans laquelle il pense pouvoir la trouver.
- **Recherche par mots clés :** l'utilisateur saisit des mots censés représenter la donnée recherchée. Il dispose souvent d'une série de termes prédéfinis pour formuler sa requête.
- **Recherche par l'exemple :** pour une telle requête, l'utilisateur fournit une donnée exemple et le logiciel recherche dans la base, les données qui ressemblent à nos données exemple.

La Figure A.1 montre bien, comment une requête interroge des documents indexés dans un système d'information.

A.1.1 Indexation des images

L'indexation d'images est rendue possible, grâce à la fusion de la base de données et le traitement d'images. La recherche textuelle n'est plus utilisée actuellement, les utilisateurs s' intéressent beaucoup plus à une recherche qui vise directement le contenu.

La Figure A.2 montre l'exemple d'un système qui s'exécute en deux étapes : l'étape d'indexation et l'étape de recherche.

Dans l'étape d'indexation, des caractéristiques sont automatiquement extraites à partir de la base d'images et stockées dans un vecteur numérique appelé descripteur visuel. Grâce aux techniques de base de données, nous pouvons stocker ces caractéristiques et les récupérer rapidement et efficacement.

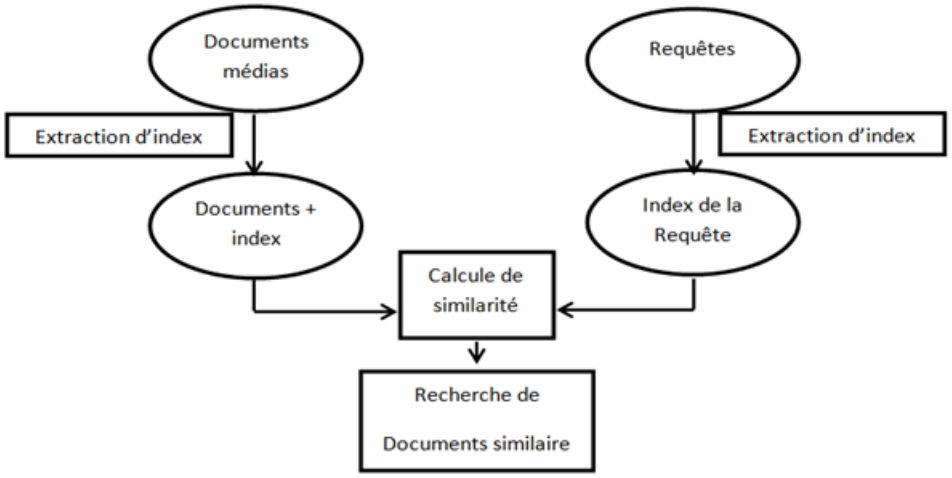


FIGURE A.1 – Modèle général d'un système d'indexation et de recherche d'information

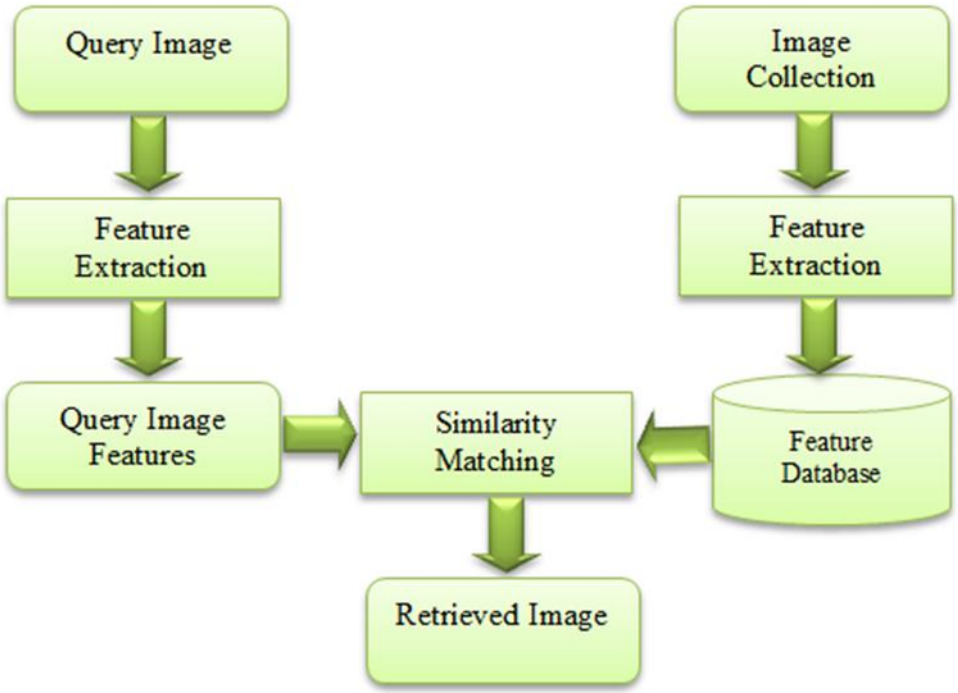


FIGURE A.2 – L'architecture d'un système d'indexation et de recherche d'images[39]

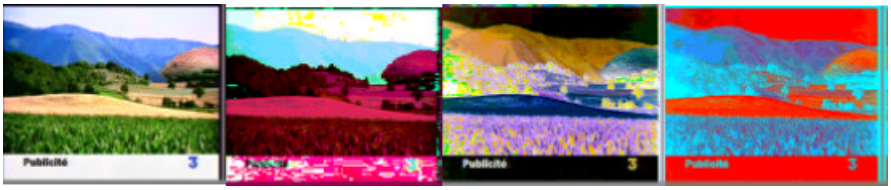


FIGURE A.3 – La même image en RGB, YIQ, YUV, CMY.

Dans l'étape de recherche, le système prend une ou des requêtes de l'utilisateur et leurs associe un résultat qui correspond à une liste d'images ordonnées en fonction de la similarité entre leurs descripteurs visuels et celui de l'image requête en utilisant une mesure de distance.

A.1.2 Les descripteurs

Un descripteur est une description numérique du contenu. Il doit être compact et précis. De plus une distance ou une mesure de similarité doit permettre de comparer les descripteurs et indirectement le contenu qu'ils représentent.

Nous pouvons distinguer plusieurs catégories de descripteurs visuels :

A.1.2.0.1 Couleur

La couleur représente le descripteur visuel le plus employé. Cependant, les grands problèmes soulevés par le choix de bons descripteurs de couleurs sont l'identification de l'espace de couleur le plus discriminant, la prise en compte des problèmes d'invariance aux conditions d'illumination et de prise de vues, ainsi que sa combinaison avec des descripteurs complémentaires (textures, formes, etc.). Il existe plusieurs espaces colorimétriques qui ont chacun certaines caractéristiques intéressantes.

L'espace RGB est un choix colorimétrique orienté matériel très simple a utilisé, car c'est celui employé par de nombreux appareils de capture d'image qui effectuent leurs échanges d'information uniquement en utilisant les triplets (R, V, B). Cependant, ces trois composantes sont fortement corrélées (par exemple, si l'on diminue la composante verte, la teinte parait plus rouge), l'espace RGB est sensible aux changements d'illumination, et ne correspond pas au processus de perception humaine.

Il existe d'autres espaces couleur comme nous montres la Figure A.3 :

- **YIQ :** (Le modèle YIQ ou plus précisément Y'IQ) est un espace colorimétrique à trois composantes. Il est utilisé dans le standard de télévision analogique National Television System Committee (NTSC). Y' représente la luma , I la composante en phase et Q la composante en quadrature de la chrominance.
- **YUV :** (Y, représente la luminance et les deux autres, U et V, représentent la chrominance), utilisé dans le système de diffusion télévisuelle PAL et SECAM appartenant aux standards de la télévision couleur.
- **CMY :** CMY et Cyan Magenta Yellow Key (CMYK) utilisés pour les imprimantes couleurs à impact et à jet d'encre.

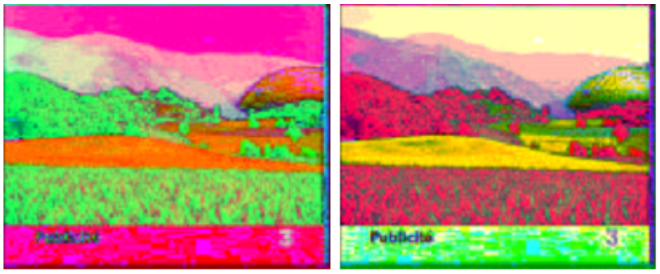


FIGURE A.4 – La même image en HSV, et HLS.

