Découverte et exploitation d'objets visuels fréquents dans des collections multimédias

Pierre Letessier

Doctorat ParisTech

THESE

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité Signal et Image

présentée et soutenue publiquement par

Pierre LETESSIER

le 28 mars 2013

Découverte et exploitation d'objets visuels fréquents dans des collections multimédias

Directrice de thèse : **Nozha BOUJEMAA**
Encadrement de la thèse : **Olivier BUISSON** et **Alexis JOLY**

Jury

Dr. Georges QUENOT (HDR), Responsable du groupe MRIM, LIG	Président et Rapporteur
Pr. Stéphane MARCHAND-MAILLET, Responsable du groupe VIPER, Université de Genève	Rapporteur
Dr. Patrick PEREZ (HDR), Technicolor	Examinateur
Dr. Nozha BOUJEMAA (HDR), Directrice de l'INRIA Saclay	Examinatrice
Dr. Olivier BUISSON, Groupe de recherche audiovisuelle, Ina	Examinateur
Dr. Alexis JOLY, INRIA Sophia-Antipolis, équipe ZENITH	Examinateur

Table des matières

Table des figures

Liste des tableaux

ELSEVIER

http ://thecostofknowledge.com

Par cette occasion, je souhaite témoigner de mon refus de participer à l'enrichissement de certains éditeurs scientifiques comme Elsevier, Springer Verlag, Association for Computing Machinery, IEEE Publications, ou tout autre éditeur ne pratiquant pas des tarifs à la mesure de ses frais de fonctionnement.

Remerciements

Je tiens tout d'abord à remercier chaleureusement Alexis Joly et Olivier Buisson, qui m'ont encadré pendant ces trois dernières années. J'ai énormément appris à leurs cotés, et je leur en suis extrêmement reconnaissant.

J'aimerais également témoigner toute ma gratitude à Nozha Boujemaa, qui a accepté de diriger ma thèse et qui m'a accueilli dans l'équipe IMEDIA de l'INRIA Rocquencourt.

Merci à Marie-Luce Viaud, Agnès Saulnier, Benjamin Renoust et Jean-Étienne Noiré pour les bons moments passés à l'Ina.

Et merci à ceux qui m'ont aidé et conseillé au cours de ma thèse, je veux parler de Félicien Vallet, Nicolas Hervé, Louis Laborelli et beaucoup d'autres qui je l'espère ne m'en voudront pas.

Merci à Amel Hamzaoui et Julien Champ avec qui il a été un réel plaisir de collaborer.

J'aimerais remercier Gilbert Garnier, Isabelle Bloch, Séverine Dubuisson et Najib Gadi, pour m'avoir aidé et poussé à aller plus loin tout au long de mes stages et études.

Enfin, je ne peux pas finir ces remerciements sans penser à mes parents et à mon frère, qui m'ont soutenu depuis le début, et qui m'ont donné la chance de pouvoir m'accomplir.

Merci à tous ceux qui auront le courage de lire cette thèse en entier, et en particulier mes rapporteurs !

Chapitre 1

Introduction

1.1 Motivations

L'Ina [1] gère actuellement environ 5 millions d'heures de télévision et radio, et plus d'un million de photographies. Chaque année s'y ajoutent 800 mille heures de plus, soit plus de 2 000 heures par jour. Flickr détient environ 5 milliards de photographies, Facebook plus de 10 milliards, avec une croissance de cent millions chaque mois. Environ 35 heures de vidéos sont envoyées sur YouTube chaque minute.

L'accroissement du nombre de données visuelles à archiver impose des contraintes de productivité nouvelles sur la documentation. Il devient ainsi de plus en plus difficile d'annoter manuellement les documents archivés, de par le volume de données, mais également par l'évolution de la finesse de description. Plus la masse d'informations augmente, et plus le nombre d'occurrences d'un objet visuel donné augmente en conséquence. Un élément jusqu'alors inconnu, devient alors potentiellement remarquable, à partir du moment où ce nombre d'apparitions dépasse un certain seuil.

A l'Ina, plus d'une centaine de documentalistes travaillent chaque jour à l'annotation manuelle des programmes à archiver. L'une de leurs activités consiste, pour chaque programme télévisé, à fournir des mots clés (personnalités, lieux, thèmes) issus d'un thésaurus, ainsi que des images représentatives du programme. Les plateformes d'hébergement de vidéos (comme YouTube, Dailymotion, etc.) utilisent généralement un système de métadonnées générées par les utilisateurs eux-mêmes. Le développement d'outils automatiques et semi-automatiques pour assister ce travail d'annotation est désormais indispensable afin d'exploiter au mieux la très grande quantité d'informations disponibles.

L'un des systèmes de fouille d'images ayant acquis une certaine notoriété auprès du public est le logiciel Picasa [2], permettant la détection et la reconnaissance de visages. Bien que remplissant parfaitement son rôle, l'application reste limitée aux visages, et ne gère pas la découverte de lieux fréquents (façades de bâtiments, décors, scènes, etc.) ou d'objets d'intérêts divers, autant d'informations qui pourraient permettre aux utilisateurs de mieux naviguer dans leurs milliers de photographies.

1. Institut National de l'Audiovisuel : http ://www.ina.fr
2. http ://picasa.google.com/

Outre les photographes et les fournisseurs de contenu visuel, la fouille d'objets visuels peut également intéresser les journalistes et les sociologues désireux d'analyser le contenu visuel des médias, afin de détecter les événements médiatiques, de mesurer le suivi de l'information, etc.

1.2 Problématique

Ces dernières années, sont apparues des techniques d'analyse et d'indexation de contenu visuel ou audio, permettant d'effectuer des recherches et de la fouille de données dans de grands volumes de contenu vidéo.

L'objectif principal de cette thèse est la découverte d'objets visuels fréquents dans de grandes collections multimédias (images ou vidéos). Comme dans de nombreux domaines (finance, génétique, etc.), il s'agit d'extraire une connaissance de manière automatique ou semi-automatique en utilisant la fréquence d'apparition d'un objet au sein d'un corpus comme critère de pertinence. Dans le cas visuel, le problème est différent de la fouille de données classique (ADN, textuel, etc.) puisque les instances d'apparition d'un même objet ne constituent pas des entités identiques mais doivent être appariées. Cette difficulté explique également pourquoi nous nous focalisons sur la découverte des objets rigides (logos, objets manufacturés, décors, bâtiments, etc.), et non des classes (catégories) d'objets de plus haut niveau sémantique (maison, voiture, chien, etc.). Tout comme en programmation orientée objet, une classe est une catégorie définissant un concept (pouvant apparaître sous différentes formes), tandis qu'une instance d'objet est une représentation de cet objet. Ainsi, on distingue par exemple la classe « maison » (voir figure 1.1), de l'instance « la maison blanche » (voir figure 1.2 page suivante).

(a) certains droits réservés, licence CCA 2.0, Jean-Pol GRANDMONT

(b) certains droits réservés, licence CC-BY-SA-3.0, Ji-Elle

(c) certains droits réservés, licence CC-BY-SA-3.0, Spedona

FIGURE 1.1 – Plusieurs instances d'objets de la classe « maison »

Bien que les techniques de recherche d'objets rigides aient atteint une certaine maturité, le problème de la découverte non supervisée d'instances d'objets dans des grandes collections d'images est à l'heure actuelle encore difficile. D'une part parce que les méthodes actuelles ne sont pas assez efficaces et passent difficilement à l'échelle. D'autre part parce que le rappel et la précision sont encore insuffisants pour de nombreux objets. Particulièrement ceux ayant une taille très restreinte par rapport à l'information visuelle contextuelle qui peut être très riche (par exemple le logo d'un parti politique

apparaissant ponctuellement dans un sujet de journal télévisé).

(a) certains droits réservés, licence CC-BY-SA-3.0, Matt Wade (b) certains droits réservés, licence CC-BY-SA-3.0, Zach Rudisin (c)

FIGURE 1.2 – Plusieurs instances de l'objet « la maison blanche »

1.3 Contributions

Une première contribution de la thèse est de fournir un formalisme aux problèmes de découverte et de fouille d'instances d'objets visuels fréquents. Ces deux problèmes sont en effet définis de manière très confuse dans les quelques travaux récents de la littérature les abordant. Cette modélisation nous a permis entre autres choses de mettre en évidence le lien étroit qui existe entre la taille des objets à découvrir et la complexité du problème à traiter.

La deuxième contribution de la thèse est une méthode générique de résolution de ces deux types de problèmes, reposant d'une part sur un processus itératif d'échantillonnage d'objets candidats, et d'autre part sur une méthode efficace d'appariement d'objets rigides à large échelle. L'idée est de considérer l'étape de recherche d'instances proprement dite comme une simple boite noire à laquelle il s'agit de soumettre des régions d'images ayant une probabilité élevée d'appartenir à un objet fréquent de la base. Une première approche étudiée dans la thèse consiste à simplement considérer que toutes les régions d'images de la base sont équiprobables, avec comme idée conductrice que les objets les plus instanciés sont ceux qui auront la couverture spatiale la plus grande et donc la probabilité la plus élevée d'être échantillonnés. En généralisant cette notion de couverture à celle plus générique de couverture probabiliste, il est alors possible de modéliser la complexité de notre méthode pour tout score de vraisemblance donné en entrée, et de montrer ainsi l'importance de cette étape.

La troisième contribution de la thèse s'attache précisément à construire un score de vraisemblance s'approchant au mieux de la distribution parfaite, tout en restant *scalable* et efficace. Cette dernière repose sur une approche originale de hachage à deux niveaux, permettant de générer efficacement un ensemble d'appariements visuels dans un premier temps, et d'évaluer ensuite leur pertinence en fonction de contraintes géométriques faibles. Les expérimentations montrent que contrairement aux méthodes de l'état de l'art notre approche permet de découvrir efficacement des objets de très petite taille dans des

millions d'images.

Pour finir, plusieurs scénarios d'exploitation des graphes d'appariement visuel produits par notre méthode sont proposés et expérimentés. Ceci inclut la détection d'évènements médiatiques transmédia et la suggestion de requêtes visuelles.

1.4 Organisation de la thèse

La thèse est organisée en trois parties. La première a pour objectif de passer en revue les différentes méthodes de recherche (cf. chapitre 2 page 21) et découverte d'objets visuels (cf. chapitre 3 page 47) de l'état de l'art. On s'attachera en particulier dans le chapitre 2 à détailler les différentes étapes composant la recherche d'objets, soit la description, le partitionnement des vecteurs, l'indexation et la recherche, ainsi que la vérification de la cohérence géométrique et l'extension de requêtes.

Dans la deuxième partie, on présentera les contributions principales de cette thèse, à travers trois chapitres. Le chapitre 4 page 55 commence par formaliser les problèmes de découverte et fouille d'instances, puis propose une architecture générale pour résoudre ces problèmes. Le chapitre 5 page 71 a pour objectif de proposer différentes approches de calcul de scores de vraisemblance issues de méthodes état de l'art, alors que le chapitre 6 page 81 expose notre nouvelle méthode de calcul de scores basée sur le hachage des descripteurs visuels et des attributs de géométrie faible. Ces deux chapitres offrent tous deux des expérimentations quantitatives afin de comparer les performances de chaque méthode.

Dans la troisième et dernière partie, nous proposons différentes expérimentations et contextes applicatifs. Le chapitre 7 page 95 s'attache à la découverte de logos, et permet d'évaluer quantitativement l'ensemble de notre système de découverte d'objets visuels sur une collection d'images adaptée au problème. Le chapitre 8 page 111 montre les résultats de notre système de découverte dans une collection de plus de 2 000 heures de journaux télévisés, avec un filtrage transmédia permettant de détecter les principaux événements médiatiques. Enfin, le chapitre 9 page 119 décrit notre proposition de système de suggestion de requêtes visuelles à partir des résultats de fouille d'objets.

Première partie

Etat de l'art

Chapitre 2

État de l'art des méthodes de recherche d'objets visuels

2.1 Introduction

Il est important de bien distinguer les différences entre les systèmes de recherche d'objets visuels et d'autres systèmes comme ceux concernant la détection de copies [1], la recherche d'images similaires [2, 3, 4], la reconnaissance de concepts sémantiques [5]. Dans cet état de l'art, nous nous focaliserons sur les méthodes de recherche d'objets visuels, autrement nommées recherche d'instances (ainsi défini par la campagne d'évaluation Trecvid [1]). La plupart des méthodes que nous présenterons seront pour autant tout à fait adaptées à la recherche d'images similaires, bien que parfois sur-dimensionnées par rapport au besoin.

La figure 2.1 page suivante présente le schéma général de l'architecture d'un système de recherche d'images par le contenu. Les images sont tout d'abord représentées par un ensemble de descripteurs (cf. section 2.2). Ces descripteurs sont ensuite partitionnés (cf. section 2.3 page 27), puis indexés pour être efficacement retrouvés lors de la recherche (cf. section 2.4 page 36). La recherche d'une image (ou zone d'image) produit en sortie une liste d'images contenant des descripteurs similaires à ceux requêtés. Une vérification de la cohérence géométrique des appariements est ensuite souvent appliquée (cf. section 2.5 page 39). Enfin, une extension de requête est optionnellement utilisée afin d'améliorer le rappel (cf. section 2.6 page 42).

2.2 Description d'images

2.2.1 Descripteurs bas niveau

Le contenu visuel des images peut être décrit par des caractéristiques bas niveau (aussi appelées descripteurs) représentant statistiquement le signal image, comme la texture, la forme, etc. ou par des concepts visuels de plus haut niveau, comme des objets reconnaissables. Ces concepts visuels de plus

1. http ://www-nlpir.nist.gov/projects/tv2012/tv2012.html#ins

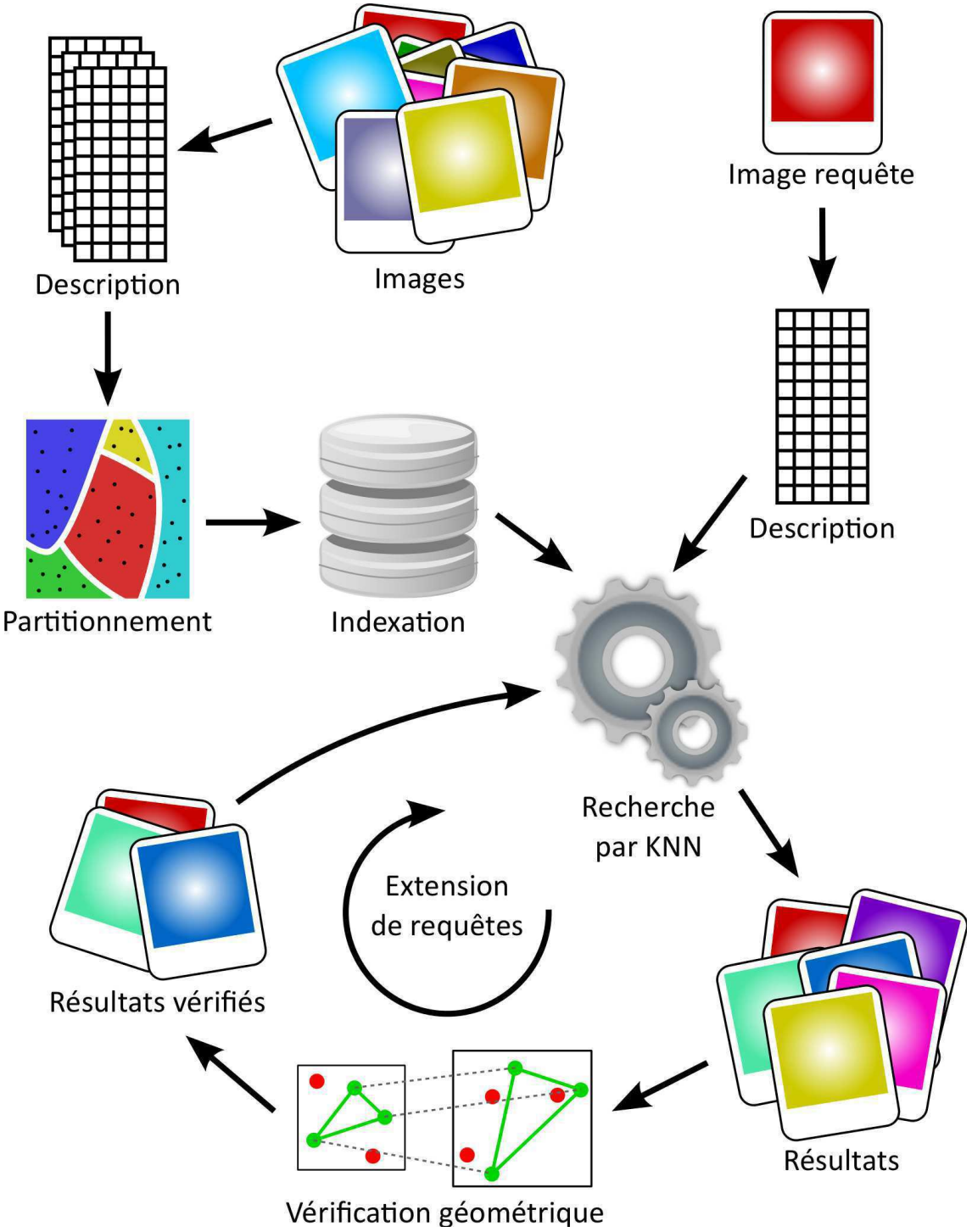


FIGURE 2.1 – Architecture générale d'un système de recherche d'images par le contenu

haut niveau sont généralement obtenus à partir des descripteurs bas niveau des images, et non pas directement à partir des images brutes. Par la suite, nous passerons en revue les descripteurs bas niveau les plus utilisés dans le domaine de la recherche d'objets visuels (sans nous arrêter sur les concepts visuels de haut niveau).

Il existe deux catégories principales de descripteurs bas niveau : ceux opérant à l'échelle de l'image (les descripteurs globaux), et ceux opérant à l'échelle de portions d'images (les descripteurs locaux).

2.2.1.1 Descripteurs globaux

Couleurs, textures et formes sont connues pour être les principales caractéristiques visuelles pouvant décrire le contenu global des images. Swain et Ballard [6] furent parmi les premiers à proposer d'utiliser les histogrammes de couleurs pour l'indexation d'images. De nombreuses améliorations furent ensuite apportées par [7, 8, 9, 10, 11]. La transformation en ondelettes de Daubechie a été utilisée pour la description de texture dans [12]. Les descripteurs visuels inclus dans le standard MPEG7 [13] sont parmi les plus connus. Ils consistent en un ensemble de descripteurs de couleur et de texture adaptés aux images et vidéos naturelles [14].

L'un des descripteurs globaux les plus utilisés est GIST [15], qui propose une représentation de l'image en faible dimension. Les images sont divisées en 4x4 zones spatiales, dans lesquelles sont extraits des histogrammes d'orientation.

Les descripteurs globaux ont été longtemps utilisés, mais le sont de moins en moins, au profit des descripteurs locaux. La raison principale de cette disparition provient de l'impossibilité pour ces descripteurs de représenter des objets couvrant une faible portion d'image. Les caractéristiques de ces petits objets sont ainsi noyés dans un seul descripteur global par image.

Si certains systèmes continuent aujourd'hui d'employer de tels descripteurs, c'est en raison de leurs deux principaux avantages : ils sont très peu coûteux en mémoire, et très rapides à calculer. Mais d'autres avantages sont également à prendre en considération : la plupart de ces descripteurs sont des histogrammes, et il est donc très facile de les combiner. Enfin, ils font toujours partie des outils les plus efficaces dans les domaines de la catégorisation de scènes ou encore de la reconnaissance de type d'image (photographie naturelle, clip-art, dessin au trait,...).

2.2.1.2 Descripteurs locaux

Comme on l'a vu précédemment, l'intérêt d'utiliser des descripteurs locaux plutôt que des globaux provient de leur capacité à représenter des petites portions d'images centrées autour d'un point ou d'une région d'intérêt, et donc à décrire beaucoup plus finement les images. Une très large variété de descripteurs locaux a été proposée, certains étant dépendants de la localisation de points ou de régions d'intérêts, d'autres non (cf. approches denses où les descripteurs sont extraits depuis des positions aléatoires [16], ou depuis une grille [17]).

On trouve de nombreuses méthodes de calcul de point d'intérêt dans la littérature. La tâche principale est de détecter une zone locale en utilisant des informations invariantes à certaines transformations.

Parmi ces transformations, on distingue les transformations géométriques (rotation, changement d'échelle…), les transformations photométriques (changements d'illumination), le bruit, la compression…

Certaines sont basées sur la détection de coins comme les méthodes de Harris [18], Shi et Tomasi [19], SUSAN [20] ou FAST [21]. D'autres sont basées sur la détection de blobs utilisant des laplaciens de gaussiennes (LoG) [22], des différences de gaussiennes (DoG) [23], ou des ondelettes de Haar (SURF) [24]. Les points d'intérêts les plus utilisés dans les systèmes de recherche d'images par le contenu sont généralement ceux résistant à une transformation affine, comme MSER [25], Harris-affine [26], ou encore Hessian-affine [27, 28].

On peut distinguer trois catégories de descripteurs locaux : ceux utilisant des histogrammes, ceux utilisant les différentielles et les moments, et enfin ceux basés sur les fréquences spatiales.

2.2.1.2.1 Descripteurs basés histogrammes

Ces descripteurs intègrent dans un histogramme différentes caractéristiques visuelles comme la couleur, la luminance, les contours, les orientations de gradients, les distances radiales, ou la texture. Ainsi, [29, 30, 31] proposent d'utiliser des histogrammes représentant la distribution des intensités des pixels. Dans [32], Zabih et Woodfill proposent une méthode décrivant les textures, robuste aux changements d'illumination, basée sur la relation entre les intensités des pixels plutôt que directement sur les intensités.

Belongie et al. [33] calculent un histogramme de contours extraits par le détecteur de Canny et la position est quantifiée dans le domaine log-polaire. Ce descripteur, appelé « shape-context », a récemment montré d'excellente performances dans le domaine de la reconnaissance foliaire, lors de la tâche « identification de plante » à ImageCLEF 2012 [2].

Lowe [23] propose un descripteur invariant au changement d'échelle (SIFT), ainsi qu'à la rotation et à l'illumination. Le descripteur est représenté par un histogramme 3D de gradients locaux orientés, pondérés par l'amplitude du gradient. Il est stocké dans un vecteur à 128 dimensions (8 intervalles d'orientation pour 4x4 zones spatiales autour du point d'intérêt). La richesse de l'information contenu dans SIFT en fait un descripteur robuste aux petites distorsions géométriques et aux petites erreurs de détection du point d'intérêt. Ce descripteur est de loin le plus populaire de ceux actuellement utilisés.

Diverses améliorations de SIFT ont été suggérées. Par exemple, Ke et Sukthankar [34] proposent PCA-SIFT, une méthode dans laquelle ils appliquent une PCA sur les patchs de pixels. Micolaczyk et al. [35] proposent une variante de SIFT appelée GLOH (Gradient Location-Orientation Histogram) dont le but est d'améliorer la robustesse et la distinctivité, en utilisant une grille en log-polaire avec 3 intervalles dans la direction radiale et 8 dans la direction angulaire. Les orientations des gradients étant quantifiées sur 16 intervalles, ils appliquent ensuite une PCA pour retrouver un total de 128 bits.

Plus récemment, Morel et Yu [36, 37] ont introduit une version affine de SIFT (ASIFT). En prenant en compte les paramètres de la caméra (angles) en plus des paramètres de zoom, rotation et translation utilisés dans SIFT, ils obtiennent un descripteur invariant aux changements de point de vue de la

2. http ://www.imageclef.org/2012

caméra. Ceci permet donc d'augmenter le nombre de correspondances entre deux images similaires. L'inconvénient de ASIFT est que sa complexité est environ 13 fois celle de SIFT, et que le nombre de descripteurs obtenus dans une image est également beaucoup plus élevé que pour SIFT.

Associé à leur point d'intérêt (SURF), Bay et al. [24] proposent un autre descripteur également robuste aux changements d'échelle et de rotation. Comme le point d'intérêt, le descripteur est calculé à partir des coefficients des ondelettes de Haar.

2.2.1.2.2 Descripteurs basés sur les différentielles et les moments

Ce type de descripteurs incorpore l'information relative aux dérivées des variations de l'image autour des points d'intérêt. Tout comme précédemment, les variations considérées sont du type couleurs, contours, ...Les moments invariants peuvent également être utilisés pour modéliser la structure des régions. Les propriétés des dérivées locales (Local Jet) ont été étudiées par Koenderink et van Doorn [38]. Florack et al. [39] dérivent les invariants différentiels, qui combinent les composantes des Local Jets pour obtenir l'invariance en rotation. Van Gool et al. [40] combinent également les Local Jets de différentes façons pour obtenir également l'invariance aux transformations affines et aux changements photométriques.

Freeman et Adelson proposent dans [41] d'utiliser des filtres orientables leur donnant une invariance en rotation.

Joly [42] utilise des dipôles dissociés pour construire des descripteurs normalisés de dimension 20. Ces dipôles dissociés sont des opérateurs différentiels non locaux, construits à partir d'une paire de lobes gaussiens. Les expérimentations dans [42] montrent que ces descripteurs donnent de meilleurs résultats que SIFT (en termes de mean Average Precision) sur la recherche d'images de copies d'images sur le web.

BRIEF [43] est un descripteur basé sur de simples tests binaires entre pixels dans une image lissée, permettant une extraction très rapide. Ses performances sont similaires à celles de SIFT sur de nombreux aspects comme la robustesse aux changements d'illumination, au flou. Il est par contre très sensible aux rotations dans le plan et aux changements d'échelle, ce qui le rend peu adapté à la recherche ou découverte d'instances, en milieu naturel.

2.2.1.2.3 Descripteurs basés sur les fréquences spatiales

Ces descripteurs modélisent la fréquence spatiale dans le voisinage des points d'intérêts en transformant l'image en un ensemble de coefficients fréquentiels obtenus par différentes transformations (Fourier, TCD, Gabor, ondelettes). Une comparaison des descripteurs de texture basés sur Gabor est présentée dans [44]. Les descripteurs basés fréquences présentent l'avantage d'être proche des caractéristiques utilisées par le cerveau humain, mais l'inconvénient d'être coûteux en mémoire.

2.2.1.2.4 Analyse

Une comparaison de plusieurs de ces descripteurs est disponible dans [45, 27, 35, 46]. La conclusion que l'on peut tirer est que SIFT et ses nombreuses variantes sont un excellent compromis pour la

recherche d'objets visuels. Ces descripteurs sont à la fois capables de stocker une grande quantité d'informations représentant au mieux la zone d'image décrite, et à la fois capable de résister à de nombreuses transformations, qu'elles soient géométriques ou photométriques. SIFT semble aujourd'hui le détecteur et descripteur de points d'intérêt le plus utilisé dans les systèmes de recherche d'objets visuels. L'usage de ASIFT pourrait permettre d'améliorer significativement les résultats de recherche de petits objets, où le nombre d'appariements est souvent très faible. Cependant, ses coûts mémoire et calculatoire sont un frein à son utilisation sur de grandes bases.

2.2.2 Représentation des images

A l'issue de l'étape d'extraction de descripteurs bas niveau, deux grandes stratégies de représentation des images s'opposent à nouveau : une stratégie locale, et une stratégie globale (sac de mots visuels, Fisher vectors) .

2.2.2.1 Représentation totale (locale)

Cette stratégie de représentation, issue de l'appariement d'images consiste à conserver toute l'information locale des images. La recherche d'une image (ou zone d'image) est alors effectuée en recherchant les descripteurs similaires à chacun des descripteurs composant la requête. Le score de similarité pouvant alors être le nombre de correspondances similaires. Toutefois, ce score est souvent revu lors d'une phase de vérification de la cohérence géométrique.

2.2.2.2 Sac de mots visuels

La stratégie du sac de mots visuels, introduite par Sivic et Zisserman [3] est la première à s'être imposée dans les systèmes de recherche d'images par le contenu à grande échelle. Le terme mot visuel tient son origine de l'analogie que l'on peut faire avec les moteurs de recherche textuels. Dans la méthode classique, les documents textuels sont en effet représentés sous forme de sacs de mots où la faisabilité de cette approche tient du fait que le vocabulaire est défini par le langage utilisé. Il n'en va évidemment pas de même pour le domaine visuel, dans lequel il n'existe pas de vocabulaire : les descripteurs sont généralement des vecteurs de nombres flottants. Il en existe donc potentiellement une infinité. Afin de se replacer dans le cadre des moteurs de recherche textuels, Sivic et Zisserman [3] proposent donc de créer un dictionnaire de mots visuels dont la taille est maîtrisée. La création de ces mots est souvent réalisée grâce à un *clustering* de type K-Means [3].

Une image est ensuite représentée par un histogramme à K cellules comptant le nombre d'occurrences de chaque mot visuel présent.

La similarité entre deux images peut ensuite être calculée comme étant le cardinal de l'intersection entre leurs sacs de mots visuels. La mesure est donc une mesure de similarité globale, bien qu'un faible score puisse tout de même indiquer la présence d'une similarité locale.

Pyramid Match Kernel [47], proposé par Grauman et Darrell, est un algorithme étendant le modèle des sacs de mots visuels en calculant des histogrammes multi-résolutions. Pour cela, l'espace des

descripteurs est quantifié en utilisant des cellules de taille croissante. Ce partitionnement de l'espace permet à certaines résolutions, de regrouper des descripteurs qui ne l'auraient pas été autrement. Grauman et Darrell [47] présente ensuite un noyau permettant de calculer la taille de l'intersection de ces histogrammes pyramidaux. Ce calcul est linéaire en nombre de descripteurs, contrairement à d'autres approches basées noyau.

Lazebnik et al. [48] préfèrent utiliser des méthodes de quantification de vecteur standard, mais utilisent par contre des pyramides pour quantifier l'espace spatial à différentes résolutions. Concrètement, pour chaque type différent (mot visuel), on obtient par image un ensemble d'histogrammes comptant le nombre de descripteurs tombant dans chaque cellule de l'histogramme.

2.2.2.3 Vecteurs de Fisher

Les *Fisher vectors* introduits par [49], ont tout d'abord été utilisés dans le domaine de la catégorisation de scènes [50], de la réduction de dimensions [51] ou de la détection de zones saillantes [52]. Contrairement au modèle du sac de mots visuels, les *Fisher vectors* n'encodent pas la présence d'un mot visuel dans une image, mais plutôt la dissimilarité entre le contenu de l'image et le vocabulaire généré au préalable. Cet encodage génère donc des vecteurs denses puisque chaque cellule de l'histogramme contient une valeur. Cette représentation améliore le modèle du sac de mots visuels sur deux points : premièrement elle n'est pas simplement limitée au comptage des occurrences de chaque mot visuel, et deuxièmement elle encode des informations additionnelles sur la distribution des descripteurs.

Dans [53], Perronnin et al. proposent une méthode de compression adaptée afin de réduire le coût mémoire ainsi que le coût de calcul.

Les *Fisher vectors* ont récemment montré d'excellentes performances sur la base ImageNet [3] pour la classification d'objets.

2.3 Partitionnement de l'espace visuel

Préalablement à l'étape d'indexation des descripteurs, on effectue un partitionnement de l'espace des descripteurs afin de pouvoir construire les index.

Le principe du partitionnement de l'espace visuel est de réaliser la transformation $\mathbb{R}^d \mapsto u, u \in \mathbb{N}$, qui à un vecteur $\boldsymbol{x}$ associe un entier u, et qui à un vecteur $\boldsymbol{y}$ similaire à $\boldsymbol{x}$, va associer ce même entier u avec une certaine probabilité.

Cet entier u est nommé différemment selon les méthodes de partitionnement : on le nomme clé de hachage dans les méthodes de hachage, *cluster* dans les méthodes de clustering, feuille dans les méthodes utilisant des arbres, région de Voronoi pour les treillis …

Il est à noter que les « mots visuels » utilisés dans certaines méthodes sont également des parties de l'espace.

Ce point de vue est d'ailleurs partiellement partagé par Keim et al. [54], qui considèrent les méthodes de partitionnement comme des méthodes de *clustering*.

3. http ://www.image-net.org/

En réalisant un tel partitionnement, on tend vers deux objectifs principaux :

– Indexer les données ;
– Compresser les données. On avait des vecteurs de grande dimension (e.g. 128×4 octets), on a maintenant un entier (codable avec quelques dizaines ou centaines de bits) ;

Il y aurait des dizaines de manières différentes de « partitionner » toutes les méthodes de partitionnement. Un point de vue intéressant est celui qui consiste à séparer les méthodes indépendantes des données, de celles qui en sont dépendantes.

2.3.1 Partitionnement indépendant de la distribution des données

2.3.1.1 Partitionnement basé sur des structures déterministes

2.3.1.1.1 Treillis

Les treillis (*lattices* en anglais) sont des structures régulières et infinies, définies par un ensemble de vecteurs. Chaque point d'un treillis peut être identifié par un unique nombre entier. Les régions de Voronoi autour des points d'un treillis ont toutes la même forme et le même volume. Deux vecteurs ayant été encodés avec le même point d'un treillis sont séparés par une distance maximum, dépendant des paramètres du treillis. Les treillis ont été utilisés pour la quantification vectorielle dans [55] et [56]. Dans le cas d'une distribution uniforme, [55] montre que les treillis sont plus efficaces que les quantifieurs scalaires. Dans [57], Tuytelaars et Schmid utilisent un treillis en forme de grille carrée. Chaque dimension du descripteur est quantifiée sur 2 bits. Devant le très grand nombre de points composant le treillis, les auteurs proposent de ne garder que les points qui se voient affecter au moins un descripteur.

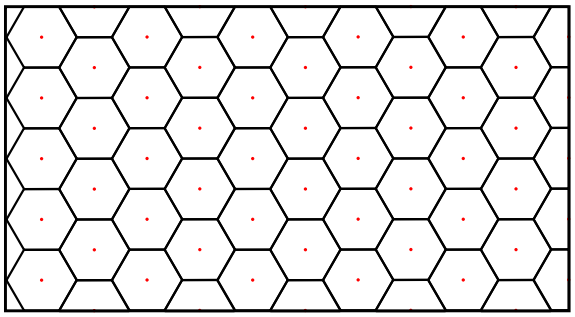


FIGURE 2.2 – Exemple d'un treillis hexagonal

2.3.1.1.2 Courbes remplissant l'espace

Les courbes remplissant l'espace (*space filling curves*, voir [58] pour une revue), comme la courbe de Hilbert ou l'ordonnancement en Z sont des courbes définies de manière récursive dans l'espace multidimensionnel et dont le tracé tend vers un recouvrement complet de l'espace lorsque l'ordre de la courbe tend vers l'infini. Elles permettent d'établir une transformation bijective entre un espace de dimension D et un espace monodimensionnel (coordonnée curviligne d'un point). Les distances entre

points ne sont pas préservées mais une certaine forme de localité est préservée. Ainsi deux points voisins sur la courbe sont proches dans l'espace. Inversement, tous les points proches dans l'espace ne le sont pas systématiquement sur la courbe. Le principe du partitionnement à base de *space filling curves* est de convertir chaque vecteur en une position sur la courbe.

Cette position est quantifiée avec un nombre de bits dépendant de l'ordre de la courbe.

Les *space filling curves* ont principalement été employées dans le domaine de la recherche d'informations pour la détection de copies [59, 60].

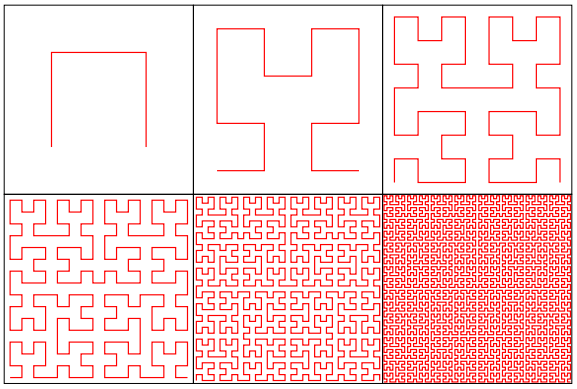


FIGURE 2.3 – Les six premières itérations d'une courbe de Hilbert (certains droits réservés, licence CC-By-SA, Braindrain0000)

2.3.1.2 Familles de hachage basées sur des transformations aléatoires

Les méthodes aléatoires sont généralement considérées comme plus adaptatives dans le cas de données ayant une distribution hétérogène, et sont donc plus efficaces que les méthodes basées sur une structuration déterministe (les descripteurs tels que SIFT ne respectent pas une distribution homogène). Ces méthodes sont un moyen de gérer les très grandes dimensions (> 30).

2.3.1.2.1 Locality Sensitive Hashing

Ces méthodes utilisent des familles de fonctions définies par des processus aléatoires, comme les fonctions de type *Locality Sensitive Hashing* (LSH), introduite par Indyk et al. [61].

Une fonction de type LSH associe une clé de hachage (un bit ou un entier) à chaque vecteur de l'espace à partitionner. Cette association est faite de telle sorte que les vecteurs similaires ont une probabilité plus forte que les vecteurs dissimilaires d'obtenir la même clé de hachage. En concaténant plusieurs clés de hachage obtenues avec différentes fonctions, on obtient une version compressée des vecteurs partitionnés.

Les fonctions de hachage utilisées sont généralement choisies pour approximer une distance particulière :

– Les fonctions de hachage par échantillonnage de bits [61] permettent d'approximer la distance de Hamming entre les vecteurs. Dans ce cas, la fonction de hachage sélectionne aléatoirement un bit i

dans le vecteur $\boldsymbol{x}$, c'est-à-dire $h = \boldsymbol{x}[i]$;

- Les fonctions de hachage de type cosinus qui approximent le produit scalaire entre vecteurs. Ces fonctions projettent les vecteurs sur des hyperplans aléatoires $\boldsymbol{r}$ passant par l'origine, et conservent le signe après projection (cosinus positif ou négatif), soit $h = \text{sign}(< \boldsymbol{x}, \boldsymbol{r} >)$;
- Les fonctions qui approximent la distance L_1 : $h = \lfloor (\boldsymbol{x}[i] - b)/w \rfloor$;
- Les fonctions qui approximent la distance L_2 [62] par projection sur des hyperplans aléatoires avec biais : $h = \lfloor (< \boldsymbol{x}, \boldsymbol{r} > -b)/w \rfloor$;
- Plus généralement, les fonctions qui approximent la distance L_p par l'utilisation de distributions p-stables [63] ;
- Les fonctions qui approximent les distances sur une hypersphère de rayon unitaire [64] (*Spherical Simplex, Spherical Orthoplex, Spherical Hypercube*) ;
- Les *Shift-invariant kernels* [65], qui approximent tout noyau invariant en translation ;
- *Min-Hashing* [66] qui estime la distance de Jacquard entre deux ensembles ;

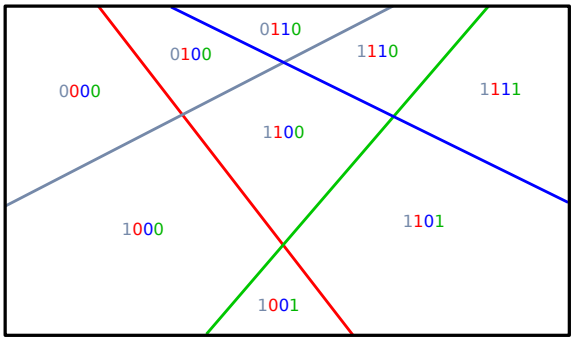


FIGURE 2.4 – Exemple de hachage en 2D avec projections aléatoires et 4 bits (4 hyperplans)

2.3.2 Partitionnement dépendant de la distribution des données

Les méthodes de partitionnement dépendant de la distribution des données sont comme leur nom l'indique, des méthodes qui s'adaptent à la distribution des données. Les descripteurs ne présentant généralement pas une distribution uniforme dans l'espace, on voit toute de suite l'intérêt de ce type de méthode. Cette distribution peut également varier en fonction de la base de données étudiée (une base contenant des photographies prises par des touristes en vacances sera différente d'une base d'images aériennes).

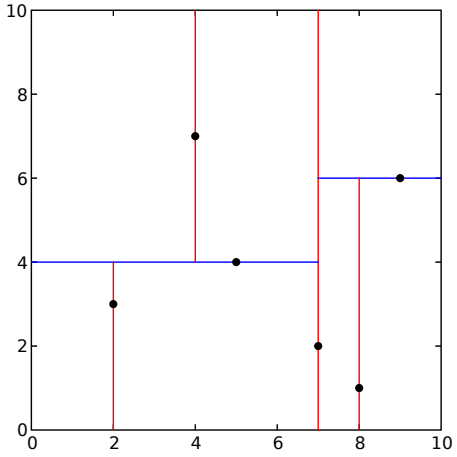
Mais il est important de noter que rien n'empêche d'apprendre un partitionnement sur un ensemble de données, et d'appliquer ensuite ce partitionnement à un autre ensemble. Cette approche est parfois utilisée dans les méthodes qui nécessitent un très long temps d'apprentissage. Avec un clustering *K-Means* par exemple, on peut apprendre la position des centres des clusters sur une petite base de données, et réaliser la phase d'affectation sur une autre base beaucoup plus grande. Cette astuce dégrade malheureusement significativement les résultats, comme le montre l'étude réalisée dans le tableau 2.1 page 45.

2.3.2.1 Partitionnement hiérarchique

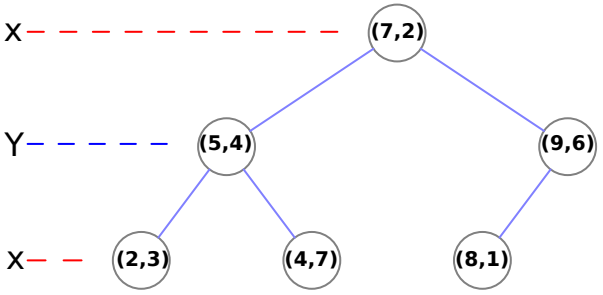
De nombreuses méthodes à base d'arbres ont été appliquées à la structuration de descripteurs visuels, comme les *R-tree* [67, 68, 69], les *SS-Tree (Similarity Search Tree)* [70], les *SR-Tree (Sphere-Rectangle Tree)* [71, 72], ou encore les *X-tree (eXtended node Tree)* [73].

