Passage à l'échelle des méthodes de recherche sémantique dans les grandes bases d'images

David Gorisse

THÈSE

Pour l'obtention du diplôme de

DOCTORAT DE L'UNIVERSITE DE CERGY-PONTOISE

en Sciences Traitement de l'Image et du Signal

Diplôme National - Arrêté du ??

ÉCOLE DOCTORALE Sciences et Technologies de l'Information et de la Communication

Présentée par

David GORISSE

Passage à l'échelle des méthodes de recherche sémantique dans les grandes bases d'images

Directeur de thèse : **M. Frédéric PRECIOSO**

Co-direction : **M. Matthieu CORD, Mme. Sylvie PHILIPP-FOLIGUET**

Soutenue le 20 décembre 2010
Devant la Commission d'Examen

JURY

M. Michel CRUCIANU	Professeur des Universités	Rapporteur
M. Stéphane MARCHAND-MAILLET	Maître de conférences	Rapporteur
M. Jean-Michel JOLION	Professeur des Universités	Examinateur
M. Florent PERRONNIN	Docteur	Examinateur
M. Frédéric PRECIOSO	Maître de conférences	Directeur de thèse
M. Matthieu CORD	Professeur des Universités	Co-Directeur de thèse
Mme. Sylvie PHILIPP-FOLIGUET	Professeur des Universités	Co-Directeur de thèse

Remerciements

Je tiens tout d'abord à exprimer ma profonde reconnaissance pour mon directeur de Thèse, Frédéric PRECIOSO, avec qui j'ai eu une grande complicité tout au long de ces trois années. Je le remercie très sincèrement pour la confiance qu'il m'a accordée en acceptant de diriger ma thèse mais également pour son soutien, sa disponibilité, ses précieux conseils et sa ténacité qui m'a aidée à me surpasser. J'ai tout particulièrement apprécié les nombreux échanges qui nous ont permis d'aboutir pas à pas aux démonstrations du χ^2-*LSH*.

Je suis également très reconnaissant envers Matthieu CORD, qui m'a accompagné tout au long de ce travail. Je le remercie pour sa disponibilité et les longues et fructueuses après midi qu'il m'a consacrées dans son bureau à Paris ou il m'a reçu de nombreuses fois. J'ai ainsi largement pu profiter de ses grandes connaissances en Machine Learning, de sa rigueur scientifique, de sa pédagogie pour présenter son travail et de ses précieux conseils qui m'ont permis de travailler dans les meilleures conditions.

Mes remerciements s'adressent aussi à Sylvie PHILIPP-FOLIGUET qui m'a également encadré. Je la remercie pour son aide ainsi que ses précieuses remarques.

J'exprime également mes vifs remerciements aux membres du laboratoire ETIS ainsi qu'aux doctorants que j'ai pu côtoyer durant ces trois dernières années et qui ont su rendre mon travail agréable. JE, c'est promis, tes blagues vont me manquer... Je n'oublie pas non plus les membres du service informatique, Laurent PROTOIS, Michel LECLERC et notre Mich(a)el JORDAN qui se sont toujours montrés très disponibles.

J'ai été très honnoré que M. Michel CRUCIANU et M. Stéphane MARCHAND-MAILLET acceptent de juger ce travail et d'en être les rapporteurs et à M. Jean-Michel JOLION et M. Florent

PERRONNIN d’en être les examinateurs.

Table des matières

I Introduction générale **5**

1 Contexte . . . 6

1.1 Techniques de recherche . . . 6

1.1.1 Approche textuelle vs approche numérique . . . 6

1.1.2 Le fossé sémantique/numérique . . . 7

1.2 Explosion de la quantité d'information à traiter . . . 7

2 Architecture des systèmes de recherche par le contenu . . . 9

2.1 Indexation . . . 9

2.1.1 Calcul d'index . . . 9

2.1.2 Fonction de similarité . . . 10

2.2 Moteurs de recherche . . . 12

2.2.1 Système de détection de copies . . . 12

2.2.2 Similarité sémantique et recherche interactive . . . 14

3 Passage à l'échelle . . . 17

3.1 Complexité de la recherche par similarité . . . 17

3.2 Complexité de la recherche interactive . . . 18

3.3 Accélération de la recherche . . . 18

4 Evaluation des systèmes de recherche . . . 19

4.1 Protocoles expérimentaux . . . 19

4.2 Bases d'évaluation . . . 20

5 Contributions et plan de la thèse . . . 20

5.1 Passage à l'échelle de la recherche par similarité basées signature globale. . . 21

5.2 Passage à l'échelle de la recherche par similarité basée signature locale. . . . 21
5.3 Passage à l'échelle de la recherche interactive par bouclage de pertinence. . . 22

I Signature globale et recherche rapide dans les grandes bases d'images 23

II LSH, une structure d'index pour accélérer la recherche par similarité 27
1 Structures d'index pour CBIR 27
2 Table de hachage 29
2.1 Adressage direct 29
2.2 Hachage 29
2.3 Collisions 30
3 Locality-Sensitive Hashing : adaptation du hachage à la recherche des plus proches voisins 31
3.1 Fonction Locality-Sensitive 31
3.2 Concaténation de M fonctions : diminution de la fausse détection 32
3.3 Utilisation de L tables de hachage : augmentation de la bonne détection . . . 32
4 Algorithmes de recherche 32
4.1 Hachage de la base 33
4.2 Etape de recherche 33
4.2.1 Problème $(R,c)NN$ 34
4.2.2 Problème $(R,1-\delta)NN$ 35
4.2.3 Problème KNN 35
4.3 Implémentation 35
5 Principales fonctions de hachage 36
5.1 Hachage pour la distance de Hamming 37
5.2 Hachage pour la similarité du cosinus 38
5.3 Généralisation du hachage pour la similarité du cosinus aux similarités noyaux 40
5.4 Hachage pour la distance l_1 et l_2 41
5.5 Diminution des ressources mémoires : Multi-Probe 41
5.6 Amélioration du l_2 43

III χ^2-*LSH* : Schéma LSH optimisé pour les signatures globales 45
1 Construction de la clé χ^2-*LSH* 46
1.1 Partitionnement de l'espace au sens de la distance χ^2 46
1.2 Construction de la clé 47

1.3 χ^2-*LSH* : une fonction Locality-Sensitive . . . 49
2 Extension du Multi-Probe : χ^2-*MPLSH* . . . 52
3 Evaluation . . . 53
3.1 Efficacité du hachage χ^2 . . . 54
3.1.1 Protocole expérimental . . . 54
3.1.2 Efficacité de la recherche approximée : recherche exacte vs χ^2-*LSH* 55
3.1.3 Ressources mémoires : recherche exacte vs χ^2-*MPLSH* . . . 58
3.2 Comparaison de la qualité de recherche entre χ^2-*MPLSH* et E^2-*MPLSH* . . . 61
3.2.1 Protocole expérimental . . . 61
3.2.2 Evaluation . . . 62
4 Conclusion . . . 63

II Signature locale et recherche rapide dans les grandes bases d'images. 67

IV Similarité entre sacs de caractéristiques locales 71
1 Formalisme . . . 72
2 Approches dictionnaires . . . 72
3 Vector of locally aggregated descriptors (VLAD) . . . 74
4 Stratégie par vote . . . 75
5 Approches noyaux . . . 76
6 Optimisation des fonctions noyaux . . . 78
6.1 K_{select} . . . 79
6.2 Ultra fast . . . 80
6.2.1 Principe . . . 80
6.2.2 Algorithme . . . 81
6.2.3 Propriétés de $UFast$ . . . 82
7 Conclusion . . . 83

V Evaluation et discussion 85
1 Validation sur VOC06 . . . 86
1.1 Protocole expérimental . . . 86
1.2 $UFast_K$ vs K_{Select} . . . 86
1.3 Evaluation de $UFast_K$ pour K_{power} . . . 87
1.4 Evaluation de $UFast_K$ pour K_{match} . . . 89
1.5 $UFast_K$ vs Vote . . . 91

1.6 Noyau sur sacs (BoF) vs Dictionnaire (BoW) . . . 93
2 Comparaison en utilisant la base INRIA Holidays . . . 94
2.1 Protocol expérimental . . . 94
2.2 Evaluation de la qualité de recherche de $UFast_K$ . . . 94
2.3 Compromis qualité / temps de recherche de $UFast_K$ . . . 95
3 Evaluation de différents détecteurs et descripteurs BoF sur VOC06 . . . 98
4 Discussions . . . 100
4.1 Ressources mémoires . . . 100
4.2 Intérêt du formalisme : BoW et noyau explicite . . . 100

III Recherche interactive rapide dans les grandes bases d'images 105

VI Méthodes d'apprentissage interactif rapides 109
1 Recherche rapide du cœur de classe . . . 110
1.1 Fonctions de similarité basiques . . . 110
1.2 Recherche par sous échantillonnage . . . 110
1.3 Recherche hiérarchique . . . 111
1.4 K-VA files . . . 111
1.5 Kernel Indexer : KDX . . . 113
1.6 Recherche dans l'espace des caractéristiques . . . 115
1.7 Synthèse . . . 115
2 Recherche des images les plus proches de la frontière . . . 116
2.1 Extension des méthodes de recherche du cœur de classe . . . 116
2.2 M-Tree . . . 116
2.3 Synthèse . . . 117
3 Stratégie d'apprentissage interactif sous-lineaire basée sur la recherche *ppv* approximée 118
3.1 Justification de la construction du pool d'images . . . 118
3.2 Sélection rapide d'un sous-ensemble d'images pertinentes . . . 119
3.3 Stratégie active . . . 120
3.4 Système . . . 121
3.5 Algorithme et complexité . . . 122

VII Evaluation de SALSAS 125
1 Plan expérimental . . . 125
1.1 Bases . . . 126

1.2 Critères d'évaluation 126
1.3 Paramétrage 127
1.3.1 Sessions de recherche 127
1.3.2 Paramètres internes des méthodes 127
1.3.3 Paramètres propres à SALSAS 128
1.3.4 Paramètres de la recherche *KNN* 129
1.4 Notations 130
2 Performances de SALSAS 130
2.1 SALSAS vs LIN_CHI2 130
2.2 Intérêt de χ^2-*LSH* : Optimisation de la structure d'index 132
2.3 Analyse détaillée des résultats 134
3 Performance du système avec d'autres paramètres 136
3.1 Multi-requête 136
3.2 Système avec un noyau l_2-*RBF* 137
3.3 Intérêt du χ^2 : comparaison des méthodes LIN_L2 et LIN_CHI2 138
4 Conclusion 138

VIII Conclusion générale et perspectives **143**
1 Bilan 143
2 Perspectives 145

IX Annexes **147**
A.1 Bases d'évaluation 147
A.2 Mesure de performances 150
A.3 Exemple de session de recherche interactive 150

Publications de l'auteur **153**

Références bibliographiques **155**

Notations

Général

Scalaire	Les scalaires sont représentés par des lettres en minuscules. Exemples : i, j, k, ...
Vecteur	Les vecteurs sont représentés par des lettres minuscules en gras. Exemple : $\mathbf{x}$. La ième composante du vecteur est représentée par l'indice i : x_i. La composante n'est plus représentée en gras puisque x_i est un scalaire.
Ensemble	Les ensembles sont représentés par des lettres majuscules calligraphiées. Exemple : $\mathcal{B}$
Espace	Les espaces sont représentés par des lettres majuscules avec une police italique. Exemple : $\mathscr{H}$
Produit scalaire	Le produit scalaire est noté $\langle .,. \rangle$.
Distance	Une distance est notée $D(\mathbf{x},\mathbf{y})$.
Complexité	La complexité des algorithmes de recherche mesure le nombre de calculs élémentaires (comparaisons entre images) nécessaire pour réaliser une recherche. Il permet de mesurer l'évolution des temps de recherche en fonction de la taille de la base. On distingue les algorithmes de complexité linéaire $O(n)$ et ceux de complexité sous-linéaire (généralement de la forme $O(n^\rho)$ avec $\rho < 1$ ou $O(log(n))$).

Base d'images

$\mathcal{B}$	Une base d'images est un ensemble d'images munies d'une signature et d'une fonction de similarité.
n	Nombre d'images dans la base.
I_j	$j^{\text{ième}}$ image de la base.
$\mathcal{S} \subset \mathcal{B}$	Représente un sous-ensemble d'images de la base.
$\mathscr{F}$	Représente l'espace des caractéristiques, espace de représentation des signatures des images.
d	Dimension de l'espace des caractéristiques.
BoW	(de l'anglais *Bag of Words*, en français, *sac de mots*) Représentation des images en utilisant une signature vectorielle. Le vecteur est un histogramme d'occurrences de mots visuels issus d'un dictionnaire. Le dictionnaire est obtenu par une étape de clustering des descriptions visuelles des images de la base. La signature de l'image I_i est alors représenté par le vecteur $\mathbf{x}_i \in \mathscr{F}$. Cette signature est dite globale car elle représente l'image dans son intégralité.
BoF	(de l'anglais *Bag of Features*, en français, *sac de descripteurs*) Représentation des images sous la forme d'un ensemble non ordonné de signatures vectorielles locales à l'image. La signature de l'image I_i est alors représentée par le sac B_i. Ce sac est composé d'un ensemble de vecteurs : $\mathbf{b}_{ri} \in \mathscr{F}$. Cette signature est dite locale car elle est composée d'un ensemble de signatures représentant chacune une partie locale de l'image.

Similarité

$f(.,.)$	Fonction de similarité utilisée pour mesurer la similitude entre deux images à l'aide de leur signature.
$\phi(.)$ ou $\Phi(.)$	Fonction d'induction qui projette les données dans un espace Hilbertien $\mathscr{H}$. La notation ϕ est utilisée pour les signatures vectorielles alors que Φ est utilisée pour les BoF.
$k(.,.)$ ou $K(.,.)$	Fonction noyau utilisée pour définir des similarités entre des données d'entrées éventuellement non vectorielles. Une des propriétés des noyaux est qu'il est possible de réécrire cette fonction sous la forme d'un produit scalaire après projection des données dans un espace induit via une fonction ϕ : $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$. On note k lorsque le noyau est défini avec des entrées vectorielles et K pour des entrées de types BoF.

Structure d'index LSH

Table de hachage	Structure d'index où chaque donnée de la base est stockée dans une case, appelée *bucket* accessible avec une fonction de hachage.
$h(.)$	Fonction *locality sensitive*. Cette fonction permet de calculer l'empreinte (ou le hash) d'une image à partir de sa signature.
$\mathcal{H}$	Ensemble des fonctions *locality sensitive*.
$g(.)$	Fonction de hachage obtenue en concaténant M fonctions $h(.)$.
$\mathcal{G}$	Ensemble des fonctions de hachage.
ppv	(les plus proches voisins) Algorithme de recherche des images les plus proches (où les plus similaires) à une image requête. On distingue la recherche *RNN* qui consiste à retrouver les images dans une rayon de recherche R de la requête et la recherche *KNN* qui consiste à retrouver les K images les plus proches. La notion de proximité entre images est définie à l'aide d'une fonction de similarité ou d'une distance et de la signature des images.

Apprentissage

y_i	annotation de l'image i : $$y_i = \begin{cases} 1 \text{ si, } \mathbf{x}_i \text{ est pertinente} \\ -1 \text{ si, } \mathbf{x}_i \text{ est non pertinente} \\ 0 \text{ s'il n'y a pas d'annotation} \end{cases}$$
$\mathcal{A}$	Ensemble des images annotées ou ensemble d'apprentissage : $\{\mathbf{x}_i \in [1 \dots n] \mid y_i \neq 0\}$
$\mathcal{A}^+$	Ensemble des images annotées positivement : $\{\mathbf{x}_i \in [1 \dots n] \mid y_i > 0\}$
$\mathcal{A}^-$	Ensemble des images annotées négativement : $\{\mathbf{x}_i \in [1 \dots n] \mid y_i < 0\}$
$\mathcal{U}$	Ensemble des images non annotées : $\{\mathbf{x}_i \in [1 \dots n] \mid y_i = 0\}$
I	Ensemble des indices des images annotées ou ensemble des indices d'apprentissage : $\{i \in [1 \dots n] \mid y_i \neq 0\}$
$\bar{I}$	Ensemble des indices des images non annotées : $\{i \in [1 \dots n] \mid y_i = 0\}$

$f_{\mathcal{A}}(.)$	Fonction de pertinence après entraînement avec l'ensemble d'apprentissage $\mathcal{A}$. Cette fonction permet de discriminer les images pertinentes des images non pertinentes.

Chapitre I

Introduction générale

Avec la révolution numérique de ces dernières décennies, la quantité de données numériques a explosé. La mise à la portée de chacun d'appareils photos numériques, scanners, webcams et téléphones portables, la montée en puissance des capacités de calcul et de stockage des ordinateurs, ainsi que la démocratisation d'Internet, ont contribué à cette explosion. De cette explosion est né le besoin de savoir gérer des immenses volumes de données pour profiter au mieux de cette masse d'information. Parmi ces données, nous nous intéressons tout particulièrement aux bases d'images numériques.

Ces bases sont souvent immenses et généralement sous-exploitées faute d'outils pour accéder à l'information qu'elles contiennent. Le traitement de ces bases a ouvert plusieurs champs de recherche :

- *Stockage*. Certaines bases d'images sont colossales, et un système matériel et logiciel spécifique est nécessaire pour pouvoir stocker sans dégradation toutes ces informations, tout en gardant à l'esprit qu'elles doivent rester accessibles.
- *Partage*. Plusieurs utilisateurs doivent pouvoir accéder aux informations de la base simultanément. Des problèmes de sécurisation des accès, de protection et d'intégrité des données se posent.
- *Recherche*. Une fois les images stockées, elles se retrouvent noyées au sein d'une très grande quantité d'information. Bien que l'on puisse retrouver une image par un index informatique, il est très difficile – voire impossible – de retrouver une image que l'on a en tête. D'où la nécessité de développer des techniques d'interrogation et de recherche dans les bases.

Dans cette thèse, nous nous intéressons aux techniques de recherche et plus généralement aux systèmes permettant d'effectuer des recherches dans les grands corpus.

1 Contexte

1.1 Techniques de recherche

Parmi les techniques de recherche, on peut distinguer deux grandes approches : l'approche textuelle (recherche par mots clés) et l'approche numérique (recherche par le contenu).

Ces deux techniques nécessitent de réaliser un pré-traitement qui consiste à indexer la base. L'*indexation* repose sur l'assignation, à chaque image de la base, d'un code qui permet de l'identifier.

1.1.1 Approche textuelle vs approche numérique

Pour la première approche, l'indexation consiste à représenter chaque donnée de la base par un mot clé (ou une liste de mots clés) qui décrit au mieux le contenu visuel de l'image. Un utilisateur peut alors effectuer une recherche en formulant une requête composée d'un ou de plusieurs termes. Le système présente à l'utilisateur les images considérées comme les plus proches par rapport aux termes introduits. La liaison entre l'espace des requêtes et celui des réponses se réalise en exploitant la similarité entre les chaînes de caractères introduites par les utilisateurs et les termes associés aux images dans les index. Cette approche a plusieurs limitations. Elle requiert un travail de saisie et de maintenance manuelle des annotations qui devient rapidement laborieux et coûteux. Dans certain cas, des solutions permettent d'automatiser l'indexation. Ainsi, sur un site Web, on peut envisager de décrire une image à partir de mots clés extraits dans les zones de texte situées autour de l'image. Cependant, il n'existe plus aucune garantie sur l'adéquation avec le contenu de l'image. De plus, avec ce type d'approche, il faut parfois être expert de la base et bien connaître les mots clés utilisés lors de l'étape d'indexation pour retrouver efficacement les images correspondant à ses besoins.

D'un autre côté, l'image porte un contenu souvent riche et permet d'illustrer un propos, une action, des émotions… mieux que beaucoup de mots. « *One picture is worth ten thousand words.* »[1] C'est pourquoi les médias et les encyclopédies utilisent les images pour décrire des objets, des personnes, des événements… L'approche numérique, à laquelle nous nous intéressons dans cette thèse, consiste à indexer les images directement à partir du contenu visuel des images. On parle alors de système de recherche d'images par le contenu (Content-Based Image Retrieval Systems ou CBIR).