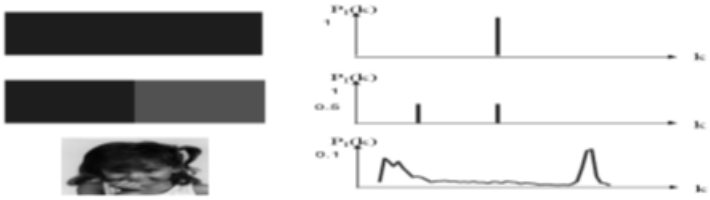


FIGURE A.5 – Exemple d'histogramme

— L'espace HSV (aussi connu sous le nom de système de cône hexagonal) sépare les informations relatives à la teinte (Hue), la saturation (Saturation) et l'intensité (Value). Cet espace est plus intuitif a utilisé car il correspond à la façon dont nous percevons les couleurs. La teinte décrit la couleur (rouge, vert, etc.), la saturation décrit l'intensité de la couleur, et la valeur décrit la luminosité de la couleur. La composante H de l'espace HSV offre une certaine invariance. Une étude récente compare six espaces calorimétriques et montre que l'espace HSV est le plus efficace pour la recherche d'images par le contenu. Cependant cet espace n'est pas perpétuellement uniforme.
— Aussi HLS sont les trois paramètres de description d'une couleur dans une approche psychologique de cette perception. Cette expression désigne des modèles de description des couleurs utilisés en graphisme informatique et en infographie, qui adaptent ces paramètres. La Figure A.4, nous montre une image avec les deux espaces colorimétrique HLS et HSV.

A.1.2.0.2 Histogramme

L'histogramme est une technique très utilisée pour la description de la couleur est l'intersection d'histogrammes [92]. Les histogrammes sont faciles et rapides à calculer, robustes à la rotation et à la translation [103], [56]. L'histogramme comme nous montre la Figure A.5 d'une image est la fonction qui associe à chaque valeur d'intensité le nombre de pixels dans l'image ayant cette valeur.

FIGURE A.6 – Problème d'histogramme lié au contraste et à la luminosité

A.1.2.0.3 Histogramme Couleur

L'histogramme couleur consiste à calculer la répartition statistique des couleurs présentes dans l'image. Pour une image I de taille M × N, l'histogramme h(c) est défini comme :

$$H(c) = \frac{1}{M \times N} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \delta(f(i,j) - c) \forall c \in C \tag{A.1}$$

— C : l'espace de couleurs possible.
— c : la fréquence de l'apparition de la couleur c dans l'image.
— δ : le symbole de Kronecker [8]. : $\delta(x,y) = 1$ si $x = y$, et 0 sinon.

Cependant, l'histogramme pose des problèmes selon les auteurs dans [79]. En effet, sa grande taille cause un manque d'efficacité et la difficulté de créer une indexation rapide. Notons aussi l'absence de l'information spatiale sur les positions des couleurs dans les histogrammes couleurs, et leurs sensibilités au changement de luminosité et de contraste comme nous montre la Figure A.6. Aussi ils sont inutilisables pour la comparaison partielle des images [68].

Différentes solutions ont été apportées pour remédier aux inconvénients des histogrammes couleurs. Par exemple les auteurs dans [44] proposent d'enrichir ces histogrammes par des informations concernant la répartition géométrique des couleurs dans l'image. La méthode se base sur une subdivision de l'image, et deux méthodes de cumul d'histogrammes locaux qui sont Histogramme cumulé multiplicatif et Histogramme cumulé additif.

A.1.2.0.4 Histogramme cumulé multiplicatif

Il est définit par :

$$H(c) = \prod_{i=1}^{N} (1 + hi(c)) \forall c \in C, \sum_{i=1}^{N} hi(c) = 0 \tag{A.2}$$

1. N : est le nombre de fenêtres.
2. hi : est histogramme de la i^{eme} fenêtre.
3. $1+hi(c)$: terme de pondération, une valeur élevée de pondération veut dire une grande présence de la couleur c dans la i^{eme} fenêtre et par conséquent, une forte contribution de c dans l'histogramme cumulé multiplicatif.

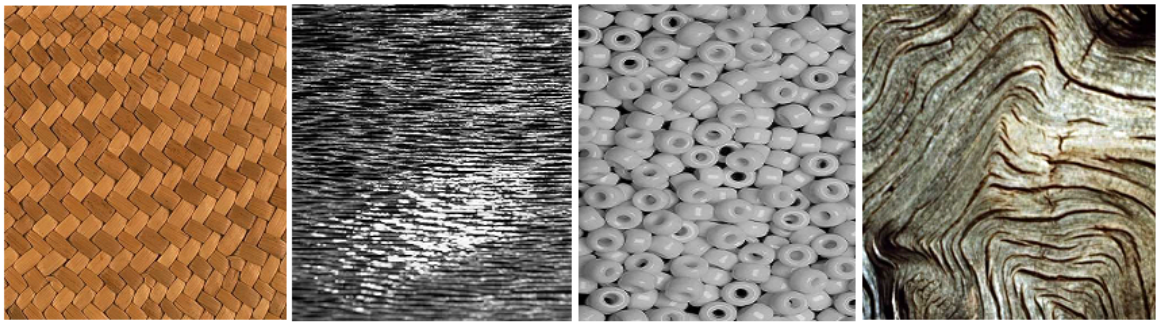


FIGURE A.7 – Exemple de texture

A.1.2.0.5 Histogramme cumulé additif

Il est définit par :

$$H(c) = \sum_{i=0}^{N} g(hi(c)) \forall c \in C \tag{A.3}$$

1. g : fonction de pondération de la présence local des couleurs.
2. $g(hi(c))$: est grand, implique l'importance de la contribution de c dans l'histogramme.
3. on choisi $g(x) = x^{\frac{1}{2}}$.

A.1.2.0.6 La texture

Dans le domaine de traitement des images, la texture est représenté par la répition d'un pixel qui est l'élément de base de l'image selon un ordre particulier. Nous pouvons trouver des exemples de textute faiclement dans la vie quotidienne par exemple : la faience, mosaïque, l'eau, les murs. La Figure A.7, nous montre un exemple de texture.

La texture composant une région est modélisée par les énergies des réponses aux 24 filtres de Gabor. Les filtres de Gabor ont la particularité d'effectuer un filtrage proche de celui réalisé par notre système de perception visuelle. Ils ont une grande sensibilité à l'orientation et à la fréquence. De plus, ils ont l'avantage d'avoir une résolution optimale conjointe en fréquence et dans l'espace. Un ensemble de filtres permet alors de capturer les directions et les fréquences principales de l'image. Un filtre de Gabor est un filtre de fréquence pure modulé par une gaussienne. Dans le domaine fréquentiel, en coordonnées polaires et pour une direction i et une échelle j, il se met sous la forme :

$$G_{i,j}(\rho, \theta) = \exp \frac{(\rho - \mu_{\rho,i,j})^2}{\sigma^2_{\rho,i,j}} + \frac{(\theta - \mu_{\theta,i,j})^2}{\sigma^2_{\rho,i,j}} \tag{A.4}$$

Les paramètres sont ensuite calculés tels que les fréquences centrales $(\mu_{\rho,i,j}, \mu_{\theta,i,j})$ décroissent en octave et que les variances $(\sigma_{\rho,i,j}, \sigma_{\theta,i,j})$ permettent le chevauchement des elipses de Gabor à la moitié de leur amplitude maximale. Cette approche permet d'optimiser la couverture de l'espace tout en limitant la redondance.