Malgré la profusion de méthodes à base d'arbres, seuls les *Kd-tree* [74] et les méthodes de *K-Means* hiérarchiques sont fréquemment utilisés dans les systèmes récents.

Un *Kd-tree* est un arbre binaire divisant l'espace des descripteurs en deux, à chaque niveau de l'arbre, sur une seule dimension. Cette division garantit un partitionnement sans aucun recouvrement entre les feuilles. On voit une illustration d'un partitionnement par *Kd-tree* dans la figure 2.5. Ce type d'arbre étant mal équilibré, plusieurs variantes ont été proposées pour résoudre ce problème, comme *Adaptive KD-tree* [75], *KDB-tree* [76], *LSDh-tree* [77], *Randomized Kd-tree* [78, 79, 80].



(a) Partitionnement de l'espace (certains droits réservés, licence CC-BY-SA-3.0-MIGRATED, KiwiSunset)



(b) Le *Kd-tree* résultant

FIGURE 2.5 – Exemple de partitionnement avec un *Kd-tree*

Dans l'algorithme HKM (*Hierarchical K-Means*) [81], Nistér et Stewénius appliquent un *K-Means* avec un très faible nombre K (10 par exemple), puis recommencent un nouveau *K-Means* sur chaque sous-ensemble de descripteurs créés par les K clusters. L'algorithme itère ensuite jusqu'à obtention d'un nombre de *clusters* total suffisant. Ce nombre étant égal à K^n, avec n le nombre d'itérations. HKM permet de réduite la complexité de l'algorithme à $O(N \log K)$.

Bien que beaucoup plus efficace en termes de temps de calcul, HKM ne produit pas d'aussi bons résultats de *clustering* [82] qu'un *K-Means* standard.

BIRCH (*Balanced Iterative Reducing and Clustering using Hierarchies*) [83] a l'avantage de s'adapter aux ressources disponibles (temps et mémoire). Trois étapes sont nécessaires :

– La première consiste à créer une structure hiérarchique (*CF-Tree*) qui va permettre d'isoler les vecteurs éloignés. Les feuilles d'un *CF-Tree* définissent des micro-clusters ;
– La deuxième est un *clustering* ascendant opérant sur les micro-clusters générés à l'étape précédente ;
– La troisième étape consiste en l'affectation des vecteurs de la base aux *clusters* issus de la deuxième étape ;

Berrani [84] propose d'améliorer la première phase de BIRCH, afin de résoudre plusieurs problèmes liés au coût d'insertion dans le *CF-Tree*, et à la taille des micro-clusters.

2.3.2.2 Familles de hachage dépendant des données

Dans ce type de méthode, les fonctions de hachage sont définies à partir d'une base d'apprentissage représentative des données (sous-ensemble des données à hacher). L'objectif de ces méthodes est de s'adapter au mieux à la distribution des données, pour améliorer la sélectivité et le balancement des clés de hachage, tout en préservant la sensibilité à la localité.

2.3.2.2.1 Restricted Boltzmann Machine

RBM [85] est basé sur l'apprentissage d'un réseau de neurones multi-couches. L'une des couches est composée de neurones visibles, et l'autre de neurones cachés. Les neurones n'ont pas de connexions avec ceux de leur couche. Ils sont par contre connectés avec tous les neurones de l'autre couche. Cette connexion est bidirectionnelle et symétrique.

2.3.2.2.2 Spectral Hashing

Une autre méthode souvent citée dans la littérature est *Spectral Hashing* [86], basée sur la théorie du partitionnement des graphes. Dans cette approche, les données sont binarisées de telle sorte que la distance de Hamming approxime la distance Euclidienne. Il est à noter que la méthode voit ses performances décroître lorsque plus de 512 bits sont utilisés. Cette limite vient du fait que les auteurs utilisent une analyse en composantes principales. Les premiers bits générés sont donc très utiles, mais le sont de moins en moins au fur et à mesure du hachage.

2.3.2.2.3 Spherical Hashing

Contrairement aux méthodes de type LSH qui utilisent des hyperplans pour hacher les données, Heo et al. [87] proposent une nouvelle fonction de hachage basée sur des hypersphères. Chaque hypersphère définit la valeur d'un bit (le vecteur est compris dans l'hypersphère ou non). Ils proposent également une nouvelle fonction de distance binaire appropriée à leur fonction de hachage. La méthode présentée à l'avantage de créer un partitionnement équilibré, avec des fonctions de hachage indépendantes, qui donnent d'excellents résultats lorsqu'il s'agit d'approximer la distance L2 sur des descripteurs GIST.

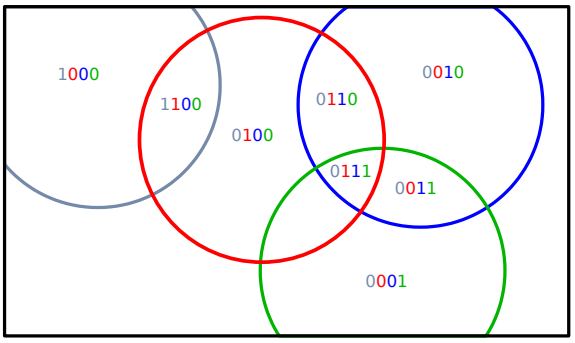


FIGURE 2.6 – Exemple de hachage en 2D avec *spherical hashing* et 4 bits (4 hypersphères)

2.3.2.2.4 *Kernelized Locality Sensitive Hashing*

Dans KLSH [88, 89], Kulis et Grauman présentent une approche légèrement différente, étant donné que leur principal objectif est de généraliser le hachage à n'importe quel noyau de Mercer plutôt que d'obtenir de meilleurs résultats. Ce type de généralisation semble connaître un certain engouement, avec des méthodes comme RMMH [90] ou une amélioration de *Product Quantization* [91].

2.3.2.2.5 *Product Quantization*

Product Quantization [92, 93] est une méthode actuellement très populaire. Les données sont tout d'abord partitionnées par un *clustering K-Means* (avec un assez faible nombre de *clusters*), puis chaque sous *cluster* et à nouveau partitionné. Ce deuxième partitionnement est effectué sur les résidus (du premier partitionnement) des sous ensembles de composantes des vecteurs. Chaque sous-ensemble quantifié produit une clé de hachage qui peut être concaténée à d'autres pour former une clé de hachage beaucoup plus discriminante. On constate toutefois que la méthode se limite à environ 64 bits, alors que certaines applications demanderaient une plus grande sélectivité. La réalisation d'un tel partitionnement peut nécessiter un coût de calcul assez important. En effet, même si l'apparition des méthodes de *K-Means* approximatif, comme AKM [94], a permis de réduire la complexité de la phase d'affectation de $O(NK)$ à $O(N \log K)$ (avec N le nombre de vecteurs et K le nombre de *clusters*), la complexité reste très importante. Que l'on ait $K = 10^3$, ou $K = 10^6$, ne change la complexité que d'un facteur 2 ($\log(10^6) = 2\log(10^3)$).

Dans [91], Bourrier et al. étendent la méthode aux *kernels* chi2, en exploitant le concept d'« *embedding explicite* » permettant de plonger les données dans un espace Euclidien, pour y effectuer la recherche.

2.3.2.2.6 *Random Maximum Margin Hashing*

Plusieurs études [93, 86, 85] ont montré qu'un hachage dépendant des données permettait l'obtention de meilleurs résultats de recherche. Mais ceci n'est généralement vrai que pour des tailles de clé de hachage limitées, ne dépassant pas 64 ou 128 bits. En effet, le problème de ces méthodes est que

le bénéfice de leur utilisation voit sa croissance diminuer au fur et à mesure qu'augmente le nombre de fonctions de hachage. Ceci s'explique par le manque d'indépendance entre les fonctions.

Dans *Random Maximum Margin Hashing* (RMMH) [90], Joly et Buisson proposent de nouvelles fonctions de hachage applicables à n'importe quel type de noyau, résolvant en partie ce problème d'indépendance. Pour chaque fonction de hachage, un sous-ensemble des données est sélectionné aléatoirement, et séparé en deux groupes de même taille. Un séparateur à vaste marge (SVM) est ensuite appris sur ces données labellisées. L'utilisation de SVM leur permet d'éviter le sur-apprentissage et offre de bonnes capacités de généralisation.

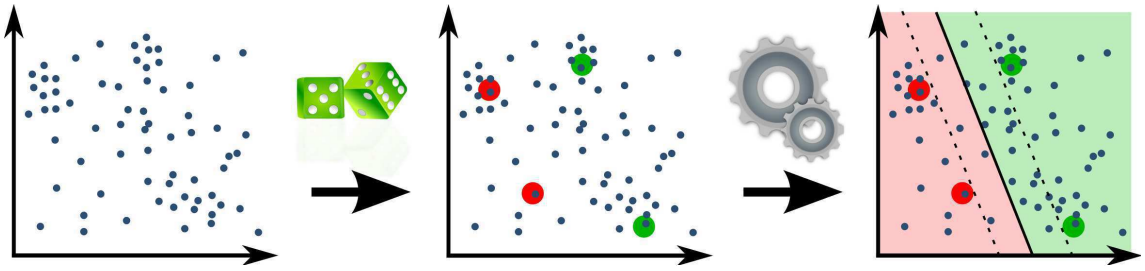


FIGURE 2.7 – Exemple de hachage en 2D avec RMMH

2.3.2.3 Partitionnement basé regroupement

2.3.2.3.1 K-Means

La méthode de structuration la plus fréquemment utilisée dans les systèmes existants est certainement celle réalisant un *clustering* des descripteurs. Bien que plusieurs algorithmes de *clustering* aient été envisagés pour cette tâche, c'est le *clustering* par *K-Means* qui semble remporter le plus de succès [3, 30, 95].

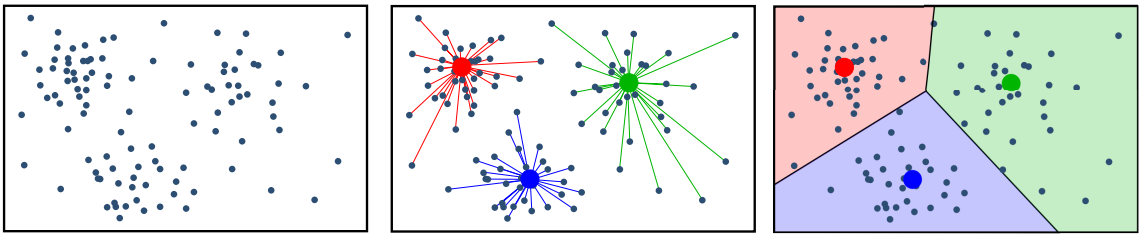


FIGURE 2.8 – Exemple de partitionnement par *K-Means*

La plupart des travaux utilisant *K-Means* emploient la version standard, dont l'utilisation devient totalement déraisonnable pour de grandes tailles de base et de vocabulaire (plusieurs millions de descripteurs), la complexité d'une itération étant de $O(NK)$, avec N le nombre de descripteurs, et K le nombre de *clusters*.

Les propositions permettant le passage à l'échelle de *K-Means* se sont principalement focalisées sur l'utilisation de *clusters* hiérarchiques [96], comme dans l'algorithme HKM (*Hierarchical K-Means*) [81] pour

lequel Nistér et Stewénius appliquent un *K-Means* avec un très faible nombre K (10 par exemple), puis recommencent un nouveau *K-Means* sur chaque sous-ensemble de descripteurs créés par les K clusters. L'algorithme itère ensuite jusqu'à obtention d'un nombre total de *clusters* suffisant. Ce nombre étant égal à K^n, avec n le nombre d'itérations. HKM permet de réduite la complexité de l'algorithme à $O(N \log K)$.

Bien que beaucoup plus efficace en termes de temps de calcul, HKM ne produit pas d'aussi bons résultats de *clustering* [82] que la version standard.

Philbin et. al [94] proposent une version approximative de *K-Means* (AKM), qui contrairement à HKM, minimise la même fonction de coût que la version standard. Ils montrent d'ailleurs que AKM donne d'aussi bons résultats qu'un *K-Means* standard, tout en conservant la même complexité que HKM, c'est-à-dire $O(N \log K)$. Ils parviennent à cette accélération en réduisant le coût principal du *K-Means* : la détermination du centre de *cluster* le plus proche pour chaque point. Pour cela, ils appliquent une méthode de recherche de plus proches voisins basée sur une forêt de plusieurs *Kd-tree* aléatoires [78, 79, 80].

2.3.2.3.2 Mean-shift

Les méthodes à base de *Mean-shift* sont très peu présentes dans la littérature. Mais on peut toutefois s'intéresser à [97], dans lequel Jurie et Triggs utilisent un *clustering* à base de *Mean-shift*. *Mean-shift* [98] est un algorithme de recherche de mode, basée sur l'analyse non paramétrique de l'espace des descripteurs. Jurie et Triggs [97] montrent que leur méthode obtient de meilleurs résultats qu'un *K-Means* standard.

2.3.3 Le problème du partitionnement « brutal »

Toutes les méthodes de partitionnement souffrent du même défaut : il n'existe pas de partitions parfaites. Qu'importe la méthode utilisée, avec des données réelles, on a toujours des descripteurs proches dans l'espace qui se retrouvent dans des partitions différentes. Cela a donc un impact négatif sur la qualité de la recherche si l'on effectue cette recherche dans une seule partie de l'espace. La figure 2.9 illustre ce problème de partitionnement. On verra dans la section 2.4.2 page suivante comment pallier à ce problème.

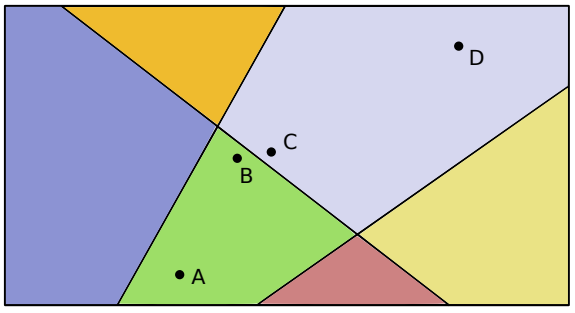


FIGURE 2.9 – Illustration du partitionnement « brutal ». Les points B et C sont proches dans l'espace mais ne sont pas dans la même partie de l'espace.

2.4 Structures d'indexation et recherche par similarité

Etant donné une méthode de partitionnement des données, il est maintenant possible d'indexer les structures pour pouvoir les retrouver efficacement. Dans cette optique, il est nécessaire de définir une structure adaptée au type de recherche effectuée.

Nous allons donc étudier les structures adaptées aux différents modes de partitionnement et les méthodes de recherche associées.

Pour cela, nous allons commencer par présenter les structures d'indexation.

2.4.1 Les structures

2.4.1.1 Tables de hachage et listes inversées

La plupart des méthodes emploient les mêmes structures basiques :

- Une première structure de type « élément / clé », qui à un vecteur x associe une partie de l'espace de l'espace : une clé de hachage, un numéro de *cluster*, etc. ;
- Une deuxième structure de type « clé / éléments », qui à une partie de l'espace associe un ensemble de vecteurs appartenant à cette même partie. On appelle cette deuxième structure « table de hachage » dans le cas des méthodes utilisant le hachage, ou « listes inversées » pour les méthodes à base de *clustering* ;

Ce type de structures concerne les méthodes de partitionnement à base de hachage, de regroupement, ou de structures déterministes.

Ces structures peuvent être implémentées par le biais de tableaux (coût mémoire élevé, mais accès très rapide), ou de conteneur associatif de type « map » (coût mémoire réduit, mais accès plus lent).

2.4.1.2 Les structures hiérarchiques

Le principe des structures hiérarchiques est de donner l'accès à un ensemble de données respectant un ensemble de critères, ces critères étant définis à chaque niveau de la hiérarchie. Dans les cas d'un *KD-tree* [74], chaque nœud définit une partie de l'espace, dont la taille réduit à mesure que l'on descend dans l'arbre. Les feuilles de l'arbre permettent donc d'obtenir l'équivalent des tables de hachage ou des listes inversées. Les structures hiérarchiques sont, comme leur nom l'indique, adaptées aux méthodes de partitionnement hiérarchiques.

2.4.2 Recherche dans une structure d'index

Il existe différentes méthodes de recherche dans une structure d'index : la recherche par accès simple, et celles qui visent à résoudre le problème de la perte d'information lors du partitionnement.

2.4.2.1 Recherche par accès simple

La recherche par accès simple dans une structure d'index opère en deux phases :

- Identifier le numéro de la partie de l'espace dans laquelle se trouve le descripteur requête ;

– Obtenir la liste des descripteurs similaires : ceux référencés par cette partie de l'espace ;

Le descripteur peut (ou pas) faire partie de la base de données. Si tel est le cas, alors on connaît généralement déjà son numéro de partie d'espace, stocké dans une structure de type « élément /clé ». Sinon, le processus dépend de la méthode de partitionnement utilisée. Pour les méthodes de hachage ou déterministes, le problème est très simple puisqu'il s'agit simplement d'appliquer la fonction mathématique utilisée lors du partitionnement.

Pour les méthodes de type *clustering*, la méthode est plus coûteuse étant donné qu'il est nécessaire d'exécuter l'étape d'affectation au *cluster* le plus proche.

Pour les méthodes à base d'arbres, comme *KD-tree*, le descripteur requête doit être évalué à chaque niveau de la hiérarchie pour savoir quel branche emprunter. Cette méthode est efficace pour la recherche dans des espaces de petite dimension. Par contre, les performances se dégradent rapidement avec l'augmentation de la dimension. Au delà de 10 dimensions, Weber et al. [99] montrent que la recherche exhaustive est plus rentable.

La recherche par accès simple est la plus rapide de celles présentées dans ce document, mais elle est aussi la moins efficace en termes de qualité de recherche, puisqu'elle ne permet pas de résoudre le problème de perte d'information lors du partitionnement vu dans la section 2.3.3 page 35. Elle ne permet généralement pas de faire du contrôle de qualité.

C'est pourquoi on préférera se tourner vers des solutions comme la recherche par accès simple dans de multiples structures, la recherche par accès multiples, ou encore vers du *soft-assignment*, dont on détaillera les avantages et inconvénients dans les sections suivantes.

2.4.2.2 Recherche par accès simple dans de multiples structures

Comme son nom l'indique, la stratégie des structures multiples vise généralement à effectuer plusieurs partitionnements des données en introduisant une part d'aléatoire dans le processus de construction de chaque index. Etant donné que bon nombre de méthodes de partitionnement possèdent déjà une part d'aléatoire, il s'agit souvent simplement de construire plusieurs partitions.

Pour les méthodes déterministes, le principe est de faire vibrer (introduire une translation aléatoire) la structure utilisée.

Pour les méthodes à base de projections aléatoires, il suffit simplement de répéter plusieurs fois le partitionnement [63], afin d'augmenter le rappel. Dans *Frequency Based Locality Sensitive Hashing* [100], Ling et Wu proposent de calculer les plus proches voisins seulement sur les points ayant un nombre minimum de collisions avec la requête, à travers les tables.

Dans les méthodes de type *KD-tree (multiple randomized KD-tree)*, l'idée est d'introduire de l'aléatoire lors du choix de la dimension qui va être séparée, ou lors de la frontière (ne plus prendre le médian, mais une valeur aléatoire autour du médian) [78, 101]. Durant la recherche, une seule file de priorités est maintenue pour tous les *KD-tree* aléatoires. Le compromis d'approximation est réglé par un nombre de nœuds parcourus. Après que ce nombre fixé de nœuds est atteint, la recherche se termine et les meilleurs résultats sont retournés. La méthode à base de *KD-tree* aléatoires est l'une des plus

performantes méthodes à base d'arbre et est très utilisée dans le domaine de la recherche approximative de descripteurs locaux. Par contre, cette méthode n'a pas de paramètres de contrôle basé sur un pourcentage de qualité par rapport à une recherche exhaustive. De plus, à large échelle, les structures d'arbres multiples ont un coût mémoire important.

Dans les méthodes à base de *clustering*, il est possible de modifier l'initialisation des centres des *clusters*, ou comme l'ont fait Aly et al. dans [102], de faire plusieurs *clustering* en utilisant des sous-ensembles de données.

Bien que l'approche de la recherche dans de multiples structures résolve en partie le problème des erreurs de partitionnement, un autre problème se pose : celui du coût mémoire. Le coût unitaire est en effet multiplié par le nombre de structures, et réduit d'autant le nombre maximum de descripteurs gérables par une machine. Il a par exemple été montré [103] que le coût mémoire de LSH (en multi-tables) est en fait sur-linéaire en fonction de la taille de la base.

2.4.2.3 Le soft-assignment

Le *soft-assignment* est une méthode utilisée dans le *clustering* qui consiste à affecter un même vecteur à plusieurs *clusters*, en donnant un poids pour chaque affectation. Elle a été appliquée pour la première fois à des descripteurs locaux quantifiés en mots visuels dans [104]. Les auteurs de [104] obtiennent une réelle progression de la qualité de recherche.

Tout comme l'approche précédente, le coût mémoire augmente linéairement avec le nombre d'affectations par descripteur.

2.4.2.4 Recherche par accès multiples

Pour réduire le coût mémoire qui augmente proportionnellement avec le nombre de structures utilisées, certaines méthodes proposent d'accéder aux parties de l'espace proches de celle contenant la requête. La difficulté est alors de déterminer les parties de l'espace qui ont la meilleure probabilité de contenir des vecteurs similaires.

Une première approche est celle proposée dans [105] où les auteurs proposent une stratégie d'accès multiples dans des courbes de Hilbert.

En ce qui concerne les méthodes de hachage, [106, 103] ont introduit des méthodes de recherche alternatives (*Multi-Probe LSH*) permettant de diminuer de manière drastique le nombre de tables, d'un facteur allant de 10 à 40. La stratégie *Multi-Probe* consiste à chercher les descripteurs similaires dans une ou plusieurs clés de hachage par table.

Dans [106], l'idée est de fixer un nombre d'accès (*probes*), et de faire des requêtes de type « rayon », tandis que [103] préfère fixer une qualité de recherche, et de s'adapter à des requêtes quelconques lors de la phase d'apprentissage (y compris des requêtes de type KNN).

Dans [93], Jégou et al. commencent par filtrer les vecteurs candidats grâce à une liste inversée, construite à partir d'un *clustering K-Means* (avec un assez faible nombre de *clusters*) sur l'ensemble des vecteurs. Ils utilisent une stratégie d'affectation multiple (c'est-à-dire à plusieurs centroïdes) pour réduire

l'effet du partitionnement « brutal » (problème vu dans la section 2.3.3 page 35). Ils obtiennent ainsi un ensemble de vecteurs candidats sur lesquels ils peuvent calculer une distance précise de façon rapide, grâce à leur quantification sur des sous-ensembles de composantes.

Dans le cas des *KD-tree*, [107] a proposé d'introduire une variable ϵ de contrôle de la qualité pour approximer les plus proches voisins. Ils ont aussi proposé une file de priorités permettant d'accélérer la recherche à travers l'arbre. Cette file de priorités leur permet de visiter les nœuds suivant leur distance par rapport à la requête courante.

2.4.2.5 Recherche avec calcul de distances asymétriques

Dans [108], Dong et al. proposent de calculer les plus proches voisins avec des distances asymétriques, c'est-à-dire entre une requête non quantifiée (dans son espace d'origine), et les vecteurs de la base ayant subi un partitionnement. Ce type de recherche, également repris par Jégou et al. [93], améliore sensiblement la qualité de recherche.

2.4.2.6 Analyse des méthodes de recherche

Comme on l'a vu précédemment, les systèmes états de l'art utilisent principalement le *soft-assigment* et les accès multiples, stratégies visant à réduire le problème du partitionnement « brutal ». En regardant attentivement, on peut observer que ces deux stratégies sont en réalité très proches. Le *soft-assignment* pourrait en effet être vu comme une stratégie d'accès multiples calculés *offline*, tandis que l'accès multiple pourrait, lui, être considéré comme la version *online* du *soft-assignment*. *Product Quantization* [93] emploie d'ailleurs une méthode proche de ces deux approches.

2.5 Vérification de la cohérence géométrique

Les méthodes de vérification géométrique servent à supprimer les fausses alarmes qui sont présentes dans la liste des résultats formée par la recherche par similarité. Les faux appariements (*outliers*), à opposer aux bons appariements (*inliers*) peuvent en effet être très nombreux, étant donné que les points d'intérêts encodent une information très locale. Il est donc normal que la recherche par similarité donne un très grand nombre d'*outliers* : de très nombreuses images peuvent présenter des zones de quelques pixels très similaires à une zone de la requête effectuée, sans pour autant contenir un objet similaire.

Une grande partie des méthodes de vérification géométrique se basent sur l'estimation de la transformation géométrique entre deux listes de points d'intérêts appariés (provenant de deux images). Cependant, on trouve également des méthodes de cosegmentation qui préfèrent s'affranchir des points d'intérêts pour descendre au niveau « pixel ». On trouve également des méthodes utilisant des contraintes géométriques faibles afin de gagner en temps de calcul.

2.5.1 Contraintes géométriques faibles

Les méthodes dites de géométrie faible (*Weak Geometry*) consistent à garantir un minimum de cohérence géométrique entre les appariements. Pour cela, [109] construit des histogrammes représentant la différence d'angle et le rapport d'échelle des appariements de descripteurs locaux, pour chaque couple d'images. Un pic dans l'histogramme tend ainsi à prouver la présence d'une transformation géométriquement cohérente au sein des points appariés. Le score d'appariement entre la requête et chaque image candidate peut donc être recalculé en fonction des histogrammes construits. Dans [110], les auteurs introduisent la translation en plus de la différence d'échelle et d'angle.

Xie et al. [111] pointe trois problèmes majeurs de ce type de technique :

- Comme la plupart des points d'intérêt ont une échelle basse, l'histogramme des différences d'échelle présente souvent un pic dans les valeurs basses ;
- La méthode ne prend pas en compte les transformations multiples ;
- Elle ne prend pas non plus en compte les changements de point de vue, ou encore les transformations non rigides ;

Xie et al. [111] propose donc l'utilisation de paires d'appariements et montre une amélioration significative des résultats.

2.5.2 Méthodes basées sur l'estimation de la transformation géométrique

Ces méthodes essaient d'estimer les paramètres d'une transformation géométrique entre deux images (entre leur points appariés). Cette transformation est généralement affine, homographique, ou plus simplement isométrique ou similaire.

En entrée, on a donc une liste d'appariements de points d'intérêts dont on connaît en général la position dans l'image, l'échelle de détection et l'angle. En sortie, elles donnent un sous ensemble de ces appariements, respectant la meilleure des transformations estimées.

Comme il est précisé dans la thèse de Rabin [112], trois approches sont envisageables afin d'estimer cette transformation : les estimateurs robustes des moindres carrés, la transformée de Hough et RANSAC.

2.5.2.1 Estimateurs robustes des moindres carrés

Toujours selon [112], l'approche la plus simple pour estimer directement la transformation d'un ensemble de correspondances est la méthode des moindres carrés. Elle consiste à définir la transformation optimale minimisant l'erreur, définie comme la somme des résidus au carré de chaque couple de points appariés. Cette approche est d'ailleurs utilisée par Lowe dans [23]. Le principal problème de ces estimateurs provient du calcul de l'erreur quadratique qui la rend très sensible aux outliers. L'estimation est perturbée dès lors qu'il y a ne serait-ce qu'un seul outlier.

Différentes améliorations proposent de calculer l'erreur médiane [113], ou de fixer le nombre d'échantillons à utiliser dans le calcul de l'erreur [114].

Il faut cependant rappeler que nous cherchons ici à étudier des méthodes potentiellement capables de retrouver des objets composés d'une dizaine de points d'intérêts dans des images en comportant

des milliers. Le nombre d'inliers est donc généralement très faible, voire nul lorsque les deux images appariées ne comportent aucun objet en commun.

Des méthodes basées sur un consensus comme Hough ou RANSAC sont donc bien plus adaptées à notre problème.

2.5.2.2 Transformée de Hough

La transformée de Hough, originellement introduite dans [115] est généralisée dans [116] pour la détection de droites. Dans cette approche, l'espace des paramètres (ceux de la transformation géométrique) est quantifié selon une grille régulière, définissant ainsi des accumulateurs. Les données sont ensuite utilisées pour voter dans ces accumulateurs. En recherchant le mode dans les accumulateurs, on trouve la transformation faisant consensus. Cette méthode est également utilisée par Lowe dans [23].

Outre la relativement faible précision de l'estimation, le principal inconvénient de Hough est le problème de la complexité qui explose lorsque le nombre de paramètres excède quatre.

A cause de cette limitation, on préfère généralement utiliser l'algorithme RANSAC et ses variantes.

2.5.2.3 RANSAC

RANSAC (*RANdom SAmple Consensus*) [117] cherche un consensus au sein d'un ensemble d'échantillons choisis aléatoirement. A chaque itération, une transformation est estimée à partir de quelques échantillons choisis aléatoirement, et le consensus est ensuite mesuré sur l'ensemble des données. L'algorithme s'arrête lorsqu'un consensus acceptable a été trouvé, qu'un nombre défini d'itérations a été atteint, ou alors que toutes les combinaisons d'échantillons possibles ont été testées.

De très nombreuses variantes de RANSAC ont été présentées depuis la publication de l'algorithme original. Une grande partie de ces variantes s'attachent à accélérer l'algorithme. RANSAC est en effet assez coûteux en temps de calcul, en raison de la nécessité de faire de nombreuses itérations pour être sûr de trouver un bon consensus. Certaines méthodes ont donc été proposées pour générer des hypothèses (échantillons aléatoires) qui sont plus enclins à faire consensus. Par exemple, dans [118, 119, 120], le but est de choisir les échantillons en exploitant les résultats des hypothèses précédentes. Dans ORSA [121] (*Optimal Random Sampling*), lorsqu'une hypothèse est jugée suffisante, les échantillons sont choisis dans le meilleur sous-ensemble trouvé. Dans [122, 123, 124], les auteurs choisissent les échantillons dans un même voisinage spatial pour maximiser la probabilité que les échantillons soient tous des *inliers*.

Dans SCRAMSAC [125], une étape de vérification de la cohérence spatiale est effectuée afin de réduire le nombre d'appariements, en conservant les plus cohérents.

Toujours dans l'idée de sélectionner des échantillons avec la meilleure probabilité possible, l'algorithme PROSAC [126] favorise les échantillons qui ont été appariés avec le meilleur score de confiance.

D'autres méthodes s'attachent davantage à contrôler la qualité des groupes de correspondances. Dans [127], l'algorithme MLESAC fait l'hypothèse d'une distribution normale des *inliers* et uniforme des *outliers*. Il introduit une nouvelle fonction de coût et une méthode pour trouver l'optimum avec l'algorithme EM

[128]. MINPRAN [129] fait lui simplement l'hypothèse de la distribution uniforme des *outliers*.

AC-RANSAC [121], introduit une méthode *a contrario* [130] afin de fixer les paramètres de RANSAC, dont le réglage peut s'avérer très sensible.

Enfin, l'algorithme MAC-RANSAC [131] généralise AC-RANSAC et propose une méthode capable de gérer les détections multiples, toujours grâce à des méthodes *a contrario*.

2.5.3 Méthodes basées cosegmentation

Le deuxième type de méthodes de vérification de la cohérence géométrique se base sur les pixels des images [132, 133]. Ces méthodes cherchent à la fois à estimer la transformation mais aussi à segmenter les deux images appariées afin de déterminer les zones communes. Ces méthodes permettent d'avoir une estimation très précise de la transformation entre les deux images. La cosegmentation donne d'excellents résultats, et permet en outre de réaliser une segmentation précise des objets détectés.

La cosegmentation suppose toutefois que le système puisse accéder aux pixels des images. Et bien évidemment, un tel accès entraîne obligatoirement un temps de calcul conséquent. Pour ces raisons, notre choix ne se portera donc pas sur ce type de méthodes, mais sur des méthodes de type RANSAC.

2.6 Extension de requête

Le concept de *Query Expansion* (QE, ou extension de requêtes) a été introduit dans le domaine de la recherche d'objets visuels dans [134]. Le but des méthodes en découlant est d'améliorer le rappel des objets recherchés. Cette méthode est inspirée des travaux effectués dans le domaine textuel utilisant des vecteurs de mots [135, 136]. Cette approche était tout à fait naturelle du fait que cette première proposition est basée sur les sacs de mots visuels. La méthode d'extension de requête à partir de sacs de mots peut être décrite de la manière suivante : à partir d'une première requête est obtenu un premier ensemble de résultats classés par leurs scores de similarités (par exemple à base d'un produit scalaire sur les vecteurs de TF-IDF). A chaque résultat est associé une image décrite par son propre vecteur de mots visuels. A partir de cet ensemble, il est possible de fusionner les différents vecteurs de mots visuels résultats afin de créer une nouvelle requête qui est ensuite exécutée dans le moteur de recherche. Ce processus peut être itéré autant de fois que désiré.

Dans le cas des requêtes globales à base de vecteurs de mots visuels, il a été proposé différentes stratégies de fusion afin de formuler la nouvelle requête [134, 137, 138]. Les différentes stratégies de fusion peuvent être décomposées en deux groupes : les génératives et les discriminantes. Les stratégies génératives [134, 137] consistent à créer une nouvelle requête en calculant la somme des vecteurs résultats. Les différentes méthodes génératives varient selon les méthodes de filtrages de résultats. La stratégie discriminante, proposée dans [138], consiste à créer un classificateur discriminant à partir d'un SVM appris sur les vecteurs résultats de la première requête. Le classificateur sert ensuite à classer les images de la base. Les images ayant un score positif sont gardées comme résultats. Pour l'apprentissage du SVM, les échantillons positifs sont les vecteurs les plus similaires du premier ensemble de résultats et les échantillons négatifs sont les vecteurs les moins similaires.

Pour les requêtes locales, une stratégie de QE a été proposée dans [4]. Dans cette stratégie, la liste de résultats est considérée comme une nouvelle liste de requêtes à exécuter. Cette liste de résultats peut potentiellement contenir des fausses alarmes. En effet, si l'extension de requêtes a pour effet d'améliorer le rappel, il est primordial de vérifier qu'elle ne dégrade pas la précision. Afin d'éviter d'exécuter les requêtes « fausses alarmes », les auteurs de [4] proposent un seuillage adaptatif permettant d'avoir une sélection très précise des résultats de la requête initiale. Cette stratégie de seuillage adaptatif est basée sur une méthode a contrario [130].

Ces méthodes d'extension requêtes [134, 137, 4] sont essentielles pour garantir un rappel important dans le cas de la recherche d'objets visuels. Elles entraînent par contre un sur-coût calculatoire qui peut atteindre des facteurs supérieurs à dix. Ces méthodes doivent donc être utilisées lorsqu'elles sont essentielles. Par exemple, elles ne sont pas essentielles lors des phases de découvertes d'objets visuels, mais elles le sont dans le cas où l'on veut réaliser des statistiques précises.

2.7 Recherche d'objets visuels à grande échelle

Le tableau 2.1 page 45 présente quelques unes des meilleures solutions actuelles de recherche d'images par le contenu. La performance de ces systèmes est évaluée par le score de mAP (mean Average Precision) obtenu sur la base Oxford Buildings [4] (qui fait référence dans ce domaine), ainsi que sur la base BelgaLogos [5] (qui permet l'évaluation de la recherche de petits objets).

Ce tableau montre que la plupart des systèmes sont basés sur l'approche par sacs de mots visuels (BoW) [3], dont les mots visuels ont été créés avec un *clustering* par *K-Means* approximatif (AKM) [94]. Bien que AKM rende possible la création de larges vocabulaires, ce type de méthode pose tout de même certains problèmes. Premièrement, la complexité d'AKM reste sur-linéaire en fonction de la taille de la base, contrairement aux méthodes de hachage. Deuxièmement, la finesse du partitionnement n'est pas réglable a posteriori. Il faut donc réeffectuer un *clustering* pour changer la taille du vocabulaire, contrairement aux méthodes de hachage qui peuvent calculer de longues clés et en utiliser une sous-partie. Troisièmement, les méthodes utilisant *K-Means* ont des performances très différentes selon que le partitionnement est effectué indépendamment ou non des données.

On remarque que le descripteur SIFT [23] est le plus utilisé, mais que son calcul est effectué autour de points d'intérêt détectés par la méthode Hessian affine de Mikolajczyk [27], ou plus récemment par l'amélioration de Perdoch [28].

Tous les systèmes étudiés emploient une vérification géométrique des appariements entre descripteurs locaux. La plupart emploie également une méthode d'extension de requêtes. On constate que l'emploi de ces méthodes améliore nettement les résultats. Malgré cette efficacité, l'extension de requête reste généralement très coûteuse en temps de calcul, la rendant souvent inappropriée à une utilisation *online*.

Malgré de nombreux téléchargements de la base BelgaLogos (environ une centaine d'équipes de

4. http ://www.robots.ox.ac.uk/ vgg/data/oxbuildings/
5. http ://www-sop.inria.fr/members/Alexis.Joly/BelgaLogos/BelgaLogos.html

recherche à ce jour), peu de travaux ont été publiés. Seul Revaud et al. [139] en 2012 ont récemment battu le score original de Joly et Buisson en 2009. Ceci est probablement dû au fait que la plupart des approches sont basées sur l'utilisation de sacs de mots visuels. Cette approche consistant à créer une description globale des images à l'aide de descripteurs locaux, elle est de fait relativement inadaptée à la recherche de petits objets (représentant une très faible proportion des descripteurs). On constate d'ailleurs que Revaud et al. ont préféré utiliser la méthode Hamming Embedding [109] plutôt que des sacs de mots visuels.

Un des travers de la méthode de Revaud et al. [139] est qu'elle requiert une base d'images « négatives » (images ne comportant aucunes des instances à rechercher) pour réaliser l'apprentissage de la *correlation-based burstiness*. Une telle base est en pratique difficile à construire dès lors que l'on souhaite ajouter de nouvelles instances à rechercher : il faut reparcourir toutes les images afin de vérifier que ces instances ne sont pas présentes dans la base de négatifs. La méthode est donc efficace pour les logos considérés, mais elle n'est pas automatiquement générique.

Toujours dans [139], les auteurs sont les premiers à soulever un problème qui, pour certains logos, limite fortement le rappel. Ce problème est celui des logos existants en deux versions (noir sur blanc ou blanc sur noir, ou plus généralement clair sur sombre et inversement). Ceci est le cas des logos Adidas, Nike ou Puma par exemple. Le descripteur SIFT n'étant pas invariant à l'inversion de couleur, Revaud et al. proposent donc de doubler chaque requête, c'est-à-dire d'effectuer la requête normale, et également la requête en inversion de couleur.

Système	Description	Partitionnement	Indexation & Recherche	Vérification géométrique	Extension de requêtes	mAP Oxford Buildings				mAP BelgaLogos	
						P. indep		P. dep		P. dep / P. indep	
						~~QE~~	QE	~~QE~~	QE	~~QE~~	QE
Philbin et al. [104] (2008)	Hessian affine [27] + SIFT [23]	AKM [94]	BoW + tf-idf + SA	RANSAC	Average expansion [134]	0,598	0,718	0,731	0,825		
Joly et al. [4] (2009)	SIFT [23]	LSH [63]	APMPLSH [103]	RANSAC	QE avec seuillage adaptatif a contrario	0,608	0,807			0.208	0.341
Arandjelovic et al. [138] (2012)	Perdoch [28] + RootSIFT [138]	AKM [94]	BoW + tf-idf	RANSAC	Discriminative QE		0,809		**0,929**		
Perdoch et al. [28] (2009)	Perdoch [28]	AKM [94]	BoW + tf-idf + SA	RANSAC	?	0,725	**0,822**	0,846	0,916		
Chum et al. [137] (2011)	Hessian affine [27] + SIFT [23]	AKM [94]	BoW + tf-idf	RANSAC	Contextual QE	**0,785**	0,82				
Notre méthode (2012)	SIFT [23]	RMMH [90]	APMPLSH [103] + HE	RANSAC	QE avec seuillage adaptatif a contrario			**0,851**	0,896	**0,419**	**0,51**
Revaud et al. [139] (2012)	Hessian affine [27] + SIFT [23]	K-Means	Liste inversée + HE [109]	RANSAC + Correlation-based burstiness	~~QE~~					0,414	

TABLE 2.1 – Comparaison des systèmes de recherche d'images par le contenu sur Oxford Buildings et BelgaLogos.
BoW = Bag of visual Words (Sacs de mots visuels), tf-idf = term frequency - inverse document frequency, SA = soft assignment, HE = Hamming Embedding, QE = Query Expansion (Extension de requêtes), ~~QE~~ = sans extension de requêtes, mAP = mean Average Precision, P. dep/indep = Partitionnement dépendant/indépendant des données

Chapitre 3

État de l'art des méthodes de découverte et fouille d'objets visuels

Dans le chapitre précédent, nous avons détaillé les méthodes utilisées dans les systèmes de recherche d'images par le contenu. Comme nous l'avons vu, cette recherche vise à retrouver des objets similaires entre une image requête, et une collection d'images ou de vidéos, à la demande d'un utilisateur, ou éventuellement d'un système automatique.

Contrairement à la recherche d'images, la découverte (ou fouille) d'images est un processus automatique, dans lequel l'utilisateur ne donne aucune information sur les objets à rechercher (exceptés quelques éventuels réglages). L'objectif est d'extraire une connaissance à partir de grandes quantités de données, trop grandes pour être rapidement analysées par un opérateur humain. Dans le domaine visuel, il s'agit donc de faire émerger des informations visuelles permettant à un utilisateur d'effectuer une analyse ou un traitement en un temps raisonnable.

Les informations à extraire dépendent directement du cas d'utilisation. Ces cas peuvent être divers et variés, allant du résumé de collections à la reconstruction d'objets en 3D. On remarque que pour la plupart de ces cas d'utilisation, on porte une attention particulière aux objets dont la taille et la fréquence sont importantes. Techniquement, on peut considérer deux problèmes principaux. Le premier est celui de la découverte d'objets, qui consiste à détecter la présence d'un objet, c'est-à-dire à découvrir plusieurs (au moins deux) instances d'un même objet. Le deuxième est celui du regroupement (*clustering*) d'instances, qui vise à regrouper toutes les instances d'un même objet entre elles.