L'accès aux données à travers des images exemples est une technique introduite dans beaucoup de travaux de recherche, mais qui tarde à s'imposer dans les systèmes de recherche d'images sur

[1] D'après [Bur87], cette citation a été inventée par Frederick R. Barnard pour un slogan publicitaire des années 20. Pour être pris au sérieux, l'auteur a présenté cette phrase comme la traduction d'un proverbe chinois. Ce proverbe a ensuite été populairement attribué à Confucius.

Internet comme une alternative ou un complément à l'interrogation par mots clés du fait du *fossé sémantique/numérique*.

1.1.2 Le fossé sémantique/numérique

Le fossé sémantique/numérique correspond à l'inadéquation entre la similarité calculées numériquement et la similarité entre les images attendue par l'utilisateur. Le premier type de similarité tente de modéliser des caractéristiques perceptuelles de l'image, tandis que le deuxième est principalement conceptuel. En d'autres termes, on distingue deux niveaux d'analyse :

- le bas-niveau dit numérique, qui fait référence au contenu signal de l'image
- le haut-niveau dit sémantique, qui fait référence au contenu interprétable de l'image, autrement dit l'ensemble des objets et significations que l'on peut y associer.

Il existe souvent entre la description bas-niveau et haut-niveau une différence forte, que l'on appelle le fossé *sémantique/numérique* : « *The semantic gap is the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for a user in a given situation.* »[2] Outre le fossé qui peut exister entre la description visuelle d'une image et sa sémantique, une image est bien souvent polysémique [SWS+00]. Seule la contextualisation permettra d'identifier le type de sémantique recherchée. Selon le contexte de la recherche, la similarité sémantique attendue peut être différente. Ceci montre l'importance du fossé qui existe entre le bas niveau et le haut niveau, puisqu'au delà de la difficulté à déterminer des caractérisations visuelles correspondant à la sémantique, le contexte est aussi un facteur important.

1.2 Explosion de la quantité d'information à traiter

Outre les problèmes d'extraction de la sémantique, les problèmes d'échelle de traitement deviennent prédominants en CBIR.

Comme l'augmentation des ressources de calcul des ordinateurs est exponentielle (comme le prédit la loi de Moore), on pourrait penser qu'aucun soin particulier ne doit être pris pour manipuler des bases de données toujours plus grandes : l'augmentation de la vitesse des processeurs permettra à n'importe quel moteur d'effectuer des recherches dans des bases toujours plus grandes (passer à l'échelle). Une analyse rapide quant à l'évolution des ressources prouve que cette affirmation est fausse.

Comme prévu selon la loi de Moore, le nombre de transistors qui peuvent être placés à bon marché sur un circuit intégré double approximativement tous les deux ans depuis les années 60. Comme le

[2] A.W.M. Smeulders [SWS+00]

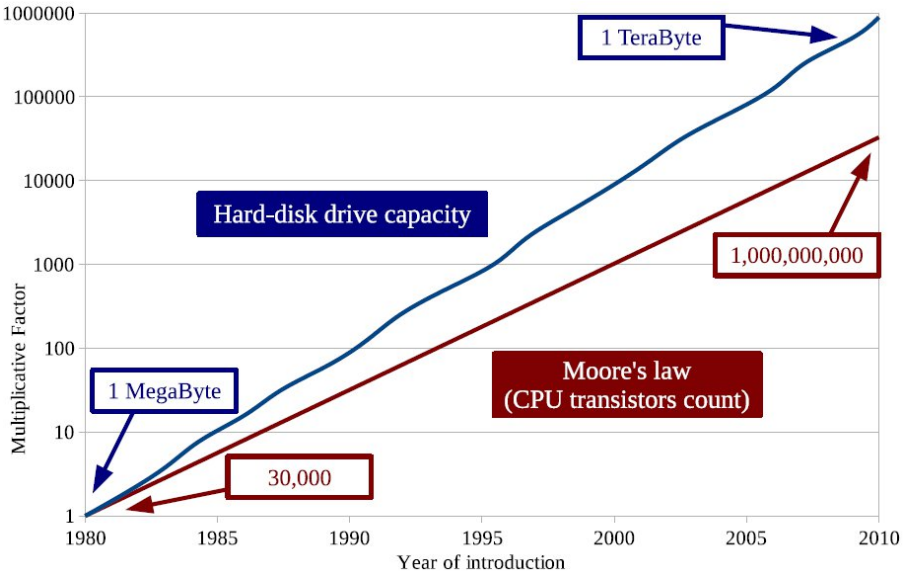


Figure I.1 – Évolution des ressources de calcul et de stockage. Comparaison de la croissance exponentielle de la capacité des disques durs (bleue) et du nombre de transistors intégrés dans les processeurs (loi de Moore) (rouge) en fonction de l'année d'introduction sur le marché. L'axe vertical représente les facteurs multiplicatifs depuis 1980 suivant une échelle logarithmique. Le nombre de transistors double tous les deux ans tandis que la capacité des disques durs double tous les 18 mois [Bor10].

montre la Figure I.1, cette loi se vérifie bien entre la période 1980-2010 (courbe en rouge) et reflète l'augmentation exponentielle de la puissance de calcul de nos ordinateurs. D'autre part, depuis les années 80, les capacités de stockage des disques durs doublent tous les 18 mois, plus ou moins comme le montre la courbe bleue de la Figure I.1. Il apparaît que le volume de données que l'on peut stocker sur nos machines augmente plus vite que la vitesse de calcul des ordinateurs. Des ordinateurs bons marchés, pouvant être reliés en réseau, permettent maintenant de rassembler et de stocker des images plus rapidement que de les analyser.

Voici quelques chiffres pour donner un ordre d'idée de l'augmentation phénoménale du nombre de photos numériques disponibles : Google annonce en août 2005, que plus de 2 milliards d'images[3] sont indexées sur son moteur de recherche Google Images. Ce chiffre est passé à plus d'un trillion d'images en juillet 2008[4]. En octobre 2009, 4 milliards d'images ont été recensées sur Flickr[5], le site web de partage de photographies le plus connu. Environ 3 000 nouvelles images sont soumises toutes les minutes.

Pour gérer ces immenses bases, il va donc falloir adapter les systèmes de recherche existants.

[3] http ://www.webrankinfo.com/actualites/200508-index-google.htm
[4] http ://www.techdigest.tv/2008/07/google_working_1.html
[5] http ://blog.flickr.net/en/2009/10/12/4000000000/

2 Architecture des systèmes de recherche par le contenu

Le principe général de la recherche d'image par le contenu se déroule en deux temps. Lors d'une première phase hors ligne (étape d'indexation) le système décrit automatiquement le contenu des images et stocke ces informations dans une base de données. Lors de la seconde phase, l'utilisateur interroge la base à l'aide d'une requête. Le système recherche en ligne les images de la base qui corresponde à la demande de l'utilisateur. Dans cette thèse, nous considérons les systèmes où les requêtes sont formulées à l'aide d'images exemples.

2.1 Indexation

2.1.1 Calcul d'index

Pour décrire les images, la première étape consiste à extraire des primitives visuelles. Elles peuvent être définies par des points d'intérêt, des régions, des contours, *etc*. Un autre choix courant consiste à considérer tous les pixels de l'image ou un sous-échantillonnage régulier, on parle alors de grille dense.

Une seconde phase, l'extraction des *caractéristiques visuelles* (ou calcul des *descripteurs visuels*), est utilisée pour décrire le contenu associé à chaque primitive. Les caractéristiques visuelles sont très variées. Smeulders *et al.* [SWS+00] proposent un état de l'art des principaux descripteurs utilisés pour les systèmes CBIR. Ces descripteurs sont regroupés selon qu'ils s'intéressent plus particulièrement à la couleur [Car99], à la texture [HKLO+00] ou à la forme. Fournier [Fou02] ajoute une quatrième catégorie distinguant les descripteurs SIFT [Low04] de la texture. Le descripteur SIFT est un descripteur très populaire qui a été introduit par Lowe [Low04] pour décrire des points d'intérêt. Cette description est obtenue en concaténant les histogrammes d'orientation des gradients de zones de l'image autour du point d'intérêt. En général, les SIFT sont calculés sur une grille de 4x4 et des histogrammes de 8 orientations de gradient, et sont donc composés de 128 valeurs. Ces descripteurs peuvent également être utilisés pour décrire des régions statiques (grille régulière). Il a également inspiré le descripteur GIST [OT06] qui permet de décrire une image dans sa globalité. Des détecteurs SIFT couleurs ont été introduits par [BG09, vdSGS10]. L'information couleur est obtenue en calculant un SIFT par canal couleur. Les SIFT couleurs sont donc de taille 384.

La dernière étape consiste à calculer les *signatures* ou *index*. Les *caractéristiques visuelles* ne représentent pas toujours directement des signatures d'images pertinentes, et peuvent nécessiter d'être traitées pour fournir cet *index* ou cette *signature*. Une possibilité est de représenter chaque image par un vecteur, mais on rencontre aussi des systèmes où les signatures sont des ensembles de vecteurs, des graphes, *etc*. Des informations spatiales peuvent également être ajouter lors de cette étape [RLJ09].

Deux approches courantes sont possibles pour construire les signatures, l'approche globale et l'approche locale.

L'approche globale consiste en l'accumulation de l'information sur toute l'image pour la formation de la signature. Un type de signature globale très courant est l'histogramme ou le *sac de mots* (*Bag of Word* ou *BoW* en anglais). Le calcul de ces signatures se fait en utilisant un dictionnaire de mots visuels, par exemples des couleurs [MM99], des textures [FCPF01] ou des mots SIFT [SZ03], *etc*. Ainsi, pour chaque image de la base, il est possible de calculer un histogramme sur cette référence commune. La détermination de ce dictionnaire se fait généralement par une quantification vectorielle de l'espace des caractéristiques visuelles de toute la base [GCPF08b]. Cette étape analyse la description de toutes les images afin de détecter les caractéristiques communes à toute la base ; par exemple, une palette des couleurs les plus utilisées dans la base. A un facteur de normalisation près, l'histogramme constitue une approximation de la densité associée à l'image, vue comme une variable aléatoire. Cette approche statistique est proche de la recherche textuelle pour laquelle les documents sont couramment caractérisés par la fréquence d'apparition de mots clés.

Une image peut être recadrée, avoir subi des incrustations et on veut pouvoir la reconnaître tout de même. Les descripteurs globaux sont mal adaptés pour faire face à ces transformations.

L'approche locale consiste à construire une signature sous la forme d'un ensemble de caractérisations pour chaque primitive de l'image, par exemple la couleur pour chaque région de l'image ou un descripteur SIFT pour chaque point d'intérêt. On parle alors de *sacs de caractéristiques* (*Bag of Feature* ou *BoF* en anglais).

2.1.2 Fonction de similarité

Afin de comparer l'image requête avec les images de la base, il est nécessaire de définir une fonction de similarité. Cette fonction permet de mesurer finement la similitude entre deux images à partir de leur signature en leur attribuant un score de similarité. Il existe de nombreuses fonctions de similarité, allant d'une simple distance entre vecteurs, à des fonctions *ad hoc* dédiées à un type particulier de caractérisation visuelle [6].

Distance. Lorsque la base est indexée avec des signatures vectorielles, telles que des histogrammes, une pratique courante est d'utiliser des distances, composante à composante, telles que les normes l_p. Cette pratique suppose que les composantes des histogrammes à comparer sont dans le même référentiel. Cependant, cette hypothèse est violée dans de nombreux cas à cause de la quantification, du changement de luminosité des images... Les distances composante à composante dépendent de la

[6] On renvoie le lecteur à [LRB09] pour un panorama détaillé sur la problématique de la similarité.

taille des vecteurs. Si la taille est faible, la distance est robuste mais elle est peu discriminante. Si la taille est importante, la distance est discriminante mais elle n'est pas robuste.

Les distances, telles que la distance EMD (de l'anglais Earth Mover's Distance) [RTG00], qui prennent en compte les relations croisées entre composantes permettent d'être à la fois robustes et discriminantes. Cependant, elles ont une complexité calculatoire très importante.

Dans beaucoup de représentations sous forme d'histogrammes, la différence entre les composantes ayant de grandes valeurs est moins importante que la différence entre les composantes ayant de petites valeurs. La distance χ^2 permet de prendre en compte ce problème. Elle est définie par :

$$score(\mathbf{x},\mathbf{y}) = \chi^2(\mathbf{x},\mathbf{y}) = \sum_i \frac{(\mathbf{x}_i - \mathbf{y}_i)^2}{(\mathbf{x}_i + \mathbf{y}_i)} \tag{I.1}$$

La distance χ^2 est issue du test statistique du χ^2 [SC67]. Ce test permet de mesurer l'adéquation entre deux variables (dans les tables de contingence). Tout d'abord introduit, dans le contexte de la recherche textuelle, par LeBart dans [Esc78], la distance χ^2 a été utilisée avec succès dans le cadre de la classification d'images [SC96, CHV99, GCPF08b].

Dans le cas où $\mathbf{x}$ et $\mathbf{y}$ sont des vecteurs de $\mathbb{R}^d$, une fonction de similarité très simple est un produit scalaire $\langle .,. \rangle$ dans $\mathbb{R}^d$: $score(\mathbf{x},\mathbf{y}) = \langle \mathbf{x},\mathbf{y} \rangle$.

Pour pouvoir utiliser un produit scalaire dans le cas où les signatures ne sont pas des vecteurs, il faut d'abord projeter les signatures dans un espace hilbertien $\mathcal{H}$ a l'aide d'une fonction d'induction ϕ, puis opérer un produit scalaire dans cet espace : $score(\mathbf{x},\mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$.

Similarité noyau. Cette transformation ϕ peut aussi permettre de « réarranger » les données de manière à ce que la similarité soit plus satisfaisante par rapport à la sémantique associée aux images.

On peut faire la combinaison des deux étapes (ϕ puis $\langle .,. \rangle$) en une seule fonction appelée noyau [SS02] :

$$k(\mathbf{x},\mathbf{y}) = score(\mathbf{x},\mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$$

La théorie des noyaux est un formalisme puissant qui permet de définir des similarités entre des éléments d'une manière très élégante, dont l'intérêt est triple. En premier lieu, il masque la fonction d'induction ϕ, ce qui évite d'avoir à expliciter l'induction ϕ vers un espace hilbertien $\mathcal{H}$. Il est alors possible de travailler avec des noyaux implicites pour lesquels l'existence d'une fonction ϕ est garantie mais où ϕ peut être inconnue ou de complexité trop importante pour être exploitable. Ensuite il permet de travailler avec des données d'entrées éventuellement non vectorielles. Enfin, il permet de modifier la représentation des données (les signatures) de façon non linéaire (c'est à dire que la fonction ϕ peut être non linéaire).

Ce dernier point est une des principales raisons de l'intérêt croissant des fonctions noyaux : comme une fonction noyau est un produit scalaire (dans un espace induit), on peut remplacer les produits scalaires des algorithmes de décision (tel que la régression, les réseaux de neurones, les Supports à Vaste Marge ...) par un noyau. Cette substitution permet alors de construire des algorithmes qui ont la capacité d'évaluer des fonctions de décision non-linéaire. Ces algorithmes, munis de ce nouveau potentiel, ont été utilisés avec succès dans un grand nombre de domaines utilisant la classification supervisée ou non supervisée.

La conception de noyaux est un problème complexe faisant l'objet de nombreuses études. Il existe des travaux tentant de créer des noyaux ayant des propriétés intéressantes, comme par exemple l'invariance en échelle.

Parmi les fonctions noyaux les plus populaires, nous pouvons mentionner les noyaux gaussiens (ou l_2-RBF) : $k(\mathbf{x},\mathbf{y}) = exp(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2})$, les noyaux polynomiaux : $k(\mathbf{x},\mathbf{y}) = (\mathbf{x}\cdot\mathbf{y}+1)^q$ et les noyaux χ^2-RBF : $k(\mathbf{x},\mathbf{y}) = exp(-\frac{\chi^2(\mathbf{x},\mathbf{y})}{2\sigma^2})$. Récemment, des noyaux ont été définis pour différents types de signatures, comme les BoF [GCPF07].

2.2 Moteurs de recherche

2.2.1 Système de détection de copies

Des institutions comme les musées, les agences de presse, les agences de photos, les archives, disposent de grands jeux d'images associées à des méta-données renseignant sur la nature des documents. Il leur est nécessaire d'identifier des photos de presse, ou d'œuvres d'art... ayant des références trop sommaires ou parfois incorrectes, voire inexistantes. Il peut également leur être demandé de vérifier si une image est protégée par un copyright, de retrouver le lieu où a été pris une photo... Dans ces cas, la seule source disponible ou fiable pour l'identification de l'original et la récupération de ses méta-données est le document visuel lui-même.

Comme les collections contiennent une grande quantité d'images, la recherche manuelle est impossible. Il est alors nécessaire de concevoir des systèmes de recherche capables de retrouver rapidement l'image désirée dans un grande collection à partir d'une image requête. L'image requête est rarement une stricte copie de l'image à retrouver. En effet, elle a pu subir des déformations telles qu'un changement de taille, un recadrage, une rotation, un changement d'illumination, un étirement, une compression...

Les systèmes conçus pour répondre à ces problématiques sont appelés des systèmes de *détection de copies*.

Des premières solutions industrielles sont aujourd'hui à disposition du grand public : kooaba[7], milpix[8], ltu technologies[9]... Elles permettent, par exemple, de retrouver des informations sur un livre et de l'acheter directement à partir d'une photo de la couverture prise avec son téléphone portable.

Le succès de cette approche pour la détection de copies en a fait une candidate crédible pour la recherche d'un même objet pris dans un environnement différent ou sous des angles de vues très variés. On parle alors de systèmes *de détection de copies proches* [CPZ08].

Un système très efficace a été proposé par Lowe [Low04] : pour chaque image de la base, identifier un grand nombre de PoI, calculer un descripteur local autour de ces points et stocker les descripteurs dans un index. Les images de la base sont donc représentées avec des signatures de types BoF. Ces étapes sont réalisées en prétraitement (hors ligne). L'utilisateur interroge le système à l'aide d'une image requête. Dans un premier temps, le même processus d'extraction et de description des PoI est appliqué à l'image requête. En utilisant l'index, chaque descripteur de l'image requête est mis en correspondance avec un petit nombre de descripteurs de la base (les plus semblables). Chaque descripteur ainsi retrouvé donne un vote à l'image associée. Les images ayant le plus de votes sont candidates. La seconde étape consiste à analyser plus finement la similarité entre l'image requête et les images candidates. Cette analyse utilise un a priori fort, propre à la détection de copies : si l'image candidate est bien une copie de l'image originale, il existe une transformation géométrique entre ces deux images. La similarité est donc mesurée en utilisant un algorithme, tel que RANSAC (de l'anglais *RANdom SAmple Consensus*) [FB81], qui mesure la cohérence géométrique entre les images sélectionnées et l'image requête. L'algorithme estime la transformation géométrique entre l'ensemble des PoI de l'image requête et celui de l'image cible. Les votes issus des appariements ne correspondant pas à cette transformation sont supprimés. Puis les votes sont recomptés et les images sont triées en fonction du nombre de votes restants. Toutes ces étapes sont réalisées à chaque image requête fournie au système (en ligne).

Le schéma global d'un système de détection de copies est présenté sur la Figure I.2.

La raison pour laquelle la méthode est si robuste vient de la multitude de descripteurs utilisés pour décrire les images. En effet, si certains appariements sont perdus en raison des occultations ou des découpages de l'image, les autres descripteurs permettent de garantir de bons résultats. D'autre part, même si de faux appariements sont obtenus, donnant des votes pour des images différentes de l'image requête, seules les images correctement identifiées recoivent une quantité significative de votes.