A.1.2.0.7 La forme

La forme est une autre caractéristique visuelle importante, qui est toutefois considérée comme une tâche difficile d'être entièrement automatisée, et cela notamment pour la recherche locale. Pour extraire des caractéristiques de forme d'un objet visuel ou d'une région d'une part, les techniques de segmentation d'images sont nécessaires, puis une description géométrique des objets segmentés ou les régions sont appliquées. Dans de nombreux cas, surtout quand une détection précise est nécessaire, une intervention humaine est nécessaire. Un descripteur de forme vise à quantifier la forme visuelle d'un objet. Il existe plusieurs techniques pour caractériser les formes, comme le périmètre, la surface, la zone de délimitation, l'enveloppe convexe, etc. [48]

Parmi les techniques de description locales de formes nous pouvons citer les méthodes suivantes :

— **SIFT** que nous pouvons traduire par « transformation de caractéristiques visuelles invariantes à l'échelle », est un algorithme utilisé dans le domaine de la vision par ordinateur pour détecter et identifier les éléments similaires entre différentes images numériques (éléments de paysages, objets, personnes, etc.). Il a été développé en 1999 par le chercheur David Lowe [63].
Les points-clés (keypoints) sont détectés avec des filtres en cascade qui utilisent des algorithmes efficaces pour identifier les positions candidates. Cette première étape consiste à identifier les positions et les échelles qui peuvent être assignées d'une manière répétitive sur différentes vues du même objet. La recherche des caractéristiques stables permet de détecter les positions invariantes aux changements d'échelles. Cela est fait en utilisant la fonction continue d'espace d'échelle. L'espace d'échelle d'une image peut être défini comme la résultante de la convolution de cette image avec des noyaux gaussiens. Pour une détection efficace des positions des points-clés stables, Lowe [63] a proposé l'utilisation des extrêmes des espaces d'échelle dans la fonction différence de gaussienne convolutée avec l'image $D(x, y, \sigma)$. L'image initiale est incrémentalement convolutée avec des gaussiennes pour produire des images séparées par un facteur constant k dans l'espace d'échelle. Chaque octave de l'espace d'échelle est divisé en un nombre entier s d'intervalle $k = 2^{\frac{1}{s}}$. On produit s+3 images dans la pile des images lisses pour chaque octave. Une fois l'octave traitée, nous rééchantillonnons l'image gaussienne ayant le double de l'échelle initiale en prenant chaque deuxième pixel dans chaque ligne et colonne. La précision de l'échantillonnage relatif σ n'est pas différente de celle du début de l'octave précédente et le calcul est très réduit.
La mise en place de la méthode de Lowe [63] nécessite deux étapes principales. Premièrement, il est nécessaire d'extraire les caractéristiques d'un objet et de calculer ses descripteurs, c'est-à-dire, de détecter les caractéristiques qui sont les plus susceptibles de représenter cet objet, de le définir et de le discriminer par rapport aux autres. Deuxièmement, il faut mettre en place une procédure de mise en correspondance « matching ». C'est le but ultime de la méthode. La Figure A.8 montre les points clés de l'algorithme SIFT.

— **SURF** que nous pouvons traduire par caractéristiques robustes accélérées, est un algorithme de détection de caractéristique. SURF est un descripteur présenté par des chercheurs de l'ETH Zurich et de l'université catholique de Louvain pour la première fois en 2006, puis dans une version révisée en 2008. Il est utilisé dans le domaine de vision par ordinateur, pour des tâches de détection d'objet ou de reconstruction.

L'algorithme de SURF est partiellement inspiré du descripteur SIFT. Selon les auteurs en [18], il

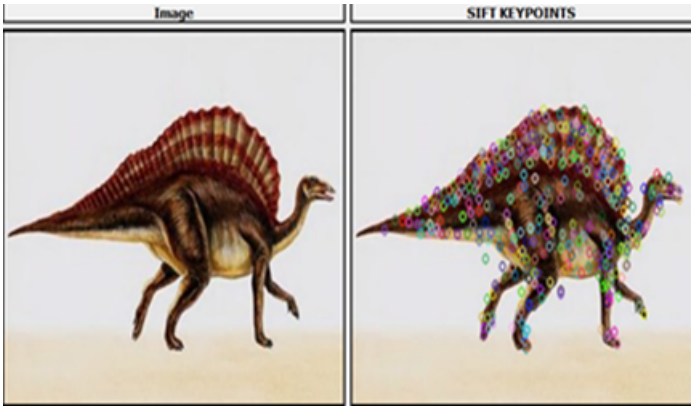


FIGURE A.8 – Les Points clés de SIFT

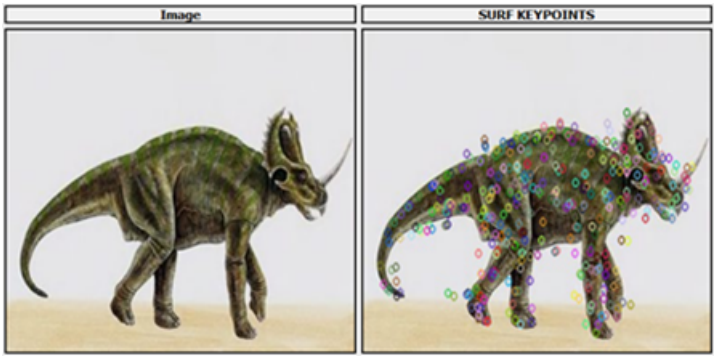


FIGURE A.9 – Les Points clés de SURF

le surpasse en terme de rapidité (temps de calcul). Il est plus robuste par rapport aux différentes transformations d'images. SURF est fondé sur des sommes de réponses d'ondelettes de Haar 2D [21] et utilise efficacement les images intégrales. En tant que caractéristique de base, SURF utilise une approximation d'ondelettes de Haar [21] et de détecteur de blob à base de déterminant hessien. La Figure A.9 montre les points d'intérêt de l'algorithme SURF [18].

Après avoir mentionné les descripteurs visuels, et en se basant sur l'histogramme couleur ainsi que SIFT et SURF, nous allons citer dans ce qui vient, trois distances de similarité pour l'histogramme couleur, et deux distances de similarité que nous allons utiliser dans notre approche.

A.1.3 Les mesures de similarité

En mathématiques et en informatique, la similarité est un critère important pour l'identification de sous-groupes dans un groupe d'objets, de valeurs (numériques ou non), de données (connues ou reconnues) dans un « espace » ou système.

Pour rechercher les images les plus similaires à une image-exemple ou pour les regrouper, il faut pouvoir mesurer la similarité (ou la dissimilarité) des images. Idéalement, les mesures doivent être capables de mesurer la similarité sémantique entre deux images, dans la pratique, elles ne sont capables que de mesurer la similarité visuelle. Cependant, beaucoup de travaux essayent de s'inspirer du système

visuel humain pour proposer des mesures plus efficaces [79].

Nous allons décrire maintenant des mesures de similarité dans le cadre de la recherche d'images similaires. Ces mesures sont également utilisables pour la recherche de régions similaires, mais aussi dans le cas de la classification supervisée ou non supervisée d'images (ou de données au sens général). D'ailleurs, d'après les auteurs dans [16], tout système ayant pour but d'analyser ou d'organiser automatiquement un ensemble de données ou de connaissances doit utiliser, sous une forme ou une autre, un opérateur de similarité dont le but est d'établir les ressemblances ou les relations qui existent entre les informations manipulées.

Il existe un grand nombre de mesures de similarité. Certaines sont des distances, c'est-à-dire des mesures qui ont les propriétés de non-négativité, réflexivité, symétrie et qui respectent l'inégalité triangulaire. Certaines mesures sont spécifiques aux histogrammes ou aux distributions.

A.1.3.0.1 Les distances (Histogramme de Couleur)

- **Chi-Square (Chi-carrée) :** il s'agit d'une mesure de distance qui peut être utilisée pour trouver la di-similarité entre deux histogrammes. Motivé par le fait que la discrimination de texture par le système de vision humain est basée sur les statistiques de second ordre [43].

$$d(H_1, H_2) = \sum_{i=1}^{255} \frac{(H_1(i) - H_2(i))^2}{H_{12}(i)} \tag{A.5}$$

- **Corrélation :** la corrélation est souvent utilisée comme un outil descriptif dans la recherche non expérimentale. Nous disons que deux mesures sont corrélées si elles ont des informations communes. L'intensité de la corrélation est exprimée par un nombre appelé coefficient de corrélation. Ce coefficient est égal au rapport de leur covariance et du produit non nul de leur écartstype. Le coefficient de corrélation est compris entre -1 et 1. Bien que généralement l'intensité de la corrélation est exprimée par un nombre appelé le coefficient de corrélation. Elle été introduite par Galton [91] et plus tard formalisée par Karl Pearson [75], puis par Ficher [54].

$$d(H_1, H_2) = \frac{\sum_i (H_1(I) - \overline{H_1})(H_2(I) - \overline{H_2})}{\sqrt{\sum_i (H_1(I) - \overline{H_1})(H_2(I) - \overline{H_2})}} \tag{A.6}$$

  Où :

$$\overline{H_k} = \frac{1}{N} \sum_j H_k(j) \tag{A.7}$$

  - N : nombre des éléments de l'histogramme.