Dans la section 4.1.2 page 56, nous formaliserons ces deux problèmes, que nous appellerons « Découverte » et « Fouille » d'objets $\{c, f\}$-fréquents, où l'on caractérisera les objets par leur couverture spatiale c, et leur fréquence d'apparition f.

3.1 Méthodes de fouille d'objets visuels par regroupement de motifs visuels

Sivic et Zisserman [140] proposent un système capable de détecter les objets principaux à l'intérieur d'une vidéo, en mesurant la fréquence d'apparition des configurations spatiales des mots visuels. Ces mots visuels sont des SIFT ou des MSER, quantifiés par un *K-Means*. Ces descripteurs peuvent ensuite être regroupés entre voisins pour former des régions plus grandes nommées voisinages. Ces voisinages sont définis comme étant l'enveloppe convexe des KNN spatiaux d'un descripteur central. Les voisinages étant représentés par des sacs de mots visuels, ils peuvent être comparés en calculant leur produit scalaire. L'algorithme opère en trois étapes. Premièrement, les voisinages sont filtrés afin de ne garder que ceux dont la fréquence d'apparition (nombre d'images) est supérieure à un certain seuil. Pour accélérer cette étape, seuls les voisinages présentant le même mot visuel central sont comparés. Deuxièmement, les voisinages restants sont regroupés par un algorithme de *clustering* progressif. Pour finir, les *clusters* générés à l'étape 2 sont fusionnés en fonction de leur recouvrement spatial et temporel. Cette approche est adaptée à la découverte d'objets à l'intérieur d'un petit corpus (comme un film), mais absolument pas à de grandes bases. En effet, la complexité de la première étape de l'algorithme reste en $O(\sum_{i=1}^{m} n_i^2)$, avec m le nombre total de mots visuels, et n_i le nombre de fois que le mot visuel i apparaît dans la base.

Comme dans [140], Quack et al. [141] recherchent les configurations de mots visuels fréquentes. La différence est qu'ils emploient une stratégie de recherche d'ensembles de motifs fréquents (*frequent itemsets*) utilisant l'algorithme Apriori [142], à laquelle ils ajoutent la localisation spatiale des mots visuels. Concrètement, les mots visuels sont filtrés sur leur fréquence et leur stabilité temporelle, puis on crée des transactions à partir de chaque mot restant. Pour une transaction, on a donc un mot visuel « central » et des KNN spatiaux, comme dans [140]. Ces KNN sont classés en quatre quadrants spatiaux afin de stocker la localisation relative des KNN. Quack et al. [141] tirent également profit du mouvement des objets dans les vidéos pour réaliser une segmentation, et ainsi filtrer les mots visuels se déplaçant indépendamment dans le temps.

Yuan et. al. [143, 144] emploient également une approche de type *frequent itemsets*. Ils proposent de mesurer la fréquence des *itemsets* relativement à une génération aléatoire d'*itemsets*. La localisation spatiale est également prise en compte, et les auteurs proposent de refaire un parcours de la base pour identifier les transactions comptées plusieurs fois au sein d'une même image. Le partitionnement des descripteurs PCA-SIFT est effectué par un *K-Means*. Ce *K-Means* peut être ré-effectué en utilisant une nouvelle métrique, apprise à partir de la phase d'identification des *frequent itemsets*. Pour finir, les *frequent itemsets* sont *clusterisés* à l'aide d'un algorithme de *normalized cut*.

Yuan et. al [145] proposent un autre type d'approche, basé cette fois sur du hachage. Les descripteurs sont hachés avec LSH, ce qui permet ensuite d'effectuer des requêtes de type rayon dans l'espace des descripteurs. En combinant cette recherche avec une recherche de KNN dans l'espace image (spatial), les

auteurs calculent un score de similarité avec d'autres images de la base, ce qui permet de découvrir des zones d'images fréquentes. Dans [146], Yuan et Wu effectuent plusieurs partitionnements spatiaux aléatoires des images afin de créer des zones d'images qui sont ensuite approximativement recherchées grâce à LSH. Un vote est ensuite appliqué pour déterminer les zones d'images les plus répétées.

Tout comme [140], la complexité de ces méthodes de découverte d'*itemsets* fréquents n'est pas adaptée à de très grands corpus.

Dans [147, 148], Anjulan et Canagarajah segmentent leurs vidéos en shots et en extraient des *tracks* (descripteurs stables temporellement). Une méthode de regroupement basée sur la distance est ensuite appliquée pour rassembler les *tracks* appartenant aux mêmes instances d'objets. Cette méthode est ainsi adaptée à la découverte d'objets dans des vidéos, mais l'inclusion de l'information temporelle empêche l'algorithme d'être appliqué à un volume important de vidéos.

3.2 Méthodes de construction de graphes d'appariement d'images contenant des objets similaires

Dans [82], Philbin propose de construire un graphe d'appariement complet, c'est-à-dire un graphe dont les nœuds sont les images et les arêtes représentent une similarité entre deux régions (vérifiées géométriquement) de ces images. Pour des raisons d'efficacité, la construction du graphe est effectuée à l'aide de mots visuels et d'une liste inversée. Chaque image est recherchée parmi les autres. Les 400 meilleurs résultats de chaque requête sont ensuite vérifiés spatialement, et seules les images comptant au moins 20 appariements avec la requête sont conservées (reliées au nœud « requête » dans le graphe). On remarque que ce seuil minimum de 20 appariements ne permet pas la découverte de petits objets, pouvant être représentés par moins d'une dizaine de descripteurs visuels. La méthode n'étant pas particulièrement robuste, Philbin [82] propose d'utiliser des méthodes capables d'identifier des *clusters* au sein de ce graphe. Par exemple, détecter les composantes connexes (en complexité linéaire grâce à un algorithme de recherche en profondeur d'abord [149]), ou calculer des mesures de type PageRank [150] ou centralité [151]. Le principal problème de la construction d'un tel graphe est le temps de calcul. La complexité de l'algorithme est certes linéaire en nombre d'images, mais l'opération de recherche reste coûteuse, à moins de fortement dégrader la qualité au profit de la vitesse. De plus, cette méthode n'envisage pas la possibilité qu'il y ait plusieurs instances d'objets différents au sein d'une même images.

3.3 Méthodes de découverte de concepts visuels fréquents par extraction de sujets latents

Les modèles de sujets latents (*latent topic models*) issus du domaine textuel, tels que l'analyse sémantique latente probabiliste (pLSA) [152] et l'allocation de Dirichlet latente (LDA) [153] ont également été appliquées au domaine visuel [17, 154, 155], bien que l'objectif visé par ces travaux était de découvrir des catégories visuelles (voiture, chien, maison, ...) dans une collection d'images, plutôt que de découvrir

des *clusters* d'instances d'objets. Ces méthodes se contentent donc de représenter les images comme des sacs de mots visuels, sans inclure d'informations géométriques sur les descripteurs visuels. Ces modèles représentent chaque image comme un mélange de K sujets, où chaque sujet correspond à une classe d'objets.

Dans [156], Philbin et al. introduisent un nouveau modèle génératif de sujets latents pour la découverte d'instances d'objets, nommé gLDA (*Geometric* LDA). A la différence des modèles précédents cités (pLSA, LDA), gLDA intègre des contraintes géométriques dans les sujets (position spatiale et forme des descripteurs visuels), ainsi qu'une transformation géométrique entre les sujets et les documents.

La principale limitation de ce type de modèles est que le nombre de sujets K doit être fixé par l'utilisateur, ce qui est loin d'être évident lorsqu'on s'intéresse à une collection d'images inconnues. Le choix du nombre de sujets est de plus un choix subjectif. Devant une photographie d'un bâtiment, doit-on considérer que le bâtiment lui-même est un sujet, ou bien qu'il est composé de plusieurs sujets (fenêtres, portes, etc.) ? Les sujets eux-mêmes peuvent être ambigus, mélangeant plusieurs objets apparaissant parfois ensemble.

Le temps de calcul nécessaire à l'estimation des paramètres de ces modèles rend ce type d'approche difficilement applicable à de grandes bases d'images.

3.4 Approximation de graphes d'appariement d'instances d'objets par hachage

Dans ce type de méthode, les instances d'objets découvertes sont localisées dans les images (avec une boîte englobante par exemple), en plus d'être détectées.

Les solutions état de l'art existantes qui approximent un tel graphe d'appariement sont basées sur min-Hashing.

Min-Hashing [66] est un algorithme fréquemment employé dans la recherche textuelle pour trouver des copies-proches [157, 158], et fonctionne en approximant l'intersection entre deux ensembles de mots.

Dans cette approche, chaque image est représentée sous forme d'un ensemble de mots visuels. Une fonction de hachage aléatoire f_j assigne un nombre à chaque mot visuel, et impose ainsi un classement spécifique à l'intérieur de chaque ensemble de mots visuels (typiquement chaque image). On peut alors obtenir un *min-Hash* $m_j(A)$ dans chaque ensemble A en sélectionnant le mot visuel ayant la valeur minimum, c'est-à-dire :

$$m_j(A) = \text{argmin} f_j(A) \tag{3.1}$$

Cette fonction a la propriété que la probabilité que deux ensembles $\mathcal{A}_1$ et $\mathcal{A}_2$ aient la même valeur de *min-Hash* est égale au recouvrement de leur ensemble, c'est-à-dire le ratio de l'intersection et de l'union de leurs ensembles :

$$\text{ovr}_1(\mathcal{A}_1, \mathcal{A}_2) = \frac{|\mathcal{A}_1 \cap \mathcal{A}_2|}{|\mathcal{A}_1 \cup \mathcal{A}_2|} \in [0, 1] \tag{3.2}$$

Cette mesure de similarité suppose que tous les mots ont la même importance. Il a été montré dans [158] que la mesure de similarité pouvait être améliorée en utilisant une version pondérée. Soit $d_w \geq 0$ l'importance du mot visuel X_w. Le recouvrement pondéré des ensembles $\mathcal{A}_1$ et $\mathcal{A}_2$ est :

$$\text{ovr}(\mathcal{A}_1, \mathcal{A}_2) = \frac{\sum_{X_w \in \mathcal{A}_1 \cap \mathcal{A}_2} d_w}{\sum_{X_w \in \mathcal{A}_1 \cup \mathcal{A}_2} d_w} \in [0, 1] \tag{3.3}$$

L'une des possibilités les plus utilisées pour choisir des poids d_w est l'approche TF-IDF (*Term Frequency - Inverse Document Frequency*) [159].

La probabilité que deux images aient la même valeur de *min-Hash* est égale à :

$$P\{m(\mathcal{A}_1) = m(\mathcal{A}_2)\} = \text{ovr}(\mathcal{A}_1, \mathcal{A}_2) \tag{3.4}$$

Pour estimer le recouvrement des mots visuels entre deux images, de multiples fonctions *min-Hash* indépendantes f_j sont utilisées. La fraction de fonctions *min-Hash* qui assignent une valeur identique aux deux ensembles donne une estimation de la similarité entre ces deux images. Pour retrouver efficacement les images ayant une forte similarité, les valeurs des fonctions *min-Hash* sont groupées en s-tuples appelés *sketches*. On obtient de meilleurs taux de rappel en répétant k fois l'étape de sélection des s-tuples. Un couple d'images est potentiellement appariable lorsqu'on obtient au moins une collision entre leurs *sketches* respectifs, pour une même itération parmi les k itérations. La probabilité qu'un couple d'images ait au moins un *sketch* en commun parmi les k générés est une fonction de leur recouvrement :

$$P\{\text{collision}\} = 1 - (1 - \text{ovr}(\mathcal{A}_1, \mathcal{A}_2)^s)^k \tag{3.5}$$

Plus on augmente la taille s des *sketches*, plus on diminue la probabilité de collision, mais plus on augmente leur distinctivité (en pratique, il est conseillé de choisir $s \leq 3$). Et plus on augmente le nombre de *sketches* k, et plus on augmente la probabilité de collision, sans autres conséquences particulières, hormis le temps de calcul. Il est donc préférable d'augmenter k plutôt que s.

Geometric min-Hashing [160] apporte une amélioration dans la génération des *sketches* en intégrant des contraintes géométriques, ou plus exactement des contraintes de voisinage. Les *sketches* sont composés de deux parties : un descripteur central et un descripteur secondaire. Le descripteur central est sélectionné par l'approche min-Hashing standard. La seule différence est que seuls les descripteurs ayant des mots visuels uniques au sein de l'image sont considérés. Le descripteur secondaire est ensuite sélectionné parmi les voisins du descripteur central, toujours par min-Hashing. L'idée sous-jacente est que deux descripteurs proches spatialement ont beaucoup plus de chance de faire partie d'un même objet (surtout si l'objet est petit), que deux descripteurs éloignés dans l'image.

Les auteurs de GmH proposent deux applications de leur méthode de hachage : le *clustering* d'images et la découverte de petits objets.

Le *clustering* d'images fonctionne en détectant les composantes connexes au sein du graphe d'appariement. Les appariements sont créés itérativement au fur et à mesure de la découverte de

collisions entre les *sketches*. Les appariements potentiels sont vérifiés géométriquement dès lors qu'ils font intervenir au moins 20 descripteurs, comme cela était suggéré dans [82].

La découverte de petits objets étant plus délicate (la probabilité de collision est très faible), la méthode de vérification des appariements est une cosegmentation [133], qui suppose d'avoir accès aux pixels, et qui est très coûteuse en temps de calcul.

Le principal inconvénient de *Geometric min-Hashing* est qu'il faut générer beaucoup de *sketches* pour avoir une probabilité suffisante de toucher les objets ayant une faible couverture spatiale. La méthode est toutefois la seule à proposer une solution efficace et *scalable* pour découvrir des objets dans de très grandes bases d'images. Sa capacité à découvrir des petits objets est plus discutable, étant donné le manque d'information quant aux réglages des paramètres, et à l'absence d'évaluation quantitative, ou de temps de calcul.

Deuxième partie

Contributions

Chapitre 4

Fouille d'objets visuels fréquents par requêtage aléatoire

4.1 Définition formelle des problèmes de fouille et de découverte d'objets visuels fréquents

4.1.1 Notations

Soit $\boldsymbol{I}$ un ensemble de N_I images I_j, $j \in 1, ...N_I$, représentées par un ensemble $\boldsymbol{X}$ de N descripteurs locaux $\boldsymbol{x}_i$, chacun étant extrait depuis une position $\boldsymbol{p}_{ij} = (I_j, \chi_{ij}, \psi_{ij})$, où χ_{ij} et ψ_{ij} sont les coordonnées spatiales dans l'image.

Considérons maintenant un ensemble $\boldsymbol{O}$ d'objets O^m, chacun étant représenté par S_m instances O^m_s. Une instance O^m_s est associée à une zone unique A^m_s (dans une seule image), et contient un ensemble de descripteurs locaux :

$$\boldsymbol{X}^m_s = \left\{\boldsymbol{x}_i \mid \boldsymbol{p}_{ij} \subset A^m_s\right\}_{1 \leq i \leq N}$$

Introduisons maintenant quelques définitions basiques qui nous aideront ensuite à formaliser les problèmes de découverte et fouille d'objets, et à modéliser une approche adaptée à la résolution de ces dits problèmes.

Définition - Couverture globale :
La couverture globale $c_{\boldsymbol{X}}(O^m)$ d'un objet O^m est définie par :

$$c_{\boldsymbol{X}}(O^m) = \frac{1}{N} \sum_{s=1}^{S_m} |\boldsymbol{X}^m_s| \tag{4.1}$$

Elle mesure le pourcentage de descripteurs locaux dans $\boldsymbol{X}$ couverts par les instances d'un objet O^m donné.

Définition - Couverture moyenne :

La couverture moyenne $c(O^m)$ d'un objet O^m est définie par :

$$c(O^m) = \frac{1}{S_m} \sum_{s=1}^{S_m} \frac{1}{N_s^m} |\boldsymbol{X}_s^m| \tag{4.2}$$

avec N_s^m le nombre de descripteurs locaux dans l'image incluant l'instance O_s^m.

Elle mesure la taille moyenne d'un objet O^m (en pourcentage moyen de descripteurs locaux couverts par une instance de l'objet à travers toutes les images).

Définition - Fréquence :

La fréquence $f(O^m)$ d'un objet O^m mesure le pourcentage d'images couvertes par une des instances de cet objet et est définie par :

$$f(O^m) = \frac{S_m}{N_I} \tag{4.3}$$

La présence de multiples instances d'un même objet dans une même image peut conduire à une fréquence $f(O^m)$ supérieure à 1.

Définition - Objet $\{c, f\}$-fréquent :

Un objet O^m est dit $\{c, f\}$-fréquent si :

$$\begin{cases} c(O^m) = c \\ f(O^m) = f \end{cases}$$

On remarque ici que si le nombre de descripteurs par image est assez stable, alors on peut considérer que :

$$c_{\boldsymbol{X}} \approx c.f$$

4.1.2 Formalisation des problèmes de découverte et fouille d'objets fréquents

A partir des définitions précédentes, il est maintenant possible de définir les deux problèmes suivants comme les deux principaux objectifs à atteindre par les méthodes de découverte et de fouille d'objets visuels.

Définition - Découverte d'objets :

Trouver au moins une instance O_s^m de tous les objets $\{c, f\}$-fréquents O^m tels que :

$$\begin{cases} c \geq c_0 \\ f \geq f_0 \end{cases}$$

Définition - Fouille d'objets :
Trouver toutes les instances O_s^m de tous les objets $\{c, f\}$-fréquents O^m tels que :

$$\begin{cases} c \geq c_0 \\ f \geq f_0 \end{cases}$$

L'idée sous-jacente est que la complexité de ces deux problèmes dépend principalement des paramètres c et f. Nous suggérons donc d'inclure ces deux paramètres dans la formulation du problème, tout comme cela est fait dans les approches classiques de fouille de données (e.g. la fouille d'item-sets fréquents [161])

4.2 Recherche itérative par échantillonnage pondéré adaptatif (RANSAS)

La méthode de découverte proposée ici est un processus itératif que l'on nommera par la suite « RANSAS » (RANdom Sample And Search model), dont chaque itération est composée de trois étapes principales. La première étape est celle de l'échantillonnage pondéré adaptatif d'une région locale d'image. La deuxième étape consiste en la recherche précise de la zone d'image sélectionnée. Enfin, la troisième étape est chargée de décider si oui ou non cette région et ses résultats devraient être considérés comme étant des instances d'un objet $\{c, f\}$-fréquent. Étant donné que les régions ainsi appariées peuvent ne représenter qu'une partie d'un objet, nous parlons parfois de « germes » (*seeds* [160]), ou dans notre première publication sur le sujet [162] de « mots visuels géométriquement cohérents ». L'algorithme répète ces trois étapes T fois ($T \leq T_{\text{max}}$, T_{max} fixé par l'utilisateur), jusqu'à ce qu'un nombre d'objets $\{c, f\}$-fréquents aient été découverts, ou qu'un temps limite ait été atteint.

Au démarrage de l'algorithme, chaque descripteur local $\boldsymbol{x}_i$ est associé à une potentielle région requête R_i, définie comme un rectangle centré autour de la position $\boldsymbol{p}_{ij}$ et dont les dimensions H_i' et W_i' sont calculées à partir des dimensions de l'image I_j de la manière suivante :

$$H_i' = \sqrt{\beta} * H_j$$

$$W_i' = \sqrt{\beta} * W_j$$

où β est un paramètre correspondant au pourcentage de la surface d'image couverte par une région requête candidate, et où W_j et H_j sont respectivement la largeur et la hauteur de l'image $\boldsymbol{I}_j$. Par exemple, $\beta = 0.10$ signifie que les requêtes auront une surface correspondant à 10% de la surface de leurs images hôtes, soit une largeur et une hauteur correspondant à un tiers de celles de l'image. L'intérêt de ce paramètre est de pouvoir adapter la taille et la forme d'une requête à celle de l'image. Cette approche est une hypothèse naïve nécessaire afin de modéliser le problème dans un premier temps, et mériterait une plus grande attention dans le futur.

Les trois étapes de l'algorithme RANSAS sont ensuite appliquées à chaque itération. Le schéma de la figure 4.1 page ci-contre présente un récapitulatif du fonctionnement de RANSAS

4.2.1 Échantillonnage pondéré adaptatif

Pour éviter de requêter toutes les régions possibles tout en conservant une couverture maximum du contenu des images, la stratégie proposée dans cette thèse repose sur un échantillonnage pondéré et adaptatif ayant pour objectif de sélectionner, à chaque itération t, une région la plus pertinente possible (en fonction du contenu de la base, et des itérations passées).

Cette $t^{\text{ème}}$ région requête candidate, que l'on note R_q^t, est centrée autour d'un descripteur local $\boldsymbol{x}_q^t$ sélectionné aléatoirement à l'aide d'une fonction de probabilité de masse $p_t(i)$, sur l'ensemble de tous les descripteurs de la base. Un tel échantillonnage peut-être obtenu facilement depuis n'importe quelle distribution p_t en utilisant la méthode de la transformée inverse [163] (également appelée transformée de Smirnov). Cette méthode consiste à transformer un nombre aléatoire (uniforme) en intégrant la fonction de probabilité de masse jusqu'à obtenir une valeur (surface) plus grande que le nombre transformé.

L'échantillonnage est une méthode statistique visant à sélectionner une sous-partie d'un ensemble d'individus dans une population, permettant d'estimer les caractéristiques de la population entière. Lorsque tous les individus ont la même probabilité d'être sélectionnés, le problème est connu comme un problème d'échantillonnage aléatoire *uniforme*.

Dans les méthodes d'échantillonnage *pondéré* [164], les individus sont pondérés individuellement et la probabilité de chaque individu est déterminée par son poids relatif.

Dans les schémas conventionnels d'échantillonnage, *uniforme* ou *pondéré*, la sélection d'un individu ne dépend pas des observations passées. Au contraire, dans un schéma d'échantillonnage *adaptatif* [165], la stratégie consiste à adapter la sélection d'un individu en tenant compte des sélections précédentes.

Notre méthode propose un échantillonnage *pondéré* et *adaptatif*. Elle débute à l'itération $t = 0$ avec une fonction de probabilité de masse $p_0(i)$ couvrant l'ensemble des descripteurs (et donc l'ensemble des régions requêtes candidates R_i). La distribution initiale peut être uniforme ou déterminée par une connaissance a priori, comme discuté dans la section 4.3 page 62, et plus en détail dans les chapitres 5 page 71 et 6 page 81.

Une fois appliquées les trois étapes de l'algorithme RANSAS, nous obtenons à l'itération $t = 0$ une région requête R_q^0 et des régions résultats R_m^0, $m \in 1, ..., M_0$ (si l'étape de décision en a retourné).

A chaque itération, la fonction de probabilité de masse est mise à jour à l'aide d'une fonction f, prenant en paramètres les probabilités issues de l'itération précédente, la requête effectuée, ainsi que les résultats :

$$p_t = f\left(p_{t-1}, R_q^{t-1}, \{R_m^{t-1}\}\right)$$

De même que dans les méthodes d'échantillonnage pondéré et adaptatif habituelles [164], la fonction de probabilité de masse p_t est en pratique calculée en normalisant une fonction de pondération z_t. Nous verrons dans la section 4.4 page 64 comment normaliser ces poids.

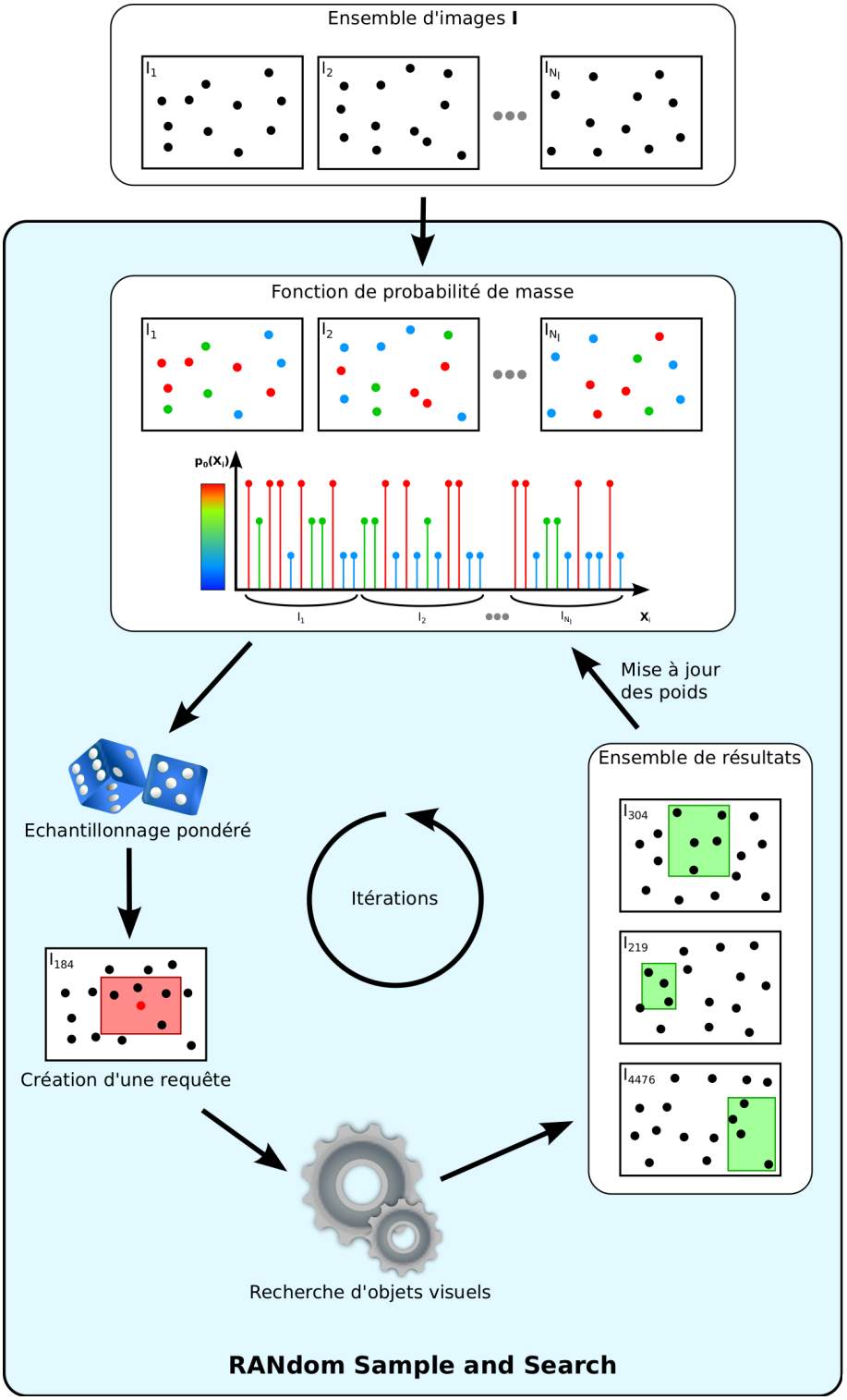


FIGURE 4.1 – Schéma récapitulatif du fonctionnement de l'algorithme RANSAS

En pratique donc, la mise à jour des poids est effectuée de manière récursive :

$$z_t = g\left(z_{t-1}, R_q^{t-1}, \{R_m^{t-1}\}\right)$$

avec la fonction g définie telle que :

$$z_t(i) = \begin{cases} 0 & \text{si } \boldsymbol{x}_i = \boldsymbol{x}_q^{t-1} \\ \alpha_1 \; z_{t-1}(i) & \text{si } \boldsymbol{x}_i \in R_q^{t-1} \\ \alpha_2 \; z_{t-1}(i) & \text{si } \boldsymbol{x}_i \in \left\{R_m^{t-1}\right\}_m \\ z_{t-1}(i) & \text{sinon} \end{cases} \tag{4.4}$$

La première condition garantit que le descripteur sélectionné pour créer la région requête candidate ne sera plus jamais sélectionné (à la manière d'un tirage sans remise).

La deuxième condition diminue le poids des descripteurs appartenant à la région requête R_q^{t-1}, afin de diminuer leur probabilité d'être sélectionnés comme centres de nouvelles requêtes. Ceci permet d'éviter de créer de nouvelles requêtes ayant un trop fort recouvrement avec celles des itérations précédentes.

La troisième condition vise à diminuer le poids des descripteurs appartenant à des régions résultats, toujours dans l'optique de créer de nouvelles requêtes différentes des précédentes. Cette condition a pour effet d'augmenter la probabilité de créer une requête à partir d'un objet différent, et non plus seulement d'une instance d'objet différente.

Enfin, la dernière des conditions garantit que les poids des descripteurs qui ne sont pas intervenus lors de l'itération courante, ne seront pas modifiés.

En pratique, α_1 et α_2 sont compris entre 0 et 1, avec $\alpha_1 < \alpha_2$.

L'utilisation de faibles valeurs pour ces paramètres accélère le processus de découverte, en maximisant les chances de sélectionner des zones d'images nouvelles. Toutefois, une trop forte diminution de α_1 et α_2 provoque une baisse globale du rappel. En effet, requêter plusieurs instances d'un même objet est le principe à la base des méthodes d'extension de requêtes [4], ce qui améliore le rappel de l'objet considéré.

Les expérimentations ont permis de déterminer empiriquement des valeurs adaptées :

$$\alpha_1 = 0.01 \qquad \alpha_2 = 0.05$$

Les courbes de la figure 4.2 page suivante montrent l'effet de la mise à jour des poids. Nous constatons par exemple que pour couvrir 90% des descripteurs, il est nécessaire d'effectuer deux fois plus d'itérations lorsqu'on utilise un échantillonnage simple (non adaptatif).

4.2.2 Recherche précise d'une région locale

L'étape de recherche locale consiste à effectuer une recherche d'objets visuels à partir de la zone sélectionnée lors de l'étape d'échantillonnage. Comme nous l'avons vu dans la section 2.7 page 43, il existe dans l'état de l'art plusieurs méthodes de recherche d'images par le contenu d'un niveau état de

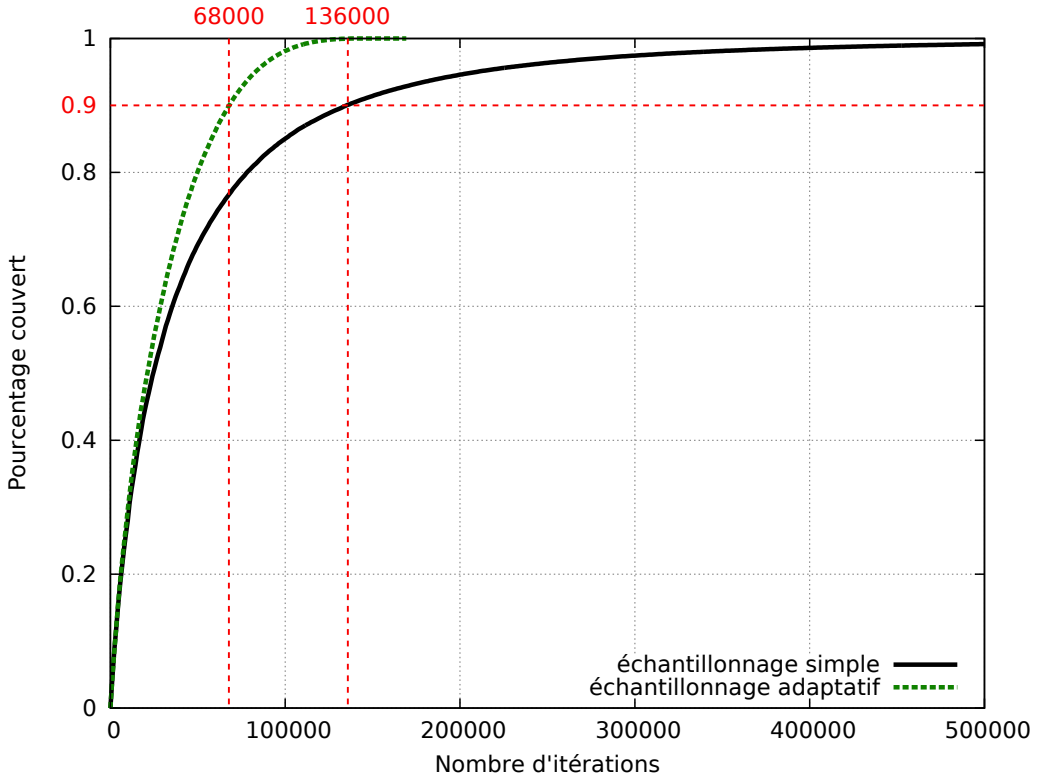


FIGURE 4.2 – Comparaison de la surface couverte (en pourcentage du nombre de descripteurs) en fonction du nombre d'itérations T de RANSAS, et du type d'échantillonnage (simple ou adaptatif).

l'art pouvant convenir à cette phase. Nous discuterons les avantages et inconvénients de ces méthodes dans la section 4.5 page 65, et nous verrons pourquoi la solution retenue est basée sur les travaux de Joly et al. [103, 4, 90].

4.2.3 Décision

La dernière étape de RANSAS est chargée de décider si oui ou non un résultat de recherche de région locale doit être conservé (dans le cadre du processus de découverte). Cette étape est très fortement couplée à la précédente puisque elle consiste principalement à post-traiter les scores de similarité (normalisation, seuillage).

Finalement, après T requêtes candidates, l'algorithme produit un vocabulaire V de $|V| \leq T$ mots visuels géométriquement cohérents v_t, ayant une forte vraisemblance d'appartenir à un objet $\{c, f\}$-fréquent. Chaque mot visuel est associé à une région d'image R_q^t et représenté par un ensemble de descripteurs locaux appartenant à R_q^t.

4.3 Modèle de coût de l'algorithme RANSAS

La fonction de probabilité de masse p_0 étant la condition initiale de l'algorithme d'échantillonnage pondéré adaptatif, la manière dont elle a été construite a un impact déterminant sur les performances globales du processus de découverte, c'est-à-dire sur le nombre T d'itérations requises pour découvrir les instances des objets $\{c, f\}$-fréquents. Comme nous le verrons dans la section 7.2.2 page 99, une bonne initialisation peut permettre de diviser par 32 le nombre d'itérations nécessaires pour atteindre les performances obtenues avec une initialisation uniforme.

Il est bien sûr évident que la rapidité et l'efficacité du système effectuant la recherche précise ont une influence sur les performances de la méthode, mais il n'en demeure pas moins que la complexité globale de l'approche reste $O(T)$, quelle que soit la vitesse de l'algorithme de recherche utilisé (bien supérieure au temps de calcul de la fonction de probabilité de masse p_0). La modélisation du nombre d'itérations T nécessaires est plus difficile à modéliser car elle dépend du rappel et de la précision de l'algorithme de recherche en lui-même.

En prenant l'hypothèse d'une recherche « parfaite », nous parvenons tout de même à modéliser le comportement global du système. Considérons une base d'images avec un seul objet $\{c, f\}$-fréquent O^m. Soit $\boldsymbol{x}_t$ le descripteur candidat sélectionné lors de la phase d'échantillonnage dans $\boldsymbol{X}$, selon une fonction de probabilité de masse p_t. Tant que $\boldsymbol{x}_t$ n'appartient pas à une instance O^m_s de O^m, p_t reste égal à p_0 (toujours si le système de recherche précise possède une précision parfaite).

D'un autre côté, si $\boldsymbol{x}_t$ appartient à une instance O^m_s de O^m, le problème de la fouille d'objets est résolu (si le système de recherche précise possède un rappel parfait).

Ainsi, le nombre d'itérations requises T correspond au nombre de tirages successifs sans succès, c'est-à-dire ne tombant pas dans une instance O^m_s. Cette variable aléatoire suit une distribution binomiale négative $BR(1, 1 - c_0(O^m))$ où la probabilité d'échec est la probabilité qu'un descripteur échantillonné appartienne à une instance O^m_s de O^m, c'est-à-dire :

$$c_0(O^m) = \sum_{\boldsymbol{x}_i \in \{O^m_s\}} p_0(\boldsymbol{x}_i)$$

Le nombre d'itérations nécessaires estimé est donc égal à l'espérance de la loi binomiale négative $BR(1, 1 - c_0(O^m))$:

$$\hat{T} = \frac{1 - c_0(O^m)}{c_0(O^m)} = \frac{1}{c_0(O^m)} - 1 \tag{4.5}$$

et la complexité de l'algorithme est $O(\frac{1}{c_0(O^m)})$.

Nous nous réfèrerons généralement à $c_0(O^m)$, comme la couverture statistique de l'objet O^m, étant donné qu'elle mesure le pourcentage de la fonction de probabilité de masse concerné par l'objet. Ceci montre que l'algorithme est fortement lié à la fonction de probabilité de masse p_0.

Si on considère par exemple une fonction de probabilité de masse parfaite, c'est-à-dire :

$$p_0(\boldsymbol{x}_i) = 0, \forall \boldsymbol{x}_i \notin \{\boldsymbol{X}^m_s\}$$

$$p_0(\boldsymbol{x}_i) = \frac{1}{|\{\boldsymbol{X}_s^m\}|}, \forall \boldsymbol{x}_i \in \{\boldsymbol{X}_s^m\}$$

avec $\{\boldsymbol{X}_s^m\}$ l'ensemble des descripteurs appartenant à l'objet O^m,
alors, on a $c_0(O^m) = 1$, et la complexité de l'algorithme est $O(1)$.

D'un autre côté, si l'on considère une fonction de probabilité de masse uniforme, telle que :

$$p_0(\boldsymbol{x}_i) = 1/N, \forall \boldsymbol{x}_i \in \boldsymbol{X}$$

avec N le nombre de descripteurs,
alors la complexité devient $O(c_{\boldsymbol{X}}(O^m))$, où $c_{\boldsymbol{X}}(O^m)$ est la couverture globale de l'objet défini dans la section 4.1.1 page 55.

Étant donné que $c_{\boldsymbol{X}}(O^m) \approx c(O^m).f(O^m)$ si le nombre de points par image est globalement stable, la complexité de l'algorithme de fouille d'objets $\{c, f\}$-fréquents avec un échantillonnage uniforme tend approximativement vers $O(\frac{1}{cf})$. Ceci illustre bien pourquoi la fouille de petits objets peu fréquents est une tâche bien plus complexe que la fouille d'objets occupant une large portion des images ($c \approx 1$).

La figure 4.3 présente des histogrammes de nombre d'instances en fonction du nombre de descripteurs SIFT pour les bases BelgaLogos et Oxford Buildings. On voit nettement que la couverture des objets de BelgaLogos est nettement inférieure à celle d'Oxford Buildings, ce qui fait des logos de cette première des objets bien plus difficiles à fouiller que les bâtiments d'Oxford. Dans la base Oxford Buildings, 20% des descripteurs appartiennent à des objets de la vérité terrain, contre seulement 0,5% dans la base BelgaLogos. Autrement dit, en sélectionnant au hasard un descripteur de la base Oxford Buildings, on a une chance sur cinq de trouver un bâtiment, contre une chance sur deux-cents pour BelgaLogos. Pour cette raison, nous nous focaliserons sur la découverte des instances issues de BelgaLogos (cf. section 7.1 page 95), représentant un véritable défi.

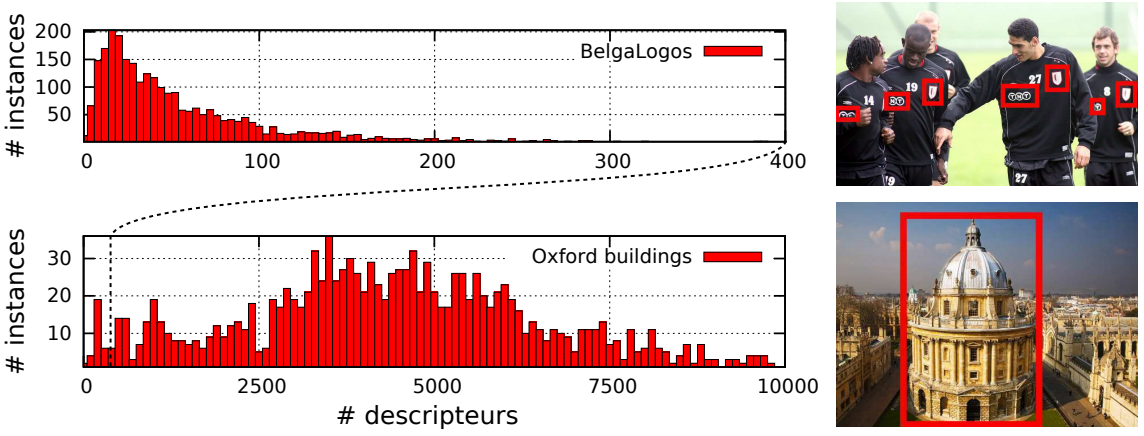


FIGURE 4.3 – Comparaison des tailles d'instances des bases BelgaLogos et Oxford Buildings

Considérons maintenant un algorithme de recherche précise « réel » plutôt qu'un algorithme « parfait », c'est-à-dire un algorithme ne retournant seulement qu'une fraction r des instances d'un

objet O^m quand un descripteur requête $\boldsymbol{x}_t$ appartenant à une des instances est sélectionné par l'échantillonnage.

Alors, un rappel élevé sera obtenu seulement si un nombre suffisant μ d'instances du même objet sont sélectionnées par l'échantillonnage. En faisant l'hypothèse que toutes les instances sont indépendantes d'une requête à l'autre, la probabilité qu'une instance soit manquée après μ requêtes est égale à $(1-r)^\mu$. Si l'on souhaite que cette probabilité soit inférieure à une probabilité ϵ, alors on obtient $\mu > \log(\epsilon)/\log(1-r)$.

Dans ce cas, le nombre d'itérations requises T suivra une distribution binomiale négative $BR(\mu, 1-c_0(O^m))$, au lieu de $BR(1, 1-c_0(O^m))$. Et le nombre d'itérations nécessaires peut être estimé par l'espérance :

$$\hat{T} = \frac{\log(\epsilon)}{\log(1-r)} \left(\frac{1}{c_0(O^m)} - 1 \right) \tag{4.6}$$

La complexité du processus global est donc toujours $O(\frac{1}{c_0(O^m)})$ pour des valeurs r et ϵ données. Pour de relativement bonnes valeurs de rappel r et une assez faible différence de performances entre les méthodes de recherche précise de l'état de l'art, on peut avancer que la fonction de probabilité de masse p_0 a un bien plus fort impact sur la complexité que l'efficacité de l'algorithme de recherche précise.