Malheureusement, la multiplication des descripteurs ainsi que l'utilisation de descripteurs de grande dimension pénalisent la durée d'exécution. Des centaines, voire des milliers, d'appariements

[7] http ://www.kooaba.com/
[8] http ://www.milpix.com/fr/solutions-technologiques-de-recherche-visuelle/
[9] http ://www.ltutech.com/fr/

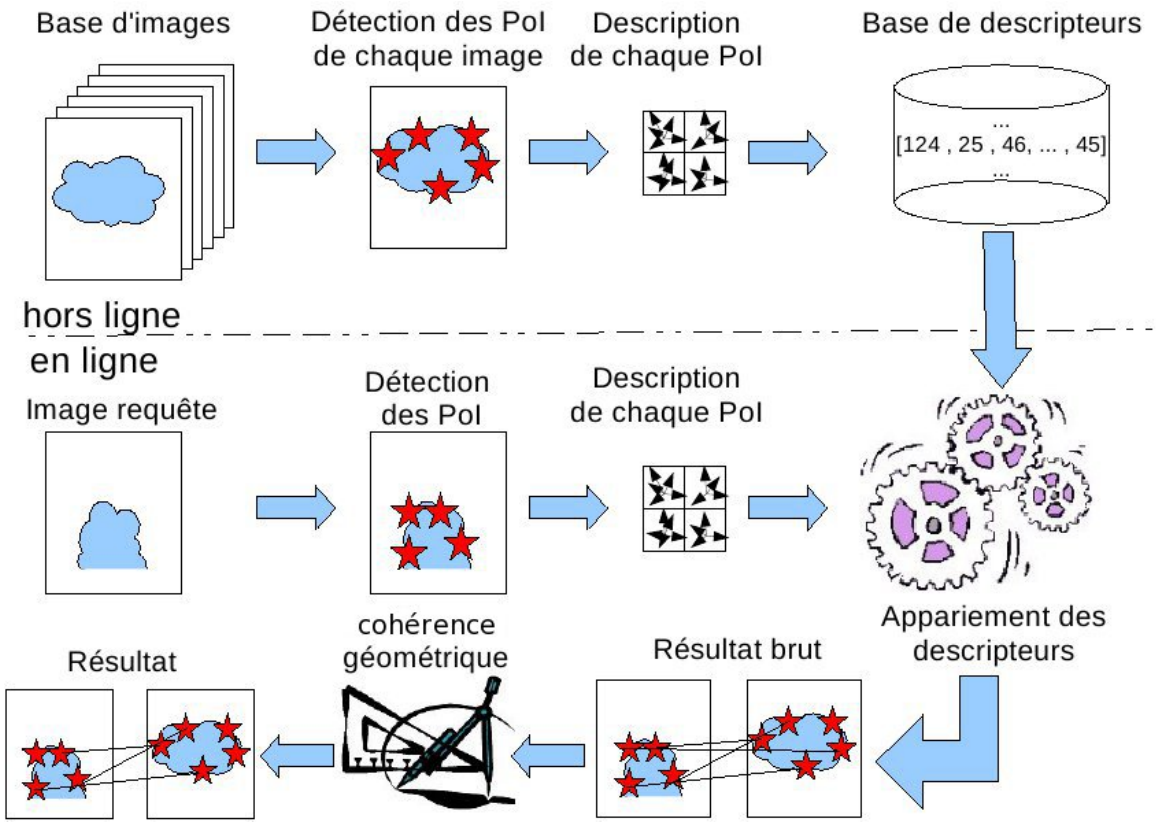


Figure I.2 – Système de détection de copies [Val08]

doivent être trouvés pour retrouver une image.

2.2.2 Similarité sémantique et recherche interactive

D'autres applications peuvent nécessiter la recherche d'images qui ne sont pas uniquement des copies de l'image requête mais qui partagent un certain nombre de similitudes. C'est le cadre de la recherche par *similarité sémantique* qui visent à trier au mieux la base à partir d'une image requête ou à extraire un sous-ensemble d'images appelé *classe* ou *concept*.

Les techniques supervisées ou semi-supervisées s'appuient sur une base d'apprentissage, un ensemble d'images exemples dont les catégories sont connues, pour construire des détecteurs à l'aide de méthodes d'apprentissage statistique, définissant ainsi un *système de classification*. L'idée est de construire un détecteur pour chaque catégorie. Une autre approche consiste à construire le détecteur « à la demande », c'est-à-dire pendant le processus de recherche. On parle alors de *système de recherche interactive*.

Les techniques de recherche interactive permettent d'améliorer itérativement la recherche en de-

mandant à l'utilisateur de fournir des informations au système dans le but de préciser la nature de sa recherche. Les informations fournies par l'utilisateur sont exploitées pour améliorer le détecteur dans une phase dite de *bouclage de pertinence* (*relevance feedback* en anglais).

Plusieurs méthodes ont été proposées dans la littérature [RHMO97, HZ01, CGJ96, MMMB07]. Dans cette thèse, nous nous intéressons aux méthodes basées classification [TC01]. Le schéma global d'un système de recherche interactive est présenté sur la Figure I.3. La manière la plus simple d'anno-

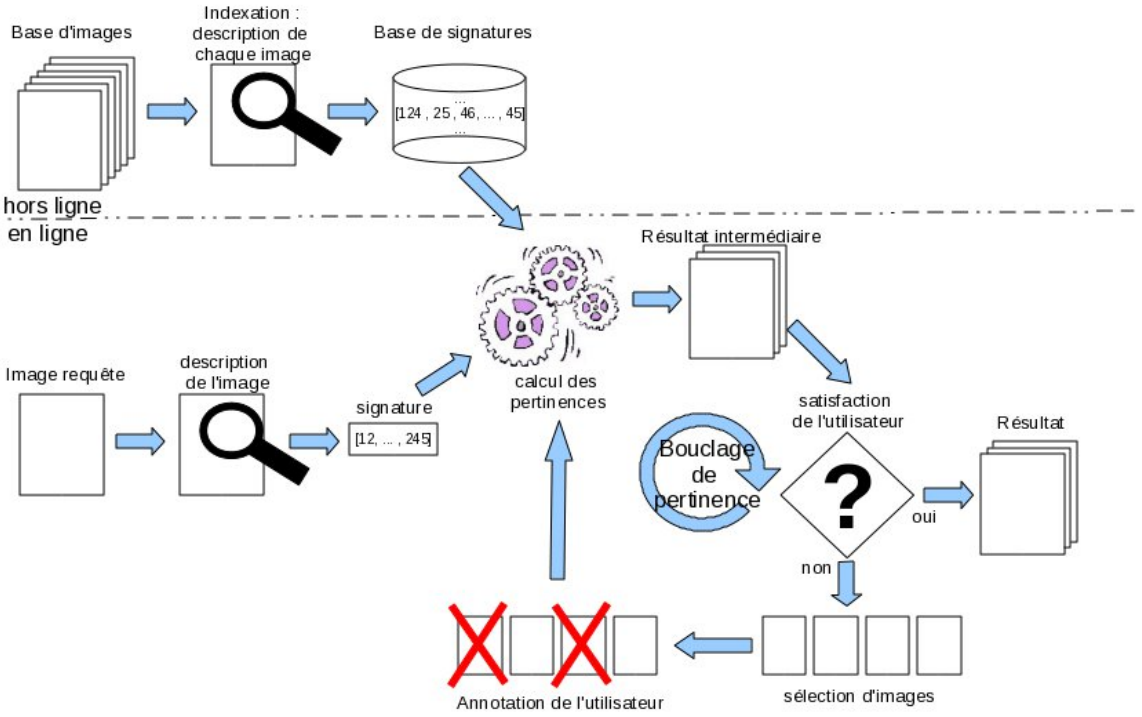


Figure I.3 – Système de bouclage de pertinence

ter une image est de spécifier si cette dernière appartient ou non à la catégorie recherchée (annotation binaire). Une image est dite pertinente lorsqu'elle appartient à la catégorie recherchée, et non pertinente dans le cas contraire. Les images annotées permettent de construire l'ensemble d'apprentissage, noté $\mathcal{A}$, formé par les couples $(\mathbf{x}_i, y_i)$, où $\mathbf{x}_i$ est l'image d'indice i et y_i son annotation :

$$y_i = \begin{cases} 1 \text{ si, } \mathbf{x}_i \text{ est pertinente} \\ -1 \text{ si, } \mathbf{x}_i \text{ est non pertinente} \end{cases}$$

Les deux étapes clefs de ce schéma sont la mise à jour de la fonction de classification et/ou de pertinence et l'étape de *sélection* des images à présenter à l'utilisateur pour annotation.

Calcul des pertinences. Lors de cette étape, l'ensemble d'apprentissage $\mathcal{A}$ est utilisé pour construire une fonction de pertinence capable d'estimer le degré d'appartenance de chacune des images de la base $\mathcal{B}$ à la catégorie recherchée par l'utilisateur.

Un grand nombre de techniques de classification supervisée ont été proposées en CBIR interactif (la classification par critère de Bayes [Vas00], K-plus proches voisins [BAG03], SVM [VV98], boosting [TV04]). Une étude comparative des techniques les plus utilisées dans la communauté de la recherche d'images est présentée dans [Gos05] et montre que le classifier SVM y est particulièrement performant [GCPF08a]. Dans ce contexte, la fonction de pertinence est définie par :

$$f_{\mathcal{A}}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$$

où $\mathbf{x}$ est l'image d'intérêt, b est un réel et $\mathbf{w}$ est la normale à l'hyperplan séparateur qui maximise la marge entre les catégories. Lorsqu'on utilise une similarité noyau, $f_{\mathcal{A}}$ s'écrit :

$$f_{\mathcal{A}}(\mathbf{x}) = \sum_{i \in I} y_i \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b \qquad \forall i, \alpha_i > 0$$

avec I l'ensemble des indices des images appartenant à $\mathcal{A}$.

La fonction de pertinence est normalisée pour être à valeurs dans $[-1,1]$. Cela permet de distinguer trois cas différents : l'appartenance à la catégorie recherchée (proche de 1), la non-appartenance à la catégorie recherchée (proche de -1), et l'incertitude (proche de 0). Ce formalisme est utile lors de l'étape de sélection des images à annoter.

Classification active : sélection des images à annoter. Il existe plusieurs manières de *sélectionner* les exemples. Une technique très courante est de présenter à l'utilisateur les images les plus pertinentes. Cependant, cette solution est peu efficace car elle n'apporte que de la redondance à l'ensemble d'apprentissage. Le cadre d'étude de l'*apprentissage actif* traite ce problème en proposant d'autres critères que la pertinence pour sélectionner les images [CGJ96, TK02, Bri03].

Très étudiée en CBIR depuis 10 ans, la classification dite *active* exploite les données non annotées pour proposer à l'utilisateur un choix particulier d'image à annoter (voir [CC08] pour un panorama de ce type de méthodes).

Une approche très répandue en apprentissage actif est la sélection basée sur l'incertitude du classifieur. Selon cette approche, l'individu le plus intéressant est celui que le classifieur a le plus de mal à classifier [TC01] :

$$\mathbf{x}^\star = \arg\min_{i \in \bar{I}} |f_{\mathcal{A}}(\mathbf{x}_i)|$$

avec $\bar{I}$ l'ensemble des indices des images non annotées.

Cette méthode autorise l'annotation d'une seule image par itération du bouclage de pertinence. Cependant, au sein d'un système de recherche d'images, vue la rapidité de décision d'un utilisateur, il peut être intéressant de sélectionner plusieurs images à chaque itération.

Afin de pallier ce problème, un critère souvent utilisé est celui de la *diversité*. L'idée est de sélectionner un lot d'images différentes, tout en préservant l'intérêt de chaque image d'être sélectionnée. Par exemple, il n'est pas intéressant de sélectionner les q images les plus proches de la frontière du SVM si ces q images sont très similaires (*i.e.* dans une même zone de l'espace de séparation linéaire). En effet, dans le cas où l'image optimale est entourée par d'autres images très proches, dans la stratégie précédente, toutes ces images seront proposées à l'utilisateur. Etant donnée la puissance des classifieurs, annoter toutes ces images donnera le même résultat que si une seule d'entre elles est annotée, d'où une perte en terme d'efficacité du système.

Dans [Bri03], Brinker choisit les images les plus orthogonales entre elles (cosinus de l'angle entre leurs vecteurs de représentations proche de 0), tout en s'assurant qu'elles soient proches de la frontière. Cette technique est nommée Diversité des Angles. Pour le choix d'une nouvelle image, la méthode minimise la somme pondérée de la distance à la frontière et du plus grand cosinus entre l'image testée et les images déjà sélectionnées :

$$s_{I^\star}(\mathbf{x}_i) = \quad \lambda |f_{\mathcal{A}}(\mathbf{x}_i)| \quad + \quad (1-\lambda) \max_{j \in I^\star} \frac{k(\mathbf{x}_i, \mathbf{x}_j)}{\sqrt{k(\mathbf{x}_i, \mathbf{x}_i) k(\mathbf{x}_j, \mathbf{x}_j)}} \tag{I.2}$$

avec $I^\star$ initialisé à $\bar{I}$, et λ un réel entre 0 et 1. Une fois qu'une image a été choisie, on ajoute à $I^\star$ l'indice de l'image choisie, puis on minimise de nouveau $s_{I^\star}$ pour obtenir une nouvelle image éloignée à la fois des images de l'ensemble d'apprentissage et des images précédemment sélectionnées. Ces opérations sont répétées autant de fois qu'il y a d'images à choisir.

3 Passage à l'échelle

Avec l'augmentation de la taille des bases d'images, trouver rapidement une image semblable à une image requête dans une grande collection est devenu un problème crucial aussi bien pour les systèmes de recherche par similarité que pour les systèmes de recherche interactive.

3.1 Complexité de la recherche par similarité

La recherche par similarité consiste à calculer un score de similarité entre l'image requête et toutes les images de la base à l'aide d'une fonction de similarité. La complexité de cette approche est en $O(n)$

où n représente le nombre de calculs élémentaires, ici, le nombre de calculs de scores de similarité. Il est intéressant de constater que n représente également la taille de la base. On a donc une estimation de l'évolution des temps de recherche en fonction de la taille de la base. La recherche est dite linéaire ou exhaustive.

3.2 Complexité de la recherche interactive

La recherche interactive comprend trois étapes principales : l'étape d'apprentissage, l'étape d'évaluation et l'étape de sélection des images à annoter.

La complexité de l'étape d'apprentissage, qui permet d'obtenir les coefficients α_i pour chacun des exemples d'apprentissage $\mathcal{A}$, est en $O(|\mathcal{A}|^2)$. Etant donné qu'en apprentissage interactif, la taille de l'ensemble d'apprentissage est très faible ($|\mathcal{A}| \ll n$), la complexité de cette étape n'est pas problématique. Cependant, si le nombre d'itérations et le nombre d'annotations par itération du bouclage de pertinence devient trop important, la taille de $\mathcal{A}$ peut devenir trop grande pour négliger la complexité de cette étape. Dans ce cas, des optimisations du SVM comme LASVM [BEWB05] peuvent être considérées.

La complexité de l'étape d'évaluation, qui consiste à calculer la pertinence de chaque image de la base $\mathcal{B}$, est, quant à elle, en $O(n)$. De même, la complexité de l'étape de sélection des images à annoter, qui consiste à calculer le degré d'incertitude de chaque image de la base, est également en $O(n)$. La complexité du système de bouclage de pertinence a donc une complexité en $O(n)$.

3.3 Accélération de la recherche

Confrontée à l'augmentation importante de la taille des bases d'images, l'utilisation d'une structure d'index qui permet de sélectionner directement un sous-ensemble contenant les images les plus similaires à l'image requête s'impose. Ainsi, lorsque l'on effectue une recherche à partir d'une image requête, le calcul de la similarité entre l'image requête et toutes les images de la base peut être approximée par le calcul sur ce sous-ensemble. La structure d'index sélectionne un pool de candidates et le système calcule uniquement la similarité entre ce pool d'images et l'image requête. La recherche a alors une complexité sous-linéaire qui s'écrit $O(n^\rho)$, avec $\rho < 1$ ou selon les cas en $O(log(n))$. n^ρ ou $log(n)$ correspond à la taille du pool d'images sélectionnées par la structure d'index.

Un grand nombre de structures d'index a été proposée dans la littérature pour effectuer des recherches rapides. Le type de structure d'index à considérer dépend directement du type de caractéristiques visuelles extraites pour décrire les images. Ainsi, lorsque les images sont décrites par des vecteurs creux (contenant majoritairement des valeurs nulles), l'utilisation de structures d'index, comme

les fichiers inversés [SMMP00], permettent d'envisager d'effectuer des recherches rapides exactes.

En raison du phénomène de la « malédiction de la dimension » [BK59, BBK01], lorsque les images sont décrites par des vecteurs denses (contenant très peu de valeurs nulles), le temps de recherche des structures d'index grandit exponentiellement avec la dimension du descripteur. Les performances des structures d'index proposées dans la littérature (KD-tree [Ben75]) pour effectuer des recherches rapides exactes deviennent plus mauvaises que la recherche exhaustive dès lors que l'espace de représentation dépasse 10 dimensions [BBK01]. Les méthodes de *recherche approximée* se sont montrées très pertinentes pour surmonter cette difficulté. Ces systèmes visent à trouver le meilleur compromis entre la diminution de la complexité de la recherche et la dégradation de la qualité des résultats. Une classification de ces méthodes a été proposée dans [Val08].

Dans cette thèse, nous nous sommes focalisés sur l'adaptation des algorithmes de recherche approximée aux différents systèmes CBIR afin d'effectuer des recherches dans les grandes collections. Nous nous intéressons tout particulièrement à la famille de méthode de hachage LSH (Locality Sensitive Hashing) [IM98].

4 Evaluation des systèmes de recherche

4.1 Protocoles expérimentaux

L'efficacité des méthodes de recherche approximée se mesure selon deux critères :

- la précision : le taux d'images retrouvées par la recherche approximée par rapport aux images retrouvées par la recherche exacte (recherche exhaustive),
- la rapidité : le gain en temps obtenu par la recherche approximée par rapport à la recherche exacte.

Ces deux critères sont liés : plus la recherche est précise moins elle est rapide (et inversement). La précision et la rapidité de recherche dépendent du paramétrage de l'algorithme de recherche. C'est pourquoi, l'analyse des performances des méthodes de recherche approximée sont données en étudiant l'effet des paramétrages.

La pertinence d'une méthode de recherche approximée dépend également de ses besoins en ressources mémoires. Les gains en temps de calcul sont souvent liés au nombre de données pré-calculées qui sont stockées en mémoire vive avant chaque recherche. Cette quantité de données dépend également de la taille des bases traitées. Comme on cherche à traiter les grands volumes de données, on doit limiter l'utilisation de cette ressource.

L'efficacité des méthodes de recherche sémantique est donnée par :

- la qualité de recherche : La métrique la plus utilisée pour mesurer ce critère est la mesure du

MAP (cf. Annexe A.2). Pour réaliser cette évaluation, la base utilisée doit être munie d'une *vérité terrain*. La vérité terrain définit la catégorie à laquelle appartient chaque image de la base. Elle est alors utilisée comme référence pour mesurer la qualité de la réponse des systèmes de recherche.

4.2 Bases d'évaluation

Avec l'essor des systèmes de recherche proposés par la communauté, des chercheurs et des instituts ont mis à disposition des bases publiques ainsi que des compétitions internationales pour répondre au besoin urgent d'évaluation des performances des systèmes de recherche d'images.

On distingue deux types de bases, les bases utilisées pour l'évaluation des systèmes de détection de copies et les bases de catégories d'images utilisées pour l'évaluation des systèmes de classification. Il n'existe pas de bases spécifiques pour les systèmes d'apprentissage interactif. Ces systèmes sont généralement évalués avec les bases de catégories d'images.

Les bases les plus utilisées par la communauté se limitent aux milliers d'images (cf. Annexe A.1). Les deux principaux freins à la construction de très grandes bases d'images (de l'ordre du million d'images) sont la construction d'une vérité terrain qui demande une annotation manuelle de chaque image et les droits d'auteur qui limitent la distribution des images.