- **Bhattacharya [3] :** En statistiques, la distance de Bhattacharyya est une mesure de la similarité de deux distributions de probabilités discrètes. Elle est reliée au coefficient de Bhattacharyya, qui est une mesure statistique du recouvrement de deux ensembles d'échantillons. Cette mesure est la plus utilisée pour la mise en correspondance entre deux observations basées sur l'histogramme de couleur. Cette mesure est régulièrement utilisée dans des problèmes de classification, en

FIGURE A.10 – Brute Force Matcher appliqué à SIFT

particulier dans le domaine de la vision par ordinateur [3].

$$d(H_1, H_2) = \sqrt{1 - \frac{1}{\sqrt{\overline{H_1}\overline{H_2}N^2}} \sum_I \sqrt{H_1(I)H_2(I)}} \tag{A.8}$$

A.1.3.0.2 Les distances (SIFT et SURF)

- **Brut-Force matcher :** Brut-Force matcher est le moyen plus simple d'utiliser le descripteur SIFT ou SURF pour la comparaison d'images. Il prend le descripteur d'une caractéristique dans le premier et l'associe à toutes les autres fonctionnalités de la deuxième série en utilisant un certain calcul de distance, et la plus proche est retournée. La Figure A.10 montre Brut-Force matcher appliqué aux descripteurs SIFT.
- **FLANN Based Matcher :** est basé sur l'algorithme de k plus proches voisins. La recherche des plus proches voisins, ou des k plus proches voisins, est un problème algorithmique classique. De façon informelle, le problème consiste, étant donné un point à trouver dans un ensemble d'autres points, quels sont les k plus proches KNN [70].

  La recherche du voisinage est utilisée dans de nombreux domaines, tels que la reconnaissance de formes, le clustering, l'approximation de fonctions, la prédiction de séries temporelles et même les algorithmes de compression. C'est en particulier l'étape principale de la méthode KNN en apprentissage automatique. La Figure A.11 montre FLANN Based Matcher appliqué aux descripteurs SURF.

A.1.4 Mesures pour évaluer un système

L'utilisation d'un système de recherche d'informations exige l'utilisation d'une mesure de performance. Les mesures les plus courantes sont souvent le temps de réponse et l'espace utilisé. Plus le temps de réponse est court, plus l'espace est petit, plus le système est considéré comme bon, mais dans le cas d'un moteur de recherche multimédia en plus de ces deux mesures, nous intéressons à la précision du système. Les deux mesures plus courantes sont le rappel et la précision (en anglais : Recall and

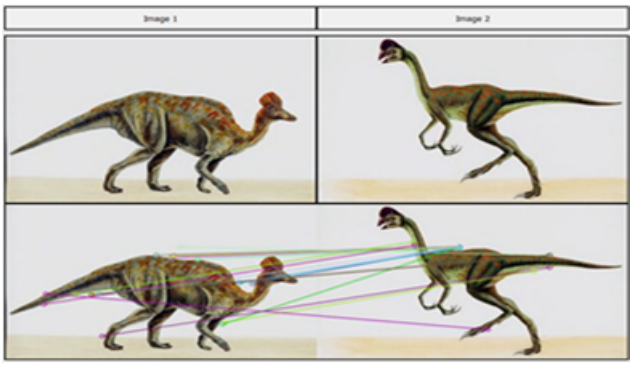


FIGURE A.11 – FLANN Based Matcher appliqué à SURF

Precision).[69]

A.1.4.0.1 Le rappel

Il s'agit du rapport entre le nombre d'images pertinentes dans l'ensemble des images trouvées et le nombre d'images pertinentes dans la base d'images.

$$Rappel = \frac{|Ra|}{R} \tag{A.9}$$

A.1.4.0.2 La précision

La précision est le rapport entre le nombre d'images pertinentes dans l'ensemble des images trouvées et le nombre d'images trouvées.

$$Prcision = \frac{|Ra|}{|A|} \tag{A.10}$$

- R : l'ensemble des images pertinentes dans la base d'images utilisées pour évaluer.
- $|R|$: le nombre d'images pertinentes dans la base d'images.
- A : l'ensemble des réponses.
- $|A|$: le nombre d'images dans l'ensemble des réponses.
- $|Ra|$: le nombre d'images pertinentes dans l'ensemble des réponses.

Parmi les travaux dans le domaine d'indexation et de recherche multimédias, nous pouvons citer les suivants :

1. AudioCycle [31] propose un prototype de navigation dans des bibliothèques de boucles musicales. Il fournit à l'utilisateur une vue graphique des extraits audio qui sont visualisés et organisés en fonction de leur similitude en termes de paramètres musicaux tels que le timbre, l'harmonie et le rythme. L'utilisateur est en mesure de naviguer dans cette représentation visuelle, d'écouter des extraits audio et de rechercher ceux qui l'intéressent. Ce projet s'appuie sur différentes technologies telles que l'analyse des fichiers audio à partir des informations musicales, la visualisation 3D et le rendu sonore spatial. L'application proposée peut être exploitée par les remixeurs, musiciens, compositeurs de bandes sonores ainsi que les artistes.

2. ImageCycle [88] permet la navigation et l'indexation d'images à partir de leur classification prenant en compte les caractéristiques de couleur, de forme, de texture, etc. Ce projet propose également des fonctions de zoom et de rotation des images. L'interface de navigation présente cinq fonctionnalités principales comme nous montre la Figure A.12a :

   (a) **interaction :** trois curseurs (coin haut à droite de la Figure A.12a) permettant à l'utilisateur de définir les poids de la forme, la couleur et la texture. Ces poids offrent une alternative pour classifier les images selon les préférences de l'utilisateur ;

   (b) **visualisation :** la fenêtre principale permet d'afficher les images de telle sorte que la distance entre chaque couple d'images reflète leur degré de similarité calculé à partir des poids définis par l'utilisateur ;

   (c) **classification :** les images sont groupées en fonction de la distance indiquée ci-dessus. L'utilisateur peut naviguer à l'intérieur ou à l'extérieur des classes ;

   (d) **affichage individuel :** lorsque la souris passe au-dessus d'une image, celle-ci devient instantanément plus grande de telle sorte que l'utilisateur peut naviguer rapidement à travers la base de données (Figure A.12a) ;

   (e) **description :** un bouton placé en bas à droite de la fenêtre principale (Figure A.12a) permet d'afficher des informations supplémentaires sur les images.

   La Figure A.12b montre un exemple illustrant l'utilisation du projet MediaCycle pour la navigation dans une base d'images. Cette figure montre les classes regroupant les images ayant des caractéristiques similaires.

3. VideoCycle [93] offre, à son tour, une application de navigation et d'indexation de séquences vidéo. La navigation dans ce cas est effectuée à partir des caractéristiques du mouvement : silhouette, zone du mouvement, moments de Hu [42], contours, etc. L'interface de navigation est similaire à celui de ImageCycle. La Figure A.13 montre une vue de l'application VideoCycle permettant la navigation dans une base de séquences vidéo à partir du calcul de leurs similarités.

A.2 Les méthodes de réduction de dimensionnalité

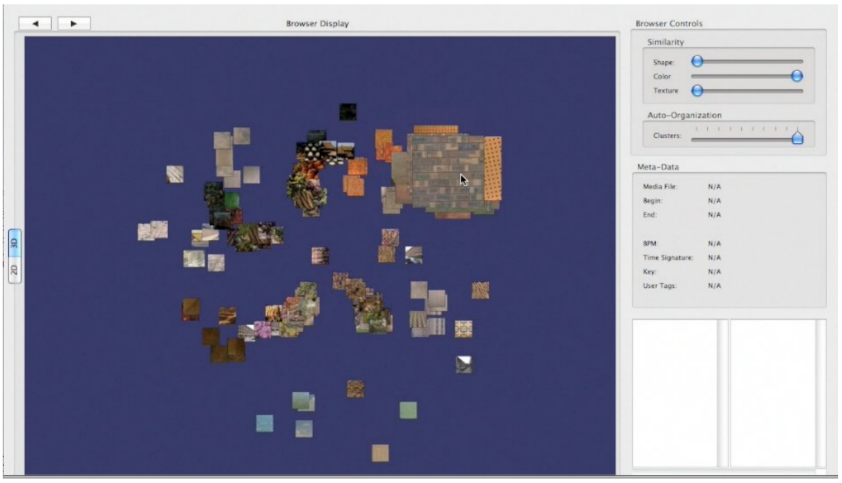
A.2.1 PCA

L'idée principale de PCA est de réduire la dimension d'un jeu de données tout en gardant un maximum d'informations. Cela est réalisé grâce à une projection qui maximise la variance tout en minimisant l'erreur quadratique moyenne de la reconstruction.