4.4 Conversion des poids en fonction de probabilité

Après que tous les poids individuels z_0 aient été calculés, ils sont convertis en fonction de probabilité de masse afin de pouvoir être utilisés pour la phase d'échantillonnage.

La conversion est effectuée par :

$$p_0 = \frac{z_0}{z_{\max}} . \frac{1}{p_z(z_0)} \tag{4.7}$$

où

$$z_{max} = \max_{\boldsymbol{x}_i \in \boldsymbol{X}} z_0(\boldsymbol{x}_i)$$

et $p_z(z)$ représente la probabilité que le poids z_0 soit égal à la valeur z dans l'ensemble des poids :

$$p_z(z) = \frac{\#\{\boldsymbol{x}_i \in \boldsymbol{X} \mid z_0(\boldsymbol{x}_i) = z\}}{N} \tag{4.8}$$

Ceci permet de distribuer de façon égale les probabilités sur l'intervalle dans lequel varient les poids z_0. En effectuant une simple normalisation linéaire des poids z_0, nous aurions conservé une masse de probabilité très importante sur les poids les plus faibles. Ces faibles poids correspondent en effet à la très grande majorité des descripteurs (qui n'appartiennent pas à des objets fréquents). Ce phénomène est illustré dans la figure 4.4 page ci-contre qui montre un histogramme des poids z_0 (c'est-à-dire $N.p_z(z_0)$) sur la base FlickrBelgaLogos (présentée dans la section 7.1 page 95) utilisée dans nos expérimentations (voir sections 5.3 page 77, 6.5 page 87 et 7.2 page 98).

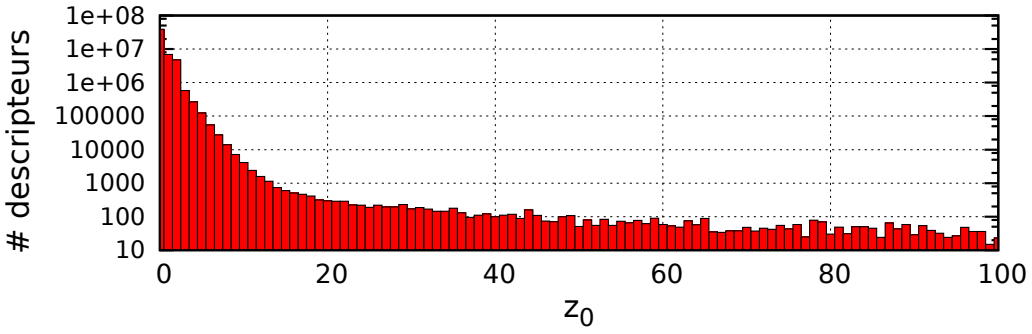


FIGURE 4.4 – Histogramme des poids z_0 (en échelle logarithmique)

4.5 Choix et implémentation de la méthode de recherche précise

Pour la recherche de régions locales, nous avons utilisé un système de recherche d'images dont les composants principaux sont décrits dans [103, 4, 90]. Les sections ci-après ont pour objectif de justifier le choix des méthodes et des implémentations de chacun des composants du système de recherche précise utilisé.

4.5.1 Description des images

L'objectif de cette thèse n'étant pas d'améliorer l'étape de description, nous avons utilisé une solution de l'état de l'art. Notre choix s'est porté sur des points d'intérêt dits DoG et des descripteurs SIFT [23]. Cette solution a l'avantage d'être générique et répandue. Il est donc ainsi plus facile d'effectuer des comparaisons, et de reproduire les contributions proposées dans cette thèse. L'utilisation du détecteur Hessian affine [27, 28] à la place de DoG permettrait sans doute d'obtenir de meilleures performances (au détriment du temps de description), mais nous n'avons pour l'instant pas poussé plus loin cette investigation.

4.5.2 Partitionnement des descripteurs SIFT

Comme nous l'avons vu dans la section 2.7 page 43, et en particulier dans le tableau 2.1, les systèmes états de l'art utilisent soit des méthodes de hachage comme RMMH [90], soit des méthodes de *clustering* comme AKM [94]. L'avantage des méthodes de hachage par rapport aux méthodes de *clustering* est leur rapidité. La complexité du hachage est linéaire par rapport au nombre de descripteurs et au nombre de fonctions de hachage, tandis qu'elle est sur-linéaire par rapport au nombre de descripteurs pour les méthodes de *clustering*. Le tableau 4.1 page suivante montre les temps de calcul pour partitionner deux différents ensembles de vecteurs et met en évidence ce problème de sur-linéarité.

L'idée fréquemment répandue qui justifie l'emploi de méthodes de *clustering* comme AKM, outre le fait qu'elle propose un très bon partitionnement (une bonne minimisation des distances intra-clusters) et un bon équilibrage des tailles de *clusters*, est que cette étape est effectuée « *offline* », une fois pour

		52M descripteurs		1G descripteurs	
Méthode	Paramètres	P. indep	P. dep	P. indep	P. dep
RMMH	128 bits	$\sim$ 647	647	$\sim$ 12 442	$\sim$ 12 442
	1 024 bits	$\sim$ 5 176	$\sim$ 5 176	$\sim$ 99 536	$\sim$ 99 536
AKM	1M mots	1 386	$\sim$ 41 580	$\sim$ 26 653	$\sim$ 799 615
	2M mots	1 680	$\sim$ 50 400	$\sim$ 32 307	$\sim$ 969 210
	4M mots	2 711	$\sim$ 81 330	$\sim$ 52 134	$\sim$ 1 564 020
	8M mots	5 297	$\sim$ 158 910	$\sim$ 101 865	$\sim$ 3 055 950

TABLE 4.1 – Comparaison des temps de calcul (en secondes) de AKM et RMMH.
La comparaison est effectuée sur une base de 52 millions de descripteurs SIFT, et sur une base d'un milliard de descripteurs. « P. dep / indep » signifie « partitionnement dépendant / indépendant des données ». On considère que le temps de calcul de RMMH est identique selon que l'on réalise son apprentissage sur les données partitionnées ou non, tandis que pour AKM, on considère le cas « dépendant » : on effectue le *clustering* sur la base elle même (avec 30 itérations), ou « indépendant » : on possède un vocabulaire, et on effectue seulement l'étape d'affectation. Seuls les temps colorés en vert ont réellement été mesurés, les autres sont des estimations inférées à partir de la complexité des algorithmes utilisés et des temps réels en vert. On considère que les temps sont proportionnels au nombre de descripteurs, et au nombre d'itérations de AKM. Les mesures ont été effectuées sur un serveur équipé de deux processeurs hexa-cores (Intel X5660).

toutes, et qu'elle peut être facilement distribuée sur de multiples serveurs de calcul. Ce temps ne devrait donc pas être considéré comme problématique. Si l'argument peut être accepté pour un processus de recherche d'images par le contenu (les temps de description, de partitionnement, d'indexation, etc. sont minoritaires par rapport au temps passé à effectuer les nombreuses requêtes des nombreux utilisateurs), il n'est plus valable dès lors que l'on se place dans le cas d'un processus de découverte d'objets visuels. En effet, on cherche alors à obtenir un résultat le plus rapidement possible (relativement à la qualité attendue), et ces étapes font parties intégrantes du processus.

Pour toutes ces raisons, nous avons donc choisi d'utiliser RMMH pour partitionner nos descripteurs, en tant que méthode de hachage état de l'art.

4.5.3 Indexation et recherche des descripteurs SIFT

Chacun des descripteurs $\boldsymbol{x}_{qm}$ composant une requête donnée Q est recherché dans l'ensemble des descripteurs de la base grâce à la méthode de recherche approximative décrite dans APMP-LSH [103], utilisant une stratégie *Multi-Probe* (recherches multiples dans une même structure d'index), combinée avec un plongement dans l'espace de Hamming (*Hamming Embedding* [108, 109]).

Le plongement dans cet espace a l'intérêt de réduire considérablement l'espace mémoire alloué aux descripteurs locaux. Les expériences ont montré que des descripteurs SIFT représentés par 128 bits dans l'espace de Hamming permettent d'obtenir des performances équivalentes à 128x32 bits dans l'espace Euclidien. Nous obtenons donc un facteur de compression de 32.

L'utilisation de la distance de Hamming permet de plus de réduire le temps de calcul en comparaison

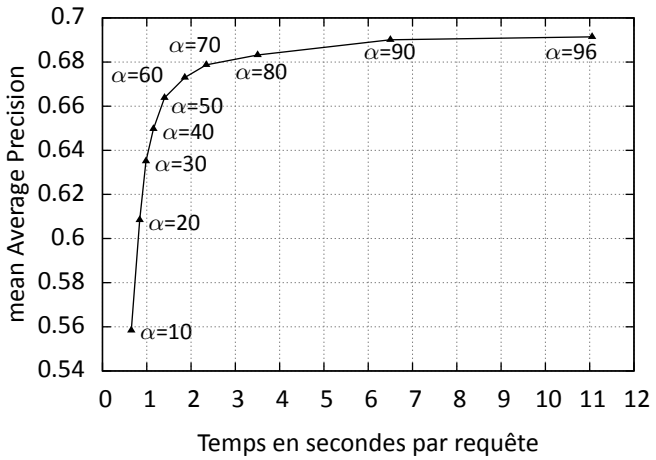


FIGURE 4.5 – Compromis entre qualité (mesurée par la mAP sur Oxford Buildings) et temps de calcul

de la distance Euclidienne. Cette distance peut en effet être calculée à partir de deux opérations extrêmement rapide : un XOR (OU exclusif) entre deux descripteurs, et le comptage du nombre de bits égaux à 1, résultant de ce XOR. Cette deuxième opération peut être effectuée très efficacement par le biais de l'instruction SSE4.2 « popcnt » qui divise par 4 le temps de calcul comparé à l'approche utilisant des tables de correspondances.

L'utilisation de APMP-LSH[103] a le gros avantage de permettre de régler la qualité de recherche α (paramètre réglant le pourcentage de « vrais » plus proches voisins que l'on est statistiquement assuré de retrouver). Ceci permet dont d'adapter le système pour utiliser un bon compromis qualité / temps de calcul. La figure 4.5 montre la variation de la *mean Average Precision* et du temps de calcul obtenus sur la base Oxford Buildings, en fonction du paramètre α. On constate qu'il n'est pas très rentable d'utiliser une valeur de α supérieure à 60% ou 80%. Le temps de calcul indiqué n'a de valeur qu'à titre comparatif. De plus, APMP-LSH garantit une évolution sous-linéaire du temps de calcul avec la croissance de la base de descripteurs.

Cette étape produit en sortie une liste de M_j descripteurs candidats $\boldsymbol{x}_{jm}$ pour chaque descripteur requête $\boldsymbol{x}_{qm}$.

4.5.4 Vérification de la cohérence géométrique

Pour chacune des images I_j contenant plus d'un nombre minimum de descripteurs candidats $\boldsymbol{x}_{jm}$ (en général 3), nous calculons un score de cohérence géométrique $S_Q(I_j)$ en estimant le modèle d'une transformation affine (A_j, B_j) à cinq degrés de liberté entre l'image requête et chaque image candidate grâce à un algorithme de type RANSAC décrit dans [4].

Le score de similarité d'une image I_j pour une requête Q est donné par le nombre d'*inliers* respectant

le meilleur modèle de transformation (selon RANSAC) :

$$S_Q(I_j) = \sum_{m=1}^{M_j} \delta(\| \begin{bmatrix} \chi_{qm} \\ \psi_{qm} \end{bmatrix} - \mathbf{A}_j \begin{bmatrix} \chi_{jm} \\ \psi_{jm} \end{bmatrix} + \mathbf{B}_j \| \geq t) \tag{4.9}$$

où $\delta(d \geq t)$ est égal à 1 si $d \geq t$, et 0 sinon. t est un seuil fixe gérant la tolérance sur l'erreur de position ($t = 8$). M_j est le nombre de descripteurs candidats dans l'image I_j, $\begin{bmatrix} \chi_{qm} \\ \psi_{qm} \end{bmatrix}$ et $\begin{bmatrix} \chi_{jm} \\ \psi_{jm} \end{bmatrix}$ sont les coordonnées spatiales des points d'intérêts du $m^{\text{ème}}$ appariement entre l'image requête I_q et l'image candidate I_j.

Afin de décider si la région requête R_q^t et ses résultats doivent être validés comme étant un mot visuel géométriquement cohérent (c'est-à-dire une partie d'un potentiel objet $\{c, f\}$-fréquent), nous filtrons les résultats en normalisant les scores S_Q obtenus précédemment, par une technique a contrario [4].

Cette technique de normalisation a contrario permet de contrôler précisément le risque de fausses alarmes, en estimant leur distribution $\hat{N}_{fa}(S)$, avec la variable discrète $S = S_Q(I_l)$ et $I_l \in \boldsymbol{I}$. Selon l'équation 4.9, S_Q dépend uniquement de l'ensemble des M_j appariements de descripteurs (c'est-à-dire de leurs coordonnées spatiales). Les fortes valeurs de S_Q sont alors directement liées à la dépendance statistique entre les positions spatiales des descripteurs requêtes et candidats. Aussi, nous définissons notre modèle a contrario par la fonction de probabilité de masse $\hat{p}_{fa}(S)$ de la variable S sous l'hypothèse $\mathcal{H}_0^Q$ que $\begin{bmatrix} \chi_{qm} \\ \psi_{qm} \end{bmatrix}$ et $\begin{bmatrix} \chi_{jm} \\ \psi_{jm} \end{bmatrix}$ sont des variables aléatoires indépendantes pour chaque Image I_j :

$$\hat{p}_{fa}(S) = \mathsf{Pr}[S_Q(I_l) = S \mid \mathcal{H}_0^Q]$$

La fonction de distribution cumulée $\hat{N}_{fa}(S)$ est obtenue par :

$$\hat{N}_{fa}(S) = \sum_{s=0}^{S} \hat{p}_{fa}(s)$$

Nous conservons finalement comme score normalisé $\hat{S}_Q(I)$ une estimation de la précision des résultats selon $\hat{N}_{fa}(S)$:

$$\hat{S}_Q(I_l) = \frac{\#\{I_j \in \Omega, S_Q(I_j) > S_Q(I_l)\} - N.\hat{N}_{fa}(S_Q(I_l))}{\#\{I_j \in \Omega, S_Q(I_j) > S_Q(I_l)\}}$$

En pratique, nous estimons la fonction de probabilité de masse $\hat{p}_{fa}(S)$ pour chaque requête Q par une simulation de Monte Carlo. Nous générons des coordonnées spatiales indépendantes $\begin{bmatrix} \chi_{qm} \\ \psi_{qm} \end{bmatrix}$ pour les descripteurs requêtes, et on conserve les coordonnées $\begin{bmatrix} \chi_{jm} \\ \psi_{jm} \end{bmatrix}$ des descripteurs candidats tels quels.

Plus précisément, on affecte à un descripteur requête $\boldsymbol{x}_{qi}$ une nouvelle position spatiale $\boldsymbol{x}_{qi'}$ sélectionnée aléatoirement parmi les autres points de la requête. Comparée à une génération aléatoire purement uniforme des coordonnées spatiales, cette méthode a l'avantage de préserver une certaine connaissance a priori de la distribution des points, comme les limites ou les orientations principales.

Nous pouvons ensuite estimer $\hat{p}_{fa}(S)$ en comptant le nombre de résultats ayant un score S_Q supérieur à S.

4.5.5 Extension de requêtes

L'extension de requêtes est un processus très lent puisqu'il nécessite de multiplier le nombre de requêtes effectuées. Son but premier étant d'améliorer le rappel, elle n'est pas nécessaire pour le problème de la découverte des objets visuels fréquents. Elle pourrait l'être pour la fouille, afin de trouver un maximum d'instances de l'objet requêté, mais il est plus rentable de multiplier les requêtes issues de notre processus d'échantillonnage adaptatif et pondéré que d'augmenter le rappel individuel de chaque requête.

Chapitre 5

Calcul des scores de vraisemblance par des méthodes de l'état de l'art

Comme nous l'avons vu au chapitre précédent, l'algorithme RANSAS nécessite d'être initialisé à l'aide d'une distribution a priori p_0 calculée à partir de poids ou scores de vraisemblance z_0.

Ce chapitre a pour objectif de proposer des méthodes de calcul de ces scores de vraisemblance à partir de méthodes de l'état de l'art. Nous proposerons tout d'abord des méthodes spécifiques à un type d'objet, puis des méthodes généralistes basées sur des heuristiques simples, un simple comptage des appariements, ou une solution plus sophistiquée basée sur Geometric min-Hashing.

5.1 Calcul de scores de vraisemblance spécifiques à un type d'objet

Dans cette section sont présentées les scores de vraisemblance spécifiques à certains concepts visuels. Deux exemples illustratifs des possibilités offertes par ces méthodes sont proposés. Le premier génère des scores différents selon la position dans l'image, et le deuxième se focalise sur les visages détectés. On pourrait bien entendu proposer de nombreuses autres alternatives, dans la catégorie « détecteur » (détecteur de texte, détecteur de mouvement dans le cas des vidéos …), ou dans la catégorie « a priori » (couleur, texture, …).

5.1.1 Scores de vraisemblance basés sur la position dans les images

Le cadrage des photographies ou vidéos obéit généralement à certaines règles esthétiques : les objets d'intérêt sont placés au centre du cadre, sur des lignes placées sur les tiers de l'image, etc.

L'idée de cette distribution spécifique à la position est donc d'utiliser ces règles afin que pour chaque descripteur, sa probabilité d'être échantillonné par l'algorithme RANSAS dépende directement de sa distance au centre de l'image ou au tiers le plus proche.

Si l'on prend l'exemple d'une distribution calculée par rapport au centre de l'image, on a :

$$z_0^P(\boldsymbol{x}_i) = \mathcal{K}_\sigma(\mathbf{c}_{I_j}, \boldsymbol{p}_{ij})$$

où $\mathbf{C}_{I_j}$ est le centre de l'image I_j, $\boldsymbol{p}_{ij} = (I_j, \chi_{ij}, \psi_{ij})$ est la position du point $\boldsymbol{x}_i$ et $\mathcal{K}$ le noyau RBF paramétré dans le cadre de nos expérimentations avec $\sigma_j^{\chi} = \frac{W_j}{8}$, $\sigma_j^{\psi} = \frac{H_j}{8}$.

La figure 5.1 présente un exemple d'un objet centré dans l'image.

FIGURE 5.1 – Score de vraisemblance basé sur la position centrale dans les images. Plus la couleur est chaude, et plus on est proche du centre de l'image

La figure 5.2 page ci-contre montre qu'en calculant un score de vraisemblance basé sur la position centrale dans l'image, on améliore significativement la mAP par rapport à une distribution uniforme. Cette expérimentation (initialement publiée dans [162]) a été réalisée sur la base Oxford Buildings. Cette base se prête particulièrement à l'évaluation de cette méthode puisque les bâtiments sont généralement centrés dans les photographies. Le nombre de mots $|V|$ est le nombre de mots visuels géométriquement cohérents extraits par l'algorithme RANSAS. La mAP est calculée en effectuant les requêtes associées à la vérité terrain d'Oxford Buildings, en utilisant une représentation des images par sacs de mots visuels, dont les mots visuels sont les mots visuels géométriquement cohérents de RANSAS. Le calcul de distances entre images se fait ensuite par un simple produit scalaire.

5.1.2 Scores de vraisemblance basés sur la détection de visages

Dans cette méthode, un détecteur de visages (celui d'OpenCV [1] dans le cadre de nos expérimentations) est passé sur les images et nous affectons une plus forte probabilité aux zones détectées comme des visages. Bien évidemment, n'importe quel type de détecteurs aurait pu être utilisé.

La position et l'échelle des visages permettent de calculer la probabilité :

$$z_0^F(\boldsymbol{x}_i) = \sum_{l=0}^{N^f-1} \mathcal{K}_{\sigma_l^f}(\mathbf{p}_{ij}, \mathbf{p}_l^f)$$

où $\mathcal{K}$ est le noyau RBF, N^f est le nombre de visages détectés dans l'image I_j, $\mathbf{P}_l^f$ et σ_l^f sont respectivement le centre et l'échelle du $l^{\text{ème}}$ visage dans l'image courante I_j.

La figure 5.3 page 74 montre un résultat de ce type de distribution dans une image.

1. http ://opencv.org

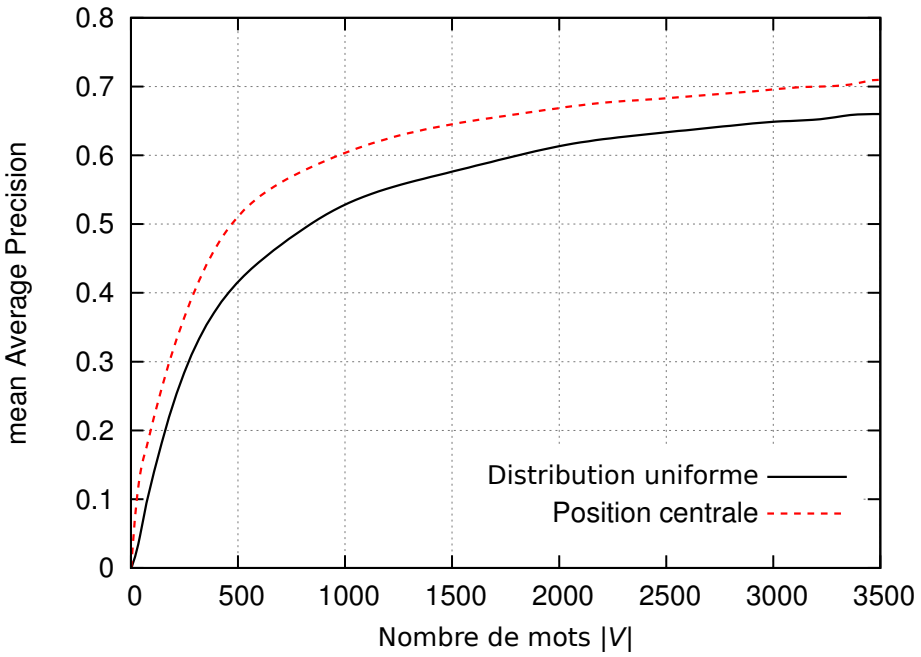


FIGURE 5.2 – Apport du score de vraisemblance « position centrale » par rapport à l'uniforme

Pour observer l'effet du score de vraisemblance basé sur la détection de visages, nous avons à trois reprises généré 600 mots visuels géométriquement cohérents sur la base BelgaLogos. Seuls les mots composés d'un minimum de cinq instances d'objets ont été pris en compte. Nous avons ensuite compté le nombre moyen de détections dans chaque catégorie (divers, logos, texture ou visage). Le tableau 5.1, issu de notre première publication [162], montre qu'en utilisant une distribution uniforme, on détecte peu de visages, et principalement des textures, tandis qu'avec une distribution basée sur la détection de visages, on augmente nettement le nombre de visages obtenus.

	Distribution uniforme	**Distribution « visage »**
Divers	31	20
Logos	46	27
Textures	**72**	23
Visages	12	**47**

TABLE 5.1 – Nombre de détections par catégorie

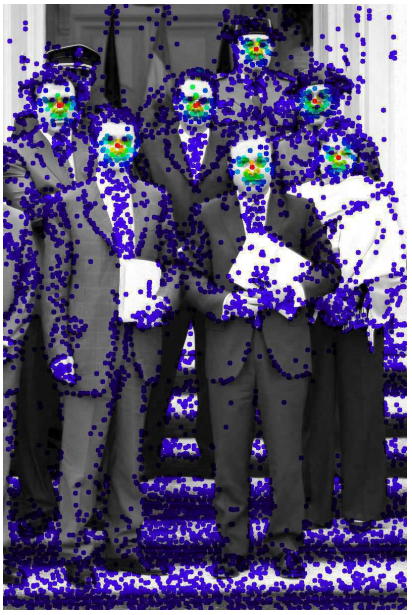


FIGURE 5.3 – Score de vraisemblance basé sur la détection de visages. Plus la couleur est chaude et plus on est proche d'un visage détecté.

5.2 Calcul de scores de vraisemblance généralistes

5.2.1 Calcul de scores de vraisemblance par des heuristiques simples

Dans cette section sont présentés deux exemples de scores de vraisemblance généralistes. Le premier est basé sur la densité spatiale des points d'intérêts, et le deuxième, sur la variabilité dans l'espace des descripteurs.

5.2.1.1 Scores de vraisemblance basés sur la densité spatiale des points d'intérêts

Le principe de cette mesure de saillance est de mettre l'accent sur les centres des régions denses (en termes de points d'intérêts), puisqu'on peut considérer qu'elles sont plus sujettes à contenir des objets d'intérêts car plus informatives. Concrètement, nous mesurons pour chaque point d'intérêt $\boldsymbol{x}_i$ la densité spatiale grâce ses points voisins, à l'aide d'une fenêtre de Parzen :

$$z_0^D(\boldsymbol{x}_i) = \sum_{l=0}^{|\boldsymbol{X}_q|-1} \mathcal{K}_\sigma(\boldsymbol{p}_{ij}, \boldsymbol{p}_l)$$

où $\mathcal{K}_\sigma$ est typiquement un noyau RBF, et $\boldsymbol{X}_q$ l'ensemble des points d'intérêts appartenant à l'image I_j. Les expériences présentées dans la section 5.3.3 page 80 utilisent des valeurs différentes pour σ selon qu'il s'agit des coordonnées horizontales ou verticales ($\sigma_\chi = \frac{W_i}{3}$ et $\sigma_\psi = \frac{H_i}{3}$). Un exemple du résultat de cette méthode est donné dans la figure 5.4 page ci-contre, et des expérimentations quantitatives sont menées dans la section 5.3.3 page 80.

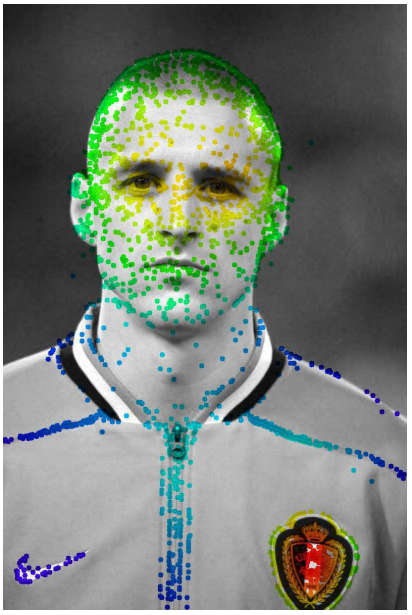


FIGURE 5.4 – Densité spatiale des points d'intérêts. Plus la couleur est chaude et plus la densité est forte.

5.2.1.2 Scores de vraisemblance basés sur la variabilité dans l'espace des descripteurs

L'un des problèmes fréquemment rencontrés dans les techniques d'appariement est la dégradation des performances en présence de structures (motifs) répétées (comme certaines textures). Ce problème vient de la forte probabilité d'obtenir des appariements multiples pour chacun des descripteurs [131, 166, 139]. Pour diminuer la probabilité de sélectionner de telles régions lors de l'étape d'échantillonnage, nous proposons donc une mesure calculant la variabilité des descripteurs, dans l'espace des descripteurs. Les descripteurs considérés pour la mesure de saillance de $\boldsymbol{x}_i$ appartiennent à la région candidate R_i, rectangle de dimension proportionnelle aux dimensions de l'image et centré autour de $\boldsymbol{x}_i$. La variabilité est calculée comme étant la moyenne des variances sur chaque dimension de l'espace des descripteurs, dans la région spatiale considérée :

$$z_0^{\delta}(\boldsymbol{x}_i) = \frac{1}{d}\sum_{l=0}^{d-1} \mathsf{Var}_{R_i}(\boldsymbol{x}\,[l])$$

où d est la dimension des descripteurs.

Un exemple de cette mesure de saillance est illustré par la figure 5.5 page suivante et des expérimentations quantitatives sont menées dans la section 5.3.3 page 80.

5.2.2 Calcul du score de vraisemblance par comptage des appariements

Plutôt que d'utiliser des mesures de saillance calculées au niveau image, comme vu précédemment, il peut être beaucoup plus efficace d'employer des méthodes tenant compte de l'ensemble de la base d'images. Le coût de calcul est bien entendu plus important qu'en travaillant à l'échelle d'une seule image,

FIGURE 5.5 – Variabilité dans l'espace des descripteurs. Plus la couleur est chaude et plus la variabilité est importante.

mais ce coût est compensé par la phase de découverte (recherche précise) qui est fortement accélérée (on a besoin de moins d'itérations).

Le calcul des scores individuels $z_0(\boldsymbol{x}_i)$ qui après normalisation deviennent la fonction de probabilité de masse p_0, peut donc être effectué à partir d'une méthode d'appariement de descripteurs. La méthode la plus couramment rencontrée dans la littérature utilise le principe des mots visuels (chaque mot visuel est un *cluster* de descripteurs locaux). Dans cette méthode, les descripteurs appartenant à un mot visuel sont appariés avec tous les autres descripteurs dans ce même mot visuel.

Soit $\boldsymbol{w}$ le mot visuel à qui on a affecté le descripteur $\boldsymbol{x}_i$, et $|\boldsymbol{w}|$ le nombre total de descripteurs affectés à $\boldsymbol{w}$. On calcule le score individuel du descripteur $\boldsymbol{x}_i$ par :

$$z_0^w(\boldsymbol{x}_i) = |\boldsymbol{w}|$$

La section 5.3.3 page 80 présente une évaluation quantitative de cette approche.

5.2.3 Calcul du score de vraisemblance par Geometric min-Hashing

L'algorithme Geometric min-Hashing [160] (pour la découverte de petits objets) est composé de différentes étapes :

1. Création d'un vocabulaire (*clustering K-Means*) à partir des descripteurs ;
2. Filtrage des descripteurs : on conserve les descripteurs dont les mots visuels sont uniques dans leur image, et ayant des voisins satisfaisant certaines contraintes géométriques ;
3. Génération de multiples *sketches* (s-tuples de mots visuels) ;

4. Co-segmentation des régions contenant des *sketches* ayant subi des collisions ;

Afin de comparer la stratégie de GmH avec celle proposée précédemment, et notre propre méthode introduite au chapitre 6 page 81, nous verrons comment adapter Geometric min-Hashing pour générer des scores individuels de probabilité d'appartenance à un objet fréquent pour chaque descripteur. Nous reprendrons donc les trois premières étapes de l'algorithme original, puis nous proposerons une méthode pour générer les scores $z_0(\boldsymbol{x}_q)$.

Dans les expérimentations présentées dans la section 5.3, le vocabulaire visuel a été créé par un *clustering K-Means*. Afin d'accélérer cette étape, l'algorithme AKM de Philbin et. al [94] a été employé. L'étape de filtrage des descripteurs sur leur unicité dans l'image et sur leurs voisins a été conservée telle quelle. Enfin, la troisième et dernière des étapes conservées, la génération des *sketches*, a simplement été réordonnée. Dans l'algorithme original, les auteurs généraient un *sketch* par image, et recommençaient cette étape autant de fois que nécessaire. Dans l'adaptation proposée, nous générons un nombre de *sketches* donné par image, et nous nous arrêtons lorsque toutes les images ont été parcourues. Ceci permet le calcul du score de vraisemblance, sans toutefois modifier les performances de GmH (à nombre de *sketches* équivalent).

Les scores individuels $z_0(\boldsymbol{x}_i)$ sont générés comme étant le nombre de *sketches* collisionnant avec le ou les *sketches* construits à partir de $\boldsymbol{x}_i$.

5.3 Expérimentations

5.3.1 Protocole d'évaluation

Dans le but d'évaluer la qualité des scores $z_0(\boldsymbol{x}_i)$, c'est-à-dire leur adéquation avec la vérité terrain, on définit deux mesures : la précision, et le rappel. On rappelle que les scores parfaits sont ceux qui après normalisation, donnent la fonction de probabilité de masse suivante :

$$p_0(\boldsymbol{x}_i) = 0, \forall \boldsymbol{x}_i \notin \{\boldsymbol{X}_s^m\}$$
$$p_0(\boldsymbol{x}_i) = \frac{1}{|\{\boldsymbol{X}_s^m\}|}, \forall \boldsymbol{x}_i \in \{\boldsymbol{X}_s^m\}$$

avec $\{\boldsymbol{X}_s^m\}$ l'ensemble des descripteurs appartenant à des instances de la vérité terrain.

Le rappel, pour un seuil η_0 donné, est le pourcentage d'instances dans la vérité terrain ayant au moins un descripteur local $\boldsymbol{x}$ avec une probabilité telle que $p_0(\boldsymbol{x}) > \eta_0$.

La précision est le pourcentage de descripteurs ayant une probabilité supérieure à η_0, qui appartiennent à une instance de la vérité terrain.

Les courbes de précision/rappel sont ensuite construites en faisant varier η_0.

Ces deux mesures d'évaluation sont totalement indépendantes de l'étape de recherche précise, et permettent ainsi de comparer directement la qualité des différentes méthodes de calcul des scores de vraisemblance. Contrairement aux mesures de précision/rappel habituellement utilisées, les mesures proposées ici opèrent donc à des niveaux différents : le rappel opère au niveau instance, et la précision au niveau descripteur.

Les expérimentations suivantes ont été effectuées sur la base FlickrBelgaLogos [2], dont le détail est donné dans la section 7.1 page 95. Les raisons pour lesquelles cette évaluation utilise cette base sont également expliquées dans la même section 7.1 page 95.

5.3.2 Évaluation de Geometric min-Hashing comme score de vraisemblance

Cette section a pour objectif d'observer l'influence des deux principaux paramètres de l'algorithme Geometric min-Hashing : le nombre de *sketches* et la taille du vocabulaire.

Il est à noter que les courbes de précision/rappel s'arrêtent prématurément. En effet, de très nombreux descripteurs étant ignorés lors du processus de génération des *sketches*, la plupart des descripteurs se retrouvent avec un score nul. Ceci est observé même avec un très grand nombre de *sketches* (100K), comparé au nombre utilisé dans le papier d'origine. Il serait donc trompeur de prolonger les courbes jusqu'au point de rappel parfait et de précision égale à celle obtenue lorsque le seuil η_0 est inférieur à tous les scores (cette précision est égale au nombre de descripteurs contenus dans des instances de la vérité terrain, divisé par le nombre total de descripteurs de la base, soit environ 0,00355).

5.3.2.1 Influence du nombre de *sketches*

La figure 5.6 page suivante montre l'influence du nombre k de *sketches* sur la précision et le rappel. L'article original [160] prétend que 60 *sketches* sont suffisants pour effectuer un *clustering* de qualité sur la base Oxford Buildings. Cependant, sur la base BelgaLogos, on voit qu'il faut au minimum 100 000 *sketches* pour atteindre des taux de précision/rappel de l'ordre de 20%. Le mode de génération des *sketches* (par min-Hashing) montre donc clairement ses limites puisqu'avec un tel nombre de *sketches*, on peut considérer que l'on a généré une bonne partie des *sketches* constructibles. Un vocabulaire de 1 million de mots a été utilisé.

5.3.2.2 Influence de la taille du vocabulaire

La figure 5.7 page ci-contre montre l'effet de la taille du vocabulaire utilisé. On voit nettement que plus la taille augmente et plus on gagne en précision/rappel, en convergeant très doucement. On peut supposer qu'en continuant d'augmenter la taille du vocabulaire, on améliorerait également la précision, mais que le rappel finirait inexorablement par chuter au fur et à mesure que la taille des *clusters* tendrait vers 1. En pratique, l'utilisation d'un vocabulaire aussi grand est problématique. Le temps de calcul nécessaire pour effectuer le *clustering* avec 8 millions de mots, ou plus, devient réellement prohibitif (voir tableau 4.1 page 66). De plus, le vocabulaire n'est pas réutilisable pour la phase de recherche précise. On peut également remarquer que nous ne traitons ici que 50 millions de descripteurs, quand on aimerait potentiellement pouvoir en traiter plus d'un milliard. En pratique, un vocabulaire d'un million de mots est un bon compromis, à la fois pour le temps de calcul, et aussi pour la ré-utilisabilité lors de la phase de recherche précise.

2. http ://www-sop.inria.fr/members/Alexis.Joly/BelgaLogos/FlickrBelgaLogos.html

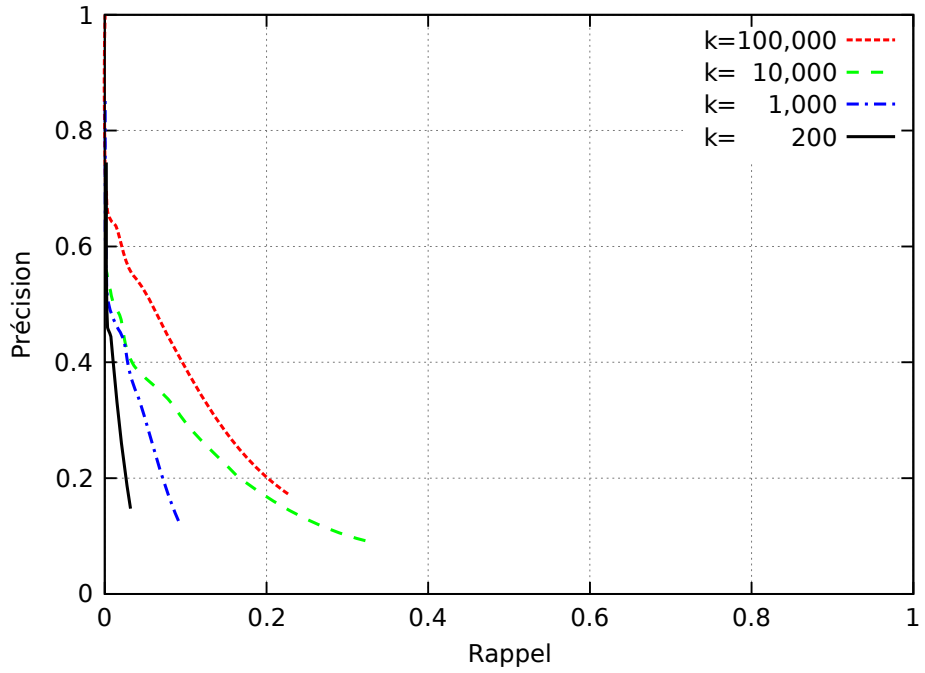


FIGURE 5.6 – Influence du nombre de *sketches* k sur la précision et le rappel. Un vocabulaire de 1 million de mots a été utilisé.

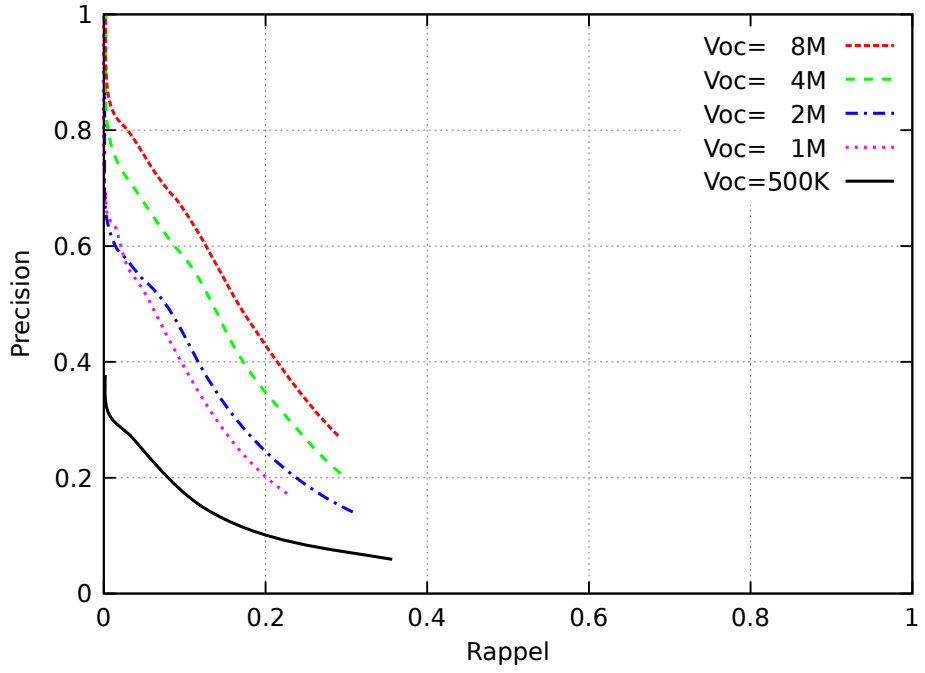


FIGURE 5.7 – Influence de la taille du vocabulaire (Voc) sur la précision et le rappel. 100 000 *sketches* ont été générés.

5.3.3 Comparaison des performances des scores de vraisemblance généralistes

Les courbes de la figure 5.8 montrent les performances des méthodes de calcul de scores de vraisemblance par des heuristiques simples (voir section 5.2.1 page 74), par comptage des appariements (voir section 5.2.2 page 75), ou par Geometric min-Hashing (voir section 5.2.3 page 76).

On constate que l'heuristique simple se focalisant sur les zones denses (en termes de points d'intérêt) obtient des performances similaires à celles de l'aléatoire, c'est-à-dire que peu importe le rappel, on obtient une précision d'environ 0,00355 correspondant au nombre de descripteurs contenus dans des instances de la vérité terrain, divisé par le nombre total de descripteurs de la base. Cette heuristique est donc totalement inutile pour cette base d'images.

L'heuristique se basant sur la variabilité des descripteurs obtient légèrement mieux, mais cette amélioration n'est pas suffisante pour justifier son calcul. La méthode de calcul de score par comptage des appariements améliore encore un peu les performances, en obtenant un compromis précision/rappel égal à 10%/10%. Cette méthode souffre clairement d'un problème de robustesse (le simple appariement des descripteurs ne permet pas d'obtenir des précisions élevées).

Geometric min-Hashing résout partiellement ce problème en rendant les appariements plus précis par l'utilisation de *sketches* (avec contraintes géométriques lors de la construction) au lieu d'appariements individuels.