Une solution pour palier à ce problème est de construire des bases à partir de requêtes textuelles sur des moteurs de recherche disponibles sur Internet comme Google Image. Des bases de plusieurs millions d'images ont ainsi été proposées comme TinyImage [TFF08] (80 millions d'images) et ImageNet [DDS+09] (3,2 millions d'images). Cependant, les images au sein d'une même catégorie peuvent être très variables et il devient alors difficile de contrôler la difficulté des bases.

La seconde solution est d'utiliser une base annotée et d'y incorporer des distracteurs, des images non annotées dont on doit juste s'assurer qu'elles ne sont pas classées dans les catégories recherchées. C'est cette seconde solution que nous avons utilisée pour construire nos bases d'évaluation.

5 Contributions et plan de la thèse

L'objectif de la thèse est de proposer des solutions pour diminuer la complexité des méthodes de recherche par le contenu dans les bases d'images. Comme vu précédemment, les fonctions de similarité utilisées sont différentes suivant que les images sont indexées avec une approche globale ou locale. D'autre part, le problème du « fossé sémantique / numérique » nécessite l'utilisation d'un moteur de recherche plus puissant comme la recherche interactive.

5.1 Passage à l'échelle de la recherche par similarité basées signature globale.

Nous nous sommes intéressés à une famille de méthodes de recherche approximée très populaires, *Locality Sensitive Hashing* (LSH) [IM98], qui fournit un cadre théorique basé sur des fonctions de hachages « *local-sensitivity* ». Le schéma LSH a été utilisé avec succès dans de nombreuses applications multimédia [KSH04, GPCPF08]. C'est pourquoi nous avons considéré cette structure d'index dans nos travaux.

Toutes ces méthodes permettent de retrouver les images les plus proches d'une requête dans un espace muni d'une métrique avec une complexité de recherche sous-linéaire par rapport au volume de données. Ces méthodes sont souvent conçues pour les distances usuelles telles que les distances l_1 et l_2, la distance du cosinus ou la distance de Hamming, ce qui limite les fonctions de similarité exploitables. Il a été montré empiriquement [SC96, CHV99, GCPF08b], que pour les tâches de recherche d'images, d'autres distances, comme la distance χ^2, sont plus performantes pour certains types de descripteurs globaux (comme les histogrammes couleur).

Après avoir présenté, au chapitre II, le principe de LSH et les principales fonctions de hachage, nous présentons, au chapitre III, une nouvelle fonction de hachage conçue pour la distance χ^2.

Ces travaux ont été publiés dans :
[GCP10c]
[GCP10b]
et soumis à :
[GCP10a]

5.2 Passage à l'échelle de la recherche par similarité basée signature locale.

La représentation des images en utilisant une signature BoF fournit une description beaucoup plus riche que les signatures globales. Cependant, cette approche nécessite de définir des fonctions de similarité plus complexes.

Il est alors nécessaire d'adapter ou de définir de nouvelles structures d'index spécifiques pour diminuer la complexité des algorithmes de recherche par similarité.

Au chapitre IV, un état de l'art des fonctions de similarité adaptées au BoF est proposé, avant de décrire deux adaptations du LSH permettant de diminuer la complexité de la fonction de similarité donnant la meilleure qualité de recherche. Dans ce même chapitre, un formalisme unifiant les différentes approches est introduit. Enfin, au chapitre V, nous présentons une évaluation de nos approches.

Ces travaux ont été publiés dans :
[GCPPF08]

5.3 Passage à l'échelle de la recherche interactive par bouclage de pertinence.

Pour combler le « fossé sémantique / numérique » plusieurs approches d'apprentissage interactif ont été proposées. Nous nous intéressons au bouclage de pertinence, avec une approche statistique.

Le problème du passage à l'échelle est bien plus critique que pour la recherche par similarité. En effet, une nouvelle recherche s'opère à chaque bouclage de pertinence. De plus, pour que l'interaction entre l'utilisateur et le système soit possible, le temps de réponse de chaque itération doit être quasiment instantané et ce quelle que soit la taille de la base. Or, le problème du passage à l'échelle se pose aussi bien pour l'étape de sélection des images à annoter que pour l'étape d'évaluation de la fonction de pertinence.

Nous présentons au chapitre VI, un état de l'art des systèmes de recherche interactive s'attaquant au problème du passage à l'échelle avant de présenter notre solution, nommée SALSAS (de l'anglais Sub-linear Active Learning Strategy with Approximate KNN Search). Enfin, au chapitre VII, nous introduisons une évaluation de notre approche.

Ces travaux ont été publiés dans :
[GCP09]
[GCP10c]
[GCP10b]

Première partie

Signature globale et recherche rapide dans les grandes bases d'images

Comme nous l'avons vu dans l'introduction, l'utilisation de structures d'index pour effectuer des recherches par similarité dans les bases d'images de grande taille s'impose. D'autre part, l'étape de description des images demande l'extraction d'un grand nombre de caractéristiques visuelles afin de réduire le « fossé sémantique / numérique ». Les images de la base sont alors décrites avec des vecteurs de grande dimension. L'efficacité des structures d'index classiques, réalisant des recherches exactes, s'effondre en grande dimension. Il est donc nécessaire de travailler avec des algorithmes de recherche approximée qui offrent un bon compromis entre rapidité de traitement et dégradation de la qualité des résultats.

Chapitre II

LSH, une structure d'index pour accélérer la recherche par similarité

Dans ce chapitre, nous nous intéressons aux structures d'index utilisées pour effectuer des recherches rapides dans les bases d'images de grande taille. Un aperçu rapide des principales méthodes de recherche est fourni avant de détailler les approches de type hachage et en particulier la famille de méthodes LSH (*Locality Sensitive Hashing*). Après avoir présenté les propriétés théoriques de LSH, l'algorithme de recherche est décrit. Enfin, nous proposons un état de l'art des méthodes appartenant à cette famille.

Sommaire

1 **Structures d'index pour CBIR** 27
2 **Table de hachage** 29
3 **Locality-Sensitive Hashing : adaptation du hachage à la recherche des plus proches voisins** 31
4 **Algorithmes de recherche** 32
5 **Principales fonctions de hachage** 36

1 Structures d'index pour CBIR

Une classification des méthodes d'indexation a été proposée dans [Val08] :

– Division spatiale : cette approche divise l'espace des caractéristiques en un certain nombre de cellules, normalement des hyper-rectangles. La distribution des données est prise en considération en divisant plus finement les régions très peuplées de l'espace. L'objectif est de limiter l'exploration de l'espace aux cellules qui sont les plus proches de la requête. Une des méthodes de division spatiale de référence est Best-Bin-First [BL02].

– Division des données : cette approche divise les données en cellules, qui peuvent se chevaucher dans l'espace. De nouveau, l'objectif est de limiter l'exploration à quelques cellules, celles qui sont les plus probables de contenir le jeu de réponses. La première méthode de division de données est le R-tree [Gut84], qui groupe les données dans des boîtes englobantes hyper-rectangulaires (avec un chevauchement possible). La plupart des méthodes de division de données souffrent de la difficulté de choisir une bonne heuristique de division. Les divisions idéales ont un nombre de points uniforme (pour faciliter le stockage), un petit volume et ne se chevauchent pas (pour optimiser la sélectivité). Cependant, si la méthode essaye trop durement d'optimiser la division, le coût de construction de l'index peut devenir prohibitif.

– Clustering : cette approche consiste à grouper les données similaires dans les mêmes cellules. L'objectif est de séparer au mieux les données en cellules les plus compactes possibles. Une des méthodes de clustering de référence est Clindex [LCGM+00].

– Métrique : cette famille de méthodes utilise seulement les distances entre les points et n'est pas limitée aux espaces vectoriels. Toutes ces méthodes séparent les données dans des groupes de points qui sont les uns proches des autres et utilisent l'inégalité triangulaire pour élaguer les groupes qui n'ont pas besoin d'être considérés pour une requête particulière. Une méthode basée métrique bien connue est le M-tree [CPZ97].

– Approximation des données : cette approche quantifie les données et considère uniquement quelques bits par dimension. Si les données sont uniformément distribuées et non corrélées, ces techniques fonctionnent très bien, puisque la sélectivité s'améliore lorsque la dimension grandit. Une des méthodes d'approximation des données de référence est VA-file [WSB98].

– Projection et hachage : ces méthodes projettent les données dans des espaces de plus petites dimensions. La recherche est alors réalisée dans les espace projetés et les résultats partiels sont agrégés pour produire le résultat final. L'espace de projection peut être obtenu de plusieurs façons : en projetant les données sur une droite [FBS06, FKS03], via une fonction de hachage,

en utilisant les Spaces-Filing Curves [GZ+01].

Dans cette thèse nous nous intéressons tout particulièrement à la famille de méthode de hachage LSH (Locality Sensitive Hashing) [IM98].

2 Table de hachage

La table de hachage est une implémentation efficace de la table d'association qui relie chaque élément à une clé permettant de l'identifier. Les éléments sont donc stockés dans une table où chaque case est accessible via une clé. En connaissant la clé, on peut récupérer l'élément qui lui est associé.

2.1 Adressage direct

La technique de recherche la plus efficace dans les tables d'association est l'adressage direct. Cette technique ne peut malheureusement s'appliquer que dans des cas très particuliers. Appelons univers l'ensemble U de toutes les clés possibles. Il faut que l'univers des clés soit de la forme $\{0,\ldots,n-1\}$, où n est un entier pas trop grand, et, d'autre part, que deux éléments distincts aient des clés distinctes. Il suffit alors d'utiliser un tableau de taille n pour représenter la table d'association contenant n éléments. L'élément associé à la clé k est ainsi stocké dans la k^{ieme} case du tableau. Cette case est appelée *bucket* k. La recherche s'effectue avec un seul accès aux données. La complexité de l'adressage direct est donc en $O(1)$ et ne dépend pas du nombre de données gérées.

Ce cas s'applique par exemple à la base de données des résultats d'une épreuve sportive qui exclut les ex-æquos. Le rang à l'arrivée d'un concurrent peut servir de clé dans la base de données. Mais, en pratique, un cas aussi simple est exceptionnel.

2.2 Hachage

L'idée fondamentale de la table de hachage est de se ramener au cas de l'adressage direct, c'est-à-dire de clés qui sont des indices dans un tableau. Soit m, entier pas trop grand, de l'ordre du nombre d'éléments d'informations que l'on compte gérer.

On se donne une fonction :

$$h : U \rightarrow \{0,\ldots,m-1\} \tag{II.1}$$

appelée fonction de hachage et dont la valeur $h(k)$ est appelée le *hash* de la clé k. L'idée est de ranger l'élément de clé k non pas dans une case de tableau $t[k]$, comme dans l'adressage direct (cela n'a d'ailleurs de sens que si k est un entier), mais dans $t[h(k)]$.

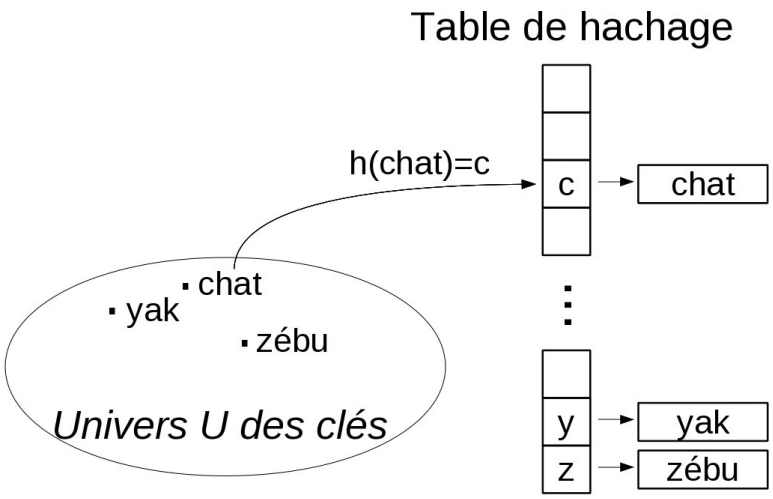


Figure II.1 – Exemple simple de hachage. L'univers U des clés est l'ensemble des chaînes de caractères. La fonction de hachage h associe à la clé, le code ASCII du premier caractère de la chaîne. L'espace d'arrivée est l'intervalle $[0,255]$ en considérant qu'un caractère est représenté par un entier entre 0 et 255.

L'intérêt de la fonction de hachage est qu'elle permet :

– d'optimiser l'occupation mémoire en utilisant un tableau de la taille du nombre d'éléments d'informations,

– d'étendre l'adressage direct aux clés non entières, comme les chaînes de caractères (Fig. II.1).

2.3 Collisions

Le fait de créer un hash à partir d'une clé peut engendrer un problème de *collision*, c'est-à-dire qu'à partir de deux clés distinctes k et k', la fonction de hachage pourrait renvoyer la même valeur de hash, $h(k) = h(k')$, et donc, par conséquent, donner accès au même bucket (la même position dans le tableau). Les collisions étant en général résolues par des méthodes de recherche linéaire qui consistent à comparer les clés des éléments ayant le même hash, une mauvaise fonction de hachage, *i.e.* produisant beaucoup de collisions, va fortement dégrader la rapidité de la recherche. Pour minimiser les risques de collisions, il faut donc choisir soigneusement sa fonction de hachage. D'autre part, il est préférable que la fonction de hachage ne soit pas de complexité élevée (demande peu de calcul).

Il existe différentes approches pour résoudre les collisions, la plus simple est le « checksum ». Par exemple, pour les chaînes de caractères, il est possible d'enregistrer sa taille dans la table. Ainsi, lors d'une recherche, on applique l'opérateur d'égalité sur les chaînes ayant le même hash que la chaîne requête et ayant aussi la même taille.

Avant d'étudier plus en détail la résolution des collisions, nous allons voir comment étendre le hachage à la recherche des plus proches voisins.

3 Locality-Sensitive Hashing : adaptation du hachage à la recherche des plus proches voisins

Dans le cadre de la recherche des plus proches voisins, une fonction de hachage doit hacher les points proches avec la même clé et les points éloignés avec des clés différentes. C'est à partir de cette notion que Indyk et Motwani [IM98] ont introduit la méthode de recherche rapide *Locality-Sensitive Hashing* (LSH). Les auteurs ont défini les propriétés que doivent respecter les fonctions de hachage pour être considérées *Locality-Sensitive*.

3.1 Fonction Locality-Sensitive

Une fonction est dite *Locality-Sensitive* si deux points proches obtiennent le même hash avec une forte probabilité et que deux point éloignés obtiennent le même hash avec une faible probabilité :

Définition 1 *une famille de fonctions* $\mathcal{H}\{h : \mathcal{F} \rightarrow U\}$ *est* (r_1, r_2, p_1, p_2)*-sensitive si on a les propriétés suivantes :*

$$\forall \mathbf{p} \in B(\mathbf{q}, r_1),\ Pr_{h\in\mathcal{H}}[h(\mathbf{p}) = h(\mathbf{q})] \geq p_1 \tag{II.2}$$

$$\forall \mathbf{p} \notin B(\mathbf{q}, r_2),\ Pr_{h\in\mathcal{H}}[h(\mathbf{p}) = h(\mathbf{q})] \leq p_2 \tag{II.3}$$

où $B(\mathbf{q}, r)$ *est la boule de centre* $\mathbf{q}$ *et de rayon* r *et* $\mathcal{F}$ *l'espace des caractéristiques.*

Les équations (II.2) et (II.3) représentent les minorants et majorants de la probabilité de bonne P_{bd} et de fausse détection P_{fd}.

La définition ne fait pas d'hypothèse quant à la mesure D utilisée entre points. En effet, cette définition s'applique pour toute mesure, aussi bien pour des similarités que pour des dissimilarités (ou distances). Lorsque D représente une distance, la boule $B(\mathbf{q}, r)$ est alors définie par $B(\mathbf{q}, r) = \{\mathbf{p} : D(\mathbf{q}, \mathbf{p}) \leq r\}$ et lorsque D est une similarité, la boule $B(\mathbf{q}, r)$ est alors définie par $B(\mathbf{q}, r) = \{\mathbf{p} : D(\mathbf{q}, \mathbf{p}) \geq r\}$. Pour qu'une fonction (r_1, r_2, p_1, p_2)-sensitive soit utilisable, il faut que $p_1 > p_2$ et que $r_1 < r_2$ lorsque D est une distance, et il faut que $p_1 > p_2$ et que $r_1 > r_2$ dans le cas d'une similarité.

Dans [Cha02], Charikar a proposé une réécriture plus resctrictive de la définition 1 dans le cas ou D représente une fonction de similarité :

Définition 2 *une famille de fonctions* $\mathcal{H}\{h : \mathcal{F} \rightarrow U\}$ *est Locality-sensitive si on a la propriété suivante :*

$$Pr_{h\in\mathcal{H}}[h(\mathbf{p}) = h(\mathbf{q})] = D(\mathbf{p}, \mathbf{q}) \tag{II.4}$$

En posant $p_2 = D(\mathbf{p},\mathbf{q}) \leq r_2$ et $p_1 = D(\mathbf{p},\mathbf{q}) \geq r_1$ on retrouve la définition 1 avec $p_1 > p_2$ et $r_1 > r_2$.

On considérera le cas général, donné en définition 1, dans la suite du document avec D représentant une distance.

3.2 Concaténation de M fonctions : diminution de la fausse détection

Pour augmenter le fossé entre bonne et fausse détection les auteurs [IM98] ont proposé de construire une famille de fonctions de hachage $\mathcal{G}\{g : \mathscr{F} \rightarrow U^M\}$ obtenue en concaténant M fonctions $h_i \in \mathcal{H}$ tel que $g(\mathbf{p}) = [h_1(\mathbf{p}) \ldots h_M(\mathbf{p})]$.

Les probabilités (II.2) et (II.3) de bonne et fausse détection se réécrivent :

$$\forall \mathbf{p} \in B(\mathbf{q}, r_1), \text{ alors } Pr_{g\in\mathcal{G}}[g(\mathbf{p}) = g(\mathbf{q})] \geq p_1^M \tag{II.5}$$

$$\forall \mathbf{p} \notin B(\mathbf{q}, r_2), \text{ alors } Pr_{g\in\mathcal{G}}[g(\mathbf{p}) = g(\mathbf{q})] \leq p_2^M \tag{II.6}$$

avec M un entier fixé (> 1).

On obtient ainsi une forte réduction de la probabilité de fausse détection : $p_2^M < p_2$ (car $p_2 < 1$). Cependant, on observe également une diminution de la probabilité de bonne détection : $p_1^M < p_1$ (car $p_1 < 1$).

3.3 Utilisation de L tables de hachage : augmentation de la bonne détection

Pour compenser cette diminution, L tables de hachage sont utilisées. Chaque table de hachage utilise sa propre fonction de hachage g_i. Les fonctions de hachage $g_1, \ldots, g_L$ sont sélectionnées aléatoirement et indépendamment dans $\mathcal{G}$. Une recherche est alors effectuée pour chaque table, le résultat de la recherche est obtenu en fusionnant les L résultats. La probabilité de bonne détection P_{bd} devient alors :

$$P_{bd} = 1 - (1 - p_1^M)^L \tag{II.7}$$

qui est supérieure à p_1^M pour tout L supérieur à 1.

4 Algorithmes de recherche

Plusieurs algorithmes basés sur LSH ont été proposés pour fournir des solutions approchées aux problèmes de la recherche des plus proches voisins (ppv). On distingue la recherche *RNN* qui consiste à retrouver tous les points à une distance inférieure à R du point requête et la recherche *KNN* (de

l'anglais « K Nearest Neighbor » ou K-ppv pour K-plus proches voisins) qui consiste à retrouver les K points les plus proches de la requête.

Toutes ces méthodes requièrent le même pré-traitement : le hachage de la base.