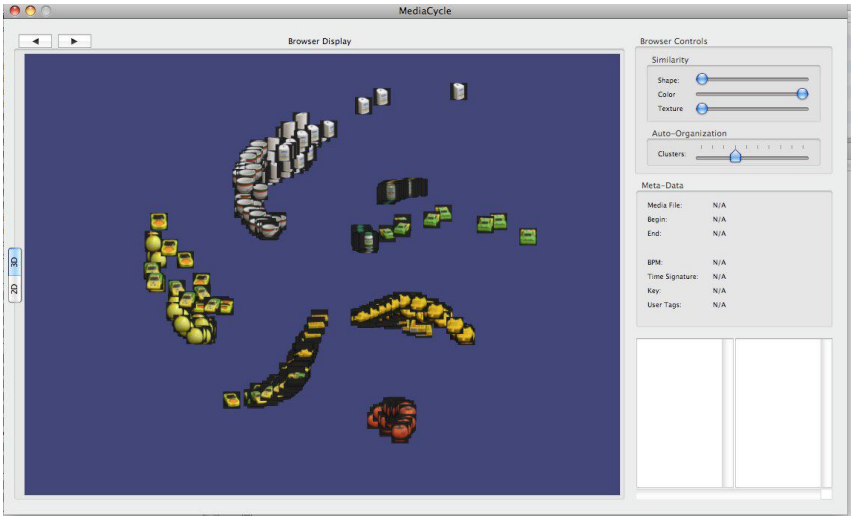
Pour la dérivation, Hotelling définit PCA comme une projection orthogonale maximisant la variance dans l'espace projeté. Étant donné n échantillons $x_i \in R^D$ et $u \in R^D$ tel que :

$$||u|| = U^T u = 1 \tag{A.11}$$

soit un vecteur ortho-normal de projection. Un échantillon x_i est projeté sur u par : $a_i = u^T x_i$ la variance



(a) Navigation dans une base d'images



(b) Indexation d'images avec ImageCycle

FIGURE A.12 – Vue d'ensemble de ImageCycle [88].

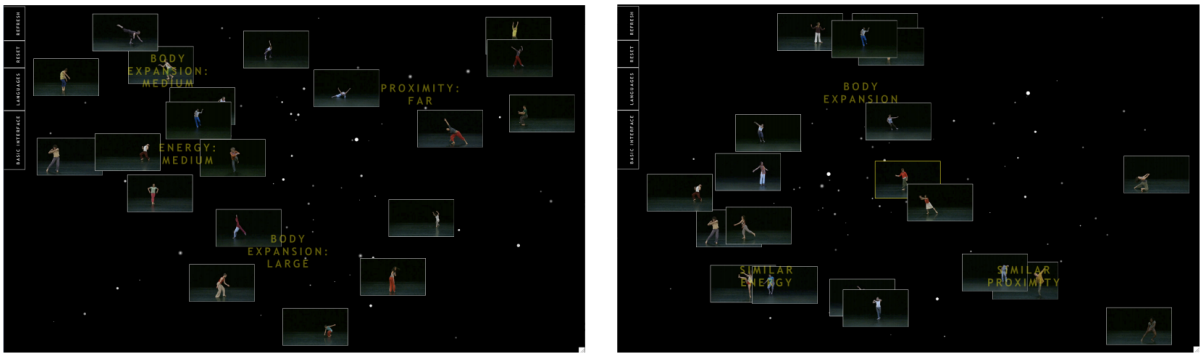


FIGURE A.13 – VideoCycle : Navigation dans des bases de séquences vidéo [93].

de l'échantillon peut donc être estimée :

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x}) \tag{A.12}$$

ou $xmoy$ est la moyenne des projetés des échantillons de la base :

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i d'ou \bar{a} = u^T \bar{x} \tag{A.13}$$

Ainsi la variance du projeté est donnée par :

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(a_i - \bar{a}) \tag{A.14}$$

$$= \frac{1}{n-1}\sum_{i=1}^{n}(u^T x_i - u^T \bar{x}) \tag{A.15}$$

$$= u^T C u \tag{A.16}$$

$$O \quad C \in R^{DxD} = \frac{1}{n-1}\sum_{i=1}^{n}((x_i - \bar{x})(x_i - \bar{x})^T) \tag{A.17}$$

est la matrice de covariance de $X = [x_1; ....; x_n] \in R^{Dxn}$. Le problème de maximisation de la variance dans l'espace projeté peut donc s'écrire : $max\ u^T C u$ avec $u^T u = 1$.

Le calcul de la solution optimale peut être réalisé grâce au multiplicateur de Lagrange :

$$f(u, \lambda) = u^T C U + \lambda(1 - u^T u) \tag{A.18}$$

$$Par\ derivation\ partielle\ selon\ u : \quad \frac{\gamma f(u,\lambda)}{\gamma u} = 2Cu - 2\lambda u = 0 \tag{A.19}$$

$$on\ obtient \qquad Cu = \lambda u \tag{A.20}$$

Ainsi, le maximum pour le multiplicateur de Lagrange est obtenu si λ est une valeur propre et u un vecteur propre de C. Ainsi la variance décrite par le vecteur de projection u est donnée par λ.

Calcul de PCA pour la mise en œuvre de méthodes : il est supposé que le jeu de données d'entraînement est disponible en entier. Ainsi nous avons un ensemble de n observations $x_i \in R^D$ organisés sous forme matricielle $X = [x_1; .....; x_n] \in R^{Dxn}$. L'estimation de la base de projection de l'ACP revient donc à estimer les éléments propres de la matrice de covariance C de X. Le calcul requiert d'abord l'échantillon moyen :

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{A.21}$$

Ensuite les échantillons sont normalisés par rapport à la moyenne x_{moy} : $\widehat{x} = \bar{x_i} = x_i - \bar{x}$ pour former la

nouvelle matrice $\widehat{x} = [\widehat{x_1}; ....; \widehat{x_n}]$. La matrice de covariance $C \in R^{DxD}$ est ensuite calculée par :

$$C = \frac{1}{n-1}\widetilde{x}\widetilde{x}^T \tag{A.22}$$

La recherche des éléments propres de C conduit à l'obtention de la base de vecteurs propres $u_i \in R^D$, pour lesquels, chacun d'eux, est associée une valeur propre λ_i. Généralement triés par ordre décroissant de valeur propre associée, les premiers vecteurs propres forment alors une base dans laquelle la plupart de l'information du jeu de données d'entraînement est gardée.

La dimension de la matrice de covariance dépend de la dimension D des vecteurs du jeu de données, qui peut être relativement grande pour certains types de données (typiquement des images). La méthode décrite plus haut devient alors difficile à appliquer, essentiellement à cause de la recherche des éléments propres de la matrice de covariance C. En effet, pour des images de taille 100×100 par exemple, la matrice de covariance C à inverser est de taille 10000×10000. Cependant, il est connu que pour toute matrice X, les produits matriciels XX^T et X^TX partagent les mêmes valeurs propres différentes de zéro.

Ainsi, le calcul des éléments propres de $C = XX^TC \in R^{DxD}$ peut se ramener au calcul des éléments propres de la matrice $M \in R^{nxn}$ ou $M = X^TX$. Soit e_i les vecteurs propres de M associés aux valeurs propres γ_i. On a donc :

$$X^TXe_i = \gamma_i e_i \tag{A.23}$$

En multipliant à gauche par X les deux côtés de l'équation, on obtient ainsi :

$$X(X^TXe_i) = X(\gamma_i e_i) \tag{A.24}$$

$$XX^T(Xe_i) = \gamma_i(Xe_i) \tag{A.25}$$

On voit donc que Xe_i est vecteur propre de XX^T tel que γ_i est la valeur propre associée, d'où

$$\begin{cases} \mu i &= Xei \\ \lambda i &= \gamma i \end{cases} \tag{A.26}$$

La matrice M étant beaucoup plus petite que la matrice C (typiquement, nous passons d'une complexité de l'ordre de la dimension des échantillons à une complexité de l'ordre du nombre d'échantillons d'apprentissage), les calculs sont donc plus efficaces. L'analyse en Composantes Principales est résumée dans Algorithme A.1.

Algorithme A.1 Calcul de l'ACP

Entrée : matrice X ;

Sortie : vecteur moyen $\bar{x}$, base de vecteurs propres U, valeurs propres associées λ_i ;

Calcul du vecteur moyen : ;

$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$

Normalisation des images d'entrées :

$\widehat{x} = x_i - \bar{x}$

$\widehat{x} = [\widehat{x}_1; .....; \widehat{x}_n]$.

si Données de grande dimension **alors**

M=$\frac{1}{n-1} \widehat{x}\widehat{x}^T$

Calcul des éléments propres de M :

E=$[e_1 ; ..... ; e_n]$

γ=$[\gamma_1 ; ..... ; \gamma_n]$

Calcul des éléments finaux :

$u_i = X e_i \; U = [u_1; .....; u_n]$

$\lambda_i = \gamma_1 \; \lambda = [\lambda_1; .....; \lambda_n]$

sinon

$C = \frac{1}{n-1} \widehat{x}\widehat{x}^T$

Calcul des éléments propres de C :

$U = [u_1; .....; u_n]$

$\lambda = [\lambda_1; .....; \lambda_n]$

fin si

retourner $\bar{x}, U, \lambda$;

Bibliographie

[1] H. Abdel-Nabi, G. Al-Naymat et A. Awajan. « Content Based Image Retrieval Approach using Deep Learning ». In : *2019 2nd International Conference on new Trends in Computing Sciences (ICTCS)*. IEEE. 2019, p. 1-8 (cf. p. 16).