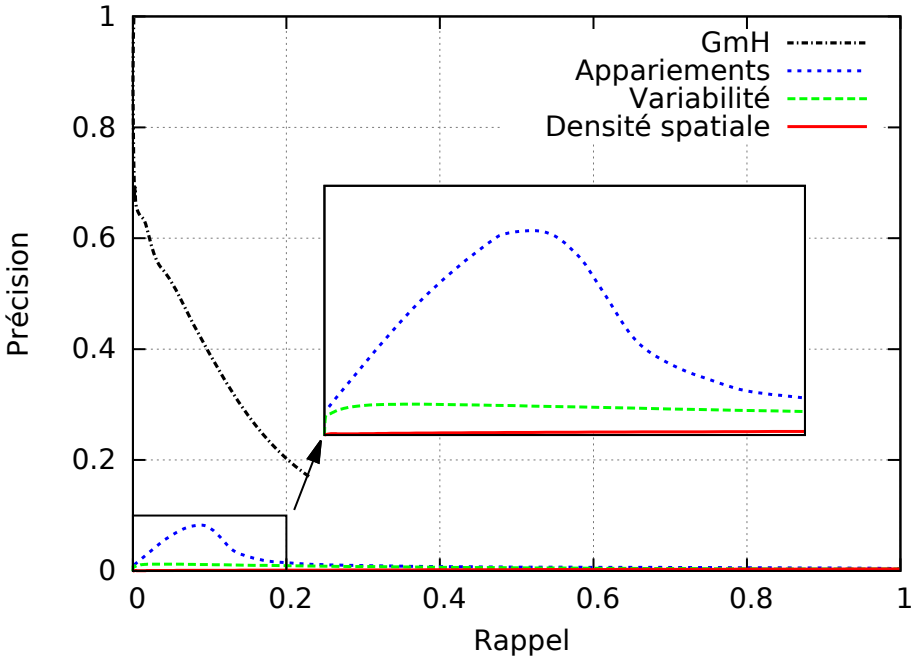


FIGURE 5.8 – Comparaison des performances des scores de vraisemblance générés par Geometric min-Hashing, par appariements de descripteurs (mots visuels), par variabilité des descripteurs, ou encore par densité spatiale des descripteurs. La courbe GmH a été obtenue avec un vocabulaire d'un million de mots visuels et 100 000 *sketches*.

Chapitre 6

Calcul des scores de vraisemblance par hachage visuel et ajout de contraintes géométriques faibles

Comme nous l'avons vu précédemment, notre objectif est de calculer une fonction de probabilité de masse p_0 sur l'ensemble des descripteurs de la base $\boldsymbol{X}$, de manière à ce que $p_0(\boldsymbol{x}_i)$ reflète au mieux la vraisemblance qu'un descripteur local $\boldsymbol{x}_i$ appartienne à un objet $\{c, f\}$-fréquent. En d'autres mots, nous cherchons à maximiser $c_0(O^m)$ pour tous les objets $\{c, f\}$-fréquents de la base (cf. section 4.3 page 62).

Plutôt que d'utiliser des mesures de saillance calculées au niveau image, nous verrons qu'il peut être beaucoup plus efficace d'employer des méthodes tenant compte de l'ensemble de la base d'images. Le coût de calcul est bien entendu plus important qu'en travaillant à l'échelle d'une seule image, mais ce coût est compensé par la phase de recherche précise dont le nombre d'itérations T est fortement réduit.

Le calcul des scores individuels $z_0(\boldsymbol{x}_i)$ repose principalement sur l'observation de la fréquence des collisions dans des tables de hachage, ce qui permet à l'algorithme présenté dans cette section de rester scalable et facilement distribuable si besoin.
Cet algorithme est composé des trois étapes suivantes :

1. Hachage visuel et filtrage des appariements candidats ;
2. Extraction et hachage des informations de géométrie faible ;
3. Calcul des scores individuels $z_0(\boldsymbol{x}_i)$ par comptage dans les tables de hachage de la géométrie faible ;

Intuitivement, le score $z_0(\boldsymbol{x}_i)$ calculé pour chaque $\boldsymbol{x}_i$ représente le nombre de correspondances visuelles qui sont à la fois dans le voisinage de p_{ij} et qui sont géométriquement cohérentes avec les appariements visuels directs de $\boldsymbol{x}_i$.

6.1 Hachage visuel et filtrage des appariements candidats

6.1.1 Construction de l'index visuel

Comme nous l'avons vu au chapitre 2 page 21, il y a deux stratégies principales pour la recherche par similarité de descripteurs locaux : un vocabulaire visuel associé à une liste inversée [82, 137], ou les méthodes de hachage [4, 90]. Ces deux types de méthodes poursuivent le même but : réduire le coût de calcul de la recherche, en partitionnant et en compressant les descripteurs.

La plupart des méthodes de recherche par similarité visuelle sont basées sur l'utilisation d'un vocabulaire visuel. Ce vocabulaire est généralement généré à l'aide d'un *clustering K-Means* [104, 134, 82, 137]. Comme nous l'avons vu dans la section 4.5 page 65, le principal inconvénient de ces méthodes est que la génération du vocabulaire est extrêmement coûteuse, et en particulier pour un processus de découverte. En effet, dans le cadre de la génération d'un index pour un moteur de recherche par similarité visuelle, cette étape n'est d'ordinaire pas considérée comme coûteuse car elle est vue comme une étape préliminaire effectuée une seule fois pour toute une base (*offline*). Le seul coût considéré est alors celui des multiples recherches *online*. Au contraire, dans un processus de découverte, le but est de faire le minimum possible de recherches dans la base. La génération de l'index devient alors un des coûts principaux.

Afin de diminuer ces coûts de calcul, nous avons donc choisi d'utiliser des méthodes de hachage pour compresser et indexer les descripteurs visuels. Les méthodes de hachage ont par le passé prouvé qu'elles pouvaient obtenir d'aussi bon résultats dans le domaine de la recherche d'objets visuels [4, 162].

De plus, comme précisé dans la section 4.5 page 65, le temps de calcul nécessaire au hachage (128 bits) d'une base de $N = 10^9$ SIFT est inférieur à 4 heures avec un serveur à 12 cœurs, tandis qu'il en faut le double pour effectuer une itération de *K-Means* (avec AKM). Cette indexation visuelle est donc particulièrement efficace et adaptée à un processus à large échelle.

Plutôt que d'utiliser des fonctions LSH classiques, la méthode employée ici est basée sur un algorithme plus efficace : *Random Maximum Margin Hashing* (RMMH, [90]), une famille de fonctions de hachage récemment introduite. La principale originalité de RMMH est d'apprendre des séparations aléatoires des données, quelle que soit la proximité des échantillons d'apprentissage (sans aucune supervision). Concrètement, la méthode fonctionne en apprenant un ensemble de classifieurs à partir d'une petite partie de la base. Pour chaque fonction de hachage, M points d'apprentissage sont sélectionnés aléatoirement dans $\boldsymbol{X}$ et sont ensuite étiquetés aléatoirement (la moitié avec -1 et le reste avec 1). Si on note x_j^+ les $\frac{M}{2}$ échantillons d'apprentissage positifs, et $\boldsymbol{x}_j^-$ les négatifs, chaque fonction de hachage est ensuite calculée par un classifieur binaire $h(\boldsymbol{x})$ tel que :

$$h(\boldsymbol{x}) = \underset{h_\theta}{\text{argmax}} \sum_{j=1}^{\frac{M}{2}} h_\theta(\boldsymbol{x}_j^+) - h_\theta(\boldsymbol{x}_j^-) \tag{6.1}$$

En utilisant un SVM linéaire comme classifieur binaire, on obtient :

$$h(\boldsymbol{x}) = sgn\left(\sum_{i=1}^{m} \alpha_i^* \boldsymbol{x}_i^* . \boldsymbol{x} + b_m\right) \tag{6.2}$$

où $\boldsymbol{x}_i^*$ sont les m vecteurs supports sélectionnés par le SVM ($\boldsymbol{x}_i^* \in \{\boldsymbol{x}_j^+, \boldsymbol{x}_j^-\}$) qui estime α_i^* et b_m.

Dans [90], les auteurs montrent que le paramètre M est en pratique assez stable sur différents types de descripteurs et différentes tailles de base. Par la suite, on notera $\boldsymbol{h}_l(\boldsymbol{x})$ la clé de hachage de longueur k produit par la concaténation de k fonctions de hachage binaire individuelles pour la $l^{\text{ème}}$ des L fonctions de hachage.

Une fois que l'index visuel a été créé, l'algorithme calcule individuellement les scores $z_0(\boldsymbol{x}_i)$ en traitant indépendamment toutes les images $I_j \in \boldsymbol{I}$ une par une (soit itérativement, soit en parallèle). Notons I_Q l'image traitée à la $Q^{\text{ème}}$ itération et $\boldsymbol{X}_Q$ l'ensemble des descripteurs locaux lui appartenant.

6.1.2 Filtrage visuel des appariements candidats

L'utilisation « naïve » de l'index visuel génère un très grand nombre de collisions entre les descripteurs, qui sont autant d'appariements candidats. Afin de limiter ce nombre, et donc de limiter le temps de calcul, nous appliquons trois stratégies de filtrage pour tous les descripteurs $\boldsymbol{x}_q \in \boldsymbol{X}_Q$:

- Filtrage sur la fréquence de collision intra-image ;
- Filtrage sur la fréquence de collision inter-tables ;
- Filtrage par les K plus proches voisins ;

6.1.2.1 Filtrage sur la fréquence de collision intra-image

Comme notifié dans [160], les descripteurs visuels qui sont uniques dans une image, sont bien plus propices à former des appariements que l'on peut considérer comme fiables. Cette observation permet de réduire une part importante des faux positifs produits par les textures et autres motifs répétés. Plutôt que de conserver seulement les descripteurs uniques, notre algorithme propose une approche plus souple qui limite la fréquence de collision intra-image à une valeur maximum ζ. De cette manière, on obtient un sous-ensemble $\boldsymbol{X}'_Q$ de descripteurs candidats à l'étape suivante, à partir de $\boldsymbol{X}_Q$:

$$\boldsymbol{X}'_Q = \{\boldsymbol{x}_q \in \boldsymbol{X}_Q \mid f_Q(\boldsymbol{x}_q) < \zeta\} \tag{6.3}$$

où la fréquence de collision intra-image f_Q est estimée par sa valeur moyenne à travers les L tables de hachage :

$$f_Q(\boldsymbol{x}_q) = \frac{1}{L}\sum_{l=1}^{L} \#\{\boldsymbol{x} \in \boldsymbol{X}_Q \mid \boldsymbol{h}_l(\boldsymbol{x}) = \boldsymbol{h}_l(\boldsymbol{x}_q)\} \tag{6.4}$$

Empiriquement, ζ a été fixé à une valeur par défaut à $\zeta = 3$.

6.1.2.2 Filtrage sur la fréquence de collision inter-tables

Comme suggéré par la méthode Frequency-based LSH [100], l'algorithme filtre ensuite l'ensemble des descripteurs de la base ayant des collisions avec $\boldsymbol{X}'_Q$. Ce filtrage s'effectue sur la fréquence de collision

à travers les L tables de hachage de l'index visuel. L'ensemble des appariements $\boldsymbol{Y}_Q$ résultant de ce filtrage peut-être exprimé de la façon suivante :

$$\boldsymbol{Y}_Q = \left\{(\boldsymbol{x}_q, \boldsymbol{x}_i) \in (\boldsymbol{X}'_Q, \boldsymbol{X}) \mid f_L(\boldsymbol{x}_q, \boldsymbol{x}_i) > \tau\right\} \tag{6.5}$$

où $f_L(\boldsymbol{x}_q, \boldsymbol{x}_i)$ est le nombre de fois où $\boldsymbol{x}_q$ et $\boldsymbol{x}_i$ ont la même clé de hachage à travers les L tables de hachage visuel.

Comme suggéré dans [100], le seuil τ peut être calculé numériquement comme une fonction de L et ϕ, où ϕ est une borne inférieure sur la fréquence de collision dans une seule fonction de hachage. Ce seuillage basé sur la fréquence approxime une requête par rayon dans l'espace original des descripteurs lorsque LSH est utilisé. Avec RMMH, il s'agit plus d'une approximation d'un seuil basé sur la densité.

Nous verrons dans la section 6.5.1.1 page 88 comment nous paramétrons ce filtrage, grâce à l'équation 6.6 donnée par Frequency-based LSH [100] :

$$\phi^* = \sum_{l=\tau}^{L} C_L^l \phi^l (1-\phi)^{L-l} \tag{6.6}$$

En estimant empiriquement une valeur optimale $\hat{\tau}$ pour un nombre de tables L, on peut alors obtenir une valeur ϕ pour une probabilité ϕ^* fixée, grâce à la méthode de Newton de résolution d'équations. ϕ^* représente la probabilité d'obtenir au moins τ collisions à travers les L tables.

6.1.2.3 Filtrage par les K plus proches voisins

Afin de réduire l'impact des descripteurs visuels ambigus qui sont très fréquents dans la base (comme dans les zones de texte par exemple), nous réduisons encore le nombre d'appariements par un filtrage par K plus proches voisins, calculé avec une distance de Hamming sur les L clés de hachage concaténées. De cette façon, l'ensemble final des appariements pour l'image I_Q est donc donné par :

$$\boldsymbol{Z}_Q = \{(\boldsymbol{x}_q, \boldsymbol{x}_m) \in \boldsymbol{Y}_Q \mid d_H(\boldsymbol{h}(\boldsymbol{x}_q), \boldsymbol{h}(\boldsymbol{x}_m)) \leq r_H(\boldsymbol{x}_q, K)\} \tag{6.7}$$

où $r_H(\boldsymbol{x}_q, K)$ est la distance de Hamming de la $K^{\text{ème}}$ plus proche clé de hachage de $\boldsymbol{h}(\boldsymbol{x}_q)$. K a été fixé empiriquement à 200 dans les expérimentations suivantes, c'est-à-dire du même ordre de grandeur que lors de l'étape de recherche précise, où K est généralement fixé à 300.

6.2 Extraction et hachage d'attributs de géométrie faible

Une fois que l'on a obtenu un ensemble $\boldsymbol{Z}_Q$ d'appariements de descripteurs visuels pour la $Q^{\text{ème}}$ image I_Q, les scores individuels $z_o(\boldsymbol{x}_q)$ sont calculés pour chaque descripteur $\boldsymbol{x}_q$ de l'image, au moyen d'un hachage des attributs de géométrie faible provenant des descripteurs.

Cette approche est inspirée de celle de Jégou et al. [109] qui estiment la cohérence géométrique des appariements, en se basant sur les métadonnées associées à chaque descripteur SIFT (orientation et échelle).

La méthode proposée ici diffère en trois points principaux :

– Premièrement, plutôt que de voter dans des histogrammes mono-dimensionnels (un par attribut), nous votons dans un histogramme multidimensionnel creux. On augmente alors la distinctivité ;
– Deuxièmement, en plus de l'échelle et de l'orientation, nous ajoutons les coordonnées spatiales des descripteurs dans l'image. Ceci permet d'inclure une contrainte de voisinage spatial, et favorise les petits objets au détriment des plus gros (copies, arrière-plans, …) ;
– Dernièrement, le vote est effectué à l'échelle de la base entière, contrairement à la version originale [109] qui s'appliquait uniquement aux paires d'images ;

6.2.1 Création de vecteurs de géométrie faible

Plus concrètement, pour chaque appariement visuel $(\boldsymbol{x}_q, \boldsymbol{x}_m) \in \boldsymbol{Z}_Q$, nous créons le vecteur de géométrie faible suivant :

$$\boldsymbol{\Delta}_{q,m} = (\Delta\theta_{q,m}, \Delta\sigma_{q,m}, \chi_q, \psi_q) \tag{6.8}$$

où (χ_q, ψ_q) représente la position de $\boldsymbol{x}_q$ dans l'image I_Q. $\Delta\sigma_{q,m} = \sigma_m - \sigma_q$ est la différence des échelles caractéristiques des points d'intérêts $\boldsymbol{x}_q$ et $\boldsymbol{x}_m$. $\Delta\theta_{q,m}$ est le facteur de rotation entre les angles caractéristiques des points d'intérêts $\boldsymbol{x}_q$ et $\boldsymbol{x}_m$, obtenu par :

$$\Delta\theta_{q,m} = \frac{\arctan(\sin(\theta_m - \theta_q), \cos(\theta_m - \theta_q))}{\pi} \tag{6.9}$$

Afin de créer un espace de vote multidimensionnel à partir de l'espace initial des attributs de géométrie faible $\boldsymbol{\Delta}_{q,m}$, nous proposons d'employer une famille de fonctions LSH adaptée de celle des fonctions LSH Euclidiennes classiques [63].

Notons que l'espace des vecteurs $\boldsymbol{\Delta}_{q,m}$ est tout d'abord normalisé par composante de manière à ce que chaque attribut varie dans l'intervalle $[-1, 1]$.

6.2.2 Hachage des attributs de géométrie faible

Les vecteurs ainsi normalisés sont ensuite hachés avec L' fonctions LSH distinctes, chacune composée de k' projections aléatoires de la forme :

$$h'(\boldsymbol{\Delta}) = \left\lfloor \frac{\boldsymbol{a}.\boldsymbol{\Delta} + b}{w} \right\rfloor \tag{6.10}$$

avec $\boldsymbol{a}$ étant un vecteur de nombre aléatoires gaussiens, et b étant un nombre aléatoire sélectionné dans $[-1; 1]$.

La légère modification introduite dans cette famille de fonctions est d'avoir une valeur adaptative w pour chaque projection aléatoire. Un hachage « L_2-sensible » n'est en effet dans ce cas pas approprié pour créer notre espace de vote. On aimerait plutôt garantir une dynamique uniforme sur chacun des axes de projection. On introduit donc les contraintes suivantes :

$$\min_{\Delta\in[-1;1]^d} \frac{\boldsymbol{a}.\boldsymbol{\Delta} + b}{w} = -2^\gamma, \qquad \max_{\Delta\in[-1;1]^d} \frac{\boldsymbol{a}.\boldsymbol{\Delta} + b}{w} = +2^\gamma \tag{6.11}$$

Ce qui nous donne $w = \frac{\|a\|_1}{2^\gamma}$ où γ est le nombre de bits utilisés pour quantifier chaque axe de projection. On obtient donc finalement :

$$h'(\boldsymbol{\Delta}) = \left\lfloor 2^\gamma \frac{\boldsymbol{a}.\boldsymbol{\Delta} + b}{\|a\|_1} \right\rfloor \tag{6.12}$$

Et on note $\boldsymbol{h}'_l(\boldsymbol{\Delta})$ la clé de hachage de longueur k' produite par la concaténation de k' fonction de hachage individuelle pour la $l^{\text{ème}}$ fonction de hachage.

6.3 Calcul des scores de vraisemblance par accumulation des collisions

6.3.1 Vote dans les accumulateurs

Le processus de vote fonctionne ensuite simplement en gérant un conteneur de type associatif H_l^Q pour chacune des L' fonctions de hachage (on peut typiquement utiliser en C++ un container std::map). L'intérêt d'utiliser une telle structure est qu'il permet d'économiser énormément d'espace mémoire, étant donné que l'accumulateur H_l^Q est extrêmement creux.

La « valeur clé » utilisée pour insérer les appariements visuels (incrémenter) dans l'accumulateur est définie par la paire :

$$\boldsymbol{h}''_l(\boldsymbol{\Delta}_{q,m}) = \left(I_{q,m}, \boldsymbol{h}'_l(\boldsymbol{\Delta}_{q,m})\right) \tag{6.13}$$

où $\boldsymbol{h}'_l(\boldsymbol{\Delta}_{q,m})$ est la clé de hachage du vecteur de géométrie faible (c'est-à-dire l'estimation de la transformation géométrique), et $I_{q,m}$ est l'identifiant de l'image contenant les descripteurs $\boldsymbol{x}_m$ appariés à $\boldsymbol{x}_q$.

L'idée est que tous les appariements $\boldsymbol{\Delta}_{q,m}$ voisins dans l'image I_Q et cohérents géométriquement se retrouvent avec la même clé de hachage (même partie de l'espace haché).

Les valeurs associées à ces clés dans l'accumulateur représentent le nombre d'occurrences dans la clé de hachage identifiée par $\boldsymbol{h}''_l(\boldsymbol{\Delta}_{q,m})$.

Tous les L' accumulateurs H_l^Q sont vides au commencement de la $Q^{\text{ème}}$ itération de l'algorithme.

Les accumulateurs sont incrémentés en parallèle au fur et à mesure que l'on parcourt les appariements $(\boldsymbol{x}_q, \boldsymbol{x}_m) \in \boldsymbol{Z}_Q$ un par un, de la manière suivante :

$$H_l^Q(\boldsymbol{h}''_l(\boldsymbol{\Delta}_{q,m})) \mathrel{+}= f_L(\boldsymbol{x}_q, \boldsymbol{x}_m) \tag{6.14}$$

où on rappelle que $f_L(\boldsymbol{x}_q, \boldsymbol{x}_m)$ est le nombre de collisions dans l'index visuel (cf. équation 6.5 page 84). L'idée d'utiliser un vote pondéré par f_L plutôt qu'une simple incrémentation unitaire est de favoriser les correspondances visuelles les plus sûres, c'est-à-dire celles dont la distance visuelle est la plus petite.

6.3.2 Calcul des scores individuels

Après que tous les appariements visuels $(\boldsymbol{x}_q, \boldsymbol{x}_m) \in \boldsymbol{Z}_Q$ aient été insérés dans les accumulateurs, les scores de vraisemblance $z_0(\boldsymbol{x}_q)$ sont calculés en comptant le nombre de collisions à travers chacun des L' accumulateurs :

$$z_0(\boldsymbol{x}_q) = \sum_{l=1}^{L'} \sum_{(\boldsymbol{x}_q, \boldsymbol{x}_m) \in \boldsymbol{Z}_Q} H_l^Q(\boldsymbol{h}_l''(\boldsymbol{\Delta}_{q,m})) \tag{6.15}$$

De cette façon, $z_0(\boldsymbol{x}_q)$ mesure le nombre d'appariements visuels dans $\boldsymbol{Z}_Q$ qui sont à la fois dans le voisinage de $\boldsymbol{x}_q$ et géométriquement cohérents avec les correspondances directes de $\boldsymbol{x}_q$ (avec les $\boldsymbol{\Delta}_{q,m}$).

Plus la valeur $z_0(\boldsymbol{x}_q)$ est forte, plus la probabilité que $\boldsymbol{x}_q$ appartienne à un objet fréquent est élevée.

6.4 Solution alternative : ajout de contraintes géométriquement faibles à GmH

Notre algorithme de vote sur les attributs de géométrie faible peut se généraliser à d'autres types d'appariements visuels en entrée. Nous proposons ainsi une variante en aval de l'utilisation d'un algorithme de type GmH.

Dans ce cas, on obtient non plus des appariements individuels entre un vecteur $\boldsymbol{x}_q$ et un vecteur $\boldsymbol{x}_m$, mais des appariements entre *sketches* (groupes de s appariements individuels), c'est-à-dire des appariements entre des *sketches* requêtes $\{\boldsymbol{x}_{q_1}, \boldsymbol{x}_{q_2}, ..., \boldsymbol{x}_{q_s}\}$ et des *sketches* candidats $\{\boldsymbol{x}_{m_1}, \boldsymbol{x}_{m_2}, ..., \boldsymbol{x}_{m_s}\}$.

Ces appariements entre *sketches* sont obtenus de la manière décrite dans le papier original de GmH [160] ainsi que dans la section 3.4 page 50. Dans chaque image, nous créons k *sketches*, composés de $s = 2$ min-Hash, dont le deuxième est choisi parmi les voisins du premier respectant certaines contraintes géométriques.

Afin d'ajouter nos contraintes géométriquement faibles à GmH, notre proposition est de créer pour chaque paire de *sketches* appariés $\{\boldsymbol{x}_{q_1}, \boldsymbol{x}_{q_2}, ..., \boldsymbol{x}_{q_s}\}$ et $\{\boldsymbol{x}_{m_1}, \boldsymbol{x}_{m_2}, ..., \boldsymbol{x}_{m_s}\}$, s vecteurs de géométrie faible $\{\boldsymbol{\Delta}_{q_1,m_1}, \boldsymbol{\Delta}_{q_2,m_2}, ..., \boldsymbol{\Delta}_{q_s,m_s}\}$. Ces s vecteurs sont ensuite utilisés individuellement pour voter dans les accumulateurs, c'est-à-dire sans aucune autre modification de notre algorithme de base.

Cette solution sera évaluée par la suite en comparaison de notre algorithme original.

6.5 Expérimentations

Le protocole d'évaluation des algorithmes présentés dans ce chapitre est le même que celui décrit au chapitre précédent dans la section 5.3.1 page 77. Nous utilisons donc la précision et le rappel calculés sur la base FlickrBelgaLogos, présentée dans la section 7.1 page 95.

6.5.1 Étude paramétrique

6.5.1.1 Paramétrage du hachage des descripteurs visuels

Les deux principaux paramètres du hachage des descripteurs visuels sont le nombre de tables de hachage (RMMH) L, et le seuil sur la probabilité de collision ϕ. Pour $L = 64$, le plus grand nombre de tables testé, on estime empiriquement un seuil optimal $\hat{\tau} = 7$. Ainsi, en fixant une probabilité raisonnable $\phi^* = 0.5$, et par la méthode de Newton appliquée à l'équation 6.6 page 84, on obtient $\phi \approx 0.10$. Ceci permet maintenant de déduire le paramètre $\hat{\tau}$ estimé en fonction du nombre de tables de hachage L. Le tableau 6.1 présente les résultats de ce calcul. En pratique, pour des raisons de temps de calcul, nous avons préféré utiliser des valeurs de τ légèrement plus élevées (voir tableau 6.1) qui permettent un filtrage plus fort, sans diminuer significativement les performances.

L	1	2	4	8	16	32	64
$\hat{\tau}$	1	1	1	1	2	4	7
τ	1	2	2	2	3	4	7

TABLE 6.1 – Réglage du paramètre $\hat{\tau}$ optimum théorique en fonction de L, et τ utilisé en pratique.

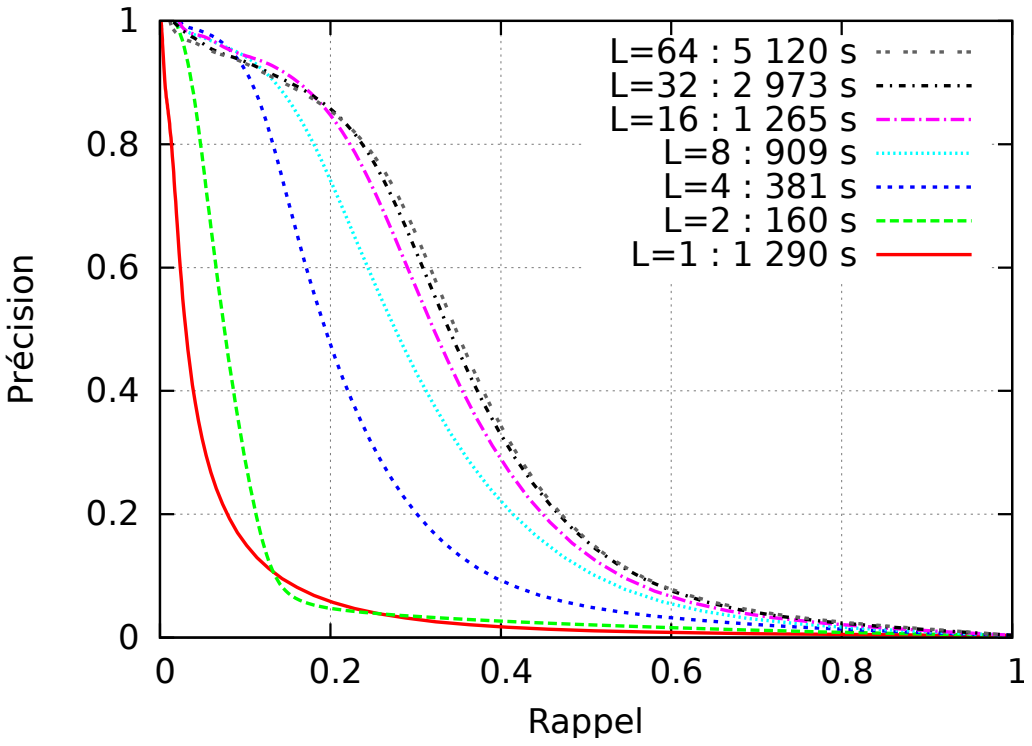


FIGURE 6.1 – Réglage du nombre de tables de hachage L

Les résultats présentés dans la figure 6.1 montrent qu'utiliser plus de 16 tables de hachage visuel n'est

pas très rentable, étant donné que la progression en termes de précision/rappel devient très faible, alors que le temps de calcul augmente linéairement.

Il faut noter que le temps nécessaire pour calculer les scores de vraisemblance avec une seule table de hachage est bien plus important qu'avec deux tables. Ceci s'explique par l'impossibilité de filtrer sur la fréquence de collision inter-tables de hachage f_L lorsque $L = 1$.

6.5.1.2 Longueur des clés de hachage des attributs de géométrie faible

Dans cette expérimentation, nous étudions l'effet du nombre de fonctions de hachage k' utilisées pour calculer les clés de hachage des attributs de géométrie faible, ainsi que le nombre de bits γ utilisés pour la quantification scalaire de chaque projection.

Pour améliorer la lisibilité du graphique de la figure 6.2, les courbes ont été groupées manuellement en quatre groupes (identifiés par quatre couleurs), selon leurs performances. Les meilleurs compromis sont obtenus avec $\gamma \in [3, 5]$ et $k' \in [3, 10]$. On remarque que l'utilisation d'une forte valeur pour γ est plutôt défavorable, étant donné que cela diminue la probabilité d'obtenir un nombre de collisions significatif. Au contraire, une faible valeur de γ peut être contrebalancée par un plus grand nombre de fonctions de hachage k', afin d'augmenter la distinctivité de l'histogramme. Dans les expérimentations suivantes, nous utiliserons les paramètres de la courbe noire ($k' = 6$ et $\gamma = 4$), pour un total de 24 bits par clé de hachage.

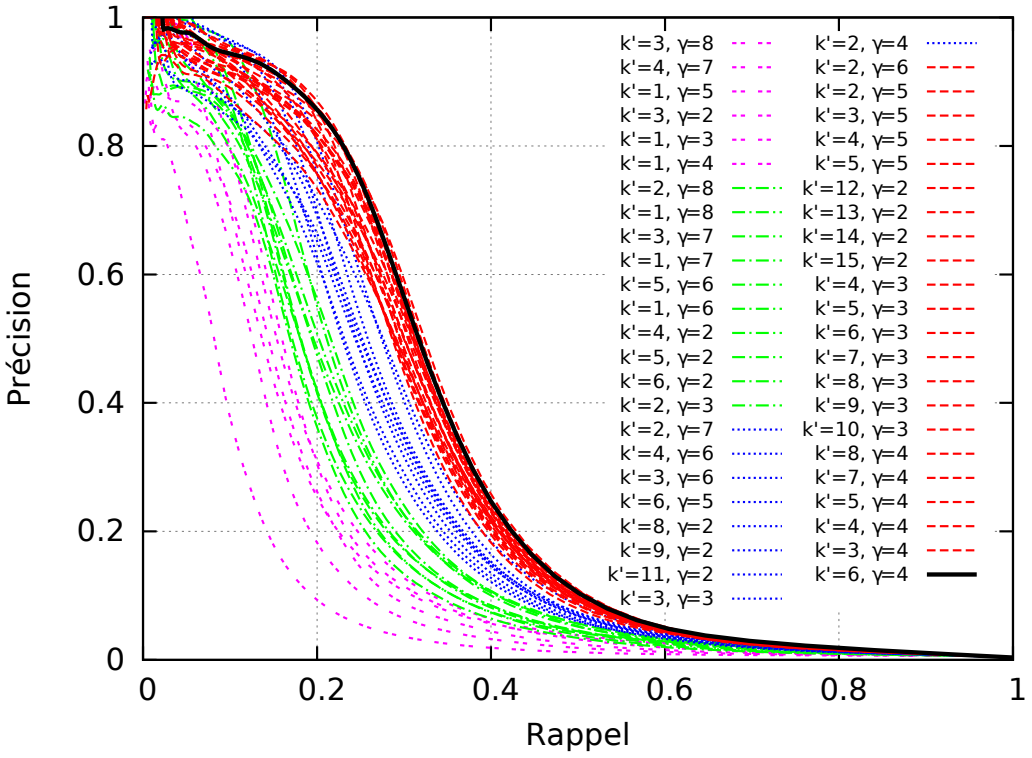


FIGURE 6.2 – Effets de la longueur des clés de hachage de la géométrie faible

6.5.1.3 Nombre d'accumulateurs

L'évaluation présentée dans la figure 6.3 compare la qualité obtenue avec différentes valeurs de L', c'est-à-dire le nombre d'accumulateurs (histogrammes). On peut également observer les performances obtenues avec une simple grille multidimensionnelle (à la place du hachage par projections aléatoires), utilisant le même nombre de bits. On remarque que les histogrammes basés LSH fonctionnent bien mieux qu'avec la grille multidimensionnelle.

En regardant la valeur de L', on voit nettement qu'il n'est pas rentable d'utiliser plus d'un accumulateur, contrairement au cas des espaces à très grandes dimensions.

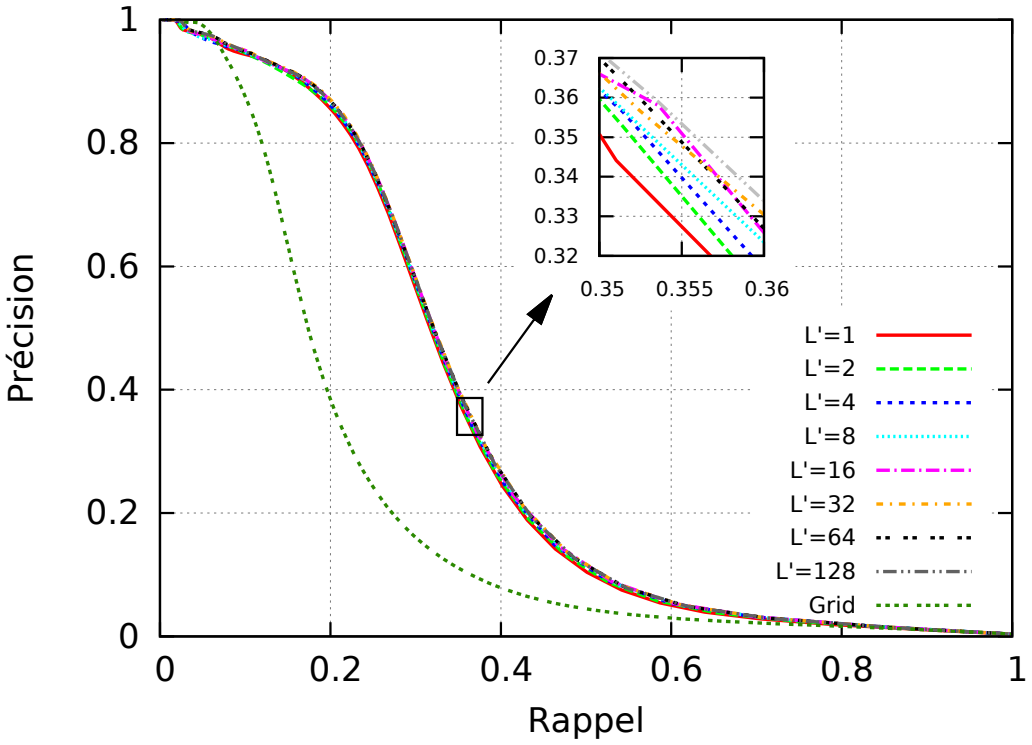


FIGURE 6.3 – Influence du nombre d'accumulateurs

6.5.2 Apport des attributs de géométrie faible

Cette évaluation vise à mesurer la contribution des différents attributs du vecteur de géométrie faible $\mathbf{\Delta}$. On peut voir dans la figure 6.4 page ci-contre que l'utilisation de la géométrie faible améliore très significativement la précision et le rappel. La courbe $0, 0, 0, 0$ mesure les performances en l'absence de géométrie faible, en utilisant simplement le hachage visuel et les trois règles de filtrage des appariements visuels. Les autres courbes ont été obtenues à partir d'un sous-ensemble d'attributs de géométrie faible (angle : θ, échelle : σ, position : χ et ψ). On observe que parmi les différents attributs, ce sont les deux attributs de position spatiale qui procurent le gain le plus significatif. On voit que tous les attributs apportent de l'information, et que cette information est complémentaire.

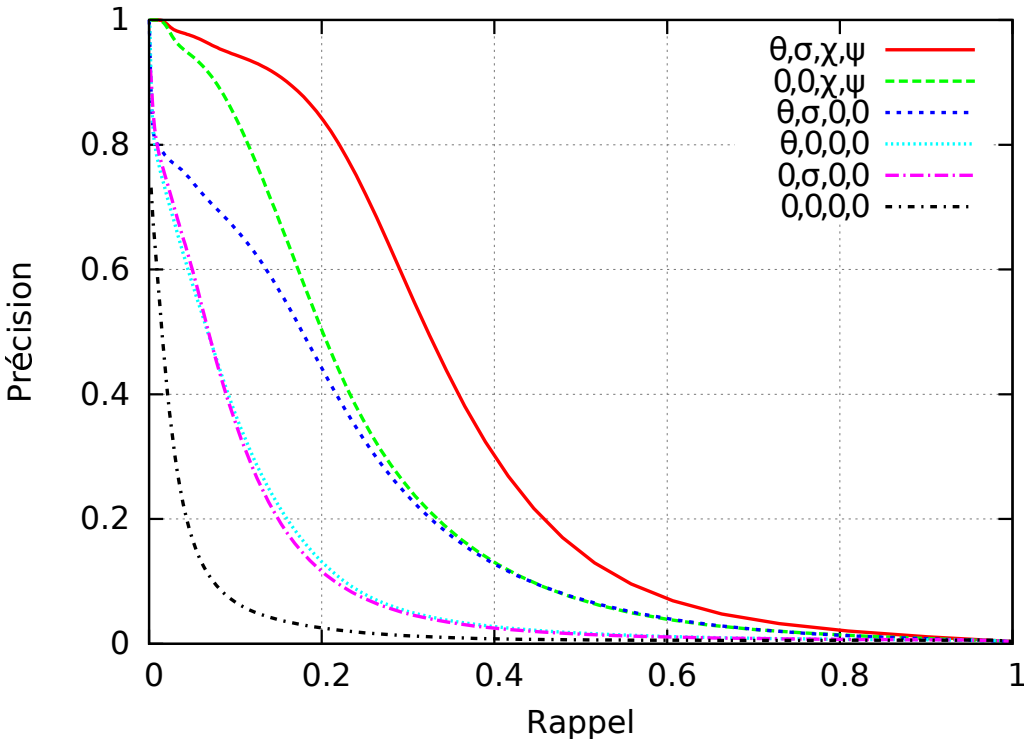


FIGURE 6.4 – Contribution des attributs de Géométrie Faible $(\theta, \sigma, \chi, \psi)$ sur la précision et le rappel, pour la méthode à base de hachage visuel.

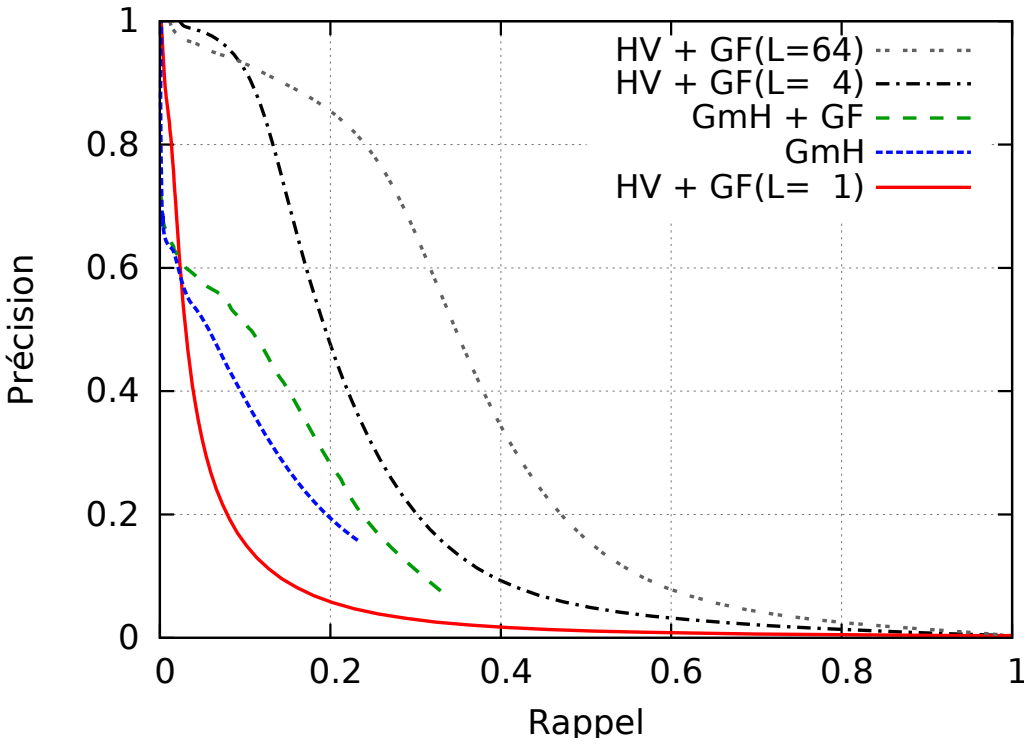


FIGURE 6.5 – Contribution de la géométrie faible sur les méthodes de calcul de score de vraisemblance. HV : Hachage Visuel, GF : Géométrie Faible, L : nombre de tables de hachage visuel, GmH : Geometric min-Hashing.