4.1 Hachage de la base

La première étape, le hachage de la base, réalisée en pré-traitement, a pour but d'initialiser les tables de hachage. Cette étape consiste à remplir les buckets des tables de hachage avec les points de la base. Ainsi, pour chaque table de hachage, une fonction de hachage g_l est sélectionnée aléatoirement dans $\mathcal{G}$. Pour chaque point $\mathbf{p}$ de la base, le hash de $\mathbf{p}$ est calculé, ce qui permet de placer le point $\mathbf{p}$ dans le bucket $g_l(\mathbf{p})$. Cette étape est réalisée pour les L tables de hachage. On peut remarquer que la complexité du pré-traitement est en $O(Ln)$. La Figure II.2 illustre l'initialisation des L tables de hachage pour le point p_1.

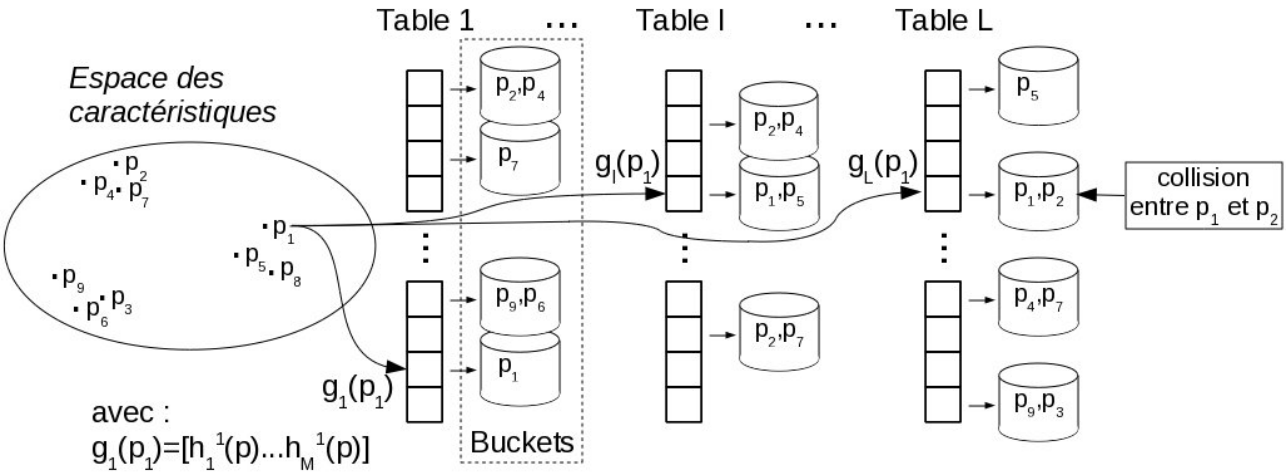


Figure II.2 – Hachage de la base : initialisation des buckets des L tables de hachage avec les points de la base. Détail du pré-traitement pour le points $\mathbf{p}_1$. Calcul du hash du point $\mathbf{p}_1$ pour chaque table l. Sélection du bucket d'intérêt $g_l(\mathbf{p_1})$ de la table l. Ajout du point $\mathbf{p}_1$ dans le bucket sélectionné. En table L, on rencontre une collision entre les points $\mathbf{p}_1$ et $\mathbf{p}_2$ qui sont placés dans le même bucket alors que ces deux points sont éloignés dans l'espace des caractéristiques.

4.2 Etape de recherche

En règle générale, l'étape de recherche s'effectue en deux opérations : la sélection des points candidats de la base et la résolution des collisions.

1. La sélection des points candidats détaillée dans l'algorithme 1 consiste à construire l'ensemble de points candidats $\mathcal{C}$. Cette sélection est obtenue en calculant le hash du point requête avec la

fonction de hachage de la première table. Les points contenus dans le bucket ainsi sélectionné permettent d'initialiser l'ensemble des points candidats $\mathcal{C}$. Tant que cet ensemble ne contient pas T candidats (doublons compris), une recherche du bucket d'intérêt est réalisée dans la table de hachage suivante.

2. Le contrôle des collisions s'opère en calculant la distance entre la requête $\mathbf{q}$ et l'ensemble des points candidats $\mathcal{C}$. Tous les points ne satisfaisant pas les conditions de recherche sont supprimés.

Le résultat de la recherche est obtenu en ordonnant les points candidats non filtrés par l'étape de contrôle des collisions ainsi qu'en supprimant les doublons.

Algorithm 1 Première étape de recherche dans les tables de hachage : sélection $\mathcal{C}$ des points candidats.

1: $l \Leftarrow 1$
2: $\mathcal{C} \Leftarrow \emptyset$
3: **while** $|\mathcal{C}| < T \quad \& \quad l \leq L$ **do**
4: $\quad i \Leftarrow g_l(\mathbf{q})$
5: $\quad B \Leftarrow$ ensemble des points appartenants au bucket i de la table l.
6: $\quad \mathcal{C} \Leftarrow \{\mathcal{C} \cup B\}$
7: $\quad l \Leftarrow l+1$
8: **end while**

Plusieurs algorithmes de recherche ont été proposés en fonction de l'approximation de la recherche des plus proches voisins considérée. Les modifications s'opèrent sur le choix du seuil T et le contrôle des collisions.

4.2.1 Problème $(R,c)NN$

L'algorithme classique de LSH [IM98, GIM99, DIIM04] s'attaque au problème $(R,c)NN$: s'il existe au moins un point dans un rayon de recherche R du point requête, retrouver au moins un point dans un rayon cR du point requête (avec $c > 1$).

L'étape de contrôle des collisions consiste donc à supprimer tous les candidats se trouvant à une distance supérieure à cR du point requête.

L'implémentation classique résout ce problème avec une complexité en $O(n^{\rho})$ avec $\rho < 1/c$. Cette complexité est obtenue en fixant correctement le seuil T ainsi que les paramètres M et L.

La recherche $(R,c)NN$ est obtenue en calculant uniquement la distance entre le point requête et T points de la base. En fixant, $T = 2L$, la complexité de la recherche est donc en $O(L)$. Pour qu'elle soit de l'ordre de $O(n^{\rho})$, il faut donc fixer $L = n^{\rho}$. Dans [IM98], les auteurs ont montré qu'en fixant $\rho = \frac{ln(1/p_1)}{ln(1/p_2)}$ et $M = \log_{\frac{p_1}{p_2}}(2n)$, visiter $2L$ candidats suffit pour s'assurer de retrouver au moins un

bon point (s'il existe) avec p_1 et p_2 définis aux équations (II.2) et (II.3) où $\frac{r_2}{r_1} = c$. Les auteurs ont également montré, que dans ce cas, la complexité de recherche est bien inférieure à $O(n^{1/c})$.

Il est intéressant de noter, que l'utilisation du seuil T permet de s'assurer de retrouver uniquement un seul « bon » point. Tous les algorithmes cherchant à retrouver plusieurs « bons » points n'utilisent donc pas ce seuil ($T = \infty$).

4.2.2 Problème $(R, 1-\delta)NN$

Un autre algorithme, tel que l'implémentation **E^2-LSH** [AI05], permet de traiter le problème $(R, 1-\delta)NN$: tous les points $\mathbf{p}$ satisfaisant $D(\mathbf{p},\mathbf{q}) \leq R$ (avec $\mathbf{q}$ le point requête) doivent être retrouvés avec une probabilité d'au moins $1-\delta$ (δ est la probabilité qu'un point ne soit pas retrouvé).

L'étape de contrôle des collisions consiste donc à supprimer tous les candidats se trouvant à une distance supérieure à R du point requête.

Il nous faut maintenant fixer T, L et M. Comme on souhaite retrouver plusieurs points, $T = \infty$. D'après l'équation (II.7), on a $1-(1-p_1^M)^L \geq 1-\delta$, ce qui implique $L \geq \frac{ln(\delta)}{ln(1-p_1^M)}$. Les auteurs proposent donc de déterminer M expérimentalement. La valeur de M optimale est celle qui minimise les temps de recherche. Une fois M fixé, on est alors capable de calculer L en fonction de δ (généralement fourni par l'utilisateur) et de p_1 (dépendant de la fonction de hachage utilisée).

4.2.3 Problème *KNN*

Dans certains problèmes, la recherche *KNN* est préférée à la recherche *RNN*. Il est alors possible de modifier l'algorithme précédent pour effectuer une recherche *KNN* approximée [VCPF08a]. Dans ce cas, l'étape de contrôle des collisions consiste à trier les candidats en fonction de leurs distances au point requête et de ne garder que les K meilleurs points. Il est également nécessaire de déterminer un rayon de recherche R englobant les K points les plus proches du point requête. R est alors obtenu expérimentalement en réalisant des tests statistiques sur la base. Dans cette thèse, on utilise essentiellement cette stratégie de recherche.

4.3 Implémentation

Rappelons que le domaine de définition de chaque fonction de hachage g_i est U^M, avec U qui dépend de la famille de hachage utilisée, des données considérées et du rayon de recherche. Une hypothèse réaliste est que U soit un entier à valoir dans l'intervalle $[0,5]$ et M soit un entier de l'ordre de 20. Dans ce contexte, le domaine de définition de chaque fonction de hachage est donc un entier entre 0 et $6^{20} = 3,6 \times 10^{15}$. Ces fonctions de hachage ne permettent donc pas de se ramener à

l'adressage direct. En effet, il n'est pas réaliste de stocker un tableau de 6^{20} entrées. De plus, même si stocker un tel tableau était envisageable, il serait ridicule d'utiliser tant de mémoire pour gérer une base d'un million d'images. En effet, seul $2,7 \times 10^{-8}\%$ du tableau seraient alors occupés.

La méthode classique pour gérer ce problème est d'utiliser un hachage universel. Une fonction de hachage $V_1^l : U^M \rightarrow \{0,\ldots,tableSize-1\}$ est associée à chaque table l et donc à chaque fonction g_l, $l = 1 \ldots L$. Cette fonction va permettre de stocker les buckets LSH dans un tableau de taille le nombre d'images dans la base (noté $tableSize$). La fonction de hachage V_1 a la forme suivante :

$$V_1(x_1,x_2,\ldots,x_M) = \left(\left(\sum_{i=1}^{M} r_i' x_i\right) mod(prime)\right) mod(tableSize) \tag{II.8}$$

où r_i' est un entier aléatoire à valeurs dans $[0, prime]$, $tableSize$ est la taille de la table de hachage (nombre d'images dans la base), $prime$ est un nombre premier (avec $prime > tableSize$) et x_i est la $i^{\text{ème}}$ valeur de la fonction g_l.

Cette fonction de hachage entraîne inévitablement l'apparition de collisions qu'il faut gérer. Une solution pour gérer les collisions est d'utiliser le double hachage. Dans ce cas, plusieurs buckets peuvent être stockés dans la même case du tableau en utilisant une liste chaînée. La fonction de hachage V_1 permet alors d'accéder à la liste chaînée contenant le bucket d'intérêt. Le bucket d'intérêt est alors identifié dans la liste en utilisant une seconde fonction de hachage V_2 qui a la forme :

$$V_2(x_1,x_2,\ldots,x_M) = \left(\sum_{i=1}^{M} r_i'' x_i\right) mod(prime) \tag{II.9}$$

où r_i'' est un entier aléatoire à valeurs dans $[0, prime]$.

Le schéma global du hachage de la base est résumé en Figure II.3.

5 Principales fonctions de hachage

Maintenant que nous avons vu les propriétés que doivent respecter les fonctions de hachage pour suivre le schéma LSH, nous allons détailler les principales fonctions proposées dans la littérature. Comme nous l'avons vu, chaque fonction de hachage est conçue pour une distance ou une similarité D donnée qui permet de définir un voisinage (Définition 1).

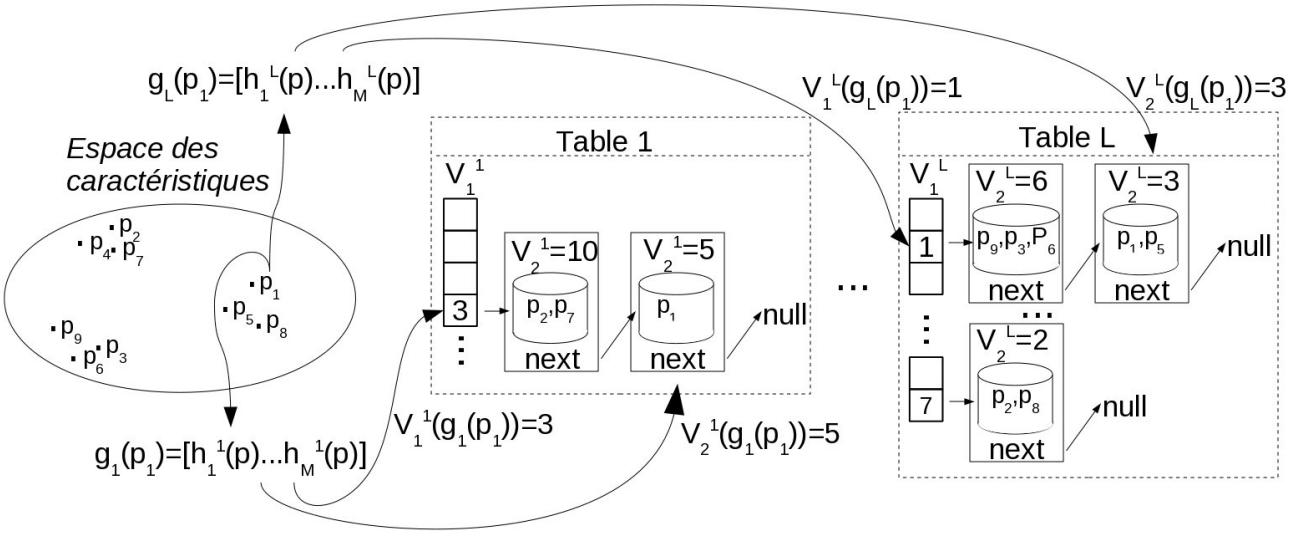


Figure II.3 – Schéma complet du hachage LSH avec résolution des collisions par double hachage universel. Pour chaque table $l = 1 \ldots L$: initialisation des buckets avec la fonction LSH g_l (regroupement des points voisins en bucket), stockage des buckets dans la table l avec la fonction de hachage universel V_1^l, mise en place des listes chaînées avec la fonction de hachage universel V_2^l.

5.1 Hachage pour la distance de Hamming

La première fonction de hachage proposée a été conçue pour approximer la distance de Hamming [IM98]. Soit $\mathbf{p}$ et $\mathbf{q}$ deux points de l'espace $\mathbb{H}^d$ (avec $\mathbb{H} \in \{0,1\}$). Rappelons que la distance de Hamming est définie par : $D(\mathbf{p},\mathbf{q}) = 1 - \frac{|\mathbf{p} \cap \mathbf{q}|}{|\mathbf{p} \cup \mathbf{q}|}$ avec $\cup$, l'opérateur OU et $\cap$, l'opérateur ET.

La famille de fonctions proposée est définie par :

$$\begin{aligned} h_i : \quad & \mathbb{H}^d \to \mathbb{H} \\ & \mathbf{p} \mapsto h_i(p_1, \ldots, p_d) = p_i \end{aligned}$$

avec i choisi par tirage uniforme dans l'intervalle $[0,d[$. La fonction h_i consiste à sélectionner, pour tout point $\mathbf{p}$ de la base, le $i^{\text{ème}}$ bit.

On peut montrer que cette fonction est $(r, r(1+\varepsilon), 1-\frac{r}{d}, 1-\frac{r(1+\varepsilon)}{d})$-sensitive. En effet, si la distance entre deux points $\mathbf{p}_1$ et $\mathbf{p}_2$ vaut r, cela veut dire que $\mathbf{p}_1$ et $\mathbf{p}_2$ ont r bits différents et $d-r$ bits en commun. La probabilité que $h(\mathbf{p}_1) = h(\mathbf{p}_2)$ correspond à la probabilité de tirer un des bits en commun. Comme i suit une loi uniforme, chaque bit à une chance $1/d$ d'être tiré. On a donc $d-r$ chances sur d que $h(\mathbf{p}_1) = h(\mathbf{p}_2)$. Il s'en suit que $p_1 = 1 - \frac{r}{d}$. La démonstration est similaire pour p_2.

Un exemple de schéma LSH utilisant cette fonction est présenté Figure II.4.

Cette fonction de hachage a été reprise par Gionis *et al.* [GIM99] pour approximer la distance l_1. Les auteurs proposent en effet d'utiliser une fonction de binarisation qui projette les données définies

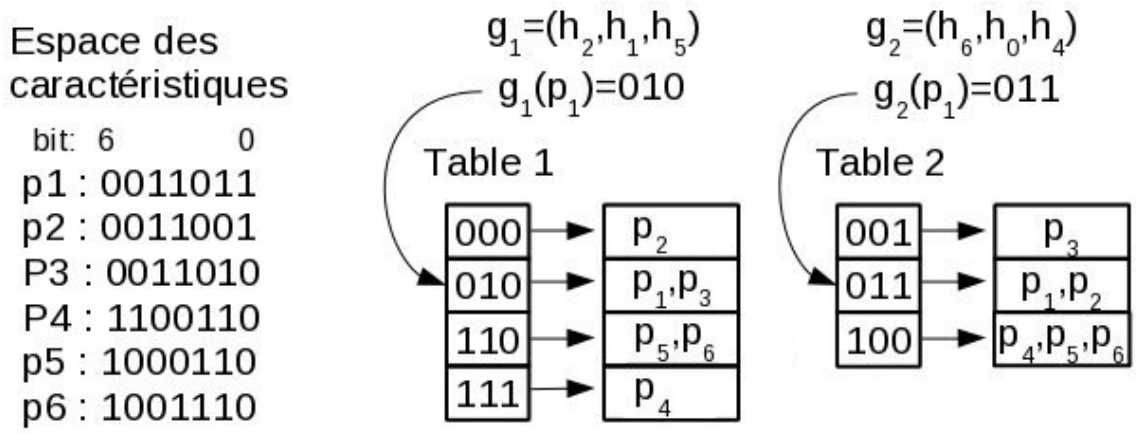


Figure II.4 – Exemple de hachage pour la distance de Hamming. Soit 6 points $\mathbf{p} \in \mathbb{H}^7$ et 2 tables de hachage. Les 2 fonctions LSH, g_1 et g_2 sont obtenues en concaténant 3 fonctions h_i. On peut remarquer que $D(\mathbf{p}_5, \mathbf{p}_6) < D(\mathbf{p}_5, \mathbf{p}_1)$, or les points $\mathbf{p}_5$ et $\mathbf{p}_6$ ont été regroupés dans les mêmes buckets contrairement aux points $\mathbf{p}_5$ et $\mathbf{p}_1$.

dans l'espace $\mathbb{Z}^{+d} \rightarrow \mathbb{H}^{bd}$ avec b le nombre maximal de bits pour coder n'importe quelle dimension de l'espace de départ. La distance de Hamming dans l'espace de projection est alors équivalente à la distance l_1 dans l'espace de départ. Le schéma LSH décrit précédemment est alors directement applicable.