[2] M. A. Abdulhayoglu et B. Thijs. « Use of locality sensitive hashing (LSH) algorithm to match Web of Science and Scopus ». In : *Scientometrics* 116.2 (2018), p. 1229-1245 (cf. p. 50).

[3] F. J. Aherne, N. A. Thacker et P. I. Rockett. « The Bhattacharyya metric as an absolute similarity measure for frequency coded data ». In : *Kybernetika* 34.4 (1998), p. 363-368 (cf. p. 115, 116).

[4] F. Alam, R. Mehmood et I. Katib. « Comparison of decision trees and deep learning for object classification in autonomous driving ». In : *Smart Infrastructure and Applications*. Springer, 2020, p. 135-158 (cf. p. 67).

[5] M. Alrahhal et K. Supreethi. « Multimedia Image Retrieval System by Combining CNN With Handcraft Features in Three Different Similarity Measures ». In : *International Journal of Computer Vision and Image Processing (IJCVIP)* 10.1 (2020), p. 1-23 (cf. p. 15).

[6] M. Aly, M. Munich et P. Perona. « Distributed kd-trees for retrieval from very large image collections ». In : *Proceedings of the British Machine Vision Conference (BMVC)*. T. 17. 2011 (cf. p. 36, 38).

[7] A. B. Ayed, M. B. Halima et A. M. Alimi. « MapReduce Based Text Detection in Big Data Natural Scene Videos ». In : *Procedia Computer Science* 53.Supplement C (2015). INNS Conference on Big Data 2015 Program San Francisco, CA, USA 8-10 August 2015, p. 216 -223. doi : https://doi.org/10.1016/j.procs.2015.07.297 (cf. p. 38).

[8] E. Becker, J.-P. Cardinal, M.-F. Roy et Z Szafraniec. « Multivariate Bezoutians, Kronecker symbol and Eisenbud-Levine formula ». In : *Algorithms in algebraic geometry and applications*. Springer, 1996, p. 79-104 (cf. p. 111).

[9] M. A. Belarbi, S. Mahmoudi et G. Belalem. « Indexing Video by the Content ». In : *Information Systems Design and Intelligent Applications : Proceedings of Third International Conference INDIA 2016, Volume 2*. Sous la dir. de S. C. Satapathy, J. K. Mandal, S. K. Udgata et V. Bhateja. T. 434. New Delhi : Springer India, 2016, p. 21-33 (cf. p. 86, 87).

[10] M. A. Belarbi, S. A. Mahmoudi, S. Mahmoudi et G. Belalem. « A new parallel and distributed approach for large scale images retrieval ». In : *International Conference of Cloud Computing Technologies and Applications*. Springer. 2018, p. 185-201 (cf. p. 56, 86).

[11] P. N. Belhumeur, J. P. Hespanha et D. J. Kriegman. « Eigenfaces vs. fisherfaces : Recognition using class specific linear projection ». In : *IEEE Transactions on pattern analysis and machine intelligence* 19.7 (1997), p. 711-720 (cf. p. 23).

[12] P. N. Belhumeur, J. P. Hespanha et D. J. Kriegman. « Eigenfaces vs. fisherfaces : Recognition using class specific linear projection ». In : *IEEE Transactions on pattern analysis and machine intelligence* 19.7 (1997), p. 711-720 (cf. p. 32).

[13] J. L. Bentley. « Multidimensional binary search trees in database applications ». In : *IEEE Transactions on Software Engineering* 4 (1979), p. 333-340 (cf. p. 28).

[14] S. Berchtold, C. Bohm, H. V. Jagadish, H.-P. Kriegel et J. Sander. « Independent quantization : An index compression technique for high-dimensional data spaces ». In : *Data Engineering, 2000. Proceedings. 16th International Conference on*. IEEE. 2000, p. 577-588 (cf. p. 20).

[15] E. Bingham et H. Mannila. « Random projection in dimensionality reduction : applications to image and text data ». In : *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2001, p. 245-250 (cf. p. 25).

[16] G. Bisson, C. Nédellec et D. Canamero. « Designing Clustering Methods for Ontology Building-The Mo'K Workbench. » In : *ECAI workshop on ontology learning*. T. 31. 2000 (cf. p. 115).

[17] S. Blott et R. Weber. « A simple vector-approximation file for similarity search in high-dimensional vector spaces ». In : *ESPRIT Technical Report TR19, ca* (1997) (cf. p. 18).

[18] H. J. Bouchech, S. Foufou et M. Abidi. « Strengthening surf descriptor with discriminant image filter learning : application to face recognition ». In : *Microelectronics (ICM), 26th International Conference on*. IEEE. 2014, p. 136-139 (cf. p. 113, 114).

[19] A. Z. Broder, M. Charikar et M. Mitzenmacher. « A derandomization using min-wise independent permutations ». In : *International Workshop on Randomization and Approximation Techniques in Computer Science*. Springer. 1998, p. 15-24 (cf. p. 26).

[20] H. Cai, X. Wang et Y. Wang. « Compact and robust fisher descriptors for large-scale image retrieval ». In : *Machine Learning for Signal Processing (MLSP), IEEE International Workshop on*. IEEE. 2011, p. 1-6 (cf. p. 24).

[21] C. Chen et C. Hsiao. « Haar wavelet method for solving lumped and distributed-parameter systems ». In : *IEE Proceedings-Control Theory and Applications* 144.1 (1997), p. 87-94 (cf. p. 114).

[22] T. Chen, M. Nakazato et T. S. Huang. « Speeding up the similarity search in multimedia database ». In : *Multimedia and Expo, 2002. ICME'02. Proceedings. IEEE International Conference on*. T. 2. IEEE. 2002, p. 509-512 (cf. p. 18).

[23] S. Cheng, L. Wang et A. Du. « An adaptive and asymmetric residual hash for fast image retrieval ». In : *IEEE Access* 7 (2019), p. 78942-78953 (cf. p. 27).

[24] P. Chhabra, N. K. Garg et M. Kumar. « Content-based image retrieval system using ORB and SIFT features ». In : *Neural Computing and Applications* 32.7 (2020), p. 2725-2733 (cf. p. 25).

[25] S. Chris, L Liu, A Sean et L Jason. « HIPI : A hadoop image processing interface for image-based map reduce tasks ». In : *University of Virginia* (2011) (cf. p. 39, 94-97).

[26] D. Comer. « Ubiquitous B-tree ». In : *ACM Computing Surveys (CSUR)* 11.2 (1979), p. 121-137 (cf. p. 28).

[27] I. Daoudi, K. Idrissi, S. Ouatik, A. Baskurt et D. Aboutajdine. « An efficient high-dimensional indexing method for content-based retrieval in large image databases ». In : *Signal Processing : Image Communication* 24.10 (2009), p. 775-790 (cf. p. 19).

[28] I. Daoudi, S. Ouatik, A. El Kharraz, K. Idrissi et D. Aboutajdine. « Vector Approximation based Indexing for High-Dimensional Multimedia Databases. » In : *Engineering Letters* 16.2 (2008) (cf. p. 20).

[29] R. Datta, D. Joshi, J. Li et J. Z. Wang. « Image retrieval : Ideas, influences, and trends of the new age ». In : *ACM Computing Surveys (Csur)* 40.2 (2008), p. 5 (cf. p. 10).

[30] R Dony et al. « Karhunen-loeve transform ». In : *The transform and data compression handbook* 1 (2001), p. 1-34 (cf. p. 22, 23).

[31] S. Dupont, N. d'Alessandro, T. Dubuisson, C. Frisson, R. Sebbe et J. Urbain. « Audio Cycle ». In : *QPSR of the numediart research program*. Sous la dir. de T. Dutoit et B. Macq. T. 1. 4. numediart Research Program on Digital Art Technologies. Déc. 2008, p. 119-127. URL : http://www.numediart.org/docs/numediart_2008_s04_p1_report.pdf (cf. p. 117).

[32] M. El Adoui, S. A. Mahmoudi, M. A. Larhmam et M. Benjelloun. « MRI Breast Tumor Segmentation Using Different Encoder and Decoder CNN Architectures ». In : *Computers* 8.3 (2019), p. 52 (cf. p. 67).