6.5.3 Comparaison de notre méthode à GmH

Dans la figure 6.5, nous comparons les performances de notre méthode « Hachage Visuel (HV) + Géométrie Faible (GF) » avec la méthode état de l'art « Geometric min-Hashing (GmH) » présentée dans la section 3.4 page 50, avec ou sans la contribution du vote sur la géométrie faible (cf. section 6.4 page 87).

On constate que notre méthode fonctionne bien mieux que Geometric min-Hashing, lorsqu'on utilise notre hachage des attributs de géométrie faible avec $L = 4$ tables de hachage visuel. Le gain apporté par la géométrie faible à GmH reste trop faible pour justifier pleinement son emploi. On peut supposer que ceci vient du fait que GmH utilise déjà une contrainte de voisinage pour générer ses *sketches*, donc le hachage sur la position devient redondant et seul le gain sur l'angle θ et l'échelle σ est visible.

On remarque que l'on peut facilement augmenter le nombre L de tables de hachage visuel afin d'améliorer encore le compromis précision/rappel.

Troisième partie

Expérimentations et Applications

Chapitre 7

Découverte et fouille de logos

7.1 Le corpus FlickrBelgaLogos

L'évaluation des performances des algorithmes de découverte et fouille d'objets visuels est un problème plus complexe que celui de la recherche d'objets visuels. En effet, la recherche d'objets visuels peut être évaluée par la réalisation d'une vérité terrain construite sur la base d'une liste de requêtes. Plus précisément, cela consiste, pour chaque requête de la liste (une requête étant une zone d'image), à fournir la liste des images (ou zones d'images) dont on considère qu'elles sont visuellement similaires à la requête.

Ce type de vérité terrain n'est malheureusement pas adapté aux problèmes de découverte et fouille d'objets visuels. En effet, il est dans ce cas nécessaire de disposer d'une vérité terrain couvrant la totalité des objets visuels se répétant dans la collection d'images, alors qu'on ne dispose que d'une vérité terrain partielle (construite sur la base d'une liste de requêtes). En outre, il faut remarquer que dans les images naturelles, il est très difficile de définir une liste exhaustive de tous les objets visuels répétés. Cela supposerait pour le créateur de la vérité terrain, qu'à chaque image, il soit capable de déterminer si l'un des multiples objets visuels composant l'image se répète ne serait-ce qu'une seule fois dans tout le reste de la collection d'images.

A notre connaissance, il n'existait pas de vérité terrain adaptée à notre problème au début de ce travail.

La construction d'une telle vérité terrain nous paraissant trop difficile à réaliser, nous avons préféré adapter la collection d'images plutôt que d'adapter la vérité terrain. L'idée était de contrôler les répétitions d'objets visuels tout en conservant une collection d'images naturelles. Pour atteindre cet objectif, nous avons introduit des répétitions dans une collection de 10 000 images n'en contenant pas, les objets visuels introduits étant des logos de la base BelgaLogos, et les images « sans répétitions » venant de Flickr.

Logo	Illustration	# Instances	
		Total	OK
Adidas		1043	147
Adidas-text		178	63
Airness		120	11
Base		248	162
BFGoodrich		308	86
Bik		270	65
Bouigues		32	14
Bridgestone		105	31
Bridgestone-text		201	64
Carglass		65	18
Citroen		242	78
Citroen-text		331	197
CocaCola		73	40
Cofidis		90	45
Dexia		626	235
ELeclerc		20	15
Ferrari		213	77
Gucci		4	2
Kia		242	141

Logo	Illustration	# Instances	
		Total	OK
Mercedes		279	86
Nike		2242	235
Peugeot		7	5
Puma		800	157
Puma-text		80	27
Quick		253	57
Reebok		66	18
Roche		2	2
Shell		236	123
SNCF		10	7
Standard-Liege		381	98
StellaArtois		28	20
TNT		183	102
Total		96	78
US_President		14	14
Umbro		659	153
Veolia		77	12
VRT		18	10
Somme totale		9842	2695

TABLE 7.1 – Logos présents dans la vérité terrain de la base BelgaLogos

L'introduction de ces logos s'est faite en copiant et collant les instances (chaque pixel de leur rectangle englobant) depuis les images de BelgaLogos vers celles de Flickr, à des coordonnées spatiales aléatoires, mais sans autres transformations géométriques du type rotation ou changement d'échelle. Les instances originales possèdent en effet déjà des angles et des échelles variés, mais surtout naturels.

La vérité terrain originale de BelgaLogos ne fournissant pas les coordonnées spatiales des instances, mais seulement les noms des images, un travail préliminaire de complètement de la vérité terrain a dû être effectué. Nous avons pour cela fourni manuellement les coordonnées des rectangles englobants de chacune des 9 842 instances appartenant à 37 objets visuels (logos) différents. Chacune de ces 9 842 instances a ensuite été évaluée par trois utilisateurs afin de la classifier (par un vote majoritaire) en « conservée » ou « rejetée » selon qu'ils étaient capables d'identifier aisément l'objet visuel ou non, sans avoir accès au contexte de l'image puisque seul le contenu du rectangle englobant l'instance était présenté. Au final, seules 2 695 instances ont été conservées. Le tableau 7.1 page ci-contre présente la liste des logos annotés dans la vérité terrain, avec le nombre total d'instances, et le nombre (OK) d'instances conservées.

Afin de garantir l'absence de répétitions (ou tout du moins les minimiser) dans les images de Flickr, celles-ci ont été sélectionnées aléatoirement dans 10 000 régions différentes du globe (c'est-à-dire une image par région) grâce aux informations de géolocalisation associées aux photographies. Ces cases géographiques sont des carrés de 2,5° de longitude et latitude. Nous garantissons également que les 10 000 images proviennent de 10 000 utilisateurs différents (grâce aux identifiants Flickr). Ces contraintes permettent ainsi de minimiser la probabilité d'obtenir des répétitions d'objets visuels. Nous minimisons principalement les répétitions architecturales (grâce à la géographie), et d'autres objets divers, car en multipliant les utilisateurs, nous augmentons la diversité des centres d'intérêts, et nous évitons les répétitions d'objets personnels (vêtements, véhicules, …).

Nous sommes toutefois bien conscients des limites de la méthode, qui par exemple ne tient pas compte du phénomène de la mondialisation économique et culturelle, qui provoque l'apparition de logos « Coca-Cola » ou « McDonald's » dans la quasi totalité des pays du monde.

Une solution pour identifier ces répétitions involontaires pourrait se baser sur la construction d'un graphe d'appariement entre toutes les images d'une collection (cf. 3.2 page 49), en requêtant exhaustivement les images avec notre système de recherche d'objets visuels. Les images représentées par les nœuds isolés pourraient alors être considérées comme n'ayant aucune similarité avec le reste de la collection. Nous pourrions également conserver une seule image par composante connexe. Même si cette solution pourrait s'avérer idéale pour notre évaluation, nous l'avons rejetée pour une raison simple : elle n'est pas objective. En effet, en usant de cette approche, nous diminuerions le nombre de faux positifs lors de l'évaluation de notre méthode, mais probablement moins que lors de l'évaluation de méthodes concurrentes, utilisant des composants différents (description, indexation, …).

La figure 7.1 présente un échantillon des images de la base FlickrBelgaLogos.

FIGURE 7.1 – Un échantillon des 10 000 images de la base FlickrBelgaLogos

7.2 Vitesse de convergence en fonction des différents scores de vraisemblance

L'objectif de cette section est d'analyser l'influence des scores de vraisemblance sur la vitesse de convergence de l'algorithme RANSAS, c'est-à-dire d'analyser les performances de l'ensemble du processus de découverte, et non plus seulement la pertinence des scores de vraisemblance. Comme nous l'avons vu précédemment, le calcul des scores de vraisemblance a pour but de focaliser la création de requêtes dans les zones à forte probabilité de contenir des objets $\{c, f\}$-fréquents. Les premières requêtes issues de l'échantillonnage pondéré ont donc de grandes chances de découvrir des objets $\{c, f\}$-fréquents, tandis que ces chances s'amenuisent au fur et à mesure des échantillonnages successifs, en raison du caractère adaptatif de la pondération (cf. section 4.2.1 page 58). Autrement dit, plus le nombre T d'itérations de RANSAS augmente, et moins l'influence des scores de vraisemblance est mesurable.

Nous verrons donc dans la section 7.2.2 page ci-contre que les meilleurs scores de vraisemblance permettent de découvrir plus rapidement des objets $\{c, f\}$-fréquents, mais que les performances finissent par converger après un grand nombre d'itérations.

7.2.1 Protocole d'expérimentations

Les serveurs utilisés dans ces expériences sont équipés de 2 processeurs hexa-cores (Intel X5660). Les temps de calcul sont donnés en temps absolu. Les mesures utilisées dans les expérimentations suivantes sont le « rappel objet » (pour évaluer le problème de la découverte d'objets), le « rappel instance » (pour

évaluer le problème de la fouille d'objets), et la « précision instance ». Ces mesures sont données en fonction du nombre d'itérations T de RANSAS. Elles dépendent en grande partie des performances de notre système de recherche d'objets visuels (cf. 4.5 page 65), mais les performances relatives sont déterminées par les différents scores de vraisemblance utilisés.

Le rappel instance est défini comme le nombre d'instances d'objets $\{c, f\}$-fréquents uniques retrouvées dans au minimum un groupe de résultats de l'étape de recherche de RANSAS, à travers tous les résultats obtenus au moment de l'itération considérée, divisé par le nombre d'instances de la vérité terrain.

Le rappel objet concerne quant à lui le pourcentage d'objets $\{c, f\}$-fréquents uniques retrouvés, de la même manière que précédemment.

La précision instance mesure le nombre d'instances d'objets $\{c, f\}$-fréquents uniques retrouvées dans au minimum un groupe de résultats de l'étape de recherche de RANSAS, divisé par le nombre de résultats retournés au moment de l'itération considérée.

La base d'images utilisée est FlickrBelgaLogos, présentée dans la section 7.1 page 95.

La recherche d'objets a été paramétrée avec :

- SIFT : octave -1, produisant 52 millions de descripteurs ;
- RMMH : noyau « produit scalaire », $M = 40$, et 128 bits par descripteur ;
- APMP-LSH : 300 KNN, $\alpha = 0.80$;
- Vérification géométrique : seuillage a contrario égal à 0.95 ;

7.2.2 Évaluation

Dans les graphiques des figures 7.2a, 7.2b page suivante et 7.3 page 101, on observe l'augmentation du rappel (objet ou instance) et la diminution de la précision en fonction du nombre d'itérations de RANSAS (en échelle logarithmique), et pour différents scores de vraisemblance, ainsi que dans le cas d'un échantillonnage uniforme, et d'un score de vraisemblance parfait.

Afin de réaliser un échantillonnage uniforme, nous utilisons un score de vraisemblance tel que sa fonction de probabilité de masse soit égale à :

$$p_0(\boldsymbol{x}_i) = 1/N, \ \forall \boldsymbol{x}_i \in \boldsymbol{X}$$

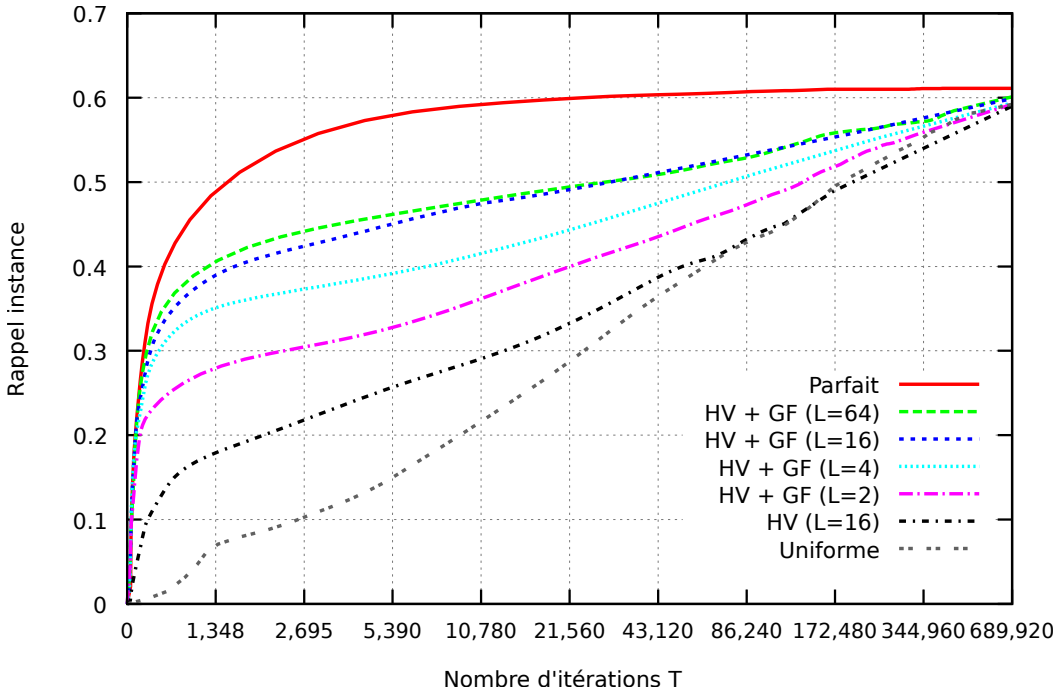
où $\boldsymbol{X}$ est l'ensemble des N descripteurs de la base.

Tandis que le score de vraisemblance parfait donne la fonction de probabilité de masse suivante :

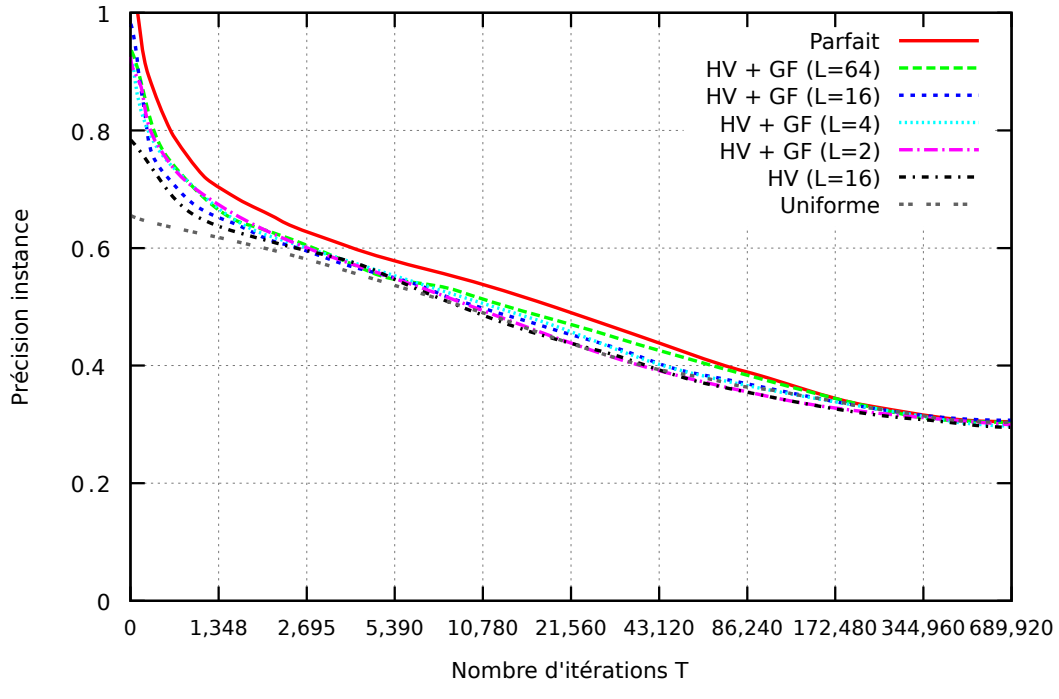
$$p_0(\boldsymbol{x}_i) = 0, \ \forall \boldsymbol{x}_i \notin \{\boldsymbol{X}_s^m\} \quad \text{et} \quad p_0(\boldsymbol{x}_i) = \frac{1}{|\{\boldsymbol{X}_s^m\}|}, \ \forall \boldsymbol{x}_i \in \{\boldsymbol{X}_s^m\}$$

avec $\{\boldsymbol{X}_s^m\}$ l'ensemble des descripteurs appartenant à des instances de la vérité terrain.

La courbe « Uniforme » représente le rappel minimum que l'on devrait obtenir avec n'importe quel score de vraisemblance. En effet, un score donnant une courbe inférieure à celle-ci signifierait que ce score défavorise les instances de la vérité terrain au profit du reste de la collection, ce qui serait



(a) Rappel instance



(b) Précision instance

FIGURE 7.2 – Rappel et de la précision instance en fonction du nombre d'itérations de RANSAS
HV : Hachage Visuel, GF : Géométrie Faible, L : nombre de tables de hachage visuel.

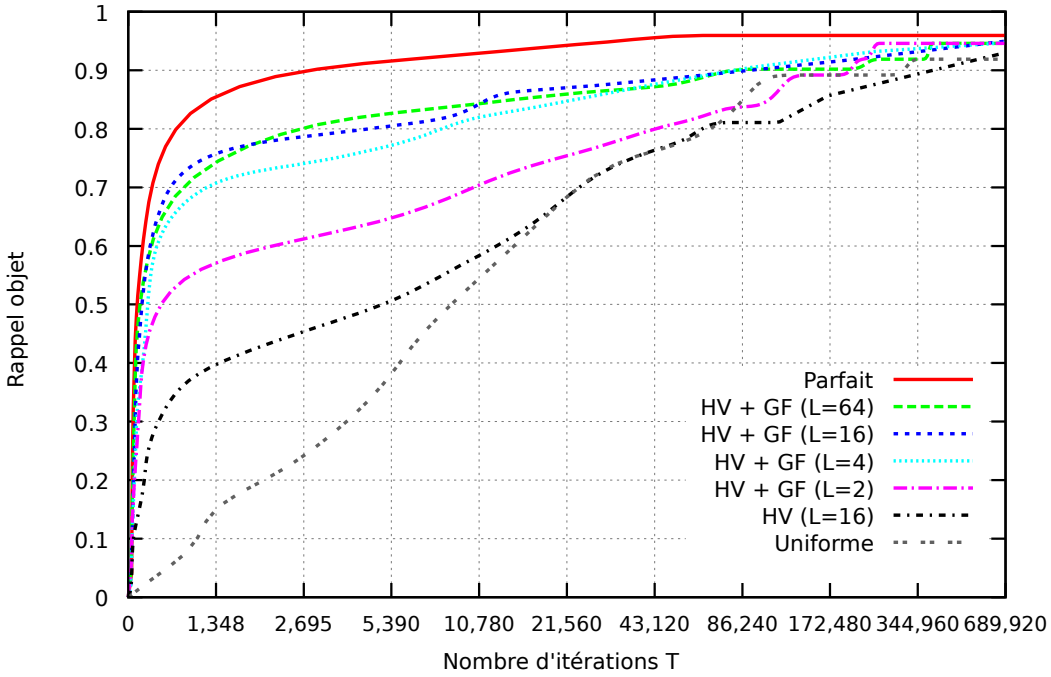


FIGURE 7.3 – Rappel objet en fonction du nombre d'itérations de RANSAS
HV : Hachage Visuel, GF : Géométrie Faible, L : nombre de tables de hachage visuel.

exactement l'inverse de ce que l'on recherche. A contrario, on ne peut espérer créer de scores de vraisemblance donnant une courbe supérieure à la courbe « Parfait », tout du moins pas sans modifier la méthode de recherche d'objets, ou ses paramètres. Ces deux courbes définissent donc l'intervalle dans lequel peuvent évoluer les courbes de rappel obtenues à partir de nos différents scores de vraisemblance.

Les courbes montrent tout d'abord que notre score de vraisemblance (avec typiquement $L = 16$) est bien plus proche du score « Parfait », que du score « Uniforme ». Ceci prouve donc que les scores $z_0(\boldsymbol{x}_i)$ quantifient bien la vraisemblance qu'un descripteur $\boldsymbol{x}_i$ appartienne à un objet $\{c, f\}$-fréquent de la vérité terrain.

On constate également que toutes les courbes finissent par converger au bout d'un grand nombre d'itérations (environ 600 000), à environ 91% d'objets découverts, et 61% d'instances découvertes. Cette convergence montre les performances limites de la méthode de recherche d'objets. Avec autant d'itérations, on a probablement requêté la plupart des instances d'objets de la vérité terrain, et il est donc inutile de poursuivre plus loin l'expérimentation.

De plus, on observe que l'augmentation du rappel est bien plus forte lorsqu'un petit nombre d'itérations ont été effectuées. Ainsi, après seulement 1 348 itérations (soit la moitié du nombre d'instances de la base FlickrBelgaLogos), nos meilleurs scores de vraisemblance ont permis la découverte de plus de 70% des objets, et de la moitié des instances retrouvables (par notre méthode), tandis qu'il faut environ 32 fois plus d'itérations pour obtenir de telles performances avec un échantillonnage uniforme.

On peut noter que la précision instance diminue au fur et à mesure des itérations de RANSAS, et

converge finalement pour tous les scores de vraisemblance vers 30% de précision. Ceci s'explique par le fait qu'à chaque itération, on diminue les poids (scores) des descripteurs correspondant à la requête et aux résultats. A l'initialisation, les scores les plus forts sont ceux représentant les descripteurs des instances les plus faciles à retrouver (par notre méthode). Après un grand nombre d'itérations, les scores les plus forts représentent en majorité des images sans instances (de la vérité terrain), ou des instances d'objets difficiles à retrouver. La précision de ces dernières requêtes est donc beaucoup plus faible. La convergence vers 30% correspond à la précision minimale atteinte par notre système de recherche d'objets visuels lorsqu'on requête principalement des zones ne comportant pas d'instances de la vérité terrain.

Le tableau 7.2 page suivante donne un aperçu du compromis temps/qualité de notre méthode comparé à un échantillonnage uniforme. On observe qu'avec tous les coûts de calcul inclus (hormis la description, qui est une constante), notre méthode permet l'obtention d'un facteur d'accélération de la découverte pouvant aller jusqu'à 18 (dans nos expérimentations). Ce tableau montre qu'il faut choisir un nombre L de tables de hachage visuel en fonction du rappel souhaité. Pour un rappel (objet ou instance) faible, on obtient les meilleurs facteurs d'accélération avec peu de tables (typiquement 2 ou 4). Par contre, pour obtenir un rappel fort, mieux vaut prendre environ 16 tables. On remarque que l'utilisation de 64 tables n'est jamais pertinente (toujours dans cette expérimentation). Le temps de calcul du score de vraisemblance est en effet dans ce cas tellement élevé qu'il est préférable de faire davantage d'itérations de RANSAS avec un moins bon score de vraisemblance. Au total, il faut environ une quinzaine d'heures pour découvrir 90% des objets de la vérité terrain, et moins de neuf heures pour découvrir 50% des instances.

Des exemples de requêtes issus de l'échantillonnage de RANSAS et de certains de leurs résultats sont présentés dans la figure 7.4 page 104.

	U	HV (L=16)	HV+GF (L=2)	HV+GF (L=4)	HV+GF (L=16)	HV+GF (L=64)
Temps de hachage	627s	1 567s	634s	641s	1 567s	7 292s
Temps de calcul du score de vraisemblance	0s	1 128s	163s	320s	1 280s	5 120s

Temps total		**U**	**HV (L=16)**	**HV+GF (L=2)**	**HV+GF (L=4)**	**HV+GF (L=16)**	**HV+GF (L=64)**
Rappel instance	**0.20**	8 858s	4 482s x2.0	**919**s x**9.6**	1 045s x8.5	2 925s x3.0	12 491s x0.7
	0.30	23 588s	14 753s x1.6	2 874s x8.2	**1 282**s x**18.4**	3 073s x7.7	12 597s x1.9
	0.40	55 825s	47 731s x1.2	19 676s x2.8	7 471s x7.5	**4 113**s x**13.6**	13 225s x4.2
	0.50	159 973s	158 105s x1.0	120 289s x1.3	60 146s x2.7	**30 751**s x**5.2**	37 066s x4.3

Temps total		**U**	**HV (L=16)**	**HV+GF (L=2)**	**HV+GF (L=4)**	**HV+GF (L=16)**	**HV+GF (L=64)**
Rappel objet	**0.30**	4 078s	3 054s x1.3	**901**s x**4.5**	1 039s x3.9	2 916s x1.4	12 467s x0.3
	0.40	5 724s	3 780s x1.5	**936**s x**6.1**	1 108s x5.2	2 932s x2.0	12 480s x0.5
	0.50	7 854s	7 111s x1.1	**1 136**s x**6.9**	1 163s x6.8	2 982s x2.6	12 520s x0.6
	0.60	14 227s	14 828s x1.0	2 515s x5.7	**1 237**s x**11.5**	3 063s x4.6	12 635s x1.1
	0.70	20 661s	23 081s x0.9	9 898s x2.1	**1 830**s x**11.3**	3 248s x6.4	12 916s x1.6
	0.80	65 831s	56 691s x1.2	37 338s x1.8	7 987s x8.2	**7 110**s x**9.3**	14 407s x4.6
	0.90	281 205s	199 128s x1.4	184 419s x1.5	**55 234**s x**5.1**	96 441s x2.9	78 261s x3.6

TABLE 7.2 – Temps et accélération (en vert) obtenus avec différents scores de vraisemblance. U : Uniforme, HV : Hachage Visuel, GF : Géométrie Faible, L : nombre de tables de hachage visuel. Le temps de hachage est celui nécessaire au hachage des descripteurs locaux. Il faut au minimum 128 bits pour la recherche d'objets. Ensuite, pour chaque valeur de rappel considérée, on donne le temps total (hachage + calcul du score + recherche d'objets). Les facteurs d'accélération sont donnés en comparaison de ceux obtenus avec un échantillonnage uniforme.

FIGURE 7.4 – Les deux premières requêtes issues de l'échantillonnage pondéré par le score de vraisemblance HV + GF (L=64), et quatre de leurs résultats de recherche

7.3 Classification des instances par un algorithme de regroupement

Notre méthode de découverte et fouille d'objets propose en sortie une liste de résultats (rectangles englobant les instances dans les images) pour chaque requête issue de la phase d'échantillonnage. Afin d'améliorer le rappel instance, nous avons vu (cf. figure 7.2a page 100) qu'il était nécessaire d'effectuer un certain nombre d'itérations, afin de rechercher plusieurs instances de chaque objet visuel.

Cette méthode génère donc en sortie des groupes d'instances représentant des sous-parties d'objets $\{c, f\}$-fréquents. Le nombre de ces groupes peut rapidement devenir difficile à analyser pour un opérateur humain, il est donc souhaitable de lui proposer de rassembler les différents groupes d'instances appartenant à un même objet $\{c, f\}$-fréquent.

Chaque instance découverte est reliée à au moins une autre instance, et on connait leur score de similarité. Il est donc possible de construire un graphe d'appariement entre instances.

La construction d'un tel graphe permet ensuite d'identifier des groupes d'instances plus complets, par l'extraction des composantes connexes. Si le graphe présente trop peu de composantes connexes, il faut alors envisager un processus de regroupement (*clustering*) plus complexe.

Dans la section suivante, nous détaillerons une solution envisageable pour résoudre ce problème, utilisant l'outil MCL-edge [167] de Stijn van Dongen, implémentant l'algorithme MCL (*Markov Cluster*

algorithm) [168].

7.3.1 L'algorithme MCL

Selon Peyronnet [169] :

> « L'algorithme MCL (*Markov Cluster algorithm*) est un algorithme de partitionnement rapide et capable de prendre en entrée de très grands graphes [millions de nœuds et centaines de millions d'arêtes]. Il est basé sur la simulation de flots stochastiques dans un graphe. Il a été mis au point par Stijn van Dongen au Centre for Mathematics and Computer Science (CWI).
> Les *clusters* naturels dans un graphe sont caractérisés par la présence d'un grand nombre d'arcs entre les membres de ce *cluster*, et on peut s'attendre à ce que le nombre de chemins de longueur supérieure entre deux nœuds arbitraires dans le *cluster* soit grand. En particulier, ce nombre devrait être grand, relativement aux paires de nœuds appartenant à des *clusters* naturels différents. Vu sous un autre angle, une marche aléatoire dans le graphe va peu fréquemment aller d'un *cluster* naturel à un autre. L'intuition derrière l'algorithme correspond donc à ce que va faire un être humain pour repérer des regroupements de nœuds : trouver les « gros » blocs de nœuds fortement connectés, groupes qui sont eux-mêmes peu liés entre eux.
>
> L'algorithme MCL trouve la structure en *cluster* d'un graphe grâce à une procédure mathématique de *bootstrapping*. Le processus calcule de manière déterministe les probabilités des marches aléatoires dans le graphe, et utilise deux opérateurs transformant un ensemble de probabilités dans un autre. Cela est rendu possible par l'utilisation du langage des matrices stochastiques (aussi appelées matrices de Markov), qui formalise le concept mathématique de marches aléatoires dans un graphe. Informellement, une matrice stochastique donne les probabilités de transition entre deux nœuds d'un graphe.
>
> L'algorithme MCL simule des marches aléatoires à l'intérieur d'un graphe en alternant deux opérations appelées expansion et inflation. L'expansion consiste à prendre la puissance d'une matrice stochastique en utilisant le produit matriciel usuel (c'est-à-dire la mise au carré d'une matrice). L'inflation consiste à prendre la puissance d'Hadamard d'une matrice, suivie d'une étape de mise à l'échelle, de telle sorte que la matrice résultante soit à nouveau stochastique, c'est-à-dire que les éléments de la matrice (sur chaque colonne) correspondent à des valeurs de probabilités. Une matrice colonne stochastique est une matrice non-négative avec la propriété que chacune de ses colonnes ait une somme égale à 1. Étant donné une telle matrice M, et un nombre réel $r > 1$, la matrice colonne stochastique résultant de l'inflation de chacune des colonnes de M avec le coefficient de puissance r est écrit $G_r(M)$, et G_r est appelé l'opérateur d'inflation de coefficient de la puissance r. En écrivant $S_{r,j}(M)$ pour la sommation de toutes les entrées de la colonne j de M pris à la puissance r (la

somme est effectué après la mise à la puissance), $G_r(M)$ est défini pour chaque élément par $G_r(M_{ij}) = M_{ijr}/S_{r,j}(M)$ Chaque colonne j de la matrice stochastique M correspond à un nœud j du graphe stochastique associé à M. L'élément de la ligne i dans la colonne j (c'est-à-dire l'élément M_{ij}) correspond à la probabilité d'aller du nœud j au nœud i. On peut observer que pour des valeurs de $r > 1$, l'inflation change les probabilités associées avec la collection de marches aléatoires partant d'un nœud particulier (correspondant à la colonne d'une matrice) en favorisant les marches les plus probables par rapport aux marches moins probables. L'expansion correspond à calculer des marches aléatoires de longueur supérieure, ce qui signifie des marches aléatoires avec beaucoup d'étapes. Il associe des nouvelles probabilités à toutes les paires de nœuds, où un nœud est le point de départ et l'autre est la destination. Puisque les chemins de longueur supérieure sont plus courants à l'intérieur d'un *cluster* qu'entre des *clusters* différents, les probabilités associées avec les paires de nœuds appartenant au même *cluster* vont, en général, être relativement grandes car il y a beaucoup de chemins allant d'un nœud à l'autre. L'inflation va donc avoir pour effet d'augmenter les probabilités de marches intra-cluster et va diminuer les marches inter-cluster. Ceci est obtenu sans connaissance à priori de la structure des *clusters*. C'est simplement le résultat de la présence d'une structure en *cluster*. Finalement, itérer les expansions et inflations résulte en la séparation du graphe en plusieurs segments. Il y a plus de chemins entre ces segments, et la collection des segments résultants est simplement interprétée comme un *clustering*. L'opérateur d'inflation peut être altéré en utilisant le paramètre r. Augmenter ce paramètre a pour effet de rendre l'opérateur d'inflation plus fort, et cela augmente la granularité des *clusters*.

Informellement, exprimé dans le langage des flux stochastiques, on peut voir que l'expansion provoque la dissipation du flux à l'intérieur des *clusters* alors que l'inflation élimine le flux entre les différents *clusters*. L'expansion et l'inflation représentent les différentes forces qui s'alternent jusqu'à ce qu'un état d'équilibre soit atteint. Un état d'équilibre prend la forme d'une matrice doublement idempotente, c'est-à-dire d'une matrice qui ne change pas avec d'autres applications des opérateurs d'expansion ou d'inflation. Le graphe associé à une telle matrice consiste en différentes composantes connexes orientées. Chaque composante est interprétée comme un *cluster*, et a une forme d'étoile, avec un attracteur au centre, et des arcs allant de tous les nœuds de cette composante vers l'attracteur. En théorie, des systèmes attracteurs possédant plus d'un nœud attracteur peuvent apparaitre, mais cela ne change pas l'interprétation en *clusters*. De plus, il peut exister des nœuds qui sont connectés à différentes étoiles, ce qui est canoniquement interprété comme des superpositions de *clusters*, ou en d'autres termes, certains nœuds peuvent appartenir à plusieurs *clusters*. Cette propriété de la méthode est d'ailleurs sans doute un avantage dans le cadre de la cartographie d'un système d'information, car on peut imaginer l'existence de nœuds qui sont connectés à de nombreux *clusters*, mais qui ne

doivent pas appartenir à un seul de ces *clusters*. Pour ce qui est de la convergence, il peut être prouvé que le processus simulé par l'algorithme converge en un temps quadratique vers l'état d'équilibre. En pratique, l'algorithme commence à converger de manière visible après peu d'itérations (environ une dizaine). La convergence globale est quelque chose de très difficile à prouver, mais on peut conjecturer que le processus converge toujours si le graphe d'entrée est symétrique. A noter qu'il n'y a pas de garantie mathématique à l'heure actuelle pour la convergence. Une chose très importante à propos de l'algorithme est le fait qu'il récupère la structure en *clusters* via la trace laissée par cette structure sur le processus du flux.

D'autres avantages de cet algorithme sont :

- il n'est pas mis en faute par des arcs liant des *clusters* différents ;
- il est très rapide et supporte bien les fortes charges (plusieurs centaines de milliers de nœuds, la limite étant la multiplication de matrices) ;
- il a un paramètre naturel influençant la granularité des *clusters* ;
- les mathématiques associées à l'algorithme montrent qu'il y a une relation intrinsèque entre le processus simulé et la structure en *clusters* du graphe ;
- sa formulation est simple et élégante ;

D'après la définition de l'algorithme MCL, on peut voir qu'il est basé sur un paradigme différent de tous les autres algorithmes basés sur des systèmes de liens. En effet, son fonctionnement est probabiliste, et sa terminaison basée sur une notion de convergence d'un processus aléatoire. »

Dans notre cas, les nœuds du graphe sont les images dans lesquelles ont a détecté des instances, et on ajoute une arête pour chaque appariement entre deux instances.

7.3.2 Protocole d'évaluation

Le regroupement des groupes d'instances donnés par RANSAS a pour objectif principal de faciliter le travail de l'utilisateur. Ce travail consiste à fusionner des groupes entre eux (parce qu'ils représentent le même objet visuel), ou à nettoyer le groupe (extraire les instances qui ne correspondent pas à l'objet visuel identifié). Il est donc préférable que le nombre de groupes (*clusters*) soit relativement faible (quelques centaines de préférence, mais pas moins que le nombre d'objets visuels attendus, 37 en l'occurrence), et surtout que les groupes soient les plus purs possibles.

Le seul paramètre réglable dans MCL-edge [167] est l'inflation. Afin de mesurer l'effet de ce paramètre sur la pureté et le nombre des *clusters* produits, nous proposons de mesurer l'évolution de la pureté moyenne (AvgPurity) en fonction du nombre de *clusters* lorsqu'on fait varier l'inflation dans l'intervalle conseillé par Van Dongen, c'est-à-dire entre 1.01 et 6.

La pureté moyenne (AvgPurity) est définie comme étant la moyenne de la pureté de chaque *cluster*,

telle que définie dans [170] par :

$$Purity(\boldsymbol{C}_j) = \frac{1}{|\boldsymbol{C}_j|} \max_{k=1,\dots c} |\boldsymbol{C}_{j,k}| \tag{7.1}$$

où $|\boldsymbol{C}_j|$ est la taille du $j^{\text{ème}}$*cluster*, et $|\boldsymbol{C}_{j,k}|$ la taille du sous groupe de $\boldsymbol{C}_j$ dont les instances appartiennent au $k^{\text{ème}}$ des c objets de la vérité terrain ($c = 37$).

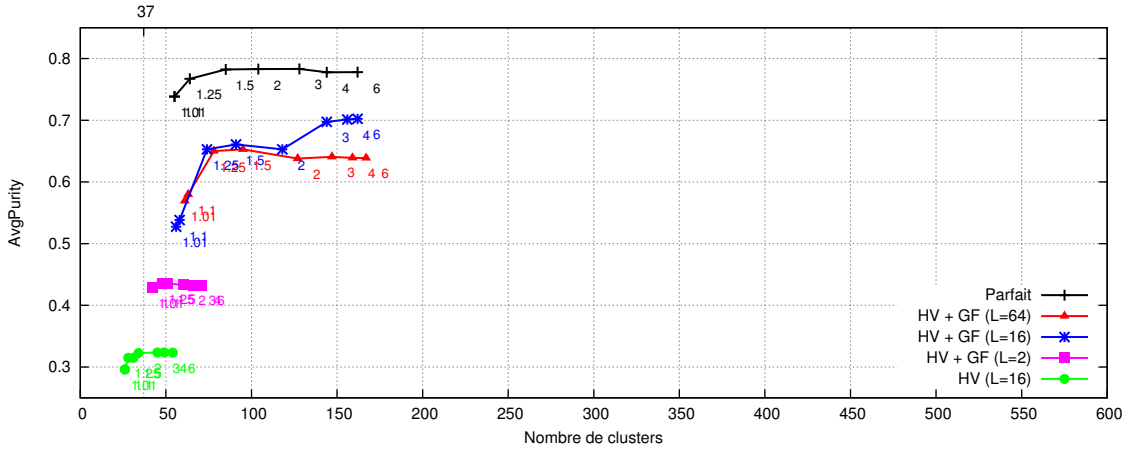
7.3.3 Évaluation

La figure 7.5 page ci-contre montre la pureté moyenne des *clusters* en fonction du nombre de *clusters* produits par MCL. On peut lire la valeur d'inflation sur les courbes. La figure 7.5 page suivante présente trois graphiques, afin de présenter les résultats sur des graphes construits avec 1 000, 10 000 ou 100 000 itérations de RANSAS. Il est ici nécessaire de rappeler que plus nombreuses sont les itérations et plus le graphe d'appariement est complet. Chaque graphique présente une courbe par score de vraisemblance utilisé dans RANSAS. On compare ici le score « parfait » (cf. section 4.3 page 62) et les scores calculés à partir de Hachage Visuel (HV), complétés ou non par le vote sur la Géométrie Faible (GF).

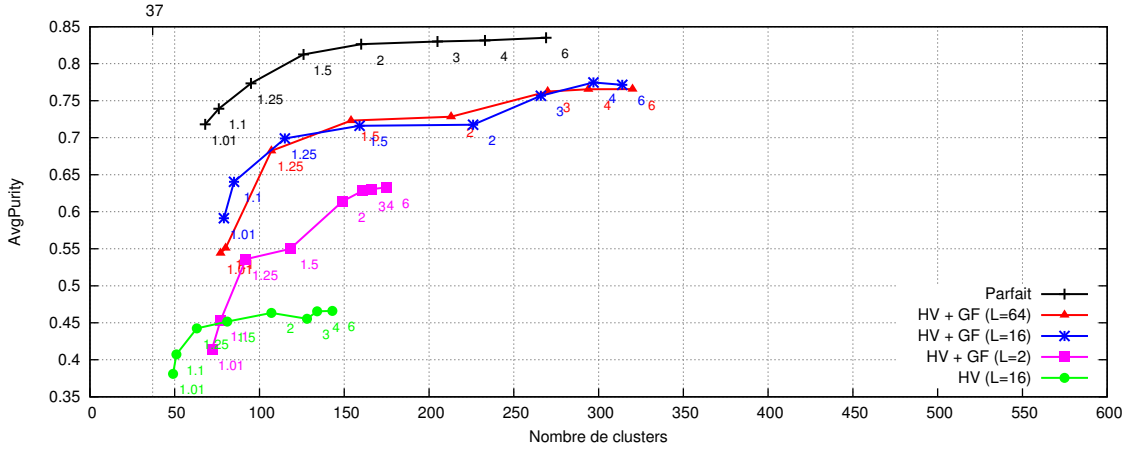
On constate que les performances globales des différents scores de vraisemblance convergent lorsqu'on utilise le graphe d'appariement construit avec 100 000 itérations de RANSAS. Comme on l'a dit précédemment dans la section 7.2.2 page 99, on est dans ce cas limité par les performances du système de recherche d'objets, et le score de vraisemblance n'a que peu d'influence.

On observe par contre sans surprise que pour seulement 1 000 itérations de RANSAS, on obtient déjà de bonnes performances avec nos meilleurs scores de vraisemblance (HV + GF et $L = \{16; 64\}$) : environ 70 *clusters* (deux fois plus que dans la vérité terrain) pour une pureté moyenne de 0.65, tandis que les autres scores ne parviennent pas à obtenir plus de 0.45.