5.2 Hachage pour la similarité du cosinus

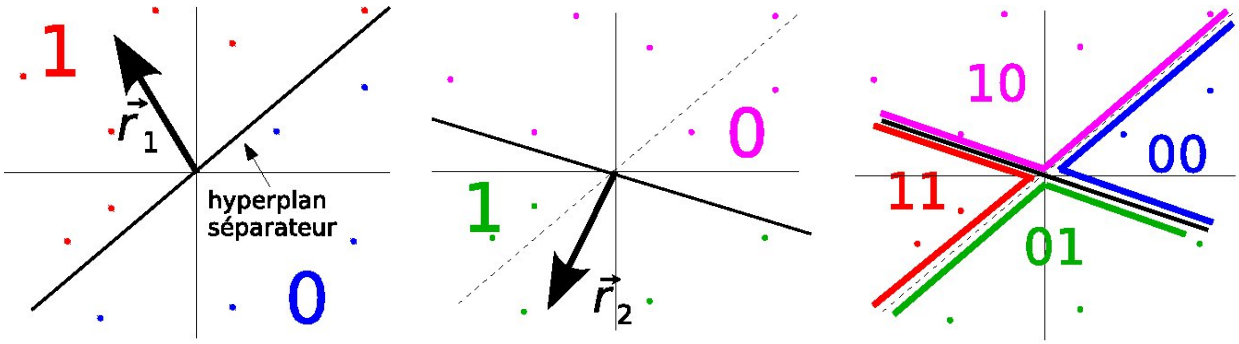
Dans [Cha02], Charikar a proposé une fonction de hachage permettant d'approximer la similarité du cosinus. Soit $\mathbf{p}$ et $\mathbf{q}$ deux points de l'espace $\mathbb{R}^d$. La similarité du cosinus est définie par : $D(\mathbf{p},\mathbf{q}) = 1 - \frac{\theta(\mathbf{p},\mathbf{q})}{\pi}$ avec $\theta(\mathbf{p},\mathbf{q}) = \cos^{-1}\left(\frac{\langle \mathbf{p},\mathbf{q}\rangle}{\sqrt{\langle \mathbf{p},\mathbf{p}\rangle\langle \mathbf{q},\mathbf{q}\rangle}}\right)$ La famille de fonctions proposée est définie par :

$$h_{\mathbf{r}}(\mathbf{p}) = \begin{cases} 1 \text{ si } \langle \mathbf{r},\mathbf{p}\rangle \geq 0 \\ 0 \text{ si } \langle \mathbf{r},\mathbf{p}\rangle < 0 \end{cases} \tag{II.10}$$

avec $\mathbf{r}$, un vecteur aléatoire de l'espace $\mathbb{R}^d$ de norme unitaire (*i.e.* la valeur de chaque coordonnée est tirée suivant une loi gaussienne puis le vecteur est normalisé).

Comme illustré par la Figure II.5, la fonction $h_{\mathbf{r}}$ utilise la direction de l'espace $\mathbf{r}$ pour définir un hyperplan séparant les données en deux ensembles, l'ensemble des données se trouvant dans la direction $\mathbf{r}$ et l'ensemble des données se trouvant dans la direction opposée à $\mathbf{r}$. La fonction de hachage g obtenue en concaténant M fonctions $h_{\mathbf{r}}$ permet de définir un découpage de l'espace en 2^M cellules.

A partir de la Figure II.6, il est aisé de montrer que la fonction $h_{\mathbf{r}}$ est bien *Locality-Sensitive*. Soit $\mathbf{p}_1^{\perp}$ et $\mathbf{p}_2^{\perp}$, les droites respectivement orthogonales à $\mathbf{p}_1$ et $\mathbf{p}_2$. Pour que les valeurs de hash, $h_{\mathbf{r}}(\mathbf{p}_1)$



(a) $h_{\mathbf{r}_1}$: séparation des données suivant la direction $\mathbf{r}_1$

(b) $h_{\mathbf{r}_2}$: séparation des données suivant la direction $\mathbf{r}_2$

(c) clusterisation de l'espace en concaténant $h_{\mathbf{r}_1}$ et $h_{\mathbf{r}_2}$

Figure II.5 – Exemple de hachage pour la similarité du cosinus. La fonction de hachage définie par la concaténation de $h_{\mathbf{r}_1}$ et de $h_{\mathbf{r}_2}$ permet de partager l'espace en 4 cellules.

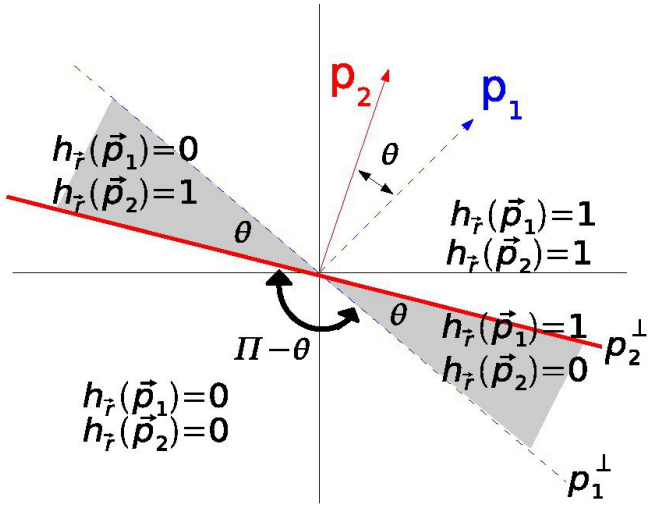


Figure II.6 – $h_{\mathbf{r}}$, une fonction *Locality-Sensitive*. Conditions sur $\mathbf{r}$ pour que $h_{\mathbf{r}}(\mathbf{p}_1) = h_{\mathbf{r}}(\mathbf{p}_2)$ suivant l'angle $\theta(\mathbf{p}_1, \mathbf{p}_2)$ noté θ.

et $h_{\mathbf{r}}(\mathbf{p}_2)$, soient différentes, il faut que le vecteur aléatoire $\mathbf{r}$ de la fonction $h_{\mathbf{r}}$ se trouve entre $\mathbf{p}_1^\perp$ et $\mathbf{p}_2^\perp$. La probabilité que $\mathbf{r}$ se trouve entre $\mathbf{p}_1^\perp$ et $\mathbf{p}_2^\perp$ (zones grisées dans la Fig. II.6) vaut $\frac{2\theta(\mathbf{p}_1^\perp, \mathbf{p}_2^\perp)}{2\pi}$. Par construction, on a $\theta(\mathbf{p}_1, \mathbf{p}_2) = \theta(\mathbf{p}_1^\perp, \mathbf{p}_2^\perp)$. On a donc : $P_{\mathbf{r}}[h_{\mathbf{r}}(\mathbf{p}_1) = h_{\mathbf{r}}(\mathbf{p}_2)] = 1 - \frac{\theta(\mathbf{p}_1, \mathbf{p}_2)}{\pi}$. On retrouve bien la Définition 2 avec D correspondant à la similarité du cosinus.

5.3 Généralisation du hachage pour la similarité du cosinus aux similarités noyaux

Dans [GD07], Grauman *et al.* ont étendu le schéma proposé pour la similarité du cosinus à une fonction de similarité noyau, *pyramid matching kernel*. Les similarités noyaux, notées k, s'écrivent via la fonction d'injection ϕ (cf. Sec I.2.1.2) : $k(\mathbf{p},\mathbf{q}) = \langle \phi(\mathbf{p}), \phi(\mathbf{q}) \rangle$. Lorsque l'on peut expliciter la fonction ϕ et tirer aléatoirement un vecteur $\mathbf{r}$, dans l'espace d'induction $\mathscr{H}$, il est possible de réécrire la famille de fonctions définie à l'équation (II.10) par :

$$h_{\mathbf{r}}(\phi(\mathbf{p})) = \begin{cases} 1 \text{ si } \langle \mathbf{r}, \phi(\mathbf{p}) \rangle \geq 0 \\ 0 \text{ si } \langle \mathbf{r}, \phi(\mathbf{p}) \rangle < 0 \end{cases}$$

On a alors :

$$P_{\mathbf{r}}[h_{\mathbf{r}}(\phi(\mathbf{p})) = h_{\mathbf{r}}(\phi(\mathbf{q}))] = 1 - \frac{\cos^{-1}\left(\frac{\langle \phi(\mathbf{p}), \phi(\mathbf{q}) \rangle}{\sqrt{\langle \phi(\mathbf{p}), \phi(\mathbf{p}) \rangle \langle \phi(\mathbf{q}), \phi(\mathbf{q}) \rangle}}\right)}{\pi}$$

Cette extension est uniquement valide pour les noyaux explicites, *i.e.* les noyaux pour lesquels il est possible d'expliciter la fonction de projection ϕ ainsi que l'espace d'induction $\mathscr{H}$. Cependant, un grand nombre de similarités noyaux sont implicites. Pour ces noyaux, on sait qu'il existe une fonction ϕ, permettant de réécrire la similarité k comme un produit scalaire dans un espace de redescription $\mathscr{H}$, mais la fonction ϕ peut être inconnue ou d'une complexité calculatoire trop importante pour être considérée. C'est par exemple le cas des noyaux gaussiens : $k(\mathbf{p},\mathbf{q}) = e^{-\frac{\|\mathbf{p}-\mathbf{q}\|^2}{2\sigma^2}}$ où l'espace d'induction est un espace de dimension infinie.

Un pas de plus a été réalisé dans [KG09]. Les auteurs ont proposé une solution pour généraliser cette fonction de hachage aux noyaux implicites. Comme il n'est pas possible de tirer aléatoirement un vecteur $\mathbf{r}$ dans $\mathscr{H}$, les auteurs proposent de construire $\mathbf{r}$ à partir des données. Soit $\mathbf{z}_T = \frac{1}{T}\sum_{\mathbf{p}_t \in S} \phi(\mathbf{p}_t)$, avec S un ensemble de T points $\mathbf{p}_t$ sélectionnés aléatoirement parmi les données. Le vecteur $\mathbf{z}_T$ suit une loi normale $\mathcal{N}(\mu, \Sigma)$ de moyenne μ et de covariance Σ. Les auteurs proposent d'approximer $\mathbf{r}$ par le vecteur $\tilde{\mathbf{z}_T}$ obtenu en centrant et réduisant $\mathbf{z}_T$. Ils ont également montré que $\tilde{\mathbf{z}_T}$ peut s'écrire $\tilde{\mathbf{z}_T} = \sum_{t=0}^{T} w_t \phi(\mathbf{p}_t)$. La famille de fonctions définie à l'équation (II.10) se réécrit alors :

$$h_{\tilde{\mathbf{z}_T}}(\phi(\mathbf{p})) = \begin{cases} 1 \text{ si } \langle \tilde{\mathbf{z}}_t, \phi(\mathbf{p}) \rangle = \sum_{t=0}^{T} w_t k(\mathbf{p}_t, \mathbf{p}) \geq 0 \\ 0 \text{ si } \langle \tilde{\mathbf{z}}_t, \phi(\mathbf{p}) \rangle = \sum_{t=0}^{T} w_t k(\mathbf{p}_t, \mathbf{p}) < 0 \end{cases}$$

Il est ainsi possible de calculer $h_{\tilde{\mathbf{z}_T}}(\phi(\mathbf{p}))$ pour tout point $\mathbf{p}$ directement dans l'espace de départ, sans

utiliser la fonction d'induction ϕ. Cependant, cette fonction a une complexité importante puisqu'elle nécessite l'évaluation de T similarités $k(\mathbf{p}_t, \mathbf{p})$. De plus, dans [KG09], une fonction de hachage g est obtenue en concaténant 300 fonctions $h_{\tilde{\mathbf{z}_T}}$ avec $T = 300$. En pratique, pour une requête $\mathbf{q}$, le calcul de $g(\mathbf{q})$ nécessite donc 9×10^4 évaluations de similarité k avant même d'avoir recours à l'étape de vérification des collisions qui consiste à calculer la similarité k entre le point requête et tous les points du bucket sélectionné.

5.4 Hachage pour la distance l_1 et l_2

Dans [DIIM04], Datar *et al.* ont proposé une famille de fonction, adaptée à la distance l_1 ou l_2, définie par :

$$h_{\mathbf{a},b}(\mathbf{p}) = \lfloor \frac{\mathbf{a}.\mathbf{p} + b}{W} \rfloor \tag{II.11}$$

où W spécifie la largeur d'une tranche, $\mathbf{a}$ est un vecteur aléatoire suivant la loi $\mathcal{N}(0,1)$ pour la distance l_2 ou la loi de Cauchy pour la distance l_1 et b est un réel choisi suivant la loi uniforme dans l'intervalle $[0, W[$.

Les auteurs ont prouvé que cette fonction est bien *Locality-Sensitive* en montrant que :

$$Pr_{h_{\mathbf{a},b} \in \mathcal{H}}[h(\mathbf{p}) = h(\mathbf{q})] = p(c) = \int_0^W \frac{1}{c} f_p(\frac{t}{c})(1 - \frac{t}{W})dt \tag{II.12}$$

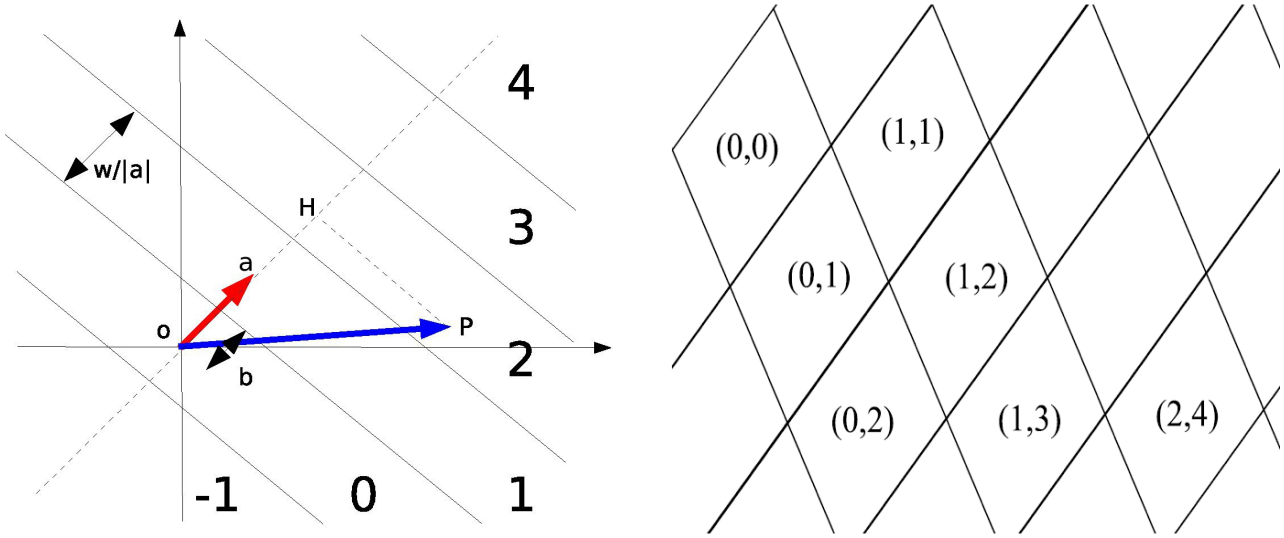
avec $c = ||\mathbf{p} - \mathbf{q}||_{l_p}$ et f_p la fonction de densité de probabilité de la valeur absolue de la distribution Normale (pour l_2) ou de Cauchy (pour l_1). Comme $p(c)$ est décroissante monotone, cette fonction est (r_1, r_2, p_1, p_2)-sensitive pour $p_1 = p(1)$, $p_2 = p(c)$ et $c = r_1/r_2$.

Comme représenté par la Figure II.7.(a), chaque projection $\mathbf{a}$ permet de découper l'espace par un ensemble d'hyperplans séparateurs parallèles et équidistants. La valeur $h_{\mathbf{a},b}(\mathbf{p})$ indique dans quelle tranche de l'espace le point $\mathbf{p}$ se trouve.

Un exemple de découpage de l'espace obtenu avec une fonction de hachage g issue de la concaténation de M fonctions $h_{\mathbf{a},b}$ est présenté Figure II.7.(b) pour $M = 2$. On peut remarquer que les projections ne sont pas orthogonales et que chaque case de la grille est indexée par un vecteur de M entiers.

5.5 Diminution des ressources mémoires : Multi-Probe

Un des problèmes de l'algorithme LSH est sa forte consommation de mémoire vive à cause de la nécessité de créer un grand nombre de tables de hachage. Une modification possible de l'algorithme est de ne générer qu'une seule table de hachage (ou moins d'une dizaine) et pour un point requête de



(a) partition de l'espace en tranches équidistantes de taille spécifiée par W suivant la direction $\mathbf{a}$. Un offset b permet de décaler le découpage

(b) exemple de partition de l'espace obtenue en concaténant deux fonctions $h_{\mathbf{a},b}$

Figure II.7 – Fonction *Locality-Sensitive* pour les distances l_1 ou l_2.

chercher ses voisins dans plusieurs cases de cette table (*i.e.* un point requête génère donc plusieurs clés de hachage).

Dans les travaux de Panigrahy [Pan06], l'auteur propose une méthode pour calculer ces clés multiples. Il propose de générer plusieurs points aléatoires $\mathbf{q}_i$, autour du point requête $\mathbf{q}$, et de chercher les voisins du point requête $\mathbf{q}$ dans les cases de la table correspondant aux clés $h(\mathbf{q}_i)$ des points aléatoires $\mathbf{q}_i$. Le défaut de cette approche est qu'on observe une perte de temps de calcul inhérente au fait qu'un grand nombre de requêtes aléatoires accèdent aux mêmes buckets. Dans ces nombreux cas, le calcul du hash du point requête est alors inutile.

Dans [LJW+07], les auteurs proposent une autre solution pour calculer ces clés multiples en s'assurant de ne pas accéder plusieurs fois au même bucket. En effet, au lieu de n'explorer qu'un bucket par table de hachage, la probabilité de bonne détection est calculée pour plusieurs buckets et seuls les buckets les plus probables sont explorés. A partir des propriétés données par la Définition 1, nous savons que si un point $\mathbf{p}$ est proche d'un point requête $\mathbf{q}$ mais qu'ils ne se trouvent pas dans le même bucket (*i.e.* ils ont une valeur de hash différente : $g(\mathbf{p}) \neq g(\mathbf{q})$), le point $\mathbf{p}$ a une probabilité importante de se trouver dans un des buckets voisins. Les auteurs définissent un vecteur de perturbation $\Delta = (\delta_1, \ldots, \delta_M)$ où $\delta_i \in \{-1, 0, 1\}$ et M correspond au nombre de fonctions h concaténées pour construire la fonction de hachage g. Pour un point requête $\mathbf{q}$ et une fonction de hachage $g(\mathbf{p}) = (h_1(\mathbf{p}), \ldots, h_M(\mathbf{p}))$, les probabilités de succès de $g(\mathbf{q}) + \Delta$ sont calculées et seuls les T buckets

dont les probabilités de succès sont les plus importantes sont explorés. Les auteurs sont ainsi capables de réduire d'un facteur 18 le nombre de tables de hachage requises pour des performances similaires. Ce schéma est appelé **E^2-MPLSH**.

5.6 Amélioration du l_2

Plusieurs améliorations de la fonction de hachage définie équation (II.11) ont été proposées dans la littérature pour effectuer des recherches suivant la distance l_2. Ces méthodes sont basées sur la remarque que le meilleur découpage de l'espace possible consiste à utiliser un partitionnement en hyper-sphères. L'idée est de créer une séquence d'hyper-sphères $B_1,\ldots,B_m$, de rayon W, réparties aléatoirement sur tout l'espace. Chaque hyper-sphère permet de définir une cellule. Cependant, il n'est pas possible d'appliquer directement cette méthode. En effet, à partir d'un point $\mathbf{p}$ donné, retrouver l'hyper-sphère contenant $\mathbf{p}$ peut demander beaucoup de calcul. Dans [AIM06], Andoni *et al.* proposent d'approximer cette solution en utilisant des réseaux géométriques (*lattice* en anglais) permettant de découper l'espace de façon régulière en utilisant des formes approximant des hyper-sphères. Nous appelons réseau, un sous-groupe additif discret de $\mathbb{R}^d$ engendré par une base $\mathcal{B}$ de m vecteurs $b_1,\ldots,b_m \in \mathbb{R}^d$, tel que :

$$\Lambda = \{x_i b_1 + \ldots + x_m b_m : x_i \in \mathbb{Z}\}$$

Λ est le sous-ensemble de $\mathbb{R}^d$ contenant toutes les combinaisons linéaires entières de sa base B. Le nombre de bases dont les propriétés sont intéressantes [CSB99] (faible recouvrement entre cellules, compacité importante) est limité. Les bases les plus connues sont la base hexagone définie en dimension 2 (voir Fig. II.8), la base E8 en dimension 8 et la base de Leech en dimension 24.

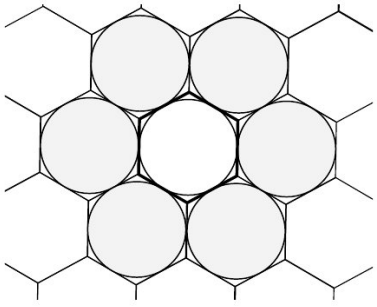


Figure II.8 – Réseau en dimension 2 : l'hexagone est la meilleure forme pour approximer, sans recouvrement, des cercles de rayons fixes en dimension 2.