[33] D. Fernandez, C. Gonzalez, D. Mozos et S. Lopez. « FPGA implementation of the principal component analysis algorithm for dimensionality reduction of hyperspectral images ». In : *Journal of Real-Time Image Processing* (2016). DOI : 10.1007/s11554-016-0650-7 (cf. p. 24).

[34] C. Fleury. « Le KD-Tree : une methode de subdivision spatiale ». In : *Universite de Rennes* 1 (2008) (cf. p. 28).

[35] H. Gaber, M. Marey, S. E. Amin et M. F. Tolba. « Content based image retrieval with hadoop ». In : *International Conference on Advanced Intelligent Systems and Informatics*. Springer. 2016, p. 257-265 (cf. p. 39).

[36] A. Gionis, P. Indyk, R. Motwani et al. « Similarity search in high dimensions via hashing ». In : *Vldb*. T. 99. 6. 1999, p. 518-529 (cf. p. 25).

[37] A. Goyal, S. B. Anandamurthy, P. Dash, S. Acharya, D. Bathla, D. Hicks, A. Bhan et P. Ranjan. « Automatic Border Surveillance Using Machine Learning in Remote Video Surveillance Systems ». In : *Emerging Trends in Electrical, Communications, and Information Technologies*. Springer, 2020, p. 751-760 (cf. p. 86).

[38] A. Guttman. *R-trees : A dynamic index structure for spatial searching*. T. 14. 2. ACM, 1984 (cf. p. 29).

[39] H. Heidari, A. Chalechale et A. A. Mohammadabadi. « Parallel implementation of color based image retrieval using CUDA on the GPU ». In : *International Journal of Information Technology and Computer Science (IJITCS)* 6.1 (2013), p. 33 (cf. p. 108).

[40] A. Henrich, H.-W. Six, F. Hagen et P. Widmayer. « The LSD tree : spatial access to multidimensional point and non-saint objects ». In : *the 5th Very Large Databases Conference*. 1989, p. 45-54 (cf. p. 31).

[41] J.-C. Heudin. *Comprendre le deep learning : une introduction aux réseaux de neurones*. Science-eBook, 2016 (cf. p. 10).

[42] M. K. Hu. « Visual Pattern Recognition by Moment Invariants ». In : *IRE Transactions on Information Theory IT-8* (1962), p. 179-187 (cf. p. 118).

[43] G. Hua, M. Brown et S. Winder. « Discriminant embedding for local image descriptors ». In : *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE. 2007, p. 1-8 (cf. p. 24, 115).

[44] J. Huang, S. R. Kumar, M. Mitra, W.-J. Zhu et R. Zabih. « Image indexing using color correlograms ». In : *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*. IEEE. 1997, p. 762-768 (cf. p. 111).

[45] Z. Huang, H. Shen, J. Liu et X. Zhou. « Effective data co-reduction for multimedia similarity search ». In : *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. ACM. 2011, p. 1021-1032 (cf. p. 21).

[46] D. M. Hussain et D Surendran. « The efficient fast-response content-based image retrieval using spark and MapReduce model framework ». In : *Journal of Ambient Intelligence and Humanized Computing* (2020), p. 1-8 (cf. p. 38).

[47] S. Ibrahim, H. Jin, L. Lu, L. Qi, S. Wu et X. Shi. « Evaluating mapreduce on virtual machines : The hadoop case ». In : *IEEE International Conference on Cloud Computing*. Springer. 2009, p. 519-528 (cf. p. 36).

[48] F. Idris et S. Panchanathan. « Review of image and video indexing techniques ». In : *Journal of visual communication and image representation* 8.2 (1997), p. 146-166 (cf. p. 113).

[49] P. Indyk et R. Motwani. « Approximate nearest neighbors : towards removing the curse of dimensionality ». In : *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. ACM. 1998, p. 604-613 (cf. p. 26).

[50] U. Jayaraman, S. Prakash et P. Gupta. « Indexing Multimodal Biometric Databases Using Kd-Tree with Feature Level Fusion ». In : *Information Systems Security : 4th International Conference, ICISS 2008, Hyderabad, India, December 16-20*. Sous la dir. de R. Sekar et A. K. Pujari. Berlin, Heidelberg : Springer Berlin Heidelberg, 2008, p. 221-234. doi : 10.1007/978-3-540-89862-7_19 (cf. p. 32).

[51] H. Jegou, M. Douze et C. Schmid. « Hamming embedding and weak geometric consistency for large scale image search ». In : *European conference on computer vision*. Springer. 2008, p. 304-317 (cf. p. 26).

[52] H. Jégou, M. Douze, C. Schmid et P. Pérez. « Aggregating local descriptors into a compact image representation ». In : *Computer Vision and Pattern Recognition (CVPR), IEEE Conference*. IEEE. 2010, p. 3304-3311 (cf. p. 24).

[53] N. Katayama et S. Satoh. « The SR-tree : An index structure for high-dimensional nearest neighbor queries ». In : *ACM Sigmod Record* 26.2 (1997), p. 369-380 (cf. p. 29-31).

[54] O. Kempthorne. « The correlation between relatives on the supposition of Mendelian inheritance ». In : *American journal of human genetics* 20.4 (1968), p. 402 (cf. p. 115).

[55] A. Khan, E. Huerta, S. Wang, R. Gruendl, E. Jennings et H. Zheng. « Deep learning at scale for the construction of galaxy catalogs in the Dark Energy Survey ». In : *Physics Letters B* 795 (2019), p. 248-258 (cf. p. 68).

[56] N Krishnan, M. S. Banu et C. C. Christiyana. « Content based image retrieval using dominant color identification based on foreground objects ». In : *Conference on Computational Intelligence and Multimedia Applications, 2007. International Conference on*. T. 3. IEEE. 2007, p. 190-194 (cf. p. 110).

[57] K. Kruthika, H. Maheshappa, A. D. N. Initiative et al. « CBIR system using Capsule Networks and 3D CNN for Alzheimer's disease diagnosis ». In : *Informatics in Medicine Unlocked* 14 (2019), p. 59-68 (cf. p. 15).

[58] X. Li, L. Chen, L. Zhang, F. Lin et W.-Y. Ma. « Image annotation by large-scale content-based image retrieval ». In : *Proceedings of the 14th ACM international conference on Multimedia*. ACM. 2006, p. 607-610 (cf. p. 21).

[59] S. Lian. « Automatic video temporal segmentation based on multiple features ». In : *Soft computing* 15.3 (2011), p. 469-482 (cf. p. 91).

[60] A. Likas, N. Vlassis et J. J. Verbeek. « The global k-means clustering algorithm ». In : *Pattern recognition* 36.2 (2003), p. 451-461 (cf. p. 88).

[61] K. Lin, H.-F. Yang, J.-H. Hsiao et C.-S. Chen. « Deep Learning of Binary Hash Codes for Fast Image Retrieval ». In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2015 (cf. p. 27).

[62] W. Lipin et P. Juncheng. « Image Retrieval based on VA-File and Multi-Resolution BOW ». In : *Applied Mathematics & Information Sciences* 9.1 (2015), p. 445 (cf. p. 22).

[63] D. G. Lowe. « Distinctive image features from scale-invariant keypoints ». In : *International journal of computer vision* 60.2 (2004), p. 91-110 (cf. p. 113).

[64] Q. Lv, W. Josephson, Z. Wang, M. Charikar et K. Li. « Multi-probe LSH : efficient indexing for high-dimensional similarity search ». In : *Proceedings of the 33rd international conference on Very large data bases*. VLDB Endowment. 2007, p. 950-961 (cf. p. 26).

[65] L. v. d. Maaten et G. Hinton. « Visualizing data using t-SNE ». In : *Journal of machine learning research* 9.Nov (2008), p. 2579-2605 (cf. p. 23).

[66] S. Maji et S. Bose. « CBIR using features derived by Deep Learning ». In : *arXiv preprint arXiv :2002.07877* (2020) (cf. p. 16).

[67] G. L. Marcialis et F. Roli. « Fusion of LDA and PCA for Face Verification ». In : *Biometric Authentication* 2359 (2002), p. 30-37 (cf. p. 24).

[68] O Marinov, M. Deen et B Iniguez. « Charge transport in organic and polymer thin-film transistors : Recent issues ». In : *IEE Proceedings-Circuits, Devices and Systems* 152.3 (2005), p. 189-209 (cf. p. 111).

[69] K. Mikolajczyk et C. Schmid. « A performance evaluation of local descriptors ». In : *IEEE transactions on pattern analysis and machine intelligence* 27.10 (2005), p. 1615-1630 (cf. p. 117).