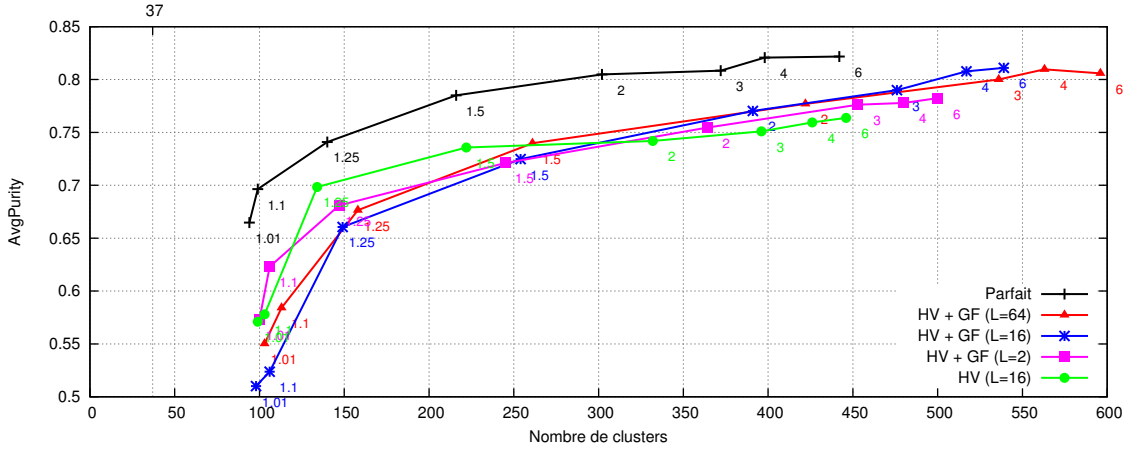
Le temps de calcul mis par MCL-edge pour regrouper les instances est totalement négligeable comparé au reste du processus (description, indexation, RANSAS). Le regroupement est en effet effectué en moins d'une dizaine de secondes, même avec une faible valeur d'inflation (1.01) qui nécessite plus d'itérations pour converger. D'après Van Dongen, MCL-edge peut supporter des millions de nœuds et des centaines de millions d'arêtes, ce qui est une taille globalement similaire à celles des graphes d'appariement construits sur des collections multimédias traitées par les systèmes état de l'art de recherche d'objets visuels.



(a) 1K itérations



(b) 10K itérations



(c) 100K itérations

FIGURE 7.5 – Pureté moyenne (AvgPurity) en fonction du nombre de *clusters* obtenus en faisant varier le paramètre d'inflation de MCL entre 1.01 et 6, pour différent scores de vraisemblance, et différents nombre d'itérations de RANSAS (1K, 10K et 100K).

Chapitre 8

Découverte d'événements saillants dans des médias d'actualité

Dans ce chapitre, nous proposons une expérimentation dans une grande collection de vidéos issues de la télévision. Cette expérimentation a pour objectif de montrer l'intérêt de l'approche transmédia dans la découverte d'évènements (médiatiques) saillants, et également d'évaluer les performances de notre méthode de découverte d'objets visuels appliquée à plus de 2 000 heures de vidéos. Nous proposons également une autre approche transmédia appliquée sur un corpus hétérogène constitué à partir de nombreuses sources de type différent.

Un évènement médiatique a plusieurs définitions possibles, selon que l'on se place du point de vue du sociologue, du journaliste, ou de l'informaticien.

Selon Marie-Luce Viaud (coordinatrice du projet ANR OTMedia [1] No. 2010CORD015) :

> Un évènement médiatique en tant qu'entité [objet visuel] détectable est constitué d'énoncés [visuels] qui relatent une même « occurrence du réel » et son importance médiatique est liée au nombre des énoncés [visuels] qui le constituent.

Parmi les évènements médiatiques français de l'année 2012, on trouve par exemple :

- Mort du journaliste Gilles Jacquier à Homs en Syrie ;
- Cinq Oscars pour le film « The Artist » ;
- Débat de l'entre-deux-tours aux élections présidentielles ;
- ArcelorMittal annonce la fermeture définitive des hauts fourneaux de Florange ;

8.1 Détection d'évènements télévisuels

8.1.1 Création d'une grande collection de journaux télévisés

Afin d'évaluer la capacité de notre méthode à traiter une grande collection multimédia, nous avons construit une nouvelle base composée de 1 809 080 images-clés, extraites de 2 051 heures de

1. www.otmedia.fr

vidéos. Nous l'avons nommée « FrenchTVFrames » dans notre dernière publication [171]. L'extraction des images-clés est réalisée de manière à ce que l'on garde au moins une image pour chaque plan, voire plusieurs si le contenu du plan est très variable (si la caméra bouge par exemple). On extrait donc bien plus d'images dans les plans des reportages que dans ceux sur le plateau du journal télévisé. En moyenne, on garde environ une image toutes les quatre secondes.

Les vidéos ont été choisies parmi sept chaînes de télévision Française : TF1, France 2, Euronews, BFMTV, France 24, i>TELE et LCI (cf. figure 8.1), entre le 10 juin 2011 et le 28 mars 2012, soit une période d'environ neuf mois. Seule une heure par jour a été conservée pour chaque chaîne, correspondant à celle proposant le plus de journaux télévisés dans la journée.

Nous avons décrit cette base avec environ 549 millions de points d'intérêt SIFT.

FIGURE 8.1 – Chaînes de télévision inclues dans la collection FrenchTVFrames

8.1.2 Approche transmédia pour la découverte d'objets informatifs

La découverte d'éléments fréquents dans les vidéos issues de la télévision pose un problème bien connu [172, 173] : les principales répétitions sont les publicités, les génériques d'émission, les décors de plateaux, etc. Or, ceci n'est pas l'objectif de notre expérimentation, qui vise à obtenir des objets visuels du type « évènement médiatique ».

Afin de défavoriser les évènements peu informatifs, et de mettre en avant les évènements médiatiques saillants, nous avons émis deux postulats :

- Premièrement, un évènement médiatique est généralement relayé par la plupart des sources d'information (la plupart des chaînes de télévision en l'occurrence), contrairement aux génériques d'émission et aux décors de plateaux qui sont presque exclusivement « mono-source ». Quant aux publicités, nous avons pu vérifier qu'elles n'apparaissent pas sur un grand nombre de chaînes ;
- Deuxièmement, un évènement médiatique est condensé dans le temps. Il est traité par les médias pendant quelques jours, avant de disparaître, tandis que les publicités sont diffusées pendant au moins plusieurs semaines, et les génériques et les décors d'émissions pendant plusieurs mois ou années ;

La stratégie que nous avons mise en place repose donc sur un classement des objets découverts. Concrètement, chaque objet détecté s'est vu affecté un score S tel que :

$$S = \frac{\#\{\mathcal{C}\} \cdot \hat{f}}{f_d} \tag{8.1}$$

où $\#\{\mathcal{C}\}$ est le nombre de chaînes (uniques) dans lesquelles on a détecté une instance de l'objet considéré, $\hat{f}$ est l'estimation de la fréquence f de l'objet (nombre de détections) et f_d est la fréquence

temporelle (nombre de jours différents parmi les instances de l'objet découvert).

Ainsi, nous privilégions les objets possédant le plus d'instances et ayant été découverts parmi un maximum de chaînes, tout en minimisant le nombre de jours de diffusion.

La figure 8.1 page suivante donne des illustrations des meilleurs évènements découverts, par catégorie de surface (petit, moyen ou gros objet) dans les 2 051 heures de télévision française, selon le classement obtenu grâce à l'équation 8.1 page ci-contre.

On constate que chaque objet visuel découvert est bien l'un des évènements médiatiques majeurs de la période couverte par notre collection de vidéos.

Le calcul du score de vraisemblance a été effectué avec $L = 8$ tables de hachage visuel et ajout de contraintes de géométrie faible, et la recherche d'objets visuels a été paramétrée avec $\alpha = 0.90$.

Cette expérimentation a été effectuée avec un serveur équipé de deux processeurs hexa-cores (Intel X5660). L'étape de hachage visuel a duré environ 4 heures, le calcul du score de vraisemblance a pris 17 heures, et utilisé 70GB de RAM. RANSAS a ensuite effectué 50 000 itérations en 18 heures. Au total, le processus complet aura donc mis 39 heures pour découvrir des objets dans plus de 1.8 millions d'images.

8.2 Découverte d'évènements transmédias

Dans cette section, nous présentons le travail [174] proposé au Grand Challenge d'ACM Multimedia 2012.

Ce travail s'inscrit dans le cadre du projet ANR OTMedia [2] No. 2010CORD015), dont l'objectif est de collecter et d'analyser les documents multimédias issus de nombreuses sources réparties sur plusieurs types de médias d'information.

Concrètement, quatre catégories de médias ont été collectés et indexés par notre système :

1. **Agences de presse** : La totalité des flux multimédias venant de l'AFP a été collectée. L'AFP est la plus ancienne agence de presse dans le monde et une des trois plus importantes actuellement. Chacune des dépêches multimédias inclut au minimum une image, un texte, et un ensemble de métadonnées ;

2. **Sites web d'information** : Environ 1 500 sites internet français ont été collectés à partir de leurs flux RSS. Les sites qui ont été sélectionnés sont des pure players (12), des sites de radios (14), de TVs (17), de presse (71), des blogs (339), ainsi que les sites des personnalités politiques comme les sénateurs, députés, candidats à la présidentielle (646), des partis politiques (263), des syndicats (19) et des institutions (62) ;

3. **Journaux d'informations** : 7 journaux nationaux ont été collectés numériquement (La Croix, Liberation, Le Monde, 20 Minutes, Direct Matin, les Echos, Le Canard Enchainé) ;

2. www.otmedia.fr

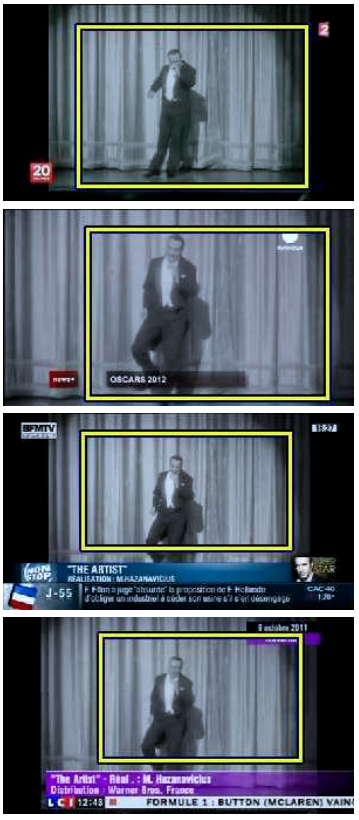
Sommet du G20 *Novembre 2011* Petit objet $< 33\%$ de la surface de l'image	**Décès de Kim Jong-il** *Décembre 2011* Objet moyen $< 66\%$ de la surface de l'image	**The Artist** *Février 2012* Gros objet $> 66\%$ de la surface de l'image
$\#\{\mathcal{C}\} = 6$ $\hat{f} = 15$ $f_d = 2$ $\Big\} S = 45$	$\#\{\mathcal{C}\} = 6$ $\hat{f} = 8$ $f_d = 1$ $\Big\} S = 48$	$\#\{\mathcal{C}\} = 7$ $\hat{f} = 39$ $f_d = 8$ $\Big\} S = 34$
		

TABLE 8.1 – Meilleurs évènements découverts (plus gros scores S), par catégorie (petit, moyen, gros), dans les 2 051 heures de télévision Française.

4. **Journaux télévisés** : Les informations et reportages de 10 chaînes de télévision ont été obtenues, à partir de 5 chaînes publiques, 3 chaînes généralistes et 2 chaînes d'information ;

L'objectif que nous recherchons ici est la découverte d'évènements médiatiques transmédias. Tout comme dans dans les vidéos issues de la télévision, les images et vidéos du web et de la presse présentent de très nombreux objets visuels très fréquents et non informatifs, tel que des publicités, des

bannières, et autres éléments graphiques divers. De même que dans la section précédente, nous partons donc du principe que les évènements médiatiques importants sont relayés par la plupart des médias d'information, et que nous pouvons les mettre en évidence par une approche transmédia, c'est-à-dire en ne considérant que les appariements visuels entre des médias de types différents.

Une image issue d'une agence de presse, et relayée par plusieurs chaînes de télévision, journaux et sites web, est ainsi sujette à représenter un évènement médiatique saillant, qu'une image très fréquemment répétée par un seul média.

Afin d'identifier ces évènements médiatiques transmédias, nous proposons de réaliser une découverte d'objets visuels avec RANSAS, puis un *clustering* MCL, pour chaque semaine calendaire de la collection constituée. En pratique, nous construisons donc un index pour chaque semaine, dans lequel nous effectuons la découverte. Les résultats de la découverte sont filtrés afin de ne conserver que les appariements transmédias. Le *clustering* MCL est ensuite effectué pour rassembler les groupes d'instances découverts. Pour finir, on trie les *clusters* selon un score de diversité calculé à partir du nombre de supports détectés. Plus concrètement, nous avons tout d'abord affecté un poids à chacun des médias collectés. Ceci a été réalisé en pondérant de façon identique les quatre catégories de médias, c'est-à-dire 0,25. Pour une catégorie de médias donnée, le poids est ensuite distribué uniformément entre tous les flux lui correspondant. Le poids du flux AFP est ainsi égal à 0,25, tandis que chacune des dix chaînes de télévision a un poids égal à un dixième de 0,25, soit 0,025. A partir de cette pondération, nous calculons le score de diversité d'un *cluster* en sommant les poids des médias détectés à l'intérieur de celui-ci.

La figure 8.2 page suivante montre un *cluster* filtré par le tri sur notre score de diversité des médias, et trois *clusters* correspondant aux meilleurs scores (meilleurs évènements transmédias) de la première semaine de septembre 2011.

La figure 8.3 page 117 présente une interface de navigation dans les meilleurs évènements transmédias détectés semaine par semaine.

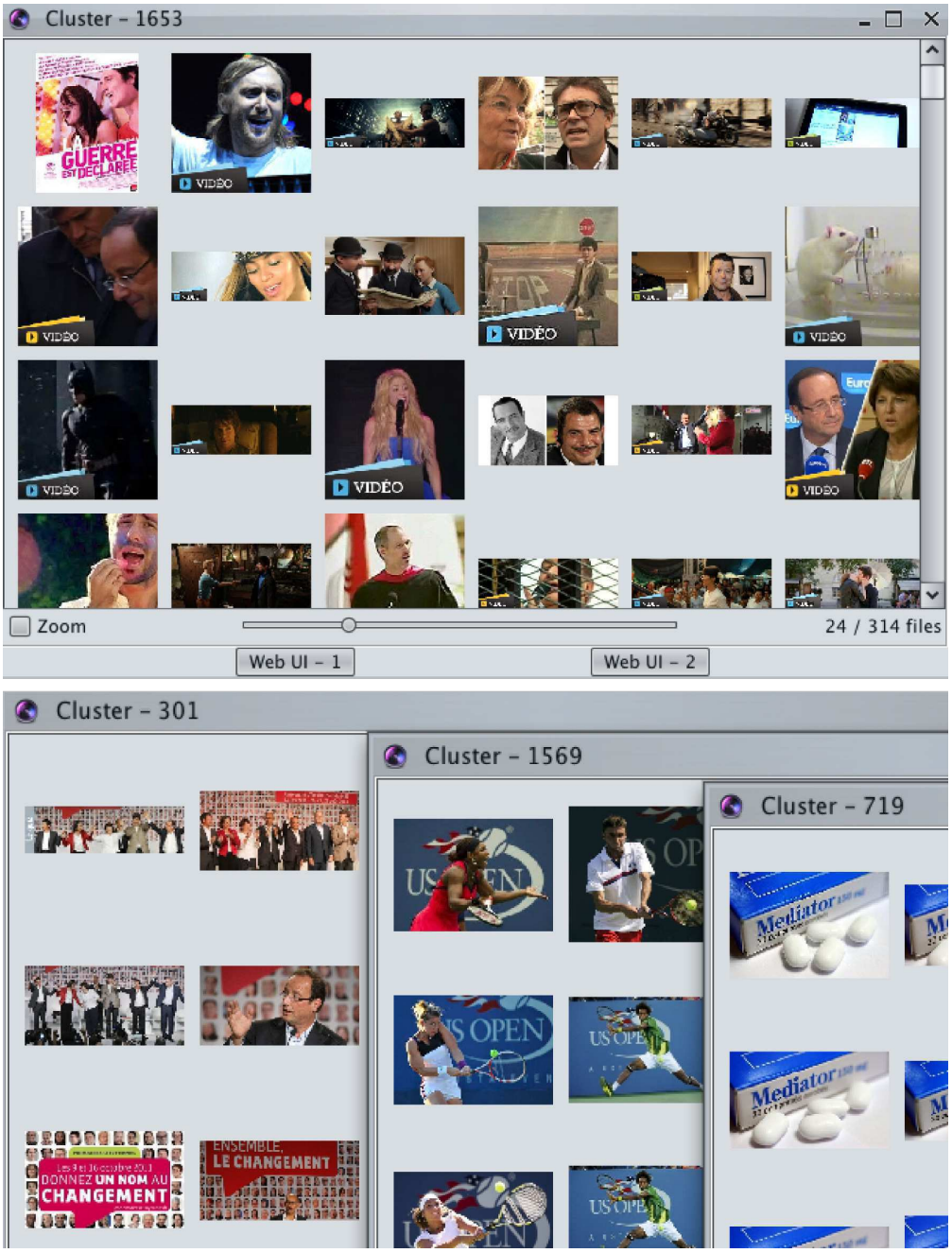


FIGURE 8.2 – Un *cluster* filtré par le tri sur le score de diversité (en haut), et les trois meilleurs *clusters* (évènements transmédias) de la première semaine de septembre 2011, représentant les primaires socialistes, l'US Open, et le scandale du Médiator.

FIGURE 8.3 – Meilleurs évènements transmédias détectés pour les semaines du 12 septembre, 3 octobre, 17 octobre et 14 novembre 2011

Chapitre 9

Suggestion de requêtes visuelles

Nous avons vu dans la section 2.7 page 43 que les systèmes de recherche d'objets visuels sont aujourd'hui capables d'atteindre des performances appréciables, y compris pour la recherche de petits objets comme des logos.

Du point de vue de l'utilisateur, cette recherche d'objets se limite généralement à la création d'une requête locale (fenêtre autour de l'objet), puisque le système se charge ensuite automatiquement d'effectuer la recherche. Cette approche souffre toutefois de deux problèmes liés à la perception de l'utilisateur :

- Premièrement, quand aucune (ou très peu) autre instance de l'objet recherché n'existe dans la collection d'images, le système retourne principalement des faux positifs, et l'utilisateur est déçu par les résultats. En effet, celui-ci est rarement en mesure de déterminer à l'avance si l'objet est fréquent ou non ;
- Deuxièmement, quand l'utilisateur sélectionne un objet déformable, ou très altéré par les conditions de prises de vue, une mauvaise résolution…, il peut également être déçu des mauvais résultats, n'ayant pas conscience des capacités limites du système utilisé. Les utilisateurs sont ainsi fréquemment tentés de rechercher des visages, des fleurs, etc. ;

Une solution envisageable pour limiter la déception de l'utilisateur est de diminuer le taux de fausses-alarmes, en utilisant par exemple une méthode de seuillage adaptatif comme la méthode *a contrario* de [4]. L'utilisateur est toutefois toujours déçu si le système ne lui retourne aucun résultat.

Nous proposons dans ce chapitre une solution alternative, qui consiste à suggérer directement à l'utilisateur, les requêtes retournant des résultats pertinents.

Une première approche de suggestion de requêtes visuelles avait déjà été proposée par Zha et al. dans [175]. Leur méthode est une extension des méthodes de suggestion de requêtes textuelles, qui sont maintenant fréquemment utilisées par les moteurs de recherche. L'argument des auteurs était que les suggestions basées sur le texte ne suffisent pas toujours à prédire précisément la requête de

l'utilisateur. En ajoutant un ensemble d'images représentatives aux suggestions textuelles, l'utilisateur peut alors spécifier plus précisément son intention. Leur méthode est principalement basée sur un calcul de similarités visuelles globales, en complément de méthodes de recherche textuelle.

Notre méthode diffère en deux points principaux :

– Nos suggestions sont purement visuelles (bien que l'on pourrait employer les informations textuelles pour affiner nos suggestions) ;
– Les suggestions représentent des instances d'objet, et non pas des images entières ou des concepts visuels (catégories), et sont de ce fait plus précises ;

9.1 Principe de la suggestion de requêtes visuelles

Dans [176], nous avons proposé, en collaboration avec Amel Hamzaoui, de suggérer des requêtes uniquement si elles contiennent des résultats dans la collection d'images. Ceci requiert d'abord de découvrir des *clusters* d'objets dans la collection, et de sélectionner ensuite les plus pertinents en fonction du nombre d'occurrences et de l'intention de l'utilisateur. Le *clustering* est effectué par un nouvel algorithme de *clustering* par voisins-partagés (*shared-neighbours*) de graphe biparti, utilisé pour rassembler les instances d'objets découvertes par RANSAS. L'algorithme MCL utilisé dans le chapitre précédent n'est en effet pas directement applicable lorsque les images peuvent contenir plusieurs instances de plusieurs objets visuels.

Plus concrètement, le concept que nous introduisons dans ce chapitre est le suivant : plutôt que de laisser l'utilisateur sélectionner une région à rechercher, nous proposons que le système se charge de lui suggérer automatiquement les régions contenant des instances d'objets $\{c, f\}$-fréquents. Lorsque un utilisateur clique sur une région suggérée, le système retourne les images contenant les autres instances de l'objet appartenant au *cluster* découvert par notre méthode. Du point de vue utilisateur, ce concept de suggestion est très différent de la méthode standard de sélection libre d'une requête. On peut ainsi surnommer notre concept de suggestion d'objet visuel par l'expression « lien visuel » (ou même hyperlien visuel, par analogie avec les hyperliens textuels).

Cette approche a également un intérêt pour le système lui-même qui est moins surchargé par des requêtes « inutiles », et qui peut répondre beaucoup plus rapidement puisque les résultats sont pré-calculés, et stockés en mémoire ou sur disque.

Ces hyperliens visuels peuvent être employés dans de nombreux scénarios. Nous nous focaliserons dans la section 9.3 page 123 sur deux d'entre eux : la « suggestion d'objets visuels au survol de la souris », et la « suggestion d'objets visuels comme facétisation d'une requête textuelle ».

9.2 Algorithme de *clustering* de graphe biparti

Comme le montre la figure 9.1 page suivante, certaines images peuvent présenter plusieurs instances différentes n'appartenant pas au même objet. L'utilisation d'un algorithme de *clustering* de graphe dont les nœuds sont les images n'est donc pas adapté, il est ainsi préférable que les nœuds soient les instances découvertes. Ceci justifie donc l'emploi d'un algorithme de *clustering* de graphe biparti, comme celui proposé par Hamzaoui [177].

Bien que les germes (mots visuels géométriquement cohérents) découverts par RANSAS correspondent bien à des motifs visuels fréquents dans la collection, ils ne peuvent pas toujours être considérés comme des objets complets pour plusieurs raisons :

- Par construction, un germe ne couvre (spatialement) qu'une sous-partie d'un objet ;
- Le rappel n'étant pas parfait, un germe correspond à un sous-ensemble des instances de l'objet découvert ;
- Plus un objet est fréquent dans la collection, plus on découvre de germes ;

La construction précise et complète d'un modèle d'objet nécessite de regrouper tous les germes appartenant à un même objet. Ceci ne peut être effectué à partir du contenu visuel des germes, étant donné que deux germes avec des contenus différents peuvent très bien être deux sous-parties d'un même objet. Une alternative à cela est de regrouper les germes fortement corrélés entre eux (c'est-à-dire apparaissant des les mêmes images), ce qui peut être formulé comme un problème de *clustering* biparti. La figure 9.1 page suivante illustre la méthode proposée pour regrouper les germes représentant le même objet.

Soit $\boldsymbol{G} = (\boldsymbol{X}; \boldsymbol{E}) = (\boldsymbol{I}, \boldsymbol{S}; \boldsymbol{E})$ le graphe biparti résultant de la découverte d'objets visuels, avec :

$$\boldsymbol{I} = \{I_i\}_{i \in [1, N_I]}$$

l'ensemble des vertex représentant les images de la collection, et :

$$\boldsymbol{S} = \{S_j\}_{j \in [1, |S|]}$$

l'ensemble des germes (ou mots visuels géométriquement cohérents) découverts par RANSAS, $\boldsymbol{X} = \boldsymbol{I} \cup \boldsymbol{S}$ et $\boldsymbol{I} \cap \boldsymbol{S} = \emptyset$

Chaque arête orientée $e_{i,j} \in \boldsymbol{E}$ a un point de départ dans $\boldsymbol{S}$ et un point d'arrivée dans $\boldsymbol{I}$, et un poids $w_{i,j}$ correspondant au score d'appariement fourni par RANSAS ($w_{i,j} = 0$ signifie qu'aucune arête ne connecte le germe S_j à l'image I_i).

L'avantage de cette représentation bipartie est qu'elle permet de formuler notre objectif de *clustering* de germes comme un problème de *co-clustering*. On vise en effet à trouver des *clusters* d'objet $\boldsymbol{O}_n = (\boldsymbol{S}^n, \boldsymbol{I}^n)$ avec $\boldsymbol{S}^n \subset \boldsymbol{S}$ étant le sous-ensemble des germes modélisant un objet donné, et $\boldsymbol{I}^n \subset \boldsymbol{I}$ étant le sous-ensemble des images contenant des instances d'objet. Un *cluster* d'objet idéal est celui dont les germes apparaissent dans les mêmes images. Il est important de remarquer l'avantage sur les méthodes de découverte d'objets créant des graphes d'appariement entre images [178, 179, 180, 148] :

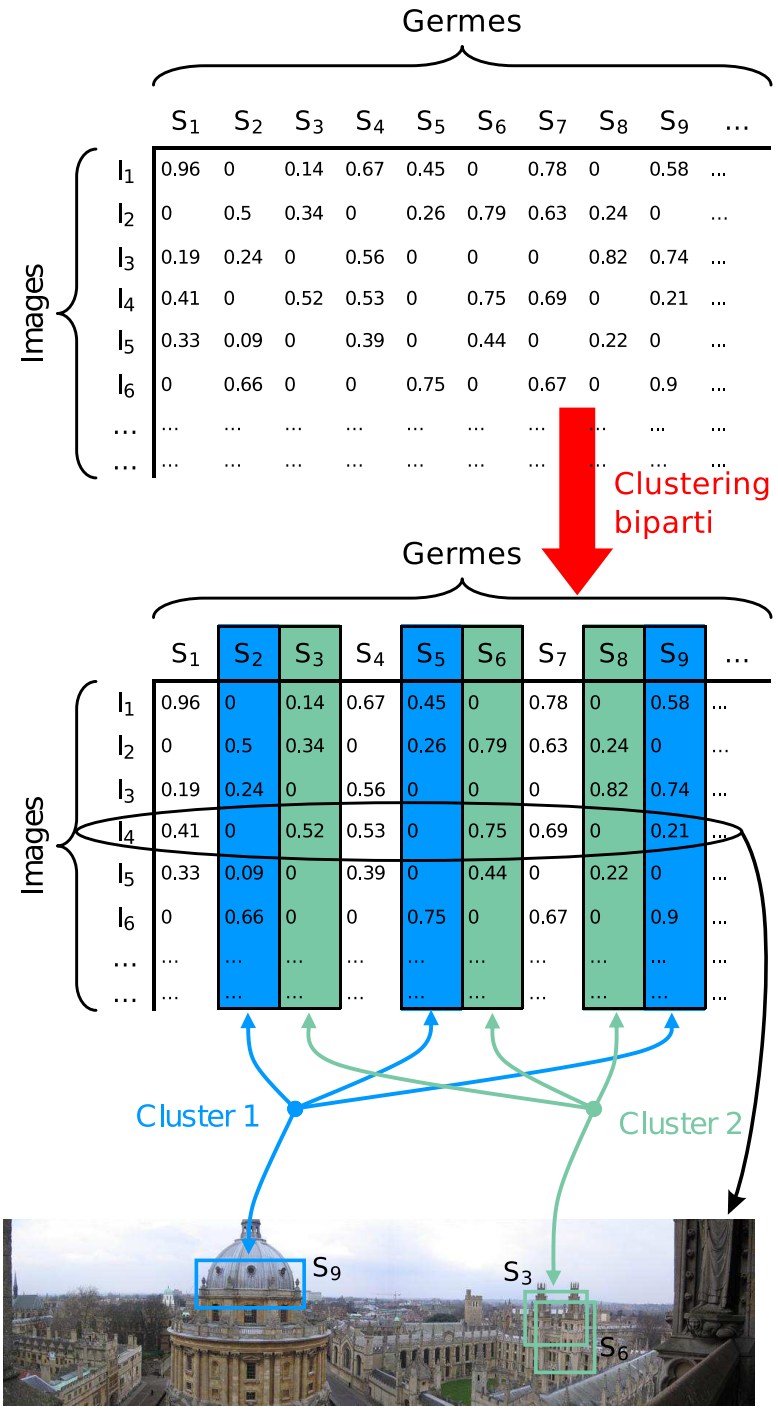


FIGURE 9.1 – Illustration de la méthode proposée pour suggérer des requêtes visuelles dans l'image I_4. Les germes S_2, S_5 et S_9 appartiennent à un premier *cluster* (en bleu), tandis que S_3, S_6 et S_8 appartiennent à un second (en vert).

une image donnée peut être affectée à plusieurs *clusters* d'objets (lorsqu'elle contient des instances de différents objets). De plus, chaque *cluster* d'objet est composé d'un ensemble unique de germes associés

aux régions appariées et localisées spatialement.

Résoudre le problème de *clustering* biparti n'est pas un problème trivial. Plusieurs travaux ont proposé des techniques spectrales appliquées au *clustering* de documents textuels [181, 182]. Ces méthodes sont utiles pour partitionner des graphes bipartis en un nombre prédéfini de *clusters* équilibrés, mais ne sont pas adaptées à notre problème. Le nombre d'objets à découvrir, tout comme le nombre de germes à regrouper peut en effet être très variable. Les résultats sont de plus très sensibles aux paramètres utilisés. C'est pourquoi nous utiliserons donc l'algorithme de *clustering* de graphe biparti proposé par Hamzaoui [177], et inspiré des méthodes de *clustering* par plus proches voisins partagés (*Shared Nearest Neighbours*). Le principe de la méthode de Hamzaoui [177] est de regrouper les éléments non pas grâce à leur similarité, mais grâce au degré de ressemblance de leurs voisinages respectifs.

9.3 Scénarios de suggestion de requêtes visuelles

Pour chacun des deux scénarios de suggestion de requêtes visuelles décrits plus haut, nous répondons aux questions suivantes : Que suggère-t-on ? Comment affichons nous les suggestions ? Que doit-on retourner lorsque l'utilisateur clique sur un objet suggéré ?

Suggestion d'objets visuels au survol de la souris :

Pour chaque image $I_j \in \boldsymbol{I}$, on suggère des requêtes dont le nombre est égal au nombre de *clusters* ayant I_j dans leur ensemble d'images duales. Chaque suggestion est représentée par une fenêtre rectangulaire calculée à partir de l'ensemble de toutes les régions appartenant au *cluster* et à l'image concernés. Cette fenêtre rectangulaire est obtenue en conservant les zones couvertes par au minimum deux régions détectées, comme cela est illustré par la figure 9.2 page suivante. Ceci permet d'être plus résistant aux *outliers* qui agrandissent par erreur les rectangles englobant des germes découverts. Dans les zones non concernées par une suggestion, l'image est légèrement opacifiée afin de guider l'utilisateur vers les zones contenant des instances d'objet découvertes. Au survol de la souris sur une suggestion, le contour de la fenêtre est coloré en vert. Lorsqu'un utilisateur clique sur l'une des requêtes suggérées, on retourne une liste d'images triées selon leur intersection avec l'objet sélectionné. L'intersection est définie comme le nombre de germes correspondant à la fois au *cluster* et à l'image. Par exemple, dans la figure 9.1 page ci-contre, l'intersection de l'image I_2 avec le *cluster* 2 est égale à 3, tandis que l'intersection de l'image I_4 avec le même *cluster* est égale à 2. La figure 9.4 page 126 montre une capture d'écran de ce scénario.

Suggestion d'objets visuels comme facétisation d'une requête textuelle :

Pour ce scénario, on suppose qu'un moteur de recherche textuel a retourné un sous-ensemble d'images $\boldsymbol{I}_x \subset \boldsymbol{I}$. On sélectionne ensuite comme requêtes suggérées les M meilleurs *clusters* de la base ayant la plus grande intersection entre les images (dans leur représentation duale) et la liste de

FIGURE 9.2 – Génération des fenêtres rectangulaires pour la suggestion d'objets visuels au survol de la souris

résultats textuels (c'est-à-dire les *clusters* représentant les objets les plus fréquents dans la liste de résultats). Chaque requête suggérée est affichée au dessus de l'interface de recherche par une miniature représentative de l'objet. Cette miniature est construite en cherchant d'abord l'image ayant la plus grande intersection avec le *cluster*, c'est-à-dire en comptant le nombre de germes. Une fois la meilleure image sélectionnée, on recadre l'objet d'intérêt dans cette image avec la même procédure que celle décrite dans le scénario précédent. La figure 9.3 page ci-contre illustre ce processus de construction des miniatures sur trois *clusters* de la base Oxford Buildings. Lorsque l'utilisateur clique sur l'une de ces suggestions, on retourne, comme précédemment, une liste d'images triées selon leur intersection avec l'objet.

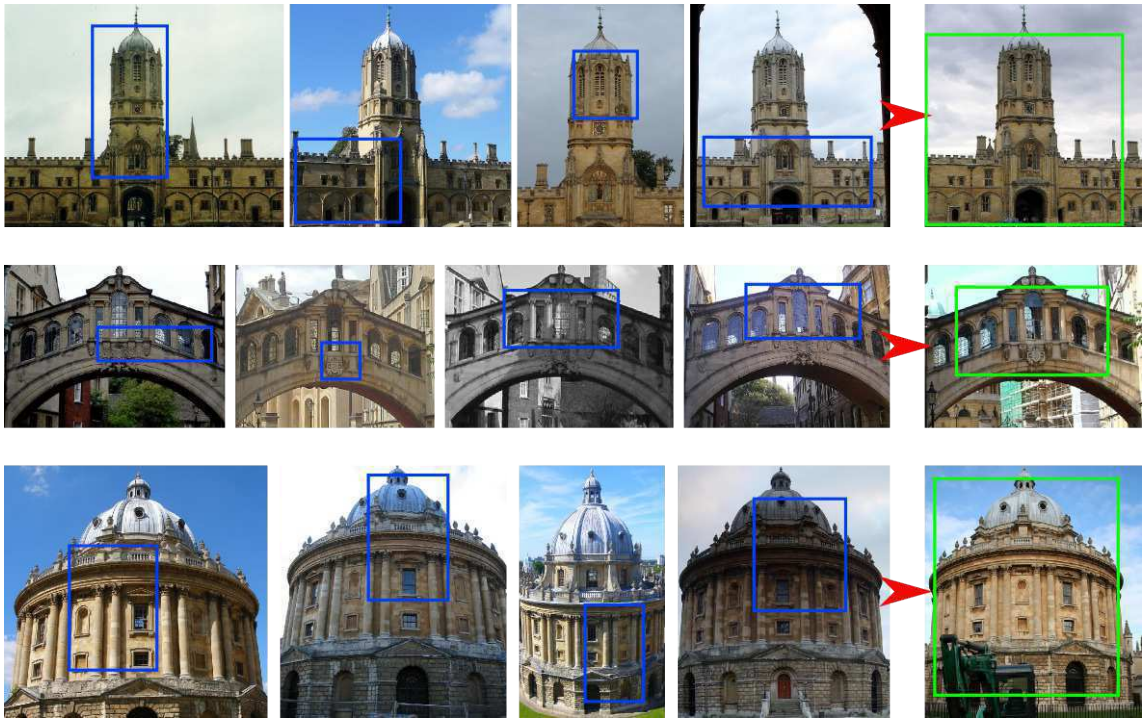


FIGURE 9.3 – Génération des miniatures par recadrage pour la suggestion d'objets visuels en complément d'une requête textuelle

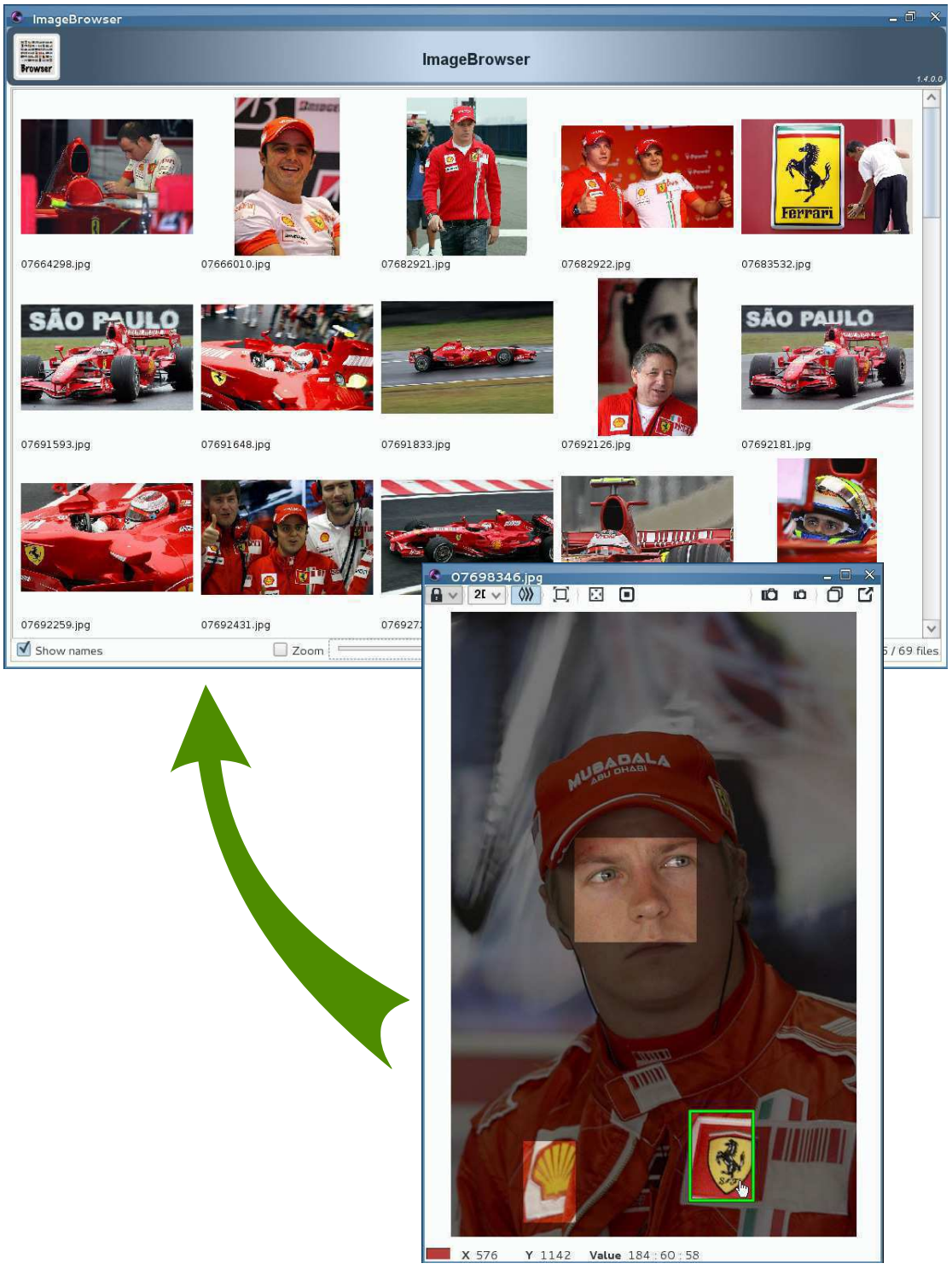


FIGURE 9.4 – Capture d'écran de l'interface de suggestion d'objets visuels au survol de la souris

Chapitre 10

Conclusion

Les travaux de cette thèse traitent du problème de la découverte et de l'exploitation d'objets visuels fréquents dans des collections multimédias.

Dans cette conclusion, nous commencerons par synthétiser les contributions. Nous dresserons ensuite un bilan des analyses menées à partir des expérimentations, et enfin nous réfléchirons aux perspectives envisageables.

10.1 Synthèse des contributions

La première contribution de cette thèse a été de fournir un formalisme aux problèmes de découverte et de fouille d'instances d'objets visuels fréquents. Ces problèmes n'avaient en effet jamais été clairement définis dans la littérature. Cette modélisation nous a permis entre autres choses de mettre en évidence le lien étroit qui existe entre la taille des objets à découvrir et la complexité du problème à traiter.

La deuxième contribution présentée dans cette thèse est l'algorithme RANSAS (*RANdom Sample And Search*), qui est une méthode générique de résolution des problèmes de fouille et découverte. RANSAS repose d'une part sur un processus itératif d'échantillonnage d'objets candidats et d'autre part sur une méthode efficace d'appariement d'objets rigides à large échelle. L'idée est de considérer l'étape de recherche d'instances proprement dite comme une simple boite noire à laquelle il s'agit de soumettre des régions d'images ayant une probabilité élevée d'appartenir à un objet fréquent de la base.

Une première approche étudiée dans la thèse consiste à simplement considérer que toutes les régions d'images de la base sont équiprobables, avec comme idée conductrice que les objets les plus instanciés sont ceux qui auront la couverture spatiale la plus grande et donc la probabilité la plus élevée d'être échantillonnés. En généralisant cette notion de couverture à celle plus générique de couverture probabiliste, il est alors possible de modéliser la complexité de notre méthode pour tout score de vraisemblance donné en entrée, et de montrer ainsi l'importance de cette étape.

La troisième contribution principale de la thèse s'attache à construire un score de vraisemblance

s'approchant au mieux de la distribution parfaite, tout en restant *scalable* et efficace. Cette dernière repose sur une approche originale de hachage à deux niveaux, permettant de générer efficacement un ensemble d'appariements visuels dans un premier temps, et d'évaluer ensuite leur pertinence en fonction de contraintes géométriques faibles.

Afin d'évaluer les problèmes de découverte et fouille d'objets $\{c, f\}$-fréquents, nous avons proposé une nouvelle collection d'images disposant d'une vérité terrain adaptée.

Enfin, nous avons également présenté un nouveau concept de suggestion de requêtes visuelles permettant d'assurer à l'utilisateur que ses requêtes auront toujours des résultats, ainsi que d'améliorer les résultats d'une recherche d'images par le texte en suggérant des objets identifiés comme fréquents au sein de la collection.