Dans [AIM06], Andoni *et al.* ont proposé d'utiliser le réseau de Leech. Cette idée a été reprise par Jegou *et al.* dans [JASG08] qui ont proposé d'utiliser le réseau E8 moins performant que le réseau de

Leech mais d'une complexité de calcul bien moins élevée.

Conclusion

Nous avons présenté dans ce chapitre le principe de LSH offrant une solution efficace de la recherche approximée des plus proches voisins dans des espaces de grande dimension. Les implémentations de LSH sont conçues pour les distances usuelles, telles que les distances de Hamming, du l_1 ou du l_2. Cependant d'autres distances, comme la distance du χ^2 éq. (I.1) ou la EMD (*Earth Mover's Distance*) se sont montrées bien plus pertinentes en CBIR [SC96, CHV99, GCPF08b].

Dans [IT03], Indyk et Thaper ont proposé une méthode qui permet d'adapter les structures d'index permettant une recherche *ppv* rapide selon la distance l_2, à la recherche *ppv* selon la distance EMD. Cette adaptation est obtenue via un pré-traitement qui consiste à projeter les données dans un nouvel espace $\mathcal{P}$. La distance l_2 entre deux points de cet espace est une approximation de la distance EMD dans l'espace d'origine. Une fois dans $\mathcal{P}$, le schéma LSH classique [DIIM04] peut alors être utilisé. Cependant, ce pré-traitement requiert de dupliquer au moins 5 fois les données ce qui entraîne une augmentation importante de l'occupation mémoire ainsi que des temps de recherche.

Dans le chapitre suivant, nous proposons une nouvelle fonction de hachage permettant d'effectuer des recherches *ppv* rapides selon la distance χ^2.

Chapitre III

χ^2-*LSH* : Schéma LSH optimisé pour les signatures globales

Dans ce chapitre, nous proposons une nouvelle structure d'index, nommée **χ^2-LSH**. Contrairement au schéma proposé pour la distance EMD par Indyk et Thaper [IT03], une nouvelle fonction de hachage conçue pour la distance χ^2 est proposée permettant d'utiliser le schéma *LSH* classique sans aucune adaptation.

Après avoir présenté notre fonction de hachage, nous prouvons que cette dernière respecte bien les propriétés énoncées par Indyk. Nous montrons ensuite que le schéma Multi-Probe [LJW+07] s'étend à cette fonction. Enfin, nous présentons une analyse empirique des performances de notre fonction de hachage ainsi qu'une comparaison avec le schéma E^2-*LSH* [DIIM04].

Sommaire

1 **Construction de la clé χ^2-*LSH*** 46

2 **Extension du Multi-Probe : χ^2-*MPLSH*** 52

3 **Evaluation** 53

4 **Conclusion** 63

1 Construction de la clé χ^2-*LSH*

Dans cette partie, nous détaillons la construction de la fonction de hachage permettant d'approximer la distance χ^2. Rappelons que cette distance est définie par $\forall(\mathbf{p},\mathbf{q}) \in \mathbb{R}^{+d}$:

$$\chi^2(\mathbf{p},\mathbf{q}) = \sqrt{\sum_{i=1}^{d} \frac{(\mathbf{p}_i - \mathbf{q}_i)^2}{\mathbf{p}_i + \mathbf{q}_i}} \tag{III.1}$$

1.1 Partitionnement de l'espace au sens de la distance χ^2

Comme pour la fonction de hachage proposée par Datar [DIIM04], le principe est de projeter les points dans un espace de plus petite dimension et de partitionner (*clusteriser*) ce sous-espace. La clusterisation doit assurer que plus 2 points sont proches, plus leur probabilité d'appartenir à un même bucket est grande. Dans notre cas, le sous-espace est défini par une demi-droite $\mathbb{R}^+$. Cette demi-droite est obtenue en projetant tous les points sur un vecteur aléatoire $\mathbf{a}$ dont chaque composante suit une loi normale. Introduisons la suite $(X_i)_{i\in\mathbb{N}}$ correspondant aux bornes des intervalles successifs de $\mathbb{R}^+$. Cette demi-droite est partitionnée de façon régulière, au sens de la distance χ^2, *i.e.* chaque intervalle $[X_i, X_{i+1}]$ possède la même longueur W (Fig. III.1).

$$\forall i,\ \chi^2(X_i, X_{i+1}) = \sqrt{\frac{(X_i - X_{i+1})^2}{X_i + X_{i+1}}} = W \tag{III.2}$$

D'un autre côté, comme le montre la Figure III.1, la longueur des intervalles $[X_i, X_{i+1}]$ n'est plus constante si on considère la distance euclidienne : $\forall i,\ l_2(X_i, X_{i+1}) \neq W$.

La Figure III.1 montre également que le partitionnement suivant la distance χ^2 permet une répartition des points entre chaque tranche plus homogène que suivant la distance l_2. Ce découpage est donc plus proche de la distribution naturelle des points de la base dans l'espace des caractéristiques. On peut donc espérer qu'en utilisant la distance χ^2, les points seront mieux répartis entre les différents buckets des tables de hachage LSH et que l'algorithme de recherche sera ainsi plus performant.

Ce partitionnement de l'espace, au sens de la distance χ^2, assure que quand deux points, après projection suivant la direction $\mathbf{a}$, sont à une distance inférieure à W, la probabilité de collision est élevée.

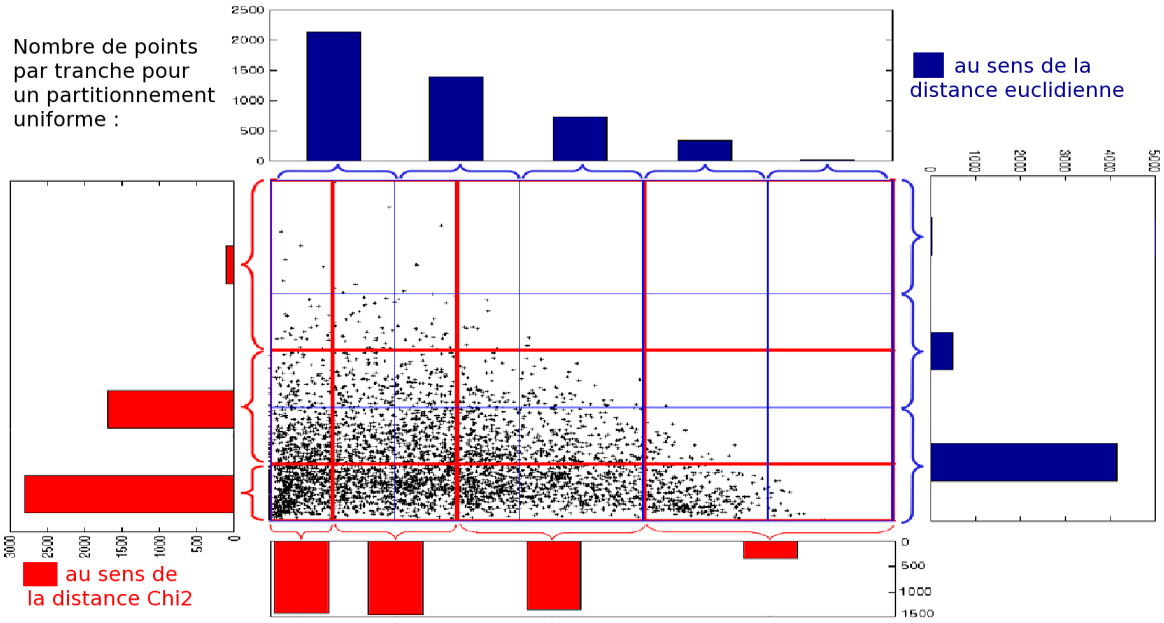


Figure III.1 – Différence entre un découpage régulier de l'espace suivant les distances l_2 et χ^2 : 4 500 images de la base COREL ont été sélectionnées aléatoirement. Les images sont représentées par un vecteur de caractéristiques couleurs et textures. L'espace des caractéristiques est découpé avec une grille régulière à deux dimensions suivant la distance χ^2 (en rouge) et l_2 (en bleu). Les histogrammes dénombrant l'occurrence des points de l'espace dans chacune des tranches ainsi formées sont reportés. Les histogrammes correspondant au découpage χ^2 ont tendance à être plus uniforme que pour le découpage l_2.

1.2 Construction de la clé

Nous cherchons une fonction $h_{\mathbf{a}}$ tel que :

$$h_{\mathbf{a}}(\mathbf{p}) = n \text{ ssi } X_{n-1} \leq \mathbf{a}.\mathbf{p} < X_n \tag{III.3}$$

où la suite $(X_n)_n$ satisfait l'équation (III.2)) avec pour valeur initiale : $X_0 = 0$. $\mathbf{a}$ est un vecteur aléatoire où chaque composante est choisie de façon indépendante comme suit : on tire une valeur suivant une loi normale centrée réduite $\mathcal{N}(0,1)$ puis on prend sa valeur absolue.

L'équation (III.2) induit à la relation suivante entre X_n et X_{n-1} :

$$X_n = X_{n-1} + W^2 \frac{\sqrt{8\frac{X_{n-1}}{W^2}} + 1}{2} \tag{III.4}$$

En fixant $X_0 = 0$, on obtient :

$$X_n = \frac{n(n+1)}{2} W^2$$

Une fois la suite $(X_n)_n$ calculée, une recherche dichotomique ou un arbre de recherche pourrait

être utilisé pour évaluer numériquement les valeurs $h(\mathbf{p})$ de la fonction précédemment décrite pour tout point $\mathbf{p}$.

Cependant, un schéma plus rapide est possible en exhibant la fonction $h_\mathbf{a}$ correspondante.

Introduisons pour cela la fonction suivante :

$$y : \mathcal{R}^+ \rightarrow \mathcal{R}^+$$

$$x \mapsto \frac{\sqrt{\frac{8x}{W^2}+1}-1}{2} \tag{III.5}$$

Alors la fonction $h_\mathbf{a}$ est donnée pour la proposition P1 suivante :

Proposition P1 : Pour la suite $(X_n)_n$ définie à l'éq. (III.3), une fonction de hachage $h_\mathbf{a}$ peut être définie par :

$$h_\mathbf{a}(\mathbf{p}) = \lfloor y(\mathbf{a}.\mathbf{p}) \rfloor \tag{III.6}$$

avec $\mathbf{p} \in \mathbb{R}^d$ et y défini à l'éq. (III.5).

Preuve : définissons tout d'abord la suite $(Y_n)_n$:

$$Y_n = y(X_n) = \frac{\sqrt{8\frac{X_n}{W^2}+1}-1}{2} \tag{III.7}$$

Nous allons prouver par récurrence sur n que $(H_n) : Y_n = n$ est vraie pour tout n.
Par définition (H_0) est vraie : $Y_0 = 0$.
Supposons (H_{n-1}) vraie, $Y_{n-1} = n-1$.
En élevant l'éq. (III.7) au carré, on obtient :

$$Y_n^2 + Y_n = 2\frac{X_n}{W^2}. \tag{III.8}$$

A partir de l'équation (III.2), il vient que : $(X_{n-1} - X_n)^2 = W^2.(X_{n-1} + X_n)$ Par construction, on a $X_n > Xn-1$ qui permet d'obtenir :

$$X_n = X_{n-1} + \frac{W^2}{2} + W^2\sqrt{2\frac{X_{n-1}}{W^2}+\frac{1}{4}}.$$

Enfin, en remplaçant X_n dans l'éq. (III.8)), on obtient :

$$Y_n^2 + Y_n = 2\frac{X_{n-1}}{W^2} + 1 + \sqrt{8\frac{X_{n-1}}{W^2}+1}. \tag{III.9}$$

L'éq. (III.7) nous permet d'écrire que : $Y_{n-1} = \frac{\sqrt{8\frac{X_{n-1}}{W^2}+1}-1}{2}$. En utilisant l'hypothèse (H_{n-1}), il vient que : $\frac{\sqrt{8\frac{X_{n-1}}{W^2}+1}-1}{2} = n-1$ qui donne en l'élevant au carré : $8\frac{X_{n-1}}{W^2}+1 = 4n^2+1-4n$ soit $2\frac{X_{n-1}}{W^2} = n(n-1)$. En reprenant l'éq. (III.9), on obtient :

$$Y_n^2 + Y_n = n(n-1)+1+\sqrt{4n^2-4n+2} = n^2-n+1+\sqrt{(2n-1)^2} = n(n+1).$$

Les 2 solutions sont $Y_n = n$ et $Y_n = -n-1$. La seule solution positive est donc $Y_n = n$. □

Comme dans [DIIM04], afin d'éviter les problèmes de bords, nous ajoutons un offset b qui permet de décaler légèrement le découpage. L'équation (III.6) devient alors :

$$h_{\mathbf{a},b}(\mathbf{p}) = \lfloor y(\mathbf{a}.\mathbf{p}) + b \rfloor \tag{III.10}$$

avec b qui suit une loi uniforme sur l'intervalle $[0,1[$.

Ayant exhibé $h_{\mathbf{a},b}$, l'accès au bucket d'intérêt est désormais de complexité $O(1)$.

1.3 χ^2-*LSH* : une fonction Locality-Sensitive

Soit $\mathcal{H}$ l'ensemble des fonctions $h_{\mathbf{a},b}$ définies par l'équation (III.10). Dans ce paragraphe, nous allons démontrer que la famille de fonctions $\mathcal{H}$ respecte la Définition 1 du schéma LSH.

Théorème 1 (χ^2-*LSH* sensitivity) *Tant que les vecteurs de l'espace d'entrée sont des distributions, la famille de fonctions $\mathcal{H}$ est (r, rc', p_1, p_2)-sensitive.*

Preuve : Soit $P = P_{\mathcal{H}}[h(\mathbf{p}) = h(\mathbf{q})] =$

$$P_{\mathbf{a},b}\left[\left\lfloor \frac{\sqrt{(8.\frac{\mathbf{a}.\mathbf{p}}{W^2}+1)}-1}{2}+b\right\rfloor = \left\lfloor \frac{\sqrt{(8.\frac{\mathbf{a}.\mathbf{q}}{W^2}+1)}-1}{2}+b\right\rfloor\right]$$

Comme $\mathcal{H}$ dépend uniquement des V.A. $\mathbf{a}$ et b, les écritures $P_{\mathcal{H}}$ et $P_{\mathbf{a},b}$ sont équivalentes.

Suivant $\mathbf{a}$, on peut distinguer deux cas : $\mathbf{a}.\mathbf{p} \leq \mathbf{a}.\mathbf{q}$ et $\mathbf{a}.\mathbf{p} > \mathbf{a}.\mathbf{q}$, d'ou :

$$\begin{aligned} P_{\mathbf{a},b}[h(\mathbf{p}) = h(\mathbf{q})] &= P_{\mathbf{a},b}[h(\mathbf{p}) = h(\mathbf{q}) \mid \mathbf{a}.\mathbf{p} \leq \mathbf{a}.\mathbf{q}].P_{\mathbf{a},b}[\mathbf{a}.\mathbf{p} \leq \mathbf{a}.\mathbf{q}] \\ &+ P_{\mathbf{a},b}[h(\mathbf{p}) = h(\mathbf{q}) \mid \mathbf{a}.\mathbf{q} < \mathbf{a}.\mathbf{p}].P_{\mathbf{a},b}[\mathbf{a}.\mathbf{q} < \mathbf{a}.\mathbf{p}] \\ &= P_{\mathbf{a},b}[h(\mathbf{p}) = h(\mathbf{q}) \mid \mathbf{a}.\mathbf{p} \leq \mathbf{a}.\mathbf{q}].\frac{1}{2} + P_{\mathbf{a},b}[h(\mathbf{p}) = h(\mathbf{q}) \mid \mathbf{a}.\mathbf{q} < \mathbf{a}.\mathbf{p}].\frac{1}{2} \end{aligned}$$

Or **p** et **q** ne représentent pas des variables aléatoires et leurs rôles sont symétriques dans la probabilité que l'on souhaite déterminer, on a donc :

$$P_{\mathbf{a},b}[h(\mathbf{p}) = h(\mathbf{q})] \quad = \quad P_{\mathbf{a},b}[h(\mathbf{p}) = h(\mathbf{q}) \mid \mathbf{a}.\mathbf{p} \leq \mathbf{a}.\mathbf{q}]$$

Ainsi :

$$P_{\mathbf{a},b}[h(\mathbf{p}) = h(\mathbf{q})] \quad = \quad \sum_n P_{\mathbf{a},b}[h(\mathbf{q}) = n \mid h(\mathbf{p}) = n, \mathbf{a}.\mathbf{p} \leq \mathbf{a}.\mathbf{q}].P_{\mathbf{a},b}[h(\mathbf{p}) = n]$$

On a construit la fonction de hachage en χ^2 parce que les données sont « plus » uniformément réparties dans chaque intervalle de hachage qu'avec la fonction proposée par Datar *et al.* [DIIM04]. On peut donc considérer que le fait que la clé de hachage du point **p** soit n est un évènement équiprobable par rapport à n. On en déduit que :

$$\begin{aligned} P_{\mathbf{a},b}[h(\mathbf{p}) = h(\mathbf{q})] \quad &= \quad P_{\mathbf{a},b}[h(\mathbf{q}) = n \mid h(\mathbf{p}) = n, \mathbf{a}.\mathbf{p} \leq \mathbf{a}.\mathbf{q}] \sum_n P_{\mathbf{a},b}[h(\mathbf{p}) = n] \\ &= \quad P_{\mathbf{a},b}[h(\mathbf{q}) = n \mid h(\mathbf{p}) = n, \mathbf{a}.\mathbf{p} \leq \mathbf{a}.\mathbf{q}] \end{aligned}$$

$$P_{\mathbf{a},b}[h(\mathbf{p}) = h(\mathbf{q})] \quad = \quad P_{\mathbf{a},b}\left[n \leq \frac{\sqrt{(8.\frac{\mathbf{a}.\mathbf{p}}{W^2}+1)}-1}{2} + b \leq \frac{\sqrt{(8.\frac{\mathbf{a}.\mathbf{q}}{W^2}+1)}-1}{2} + b < n+1 \right]$$

La relation $n \leq \frac{\sqrt{(8.\frac{\mathbf{a}.\mathbf{p}}{W^2}+1)}-1}{2} + b \leq \frac{\sqrt{(8.\frac{\mathbf{a}.\mathbf{q}}{W^2}+1)}-1}{2} + b < n+1$ nous permet de déduire les deux relations suivantes :

– Conditions sur b :

$$n - \frac{\sqrt{(8.\frac{\mathbf{a}.\mathbf{p}}{W^2}+1)}-1}{2} \leq b < n+1 - \frac{\sqrt{(8.\frac{\mathbf{a}.\mathbf{q}}{W^2}+1)}-1}{2} \tag{III.11}$$

– Conditions sur **a** :

$$0 \leq \frac{\sqrt{(8.\frac{\mathbf{a}.\mathbf{q}}{W^2}+1)}-1}{2} + b - \frac{\sqrt{(8.\frac{\mathbf{a}.\mathbf{p}}{W^2}+1)}-1}{2} - b \leq n+1 - \frac{\sqrt{(8.\frac{\mathbf{a}.\mathbf{q}}{W^2}+1)}-1}{2} - b \leq n+1-n$$

d'où

$$0 \leq \frac{\sqrt{(8.\frac{\mathbf{a}.\mathbf{q}}{W^2}+1)}-1}{2} - \frac{\sqrt{(8.\frac{\mathbf{a}.\mathbf{p}}{W^2}+1)}-1}{2} \leq 1 \tag{III.12}$$