[70] M. Muja et D. G. Lowe. « Fast matching of binary features ». In : *Computer and Robot Vision (CRV), Ninth Conference on*. IEEE. 2012, p. 404-410 (cf. p. 116).

[71] N.-V. Nguyen, C. Rigaud et J.-C. Burie. « Digital comics image indexing based on deep learning ». In : *Journal of Imaging* 4.7 (2018), p. 89 (cf. p. 15).

[72] D. Nister et H. Stewenius. « Scalable recognition with a vocabulary tree ». In : *Computer vision and pattern recognition, IEEE computer society conference on*. T. 2. Ieee. 2006, p. 2161-2168 (cf. p. 26).

[73] C. Palai, S. R. Pattanaik et P. K. Jena. « Significance of the Background Image Texture in CBIR System ». In : *Computational Intelligence in Data Mining*. Springer, 2020, p. 489-500 (cf. p. 72).

[74] B. V. Patel et B. B. Meshram. « Content based video retrieval systems ». In : *CoRR* abs/1205.1641 (2012). arXiv : 1205.1641 (cf. p. 87, 88).

[75] K. Pearson. « X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling ». In : *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 50.302 (1900), p. 157-175 (cf. p. 115).

[76] F. Perronnin, Y. Liu, J. Sánchez et H. Poirier. « Large-scale image retrieval with compressed fisher vectors ». In : *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*. IEEE. 2010, p. 3384-3391 (cf. p. 26).

[77] G. L. Priya et S Domnic. « Shot based keyframe extraction for ecological video indexing and retrieval ». In : *Ecological informatics* 23 (2014), p. 107-117 (cf. p. 90, 91, 93).

[78] P. Punitha et D. Guru. « Symbolic image indexing and retrieval by spatial similarity : An approach based on B-tree ». In : *Pattern Recognition* 41.6 (2008), p. 2068 -2085. doi : https://doi.org/10.1016/j.patcog.2007.09.012 (cf. p. 32).

[79] Y. Raja, S. J. McKenna et S. Gong. « Tracking and segmenting people in varying lighting conditions using colour ». In : *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on*. IEEE. 1998, p. 228-233 (cf. p. 111, 115).

[80] R. Ramya, K. Venugopal, S. Iyengar et L. Patnaik. « Feature Extraction and Duplicate Detection for Text Mining : A Survey ». In : *Global Journal of Computer Science and Technology* 16.5 (2017) (cf. p. 38).

[81] P. Ruamviboonsuk, J. Krause, P. Chotcomwongse, R. Sayres, R. Raman, K. Widner, B. J. Campana, S. Phene, K. Hemarat, M. Tadarati et al. « Deep learning versus human graders for classifying diabetic retinopathy severity in a nationwide screening program ». In : *NPJ digital medicine* 2.1 (2019), p. 1-9 (cf. p. 12).

[82] Ruigang Fu, Biao Li, Yinghui Gao et Ping Wang. « Content-based image retrieval based on CNN and SVM ». In : *2nd IEEE International Conference on Computer and Communications (ICCC)*. 2016, p. 638-642 (cf. p. 15).

[83] C. Ryu, D. Lee, M. Jang, C. Kim et E. Seo. « Extensible video processing framework in apache hadoop ». In : *Cloud Computing Technology and Science (CloudCom), IEEE 5th International Conference on*. T. 2. IEEE. 2013, p. 305-310 (cf. p. 34).

[84] P. Sadeghi-Tehran, P. Angelov, N. Virlet et M. J. Hawkesford. « Scalable database indexing and fast image retrieval based on deep learning and hierarchically nested structure applied to remote sensing and plant biology ». In : *Journal of Imaging* 5.3 (2019), p. 33 (cf. p. 16).

[85] Y. Sakurai, M. Yoshikawa, S. Uemura, H. Kojima et al. « The A-tree : An index structure for high-dimensional spaces using relative approximation ». In : *VLDB*. T. 2000. Citeseer. 2000, p. 516-526 (cf. p. 20).

[86] F. S. Samaria et A. C. Harter. « Parameterisation of a stochastic model for human face identification ». In : *Applications of Computer Vision, Proceedings of the Second IEEE Workshop on*. IEEE. 1994, p. 138-142 (cf. p. 32).

[87] K. Shvachko, H. Kuang, S. Radia et R. Chansler. « The hadoop distributed file system ». In : *Mass storage systems and technologies (MSST), 26th symposium on*. IEEE. 2010, p. 1-10 (cf. p. 34, 35).

[88] X Siebert, S Dupont, P Fortemps et D Tardieu. « Media Cycle : Browsing and Performing with Sound and Image libraries. » In : *in QPSR of the numediart research program* (2009), p. 19-22 (cf. p. 118, 119).

[89] V. Singh, B. Zong et A. K. Singh. « Nearest keyword set search in multi-dimensional datasets ». In : *ieee transactions on knowledge and data engineering* 28.3 (2015), p. 741-755 (cf. p. 32).

[90] B. Soni, A. Borah, P. N. L. Sowgandhi, P. Sarma et E. F. Shiferaw. « KTRICT A KAZE Feature Extraction : Tree and Random Projection Indexing-Based CBIR Technique ». In : *International Journal of Multimedia Data Engineering and Management (IJMDEM)* 11.2 (2020), p. 49-65 (cf. p. 21).

[91] S. M. Stigler. « Francis Galton's Account of the Invention of Correlation ». In : *Statistical Science* 4.2 (1989), p. 73-79 (cf. p. 115).

[92] M. J. Swain et D. H. Ballard. « Color indexing ». In : *International journal of computer vision* 7.1 (1991), p. 11-32 (cf. p. 110).

[93] D. Tardieu, R. Chessini, J. Dubois, S. Dupont, S. Hidot, B. Mazzarino, A. Moinet, X. Siebert, G. Varni et A. Visentin. « Video Navigation Tool : Application to browsing a database of dancers' performances ». In : *QPSR of the numediart research program*. Sous la dir. de T. Dutoit et B. Macq. T. 2. 3. numediart Research Program on Digital Art Technologies. Sept. 2009, p. 85-90. URL : http://www.numediart.org/docs/numediart_2009_s07_p2_report.pdf (cf. p. 118, 119).

[94] G. Tolias, R. Sicre et H. Jégou. « Particular object retrieval with integral max-pooling of CNN activations ». In : (2015). arXiv : 1511.05879 [cs.CV] (cf. p. 14, 78).

[95] L. V. Tran et R. Lenz. « PCA-based representation of color distributions for color-based image retrieval ». In : *Image Processing. Proceedings. International Conference on*. T. 2. IEEE. 2001, p. 697-700 (cf. p. 24).

[96] Y. Wang, F. Liu, Z. Pang, A. Hassan et W. Lu. « Privacy-preserving content-based image retrieval for mobile computing ». In : *Journal of Information Security and Applications* 49 (2019), p. 102399 (cf. p. 27).

[97] R. Weber, H.-J. Schek et S. Blott. « A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces ». In : *VLDB*. T. 98. 1998, p. 194-205 (cf. p. 18).

[98] D. A. White et R. Jain. « Similarity indexing with the SS-tree ». In : *Proceedings of the Twelfth International Conference on Data Engineering*. 1996, p. 516-523. DOI : 10.1109/ICDE.1996.492202 (cf. p. 32).

[99] R. Xia, Y. Pan, H. Lai, C. Liu et S. Yan. « Supervised hashing for image retrieval via image representation learning. » In : *AAAI*. T. 1. 2014, p. 2 (cf. p. 27).

[100] Y. Yan, M.-L. Shyu et Q. Zhu. « Negative correlation discovery for big multimedia data semantic concept mining and retrieval ». In : *Semantic Computing (ICSC), Tenth International Conference on*. IEEE. 2016, p. 55-62 (cf. p. 38).

[101] H.-W. Yoo, H.-J. Ryoo et D.-S. Jang. « Gradual shot boundary detection using localized edge blocks ». In : *Multimedia Tools and Applications* 28.3 (2006), p. 283-300 (cf. p. 91).

[102] L. Zhang, J. Tan, D. Han et H. Zhu. « From machine learning to deep learning : progress in machine intelligence for rational drug discovery ». In : *Drug discovery today* 22.11 (2017), p. 1680-1685 (cf. p. 67).

[103] Z. Zhang, W. Li et B. Li. « An improving technique of color histogram in segmentation-based image retrieval ». In : *Information Assurance and Security. IAS'09. Fifth International Conference on*. T. 2. IEEE. 2009, p. 381-384 (cf. p. 110).

[104] X. S. Zhou et T. S. Huang. « Edge-based structural features for content-based image retrieval ». In : *Pattern recognition letters* 22.5 (2001), p. 457-468 (cf. p. 72).