10.2 Analyse et bilan

Cette thèse a été effectuée dans le cadre d'un contrat cifre[1] à l'Ina, et l'accent a donc été porté sur l'optimisation de nos algorithmes afin qu'ils trouvent leurs applications dans les très grands volumes d'images et de vidéos détenus par l'Ina. Nous avons effectué des expérimentations sur une collection de petite taille (FlickrBelgaLogos : 10 000 images) afin d'évaluer quantitativement nos performances, et de permettre la comparaison avec de futurs travaux sur le sujet. Mais nous avons également montré que notre système pouvait s'adapter à des collections de plus de 2 000 heures de télévision (sur une seule machine).

En ce qui concerne l'algorithme de découverte RANSAS, nous avons montré l'intérêt de la mise à jour de la fonction de probabilité de masse à chaque itération. Nous avons aussi évalué l'accélération (en termes d'objets et d'instances découverts en fonction du nombre d'itérations) obtenue avec nos scores de vraisemblance par rapport à l'échantillonnage uniforme. Nous avons ainsi montré que l'utilisation de scores de vraisemblance adaptés permettait de réduire le nombre d'itérations de RANSAS d'un facteur pouvant aller jusqu'à 32. Au total, il faut environ une quinzaine d'heures pour découvrir 90% des objets de la vérité terrain, et moins de neuf heures pour découvrir 50% des instances dans la base FlickrBelgaLogos.

Pour ce qui est du calcul du score de vraisemblance, nous avons observé que chacun des quatre attributs de géométrie faible apportait une information utile, et que l'utilisation de ces quatre attributs complétait efficacement les phases de hachage et de filtrage des descripteurs visuels. Nous avons constaté que le temps de calcul de ce score restait assez faible (moins d'un demi-heure pour 10 000 images et 50 millions de descripteurs, avec 16 tables de hachage visuel). Les évaluations ont également mis en évidence que l'utilisation de plus de 16 tables de hachage visuel était peu pertinente (faible amélioration de la précision et du rappel pour un coût calculatoire important). En comparant notre

1. http ://www.anrt.asso.fr/fr/espace_cifre/accueil.jsp

approche à celle basée sur la méthode Geometric min-Hashing, considérée comme l'actuel état de l'art, nous avons montré que notre méthode était en mesure d'atteindre de bien meilleures performances.

10.3 Perspectives

Il serait intéressant d'étudier la taille et la forme des requêtes issues de la phase d'échantillonnage de RANSAS. Plus précisément, il serait utile de mesurer l'incidence que ce choix de taille et de forme peut engendrer sur les performances de RANSAS, et en particulier pour différentes tailles d'instances. En effet, des requêtes de grande taille sont adaptées aux objets ayant une forte « couverture moyenne », et inversement. Un choix inadapté a pour conséquence une sur/sous-segmentation des instances découvertes. Ce problème peut-être plus ou moins résolu par l'étape de *clustering* effectuée a posteriori, mais l'introduction de nouvelles méthodes pour sélectionner automatiquement la couverture de chaque requête pourrait améliorer nos résultats.

Dans cette thèse, nous avons adressé le problème de la découverte et de la fouille d'objets $\{c, f\}$-fréquents. Nous avons montré que notre système permettait de découvrir et de fouiller des objets visuels fréquents efficacement dans de grandes collections multimédias. L'objectif serait maintenant de pouvoir paramétrer précisément notre système en fonction de deux seuls paramètres : la couverture c_0 et la fréquence f_0 minimales. La réalisation de cet objectif supposerait que nous soyions capables de mesurer la couverture et la fréquence réelles d'un objet visuel dans une collection, ce qui est loin d'être trivial, surtout si l'on considère le fait que la plupart des méthodes que nous utilisons (ainsi que celles de l'état de l'art) sont des méthodes approximatives et basées sur de l'aléatoire.

L'une des évolutions envisageables de notre algorithme de calcul de score de vraisemblance basé sur la géométrie faible, serait d'ajouter de nouveaux attributs aux vecteurs de géométrie faible. On pourrait ainsi étudier l'intérêt de la translation, de la couleur, ou de toutes autres informations contextuelles, comme le type de média par exemple, afin de créer un score de vraisemblance filtrant directement les objets transmédias.

Afin de diminuer l'éclatement des objets visuels découverts en plusieurs sous-groupes, nous avons proposé d'effectuer un *clustering* des instances. Cet éclatement aurait pu être atténué en utilisant une extension de requêtes lors de la phase de recherche d'objets visuels de RANSAS. On a cependant vu que l'extension de requêtes était en règle générale un processus très coûteux, et donc inadapté. Une alternative à ces deux approches serait de créer des modèles d'objets visuels au fur et à mesure de la découverte. Cette solution aurait l'avantage de permettre d'effectuer des requêtes multiples à partir des modèles.

Comme toutes les méthodes de recherche ou de découverte d'objets visuels, le volume des collections

que nous traitons est principalement limité par les capacités des ordinateurs, et en particulier par l'espace mémoire. Il serait donc intéressant d'étudier la distribution de nos algorithmes sur de multiples nœuds de calcul. Une grande partie de nos algorithmes étant basés sur du hachage, cet objectif parait réaliste. On pourrait alors s'attaquer à l'échelle du web et exploiter les liens hypervisuels dans les moteurs de recherche.

Les applications proposées dans cette thèse sont actuellement à l'état de prototypes, mais sont d'ores et déjà en phase de test par des journalistes et des sociologues dans le cadre du projet OTMedia. Nous projetons ainsi de réaliser grâce à eux des évaluations qualitatives de nos applications sur des cas d'usages réels et concrets.

Bibliographie

[1] J. Law-To et al. « Video copy detection : a comparative study ». Dans : *Proceedings of the 6th ACM international conference on Image and video retrieval*. ACM. 2007, p. 371–378.

[2] M.S. Lew et al. « Content-based multimedia information retrieval : State of the art and challenges ». Dans : *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)* 2.1 (2006), p. 1–19.

[3] J. Sivic et A. Zisserman. « Video Google : A text retrieval approach to object matching in videos ». Dans : *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. Ieee. 2003, p. 1470–1477.

[4] Alexis Joly et Olivier Buisson. « Logo retrieval with a contrario visual query expansion ». Dans : *Proceedings of the seventeen ACM international conference on Multimedia*. MM '09. Beijing, China : ACM, 2009, p. 581–584. isbn : 978-1-60558-608-3. doi : http://doi.acm.org/10.1145/1631272.1631361.

[5] F. Perronnin, J. Sánchez et T. Mensink. « Improving the fisher kernel for large-scale image classification ». Dans : *Computer Vision--ECCV 2010* (2010), p. 143–156.

[6] M.J. Swain et D.H. Ballard. « Color indexing ». Dans : *International journal of computer vision* 7.1 (1991), p. 11–32.

[7] M. Flickner et al. « Query by image and video content : The QBIC system ». Dans : *Computer* 28.9 (1995), p. 23–32.

[8] T. Gevers et A.W.M. Smeulders. « Pictoseek : Combining color and shape invariant features for image retrieval ». Dans : *Image Processing, IEEE Transactions on* 9.1 (2000), p. 102–119.

[9] J.R. Smith et S.F. Chang. « VisualSEEk : a fully automated content-based image query system ». Dans : *Proceedings of the fourth ACM international conference on Multimedia*. ACM. 1997, p. 87–98.

[10] G.D. Finlayson. « Color in perspective ». Dans : *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 18.10 (1996), p. 1034–1038.

[11] J. Huang et al. « Spatial color indexing and applications ». Dans : *International Journal of Computer Vision* 35.3 (1999), p. 245–268.

[12] J.Z. Wang et al. « Content-based image indexing and searching using Daubechies' wavelets ». Dans : *International Journal on Digital Libraries* 1.4 (1998), p. 311–328.

[13] P. Salembier, T. Sikora et BS Manjunath. *Introduction to MPEG-7 : multimedia content description interface*. John Wiley & Sons, Inc., 2002.

[14] B.S. Manjunath et al. « Color and texture descriptors ». Dans : *Circuits and Systems for Video Technology, IEEE Transactions on* 11.6 (2001), p. 703–715.

[15] A. Oliva et A. Torralba. « Modeling the shape of the scene : A holistic representation of the spatial envelope ». Dans : *International Journal of Computer Vision* 42.3 (2001), p. 145–175.

[16] S. Ullman, M. Vidal-Naquet, E. Sali et al. « Visual features of intermediate complexity and their use in classification ». Dans : *Nature neuroscience* 5.7 (2002), p. 682–687.

[17] L. Fei-Fei et P. Perona. « A bayesian hierarchical model for learning natural scene categories ». Dans : *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. T. 2. IEEE. 2005, p. 524–531.

[18] C. Harris et M. Stephens. « A combined corner and edge detector ». Dans : *Alvey vision conference*. T. 15. Manchester, UK. 1988, p. 50.

[19] J. Shi et C. Tomasi. « Good features to track ». Dans : *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*. IEEE. 1994, p. 593–600.

[20] S. M. Smith et J. M. Brady. « SUSAN - A New Approach to Low Level Image Processing ». Dans : *International Journal of Computer Vision* 23 (1995), p. 45–78.

[21] E. Rosten et T. Drummond. « Machine learning for high-speed corner detection ». Dans : *Computer Vision--ECCV 2006* (2006), p. 430–443.

[22] T. Lindeberg. « Feature detection with automatic scale selection ». Dans : *International journal of computer vision* 30.2 (1998), p. 79–116.

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

[24] H. Bay et al. « Speeded-up robust features (SURF) ». Dans : *Computer Vision and Image Understanding* 110.3 (2008), p. 346–359.

[25] J. Matas et al. « Robust wide-baseline stereo from maximally stable extremal regions ». Dans : *Image and Vision Computing* 22.10 (2004), p. 761–767.

[26] K. Mikolajczyk et C. Schmid. « An affine invariant interest point detector ». Dans : *Computer Vision—ECCV 2002* (2002), p. 128–142.

[27] K. Mikolajczyk et C. Schmid. « Scale & affine invariant interest point detectors ». Dans : *International journal of computer vision* 60.1 (2004), p. 63–86.

[28] M. Perd'och, O. Chum et J. Matas. « Efficient representation of local geometry for large scale object retrieval ». Dans : *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. 2009, p. 9–16.

[29] A.E. Johnson et M. Hebert. « Recognizing objects by matching oriented points ». Dans : *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*. IEEE. 1997, p. 684–689.

[30] S. Lazebnik, C. Schmid et J. Ponce. « A sparse texture representation using affine-invariant regions ». Dans : *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*. T. 2. IEEE. 2003, p. II–319.

[31] T. Tuytelaars et L. Van Gool. « Wide baseline stereo matching based on local, affinely invariant regions ». Dans : *british Machine vision conference*. 2000, p. 412–425.

[32] R. Zabih et J. Woodfill. « Non-parametric local transforms for computing visual correspondence ». Dans : *Computer Vision ?ECCV'94* (1994), p. 151–158.

[33] S. Belongie, J. Malik et J. Puzicha. « Shape matching and object recognition using shape contexts ». Dans : *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 24.4 (2002), p. 509–522.

[34] Y. Ke et R. Sukthankar. « PCA-SIFT : A more distinctive representation for local image descriptors ». Dans : *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*. T. 2. Ieee. 2004, p. II–506.

[35] K. Mikolajczyk et al. « A comparison of affine region detectors ». Dans : *International journal of computer vision* 65.1 (2005), p. 43–72.

[36] J.M. MOREL et G. YU. « ASIFT : A new framework for fully affine invariant image comparison ». Dans : *SIAM Journal on Imaging Sciences* 2.2 (2009), p. 438–469.

[37] G. YU et J.M. MOREL. « ASIFT : An Algorithm for Fully Affine Invariant Comparison ». Dans : *Image Processing On Line* (2011).

[38] J.J. KOENDERINK et AJ VAN DOORN. « Representation of local geometry in the visual system ». Dans : *Biological cybernetics* 55.6 (1987), p. 367–375.

[39] LMJ FLORACK et al. « General intensity transformations and differential invariants ». Dans : *Journal of Mathematical Imaging and Vision* 4.2 (1994), p. 171–187.

[40] L. VAN GOOL, T. MOONS et D. UNGUREANU. « Affine/photometric invariants for planar intensity patterns ». Dans : *Computer Vision ?ECCV'96* (1996), p. 642–651.

[41] W.T. FREEMAN et E.H. ADELSON. « The design and use of steerable filters ». Dans : *IEEE Transactions on Pattern analysis and machine intelligence* 13.9 (1991), p. 891–906.

[42] A. JOLY. « New local descriptors based on dissociated dipoles ». Dans : *Proceedings of the 6th ACM international conference on Image and video retrieval*. Sous la dir. de Nicu SEBE et Marcel WORRING. Amsterdam, The Netherlands : ACM, juil. 2007, p. 573–580. ISBN : 978-1-59593-733-9.

[43] M. CALONDER et al. « Brief : Binary robust independent elementary features ». Dans : Springer, 2010, p. 778–792.

[44] S.E. GRIGORESCU, N. PETKOV et P. KRUIZINGA. « Comparison of texture features based on Gabor filters ». Dans : *Image Processing, IEEE Transactions on* 11.10 (2002), p. 1160–1167.

[45] E. KHVEDCHENYA. *Feature descriptor comparison report*. Août 2011. URL : http://computer-vision-talks.com/2011/08/feature-descriptor-comparison-report.

[46] T. DESELAERS, D. KEYSERS et H. NEY. « Features for image retrieval : A quantitative comparison ». Dans : *Pattern Recognition* (2004), p. 228–236.

[47] K. GRAUMAN et T. DARRELL. « The pyramid match kernel : Discriminative classification with sets of image features ». Dans : *Proceedings of the 10th International Conference on Computer Vision,ICCV*. T. 2. Beijing, China : IEEE Computer Society, oct. 2005, p. 1458–1465. ISBN : 0-7695-2334-X.

[48] S. LAZEBNIK, C. SCHMID et J. PONCE. « Beyond bags of features : Spatial pyramid matching for recognizing natural scene categories ». Dans : *Conference on Computer Vision and Pattern Recognition*. Sous la dir. d'IEEE Computer SOCIETY. T. 2. New York, NY, USA : IEEE Computer Society, juin 2006, p. 2169–2178. ISBN : 0-7695-2597-0.

[49] T.S. JAAKKOLA et D. HAUSSLER. « Exploiting generative models in discriminative classifiers ». Dans : *Advances in neural information processing systems* (1999), p. 487–493.

[50] F. PERRONNIN et C. DANCE. « Fisher kernels on visual vocabularies for image categorization ». Dans : *Conference on Computer Vision and Pattern Recognition*. Sous la dir. d'IEEE Computer SOCIETY. Minneapolis, Minnesota, USA : IEEE Computer Society, juin 2007, p. 1–8.

[51] M. BRESSAN, C. CIFARELLI et F. PERRONNIN. « An analysis of the relationship between painters based on their work ». Dans : *Proceedings of the International Conference on Image Processing, ICIP*. San Diego, California, USA : IEEE, oct. 2008, p. 113–116.

[52] L. MARCHESOTTI, C. CIFARELLI et G. CSURKA. « A framework for visual saliency detection with applications to image thumbnailing ». Dans : *Proceedings of the 12th International Conference on Computer Vision, ICCV*. Kyoto, Japan : IEEE, oct. 2009, p. 2232–2239.

[53] F. PERRONNIN et al. « Large-scale image retrieval with compressed fisher vectors ». Dans : *Conference on Computer Vision and Pattern Recognition*. Sous la dir. d'IEEE Computer SOCIETY. San Francisco, CA, USA : IEEE, juin 2010, p. 3384–3391.

[54] D.A. KEIM et A. HINNEBURG. « Clustering techniques for large data sets ». Dans : *Tutorial notes of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. Sous la dir. d'ACM. ACM. San Diego, CA, USA, 1999, p. 141–181.

[55] R.M. GRAY et D.L. NEUHOFF. « Quantization ». Dans : *Information Theory, IEEE Transactions on* 44.6 (1998), p. 2325–2383.

[56] JH CONWAY et NJA SLOANE. « On the enumeration of lattices of determinant one ». Dans : *Journal of Number Theory* 15.1 (1982), p. 83–94.

[57] T. TUYTELAARS et C. SCHMID. « Vector quantizing feature space with a regular lattice ». Dans : *Proceedings of the 11th International Conference on Computer Vision, ICCV*. Rio de Janeiro, Brazil : IEEE, oct. 2007, p. 1–8.

[58] H. SAGAN. *Space-filling curves*. T. 2. Springer-Verlag New York, 1994.

[59] A. JOLY. « Recherche par similarité statistique dans une grande base de signatures locales pour l'identification rapide d'extraits vidéo ». Thèse de doct. Université de la Rochelle, 2005.

[60] S. POULLOT, O. BUISSON et M. CRUCIANU. « Z-grid-based probabilistic retrieval for scaling up content-based copy detection ». Dans : *Proceedings of the 6th ACM international conference on Image and video retrieval*. ACM. 2007, p. 348–355.

[61] P. INDYK et R. MOTWANI. « Approximate Nearest Neighbors : Towards Removing the Curse of Dimensionality ». Dans : *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. Sous la dir. d'ACM. Dallas, TX, USA, 1998, p. 604–613.

[62] A. ANDONI et P. INDYK. « Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions ». Dans : *Foundations of Computer Science, FOCS'06. 47th Annual IEEE Symposium on*. Sous la dir. d'IEEE. Ieee. 2006, p. 459–468.

[63] M. DATAR et P. INDYK. « Locality-sensitive hashing scheme based on p-stable distributions ». Dans : *SCG*. ACM Press, 2004, p. 253–262.

[64] K. TERASAWA et Y. TANAKA. « Spherical lsh for approximate nearest neighbor search on unit hypersphere ». Dans : *Algorithms and Data Structures* (2007), p. 27–38.

[65] M. RAGINSKY et S. LAZEBNIK. « Locality-sensitive binary codes from shift-invariant kernels ». Dans : *The Neural Information Processing Systems* 22 (2009).

[66] A. BRODER. « On the Resemblance and Containment of Documents ». Dans : *SEQUENCES '97 : Proceedings of the Compression and Complexity of Sequences 1997*. Washington, DC, USA : IEEE Computer Society, 1997, p. 21.

[67] A. GUTTMAN. *R-trees : a dynamic index structure for spatial searching*. T. 14. 2. ACM, 1984.

[68] Y. MANOPOULOS, A. NANOPOULOS et E. TOUSIDOU. « Advanced signature indexing for multimedia and web applications (Series on advances in database systems, Vol. 27) ». Dans : *Recherche* 67 (2003), p. 02.

[69] N. BECKMANN et al. *The R*-tree : an efficient and robust access method for points and rectangles*. T. 19. 2. ACM, 1990.

[70] D.A. WHITE et R. JAIN. « Similarity indexing with the SS-tree ». Dans : *Data Engineering, 1996. Proceedings of the Twelfth International Conference on*. IEEE. 1996, p. 516–523.

[71] N. KATAYAMA et S. SATOH. « The SR-tree : An index structure for high-dimensional nearest neighbor queries ». Dans : *ACM SIGMOD Record*. T. 26. 2. ACM. 1997, p. 369–380.

[72] N. BOUTELDJA, V. GOUET-BRUNET et M. SCHOLL. « Evaluation of strategies for multiple sphere queries with local image descriptors ». Dans : *Proceedings of SPIE*. T. 6073. 2006, p. 87–98.

[73] S. BERCHTOLD, D.A. KEIM et H.P. KRIEGEL. *The X-tree : An index structure for high-dimensional data*. Bibliothek der Universität Konstanz, 1996.

[74] J.L. BENTLEY. « Multidimensional binary search trees used for associative searching ». Dans : *Communications of the ACM* 18.9 (1975), p. 509–517.

[75] J.L. BENTLEY et J.H. FRIEDMAN. « Data structures for range searching ». Dans : *ACM Computing Surveys (CSUR)* 11.4 (1979), p. 397–409.

[76] J.T. ROBINSON. « The KDB-tree : a search structure for large multidimensional dynamic indexes ». Dans : *Proceedings of the 1981 ACM SIGMOD international conference on Management of data*. ACM. 1981, p. 10–18.

[77] A. HENRICH. « The LSDh-tree : An access structure for feature vectors ». Dans : *Data Engineering, 1998. Proceedings., 14th International Conference on*. IEEE. 1998, p. 362–369.

[78] Y. AMIT et D. GEMAN. « Shape quantization and recognition with randomized trees ». Dans : *Neural computation* 9.7 (1997), p. 1545–1588.

[79] V. LEPETIT, P. LAGGER et P. FUA. « Randomized Trees for Real-Time Keypoint Recognition ». Dans : *Conference on Computer Vision and Pattern Recognition*. Sous la dir. d'IEEE Computer SOCIETY. CVPR '05. San Diego, CA, USA : IEEE Computer Society, juin 2005, p. 775–781. ISBN : 0-7695-2372-2. DOI : 10.1109/CVPR.2005.288.

[80] C. SILPA-ANAN et R. HARTLEY. « Localisation using an image-map ». Dans : *Proceedings of the Australian Conference on Robotics and Automation*. Citeseer. 2004.

[81] D. NISTER et H. STEWENIUS. « Scalable recognition with a vocabulary tree ». Dans : *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. T. 2. Ieee. 2006, p. 2161–2168.

[82] J. PHILBIN. « Scalable Object Retrieval in Very Large Image Collections ». Thèse de doct. University of Oxford, 2010. URL : http://marcade.robots.ox.ac.uk:8080/~vgg/publications/2010/Philbin10c.

[83] T. ZHANG, R. RAMAKRISHNAN et M. LIVNY. « BIRCH : an efficient data clustering method for very large databases ». Dans : *ACM SIGMOD Record*. T. 25. 2. ACM. 1996, p. 103–114.

[84] S.A. BERRANI. « Recherche approximative de plus proches voisins avec contrôle probabiliste de la précision ; application à la recherche d'images par le contenu ». Dans : (2004).

[85] R. SALAKHUTDINOV, A. MNIH et G. HINTON. « Restricted Boltzmann machines for collaborative filtering ». Dans : *Proceedings of the 24th international conference on Machine learning*. ACM. 2007, p. 791–798.

[86] Y. WEISS, A. TORRALBA et R. FERGUS. « Spectral hashing ». Dans : *Neural Information Processing Systems*. Vancouver, B.C., Canada, 2008.

[87] J. HEO et al. « Spherical Hashing ». Dans : *Conference on Computer Vision and Pattern Recognition*. Sous la dir. d'IEEE Computer SOCIETY. Providence, RI, USA : IEEE, juin 2012, p. 2957–2964.

[88] B. KULIS et K. GRAUMAN. « Kernelized locality-sensitive hashing for scalable image search ». Dans : *Computer Vision, 2009 IEEE 12th International Conference on*. Ieee. 2009, p. 2130–2137.

[89] B. KULIS et K. GRAUMAN. « Kernelized locality-sensitive hashing ». Dans : *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 34.6 (2012), p. 1092–1104.

[90] A. JOLY et O. BUISSON. « Random Maximum Margin Hashing ». Dans : *CVPR*. Colorado springs, United States : IEEE, juin 2011.

[91] A. BOURRIER et al. *Nearest neighbor search for arbitrary kernels with explicit embeddings*. Rap. tech. INRIA Project-Teams Metiss et Texmex, 2012.

[92] L. PAULEVÉ, H. JÉGOU et L. AMSALEG. « Locality sensitive hashing : A comparison of hash function types and querying mechanisms ». Dans : *Pattern Recognition Letters* 31.11 (2010), p. 1348–1358.

[93] H. JÉGOU, M. DOUZE et C. SCHMID. « Product quantization for nearest neighbor search ». Dans : *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 33.1 (2011), p. 117–128.

[94] J. PHILBIN et al. « Object Retrieval with Large Vocabularies and Fast Spatial Matching ». Dans : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2007.

[95] M. VARMA et A. ZISSERMAN. « Texture classification : Are filter banks necessary ? » Dans : *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*. T. 2. IEEE. 2003, p. II–691.

[96] K. MIKOLAJCZYK, B. LEIBE et B. SCHIELE. « Multiple object class detection with a generative model ». Dans : *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. T. 1. IEEE. 2006, p. 26–36.

[97] F. JURIE et B. TRIGGS. « Creating efficient codebooks for visual recognition ». Dans : *Proceedings of the 10th International Conference on Computer Vision,ICCV*. Beijing, China : IEEE Computer Society, oct. 2005. ISBN : 0-7695-2334-X.

[98] K. FUKUNAGA et L. HOSTETLER. « The estimation of the gradient of a density function, with applications in pattern recognition ». Dans : *Information Theory, IEEE Transactions on* 21.1 (1975), p. 32–40.

[99] R. WEBER, H.J. SCHEK et S. BLOTT. « A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces ». Dans : *Proceedings of the International Conference on Very Large Data Bases*. INSTITUTE OF ELECTRICAL & ELECTRONICS ENGINEERS. 1998, p. 194–205.

[100] K. LING et G. WU. « Frequency Based Locality Sensitive Hashing ». Dans : *Multimedia Technology (ICMT), International Conference on*. Sous la dir. d'IEEE. Hangzhou, China, juil. 2011, p. 4929 –4932.

[101] C. SILPA-ANAN et R. HARTLEY. « Optimised KD-trees for fast image descriptor matching ». Dans : *Conference on Computer Vision and Pattern Recognition*. Sous la dir. d'IEEE Computer SOCIETY. Anchorage, Alaska, USA : IEEE Computer Society, juin 2008.

[102] M. ALY, M. MUNICH et P. PERONA. « Multiple Dictionaries for Bag of Words Large Scale Image Search ». Dans : *International Conference on Image Processing (ICIP)*. Sous la dir. d'IEEE. Brussels, Belgium, sept. 2011.

[103] A. JOLY et O. BUISSON. « A Posteriori Multi-Probe Locality Sensitive Hashing ». Dans : *ACM International Conference on Multimedia (MM'08)*. Sous la dir. d'Abdulmotaleb EL-SADDIK et al. Vancouver, British Columbia, Canada : ACM, oct. 2008, p. 209–218. ISBN : 978-1-60558-303-7.

[104] J. PHILBIN et al. « Lost in Quantization : Improving Particular Object Retrieval in Large Scale Image Databases ». Dans : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2008.

[105] A. JOLY, O. BUISSON et C. FRELICOT. « Statistical similarity search applied to content-based video copy detection ». Dans : *Data Engineering Workshops, 2005. 21st International Conference on*. IEEE. 2005, p. 1285–1285.

[106] Q. LV et al. « Multi-probe LSH : efficient indexing for high-dimensional similarity search ». Dans : *Proceedings of the 33rd international conference on Very large data bases*. VLDB '07. Vienna, Austria : VLDB Endowment, 2007, p. 950–961. ISBN : 978-1-59593-649-3. URL : http://portal.acm.org/citation.cfm?id=1325851.1325958.

[107] S. ARYA et al. *An optimal algorithm for approximate nearest neighbor searching in fixed dimensions*. Rap. tech. College Park, MD, USA, 1995.

[108] W. DONG, M. CHARIKAR et K. LI. « Asymmetric distance estimation with sketches for similarity search in high-dimensional spaces ». Dans : *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2008, p. 123–130.

[109] H. JEGOU, M. DOUZE et C. SCHMID. « Hamming embedding and weak geometric consistency for large scale image search ». Dans : *Computer Vision--ECCV 2008* (2008), p. 304–317.

[110] W. ZHAO, X. WU et C.W. NGO. « On the Annotation of Web Videos by Efficient Near-Duplicate Search ». Dans : *IEEE Transactions on Multimedia* 12.5 (2010), p. 448–461.

[111] H. XIE et al. « Pairwise weak geometric consistency for large scale image search ». Dans : *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*. ICMR '11. Trento, Italy : ACM, 2011, 42 :1–42 :8. ISBN : 978-1-4503-0336-1. DOI : 10.1145/1991996.1992038.

[112] J. RABIN. « Approches robustes pour la comparaison d'images et la reconnaissance d'objets ». Thèse de doct. Telecom ParisTech, 2009.

[113] P.J. ROUSSEEUW. « Least median of squares regression ». Dans : *Journal of the American statistical association* (1984), p. 871–880.

[114] P.J. ROUSSEEUW. « Multivariate estimation with high breakdown point ». Dans : *Mathematical statistics and applications* 8 (1985), p. 283–297.

[115] P.V.C. HOUGH. « Machine analysis of bubble chamber pictures ». Dans : *International Conference on High Energy Accelerators and Instrumentation*. T. 73. 1959.

[116] D. H. BALLARD. « Generalizing the Hough Transform to Detect Arbitrary Shapes ». Dans : *Readings in Computer Vision : Issues, Problems, Principles, and Paradigms*. Sous la dir. de M. A. FISCHLER et O. FIRSCHEIN. Los Altos, CA. : Kaufmann, 1987, p. 714–725.

[117] M. A. FISCHLER et R. C. BOLLES. « Random Sample Consensus : A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography ». Dans : *Commun. ACM* 24.6 (1981), p. 381–395.

[118] P.H.S. TORR et C. DAVIDSON. « IMPSAC : synthesis of importance sampling and random sample consensus ». Dans : *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 25.3 (2003), p. 354–364.

[119] C.M. CHENG et S.H. LAI. « A consensus sampling technique for fast and robust model fitting ». Dans : *Pattern Recognition* 42.7 (2009), p. 1318–1329.

[120] O. CHUM et J. MATAS. « Optimal randomized RANSAC ». Dans : *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 30.8 (2008), p. 1472–1482.

[121] L. MOISAN et B. STIVAL. « A probabilistic criterion to detect rigid point matches between two images and estimate the fundamental matrix ». Dans : *International Journal of Computer Vision* 57.3 (2004), p. 201–218.

[122] P.H.S. TORR. « Motion segmentation and outlier detection ». Thèse de doct. University of Oxford, 1995.

[123] M. ZULIANI, CS KENNEY et BS MANJUNATH. « The multiransac algorithm and its application to detect planar homographies ». Dans : *Image Processing, 2005. ICIP 2005. IEEE International Conference on*. T. 3. IEEE. 2005, p. III–153.

[124] K. SCHINDLER et D. SUTER. « Two-view multibody structure-and-motion with outliers through model selection ». Dans : *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 28.6 (2006), p. 983–995.

[125] T. SATTLER, B. LEIBE et L. KOBBELT. « SCRAMSAC : Improving RANSAC's efficiency with a spatial consistency filter ». Dans : *Computer Vision, 2009 IEEE 12th International Conference on*. Ieee. 2009, p. 2090–2097.

[126] O. CHUM et J. MATAS. « Matching with PROSAC-progressive sample consensus ». Dans : *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. T. 1. Ieee. 2005, p. 220–226.

[127] P.H.S. TORR et A. ZISSERMAN. « MLESAC : A new robust estimator with application to estimating image geometry ». Dans : *Computer Vision and Image Understanding* 78.1 (2000), p. 138–156.

[128] A.P. DEMPSTER, N.M. LAIRD et D.B. RUBIN. « Maximum likelihood from incomplete data via the EM algorithm ». Dans : *Journal of the Royal Statistical Society. Series B (Methodological)* (1977), p. 1–38.

[129] C.V. STEWART. « MINPRAN : A new robust estimator for computer vision ». Dans : *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 17.10 (1995), p. 925–938.

[130] A. DESOLNEUX, L. MOISAN et J.M. MOREL. « Meaningful Alignments ». Dans : *Int. J. Comput. Vision* 40 (1 2000), p. 7–23. ISSN : 0920-5691. DOI : 10.1023/A:1026593302236.

[131] J. RABIN et al. « MAC-RANSAC : a robust algorithm for the recognition of multiple objects ». Dans : *Proceedings of 3DPTV 2010*. 2009.

[132] J. CECH et R. SÁRA. « Efficient Sampling of Disparity Space for Fast And Accurate Matching ». Dans : *Conference on Computer Vision and Pattern Recognition*. Sous la dir. d'IEEE Computer SOCIETY. Minneapolis, Minnesota, USA : IEEE Computer Society, juin 2007.

[133] J. CECH, J. MATAS et M. PERDOCH. « Efficient Sequential Correspondence Selection by Cosegmentation ». Dans : *IEEE Trans. Pattern Anal. Mach. Intell.* 32.9 (sept. 2010), p. 1568–1581. ISSN : 0162-8828. DOI : 10.1109/TPAMI.2009.176. URL : http://dx.doi.org/10.1109/TPAMI.2009.176.

[134] O. CHUM et al. « Total Recall : Automatic Query Expansion with a Generative Feature Model for Object Retrieval ». Dans : *Proceedings of the 11th International Conference on Computer Vision, Rio de Janeiro, Brazil*. 2007.

[135] C. BUCKLEY et al. « Automatic Query Expansion Using SMART : TREC 3 ». Dans : *TREC*. 1994, p. 0–.

[136] M. MITRA, A. SINGHAL et C. BUCKLEY. « Improving automatic query expansion ». Dans : *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 1998, p. 206–214.

[137] O. CHUM et al. « Total Recall II : Query Expansion Revisited ». Dans : *Conference on Computer Vision and Pattern Recognition*. Sous la dir. d'IEEE Computer SOCIETY. Colorado Springs, CO, USA : IEEE, juin 2011, p. 889–896.

[138] R. ARANDJELOVIC et A. ZISSERMAN. « Three things everyone should know to improve object retrieval ». Dans : *Conference on Computer Vision and Pattern Recognition*. Sous la dir. d'IEEE Computer SOCIETY. Providence, RI, USA : IEEE, juin 2012, p. 2911–2918.

[139] J. REVAUD, M. DOUZE, C. SCHMID et al. « Correlation-Based Burstiness for Logo Retrieval ». Dans : *Proceedings of the 20th ACM international conference on Multimedia*. Sous la dir. d'ACM. Nara, Japan, 2012.

[140] J. SIVIC et A. ZISSERMAN. « Video data mining using configurations of viewpoint invariant regions ». Dans : *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*. T. 1. IEEE. 2004, p. I–488.

[141] T. QUACK, V. FERRARI et L. VAN GOOL. « Video mining with frequent itemset configurations ». Dans : *Image and Video Retrieval* (2006), p. 360–369.

[142] R. AGRAWAL, R. SRIKANT et al. « Fast algorithms for mining association rules ». Dans : *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*. T. 1215. Santiago de Chile, Chile, 1994, p. 487–499.

[143] J. YUAN, Y. WU et M. YANG. « From frequent itemsets to semantically meaningful visual patterns ». Dans : *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2007, p. 864–873.

[144] J. YUAN, Y. WU et M. YANG. « Discovery of collocation patterns : from visual words to visual phrases ». Dans : *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. Ieee. 2007, p. 1–8.

[145] J. YUAN et al. « Common spatial pattern discovery by efficient candidate pruning ». Dans : *Image Processing, 2007. ICIP 2007. IEEE International Conference on*. T. 1. IEEE. 2007, p. I–165.

[146] J. YUAN et Y. WU. « Spatial random partition for common visual pattern discovery ». Dans : *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE. 2007, p. 1–8.

[147] A. ANJULAN et N. CANAGARAJAH. « A novel video mining system ». Dans : *Image Processing, 2007. ICIP 2007. IEEE International Conference on*. Sous la dir. d'IEEE. T. 1. IEEE. Brussels, Belguim, 2007, p. I–185.

[148] A. ANJULAN et N. CANAGARAJAH. « A Unified Framework for Object Retrieval and Mining ». Dans : *Circuits and Systems for Video Technology, IEEE Transactions on* 19.1 (2009), p. 63 –76.

[149] T.H. GORMEN et al. « Introduction to algorithms ». Dans : *MIT Press* 44 (1990), p. 97–138.

[150] S. BRIN et L. PAGE. « The anatomy of a large-scale hypertextual Web search engine ». Dans : *Computer networks and ISDN systems* 30.1 (1998), p. 107–117.

[151] U. BRANDES. « A faster algorithm for betweenness centrality ». Dans : *Journal of Mathematical Sociology* 25.2 (2001), p. 163–177.

[152] T. HOFMANN. « Unsupervised learning by probabilistic latent semantic analysis ». Dans : *Machine Learning* 42.1 (2001), p. 177–196.

[153] D.M. BLEI, A.Y. NG et M.I. JORDAN. « Latent dirichlet allocation ». Dans : *the Journal of machine Learning research* 3 (2003), p. 993–1022.

[154] P. QUELHAS et al. « Modeling scenes with local descriptors and latent aspects ». Dans : *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*. T. 1. IEEE. 2005, p. 883–890.

[155] J. SIVIC et al. « Discovering object categories in image collections ». Dans : (2005).

[156] J. PHILBIN, J. SIVIC et A. ZISSERMAN. « Geometric LDA : A Generative Model for Particular Object Discovery ». Dans : *Proceedings of the British Machine Vision Conference*. 2008.

[157] O. CHUM et al. « Scalable near identical image and shot detection ». Dans : *Proceedings of the 6th ACM international conference on Image and video retrieval*. CIVR '07. Amsterdam, The Netherlands : ACM, 2007, p. 549–556. ISBN : 978-1-59593-733-9. DOI : 10.1145/1282280.1282359.

[158] O. CHUM, J. PHILBIN et A. ZISSERMAN. « Near duplicate image detection : min-hash and tf-idf weighting ». Dans : *Proceedings of the British Machine Vision Conference*. T. 3. 2008, p. 4.

[159] R.A. BAEZA-YATES et B. RIBEIRO-NETO. *Modern Information Retrieval*. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 1999. ISBN : 020139829X.

[160] O. CHUM, M. PERDOCH et J. MATAS. « Geometric min-Hashing : Finding a (thick) needle in a haystack ». Dans : *Conference on Computer Vision and Pattern Recognition*. Sous la dir. d'IEEE Computer SOCIETY. Miami, Florida, USA : IEEE Computer Society, juin 2009, p. 17–24. ISBN : 978-1-4244-3992-8.

[161] C. BORGELT. « Frequent item set mining ». Dans : *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery* 2.6 (2012), p. 437–456. ISSN : 1942-4795. DOI : 10.1002/widm.1074.

[162] P. LETESSIER, O. BUISSON et A. JOLY. « Consistent visual words mining with adaptive sampling ». Dans : *ICMR*. Trento, Italy : ACM, avr. 2011, 49 :1–49 :8.

[163] L. DEVROYE et L. DEVROYE. « Non-uniform random variate generation ». Dans : (1986).

[164] F. OLKEN. « Random Sampling from Databases ». Thèse de doct. University of California, 1993.

[165] S.K. THOMPSON. « Adaptive sampling ». Dans : *The Survey Statistician*. 1995.

[166] H. JÉGOU, M. DOUZE et C. SCHMID. « On the burstiness of visual elements ». Dans : *Conference on Computer Vision & Pattern Recognition*. Juin 2009.

[167] S. van DONGEN. *MCL-edge : Analysis of networks with millions of nodes*. 2012. URL : http://www.micans.org/mcl/index.html.

[168] S. VAN DONGEN. « A cluster algorithm for graphs ». Dans : *Report-Information systems* 10 (2000), p. 1–40.

[169] S. PEYRONNET. *Méthodes de clustering*. 2010. URL : http://www.spoonylife.org/algorithms-and-computation/methodes-de-clustering.

[170] Y. CHEN, J.Z. WANG et R. KROVETZ. « Content-based image retrieval by clustering ». Dans : *Proceedings of the 5th ACM SIGMM international workshop on Multimedia information retrieval*. ACM. 2003, p. 193–200.

[171] P. LETESSIER, O. BUISSON et A. JOLY. « Scalable mining of small visual objects ». Dans : *Proceedings of the 20th ACM international conference on Multimedia*. MM '12. Nara, Japan : ACM, 2012, p. 599–608. ISBN : 978-1-4503-1089-5. DOI : 10.1145/2393347.2393431.

[172] X. WU et S. SATOH. « Temporal recurrence hashing algorithm for mining commercials from multimedia streams ». Dans : *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE. 2011, p. 2324–2327.

[173] N. PUTPUEK et al. « Unified Approach to Detection and Identification of Commercial Films by Temporal Occurrence Pattern ». Dans : *Pattern Recognition (ICPR), 2010 20th International Conference on*. IEEE. 2010, p. 3288–3291.

[174] A. JOLY et al. « Visual-Based Transmedia Events Detection ». Dans : *Proceedings of the 20th ACM international conference on Multimedia*. ACM. 2012.

[175] Z.J. ZHA et al. « Visual query suggestion ». Dans : *Proceedings of the 17th ACM international conference on Multimedia*. Sous la dir. d'ACM. Beijing, China, 2009, p. 15–24.

[176] A. HAMZAOUI et al. « Object-based visual query suggestion ». Dans : *Multimedia Tools and Applications* 11042 (2013). Sous la dir. de Springer US. DOI : 10.1007/s11042-012-1340-5.

[177] A. HAMZAOUI. « Shared-Neighbours methods for visual content structuring and mining ». Thèse de doct. 2012.

[178] J. PHILBIN et A. ZISSERMAN. « Object mining using a Matching Graph on Very Large Image Collections ». Dans : *Computer Vision, Graphics Image Processing, 2008. ICVGIP '08. Sixth Indian Conference on*. 2008, p. 738 –745.

[179] K. GRAUMAN et T. DARRELL. « Unsupervised Learning of Categories from Sets of Partially Matching Image Features ». Dans : *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. T. 1. 6. 2006, p. 19 –25.

[180] O. CHUM et J. MATAS. « Large-Scale Discovery of Spatially Related Images ». Dans : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32 (2010), p. 371–377.

[181] G. XU et al. « Co-clustering analysis of weblogs using bipartite spectral projection approach ». Dans : *Knowledge-Based and Intelligent Information and Engineering Systems* (2010), p. 398–407.

[182] H. ZHA et al. « Bipartite graph partitioning and data clustering ». Dans : *Proceedings of the tenth international conference on Information and knowledge management*. ACM. 2001, p. 25–32.

[183] IEEE Computer SOCIETY, éd. *The 25th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2012*. Providence, RI, USA : IEEE, juin 2012.

[184] IEEE Computer SOCIETY, éd. *2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007), 18-23 June 2007, Minneapolis, Minnesota, USA*. Minneapolis, Minnesota, USA : IEEE Computer Society, juin 2007.

[185] *10th IEEE International Conference on Computer Vision (ICCV)*. Beijing, China : IEEE Computer Society, oct. 2005. ISBN : 0-7695-2334-X.

ina