De l'équation (III.11), on déduit les bornes d'intégration de la variable b :

$$\begin{aligned} P = P_{\mathbf{a},b}[h(\mathbf{p}) = h(\mathbf{q})] &= \int_{\mathbf{a}} \int_{n-\frac{\sqrt{(8.\frac{\mathbf{a}.\mathbf{q}}{W^2}+1)}-1}{2}}^{n+1-\frac{\sqrt{(8.\frac{\mathbf{a}.\mathbf{q}}{W^2}+1)}-1}{2}} db f_{\mathbf{a}}(\mathbf{a}) d\mathbf{a} \\ &= \int_{\mathbf{a}} \int_{n-\frac{\sqrt{(8.\frac{\mathbf{a}.\mathbf{p}}{W^2}+1)}-1}{2}}^{n+1-\frac{\sqrt{(8.\frac{\mathbf{a}.\mathbf{q}}{w^2}+1)}-1}{2}} db f_{\mathbf{a}}(\mathbf{a}) d\mathbf{a} \\ &= \int_{\mathbf{a}} 1 - \left(\frac{\sqrt{(8.\frac{\mathbf{a}.\mathbf{q}}{W^2}+1)}-1}{2} - \frac{\sqrt{(8.\frac{\mathbf{a}.\mathbf{p}}{w^2}+1)}-1}{2} \right) f_{\mathbf{a}}(\mathbf{a}) d\mathbf{a} \end{aligned}$$

avec $f_{\mathbf{a}}$ la fonction de densité de la V.A. $\mathbf{a}$. En rappelant que $h(\mathbf{p}) = n$ et que $\mathbf{ap} \leq \mathbf{aq}$, d'après l'équation (III.3) on obtient :

$$\begin{aligned} & 0 \leq \frac{\sqrt{(8.\frac{\mathbf{a}.\mathbf{q}}{W^2}+1)}}{2} - \frac{\sqrt{(8.\frac{\mathbf{a}.\mathbf{p}}{W^2}+1)}}{2} \leq 1 \\ \Leftrightarrow \quad & (2n-1)^2 \leq 8.\frac{\mathbf{a}.\mathbf{q}}{W^2} + 1 \leq 8.\frac{\mathbf{a}.\mathbf{p}}{W^2} + 1 \leq (2n+1)^2 \\ \Leftrightarrow \quad & \tfrac{n(n-1)}{2} W^2 \leq \mathbf{ap} \leq \mathbf{aq} \leq \tfrac{n(n+1)}{2} W^2 \\ \Leftrightarrow \quad & 0 \leq \mathbf{a}(\mathbf{q}-\mathbf{p}) \leq nW^2 \end{aligned}$$

On en déduit que

$$0 \leq \frac{\sqrt{(8.\frac{\mathbf{a}.\mathbf{p}}{W^2}+1)}}{2} - \frac{\sqrt{(8.\frac{\mathbf{a}.\mathbf{q}}{W^2}+1)}}{2} \leq 1 \Leftrightarrow 0 \leq \frac{\mathbf{a}(\mathbf{p}-\mathbf{q})}{nW^2} \leq 1 \tag{III.13}$$

D'où :

$$P = \int_{\mathbf{a}} \left(1 - \frac{\mathbf{a}(\mathbf{p}-\mathbf{q})}{nW^2} \right) f_{\mathbf{a}}(\mathbf{a}) d\mathbf{a}$$

Comme dans [DIIM04], nous utilisons la propriété de la distribution 2-stable : soit deux vecteurs $\mathbf{p}$ et $\mathbf{q}$, une variable aléatoire $\mathbf{a}$ dont chaque composante suit une loi normale centrée réduite. La variable aléatoire $\mathbf{a}.(\mathbf{q}-\mathbf{p})$ a la même distribution que cT où $c = ||\mathbf{p}-\mathbf{q}||_2$ et T est une variable aléatoire qui suit une loi normale centrée réduite.

On obtient donc :

$$0 \leq \frac{\mathbf{a}(\mathbf{p}-\mathbf{q})}{nW^2} \leq 1 \equiv 0 \leq \frac{Tc}{nW^2} \leq 1 \tag{III.14}$$

D'où :

$$P = \int_T \left(1 - \frac{Tc}{nw^2}\right) f_T(T)dT$$

avec f_T la fonction de densité de probabilité de la valeur absolue de la distribution normale.

De l'équation (III.14), on déduit les bornes d'intégration de la variable T :

$$P = \int_0^{\frac{nW^2}{c}} \left(1 - \frac{cT}{nW^2}\right) f_T(T)dT$$

En posant $T = ct$, on obtient :

$$P = \int_0^{nW^2} \frac{1}{c}(1 - \frac{t}{nW^2}) f_T(\frac{t}{c})dt$$

P dépend de $c = ||\mathbf{p} - \mathbf{q}||_2$ au lieu de dépendre de $c' = \chi^2(\mathbf{p}, \mathbf{q})$. Par hypothèse sur l'espace d'entrée, $\mathbf{p}_i$ et $\mathbf{q}_i \in \mathbb{Z}^+$. Les distances χ^2 et l_2 ont donc les mêmes variations : si c' décroît, c décroît aussi et si c' croît, c croît aussi. Comme P est décroissante suivant c, P est également décroissante suivant c'. Comme $r < rc'$, il vient que $p_2 < p_1$. □

2 Extension du Multi-Probe : χ^2-*MPLSH*

Pour intégrer notre clé de hachage dans le schéma Multi-Probe, il faut être capable de calculer la probabilité de collision entre un point $\mathbf{p}$ quelconque et la requête $\mathbf{q}$ « agitée » par le vecteur de perturbation Δ (comme défini page 41) :

$$Pr[g(\mathbf{p}) = g(\mathbf{q}) + \Delta]$$

Comme toutes les composantes h_i de la fonction de hachage g sont indépendantes, nous avons :

$$Pr\left[g(\mathbf{p}) = g(\mathbf{q}) + \Delta\right] = \prod_{i=1}^{M} Pr\left[h_i(\mathbf{p}) = h_i(\mathbf{q}) + \delta_i\right] \quad \text{(III.15)}$$

On peut noter que chaque fonction h_i projette $\mathbf{q}$ sur une droite de l'espace puis divise cette droite en tranches régulières de largeur W. Nous rappelons que le découpage est fait au sens de la distance du χ^2, *i.e.* la distance entre deux délimiteurs consécutifs X_i et X_{i+1} vaut : $\chi^2(X_i, X_{i+1}) = W$. Les tranches sont ensuite numérotées, à l'aide de la fonction h_i, de sorte que deux tranches adjacentes

aient des numéros consécutifs. Soit $\mathbf{q}$ un point requête et $\mathbf{p}'$, le plus proche voisin de $\mathbf{q}$. Le point $\mathbf{p}'$ a une probabilité importante de se trouver dans la même tranche que $\mathbf{q}$. Cependant, la probabilité que $\mathbf{p}'$ se trouve dans une des deux tranches adjacentes : $h_i(\mathbf{q})-1$ ou $h_i(\mathbf{q})+1$ est très importante aussi. En fait, plus $\mathbf{q}$ est proche du délimiteur de gauche X_i, plus la probabilité que $\mathbf{p}'$ se trouve dans la tranche $h_i(\mathbf{q})-1$ est importante. Inversement, plus $\mathbf{q}$ est proche du délimiteur de droite X_{i+1}, plus la probabilité que $\mathbf{p}'$ se trouve dans la tranche $h_i(\mathbf{q})+1$ est importante. On comprend que la position du point requête $\mathbf{q}$ dans la tranche $h_i(\mathbf{q})$ permet de déterminer la probabilité de succès : $Pr[h_i(\mathbf{p}) = h_i(\mathbf{q})+\delta_i]$. Pour $\delta \in \{-1,+1\}$, soit $x_i(\delta)$ la distance entre $\mathbf{q}$ et la borne se trouvant entre les tranches $h_i(\mathbf{q})$ et $h_i(\mathbf{q})+\delta$:

$$x_i(\delta) = |k_i(\mathbf{q}) - h_i(\mathbf{q}) - \delta| \tag{III.16}$$

avec $k_i(\mathbf{q}) = y(\mathbf{a}.\mathbf{q})+b$.

Comme, dans [LJW+07], nous estimons la probabilité que $\mathbf{p}$ se trouve dans la tranche $h_i(\mathbf{q})+\delta$ par :

$$Pr[h_i(\mathbf{p}) = h_i(\mathbf{q})+\delta] \approx e^{-Cx_i(\delta)^2} \tag{III.17}$$

ou C est une constante dépendant des données.

A partir des équations (III.15) et (III.17), nous pouvons déduire le score correspondant à la chance de trouver un point proche de $\mathbf{q}$ dans le bucket défini par Δ :

$$score(\Delta) = \sum_{i=1}^{M} x_i(\delta_i)^2 \tag{III.18}$$

Plus le score est faible, plus la probabilité de trouver un point proche de $\mathbf{q}$ dans le bucket déterminé par Δ est importante.

Pour chaque requête, le calcul de la probabilité de succès de tous les vecteurs « agités » par Δ permet de sélectionner les T buckets dont la probabilité de contenir un des plus proches voisins est maximale. Ce schéma est appelé **χ^2-MPLSH**.

3 Evaluation

Dans un premier temps, nous évaluons l'efficacité du système. L'efficacité est mesurée selon la précision et la rapidité de la recherche approximée par rapport à la recherche exacte (cf. Sec. I.4.1). Notre première évaluation permet de prouver empiriquement que notre fonction de hachage permet de réduire de façon importante les temps de recherche sans trop dégrader les résultats. L'évaluation de l'efficacité du système est également donnée avec une mesure de l'occupation des ressources mé-

moires. Sachant que le schéma classique de recherche LSH demande beaucoup de mémoire vive, il est important de considérer le schéma Multi-Probe. Notre seconde évaluation permet de prouver empiriquement que le schéma Multi-Probe s'adapte bien à notre fonction de hachage. Nous montrons que ce schéma permet d'obtenir les mêmes performances que le schéma classique tout en réduisant l'occupation des ressources mémoires.

Dans un second temps, nous évaluons la qualité de la recherche (critère défini en Sec. I.4.1). Ce critère mesure le taux d'images pertinentes effectivement retrouvées. Cette évaluation permet de comparer la pertinence des distances χ^2 et l_2 quand les images sont indexées par des descripteurs globaux. Cette évaluation est réalisée à l'aide d'une tâche de classification supervisée.

3.1 Efficacité du hachage χ^2

3.1.1 Protocole expérimental

Aucun consensus n'est actuellement adopté par la communauté sur un protocole expérimental standard permettant d'évaluer les méthodes d'indexation en grande dimension. Le choix de la base de données, des requêtes et de la vérité terrain est toujours difficile. Bien que des chercheurs aient proposé d'utiliser des données synthétiques, suivant une distribution aléatoire uniforme, il est maintenant accepté que ce contexte d'évaluation est peu réaliste. En règle générale, les travaux récents évaluent leur méthode sur des données réelles. Nous évaluons les performances de notre méthode avec la base de vidéos TrecVid (cf. Annexe A.1). Chaque keyframe est représentée par un vecteur de 128 dimensions obtenu en concaténant deux histogrammes, un histogramme de 64 chrominances de l'espace CIE Lab et un second de 64 textures issues des filtres de Gabor.

Pour évaluer l'influence de la taille de la base de données, nous avons considéré 3 ensembles, le premier, de 43 616 images correspondant à la base de données Trecvid 2007 (cf. Annexe A.1), appelé DB1, le second, de 86 077 images obtenu en y ajoutant la base Trecvid 2008, appelé DB2 et la troisième, de 158 763 images, obtenu en ajoutant la base Trecvid 2009, appelé DB3. Pour chaque base de données, nous avons créé un banc de test en prenant aléatoirement 25 000 images requêtes.

Pour chaque image requête, la vérité terrain est construite en effectuant une recherche *KNN* exhaustive (image requête exclue) basée sur la distance χ^2. La vérité terrain permet, d'une part, d'obtenir le temps de recherche de référence et ainsi d'évaluer le facteur de rapidité de χ^2-*LSH*. Elle permet, d'autre part, de construire l'ensemble des k images à retrouver afin d'évaluer la précision de la recherche approximée.

Pour mesurer la précision nous utilisons la méthode d'estimation classique qui consiste à calculer la fraction d'images pertinentes retrouvées. La rapidité est mesurée par le facteur de gain en temps de recherche défini comme le ratio entre le temps de recherche de la méthode exacte et le temps de

la recherche approximée. Cette mesure a l'avantage de ne dépendre ni de la machine ni du système d'exploitation utilisé.

Le nombre d'images recherchées k peut varier suivant l'application visée, la base considérée, voire les besoins de l'utilisateur. N'ayant aucune information pour choisir k, il nous faut choisir une valeur arbitraire. Dans [GIM99], les évaluations ont été réalisée avec $k = 1$ et $k = 10$. Nous avons préféré fixer $k = 20$ qui a été utilisée dans [LJW+07], [VCPF07].

3.1.2 Efficacité de la recherche approximée : recherche exacte vs χ^2-*LSH*

L'efficacité de χ^2-*LSH* est évaluée avec deux jeux d'expérimentations qui permettent de mesurer l'influence des paramètres LSH ainsi que celle de la taille du jeu de données.

Les trois paramètres de l'algorithme *LSH* sont : le nombre de tables de hachage (L), le nombre de projections par fonction de hachage (M) et la largeur de la fenêtre de recherche (W).

Le premier jeu d'expérimentations permet d'étudier l'influence des paramètres M et W. Nous utilisons le paramètre L pour contrôler le compromis entre précision et rapidité.

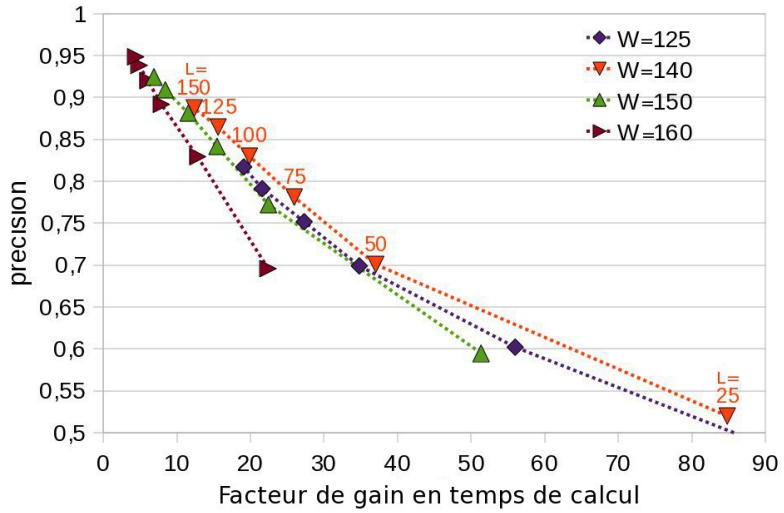


Figure III.2 – Influence du paramètre W de l'algorithme χ^2-*LSH* : évaluation de la précision en fonction du facteur de gain en temps de calcul pour 4 valeurs de W avec M fixé à 26 et 6 valeurs de L comprises entre 25 et 150.

Pour étudier l'influence de W, nous fixons M à 26 et nous faisons varier W et L. La Figure III.2 montre l'influence des deux paramètres W et L pour la base DB3. Nous faisons varier W dans l'intervalle $[125, 160]$ et L dans l'intervalle $[25, 150]$. Pour chaque paire de paramètres testée, la précision et la rapidité sont reportées.

On observe que pour une même valeur de L, augmenter la valeur de W augmente les temps de recherche. En effet, plus on augmente la valeur de W, plus le découpage de l'espace opéré par les

fonctions h_i est grossier. Par conséquent, le nombre de buckets diminue et le nombre de points par bucket augmente. De plus, plus W est grand, plus la distance moyenne entre les points d'un même bucket augmente. Il en résulte que, lors d'une recherche, le nombre de faux points positifs augmente. Le nombre de candidats à examiner lors de l'étape de vérification des collisions est plus important ce qui accroît les temps de recherche.

Cependant, augmenter W a également comme conséquence d'augmenter la précision. Comme le nombre de points par bucket augmente, la probabilité que les « bons » points (les ppv du point requête) se trouvent dans le bucket requête est donc plus importante. Le nombre de faux négatifs diminue.

Un compromis entre la diminution des faux positifs et la diminution des faux négatifs doit donc être trouvé.

Comme le montre la Figure III.2, le meilleur compromis est obtenu pour $W = 140$. En effet, pour toutes les valeurs de L testées, la courbe correspondant à $W = 140$ (en orange sur la figure) offre, pour une précision donnée, un gain en temps de calcul supérieur aux autres valeurs de W. Cependant, elle ne permet pas d'atteindre une précision supérieur à 90% pour un nombre de tables limité à 150. Etant donnée qu'il devient difficile d'envisager d'utiliser plus de 150 tables de hachage (pour des raisons de ressources mémoires : 150 $\times$ taille de la base $\times$ taille d'un pointeur vers un point) si une précision supérieure est nécessaire, il devient préférable de diminuer la valeur de W.

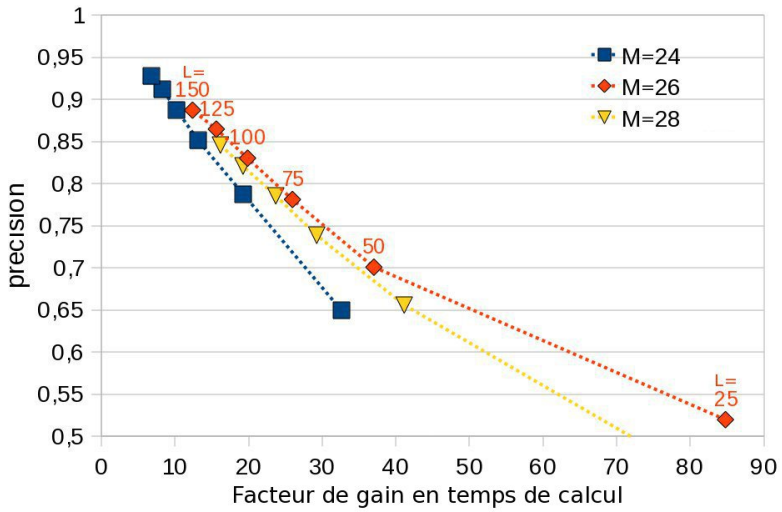


Figure III.3 – Influence du paramètre M de l'algorithme χ^2-*LSH* : évaluation de la précision en fonction du facteur de gain en temps de calcul pour 3 valeurs de M avec W fixé à 140 et 6 valeurs de L comprises entre 25 et 150.

Pour étudier l'influence de M, nous fixons W à 140 et nous faisons varier M et L. La Figure III.3 montre l'influence des deux paramètres M et L pour la base DB3. Nous faisons varier M dans l'intervalle $[24, 28]$ et L dans l'intervalle $[25, 150]$. Pour chaque paire de paramètres, la précision et la rapidité sont reportées.

On observe que pour une même valeur de L, augmenter la valeur de M permet d'accroître la rapidité de la recherche. En effet, augmenter la valeur de M entraîne une augmentation du nombre de découpages de l'espace. Par conséquent, le nombre de buckets augmente et le nombre de points par bucket diminue. Plus M est grand, plus les points appartenant à un même bucket sont proches les uns des autres. Il en résulte que, lors d'une recherche, le nombre de faux positifs diminue. Le nombre de candidats à examiner lors de l'étape de vérification des collisions diminue entraînant une diminution des temps de recherche.

Cependant, l'augmentation du nombre de découpages entraîne également la diminution de la précision de recherche. En effet, le nombre de faux négatifs, *i.e.* le nombre de points réellement proches de la requête qui ne sont pas dans les buckets d'intérêt, augmente.

Le paramètre M permet également de régler le compromis entre diminution des faux positifs et diminution des faux négatifs.

Comme le montre la Figure III.3, le meilleur compromis est obtenu pour $M = 26$. En effet, pour toutes les valeurs de L testées, la courbe correspondant à $M = 26$ (en orange sur la figure) offre, pour une précision donnée, un gain en temps de calcul supérieur aux autres valeurs de M.

Pour résumer $W = 400$ et $M = 26$ sont considérés comme paramètres par défaut puisqu'ils offrent les meilleurs compromis entre précision et rapidité.

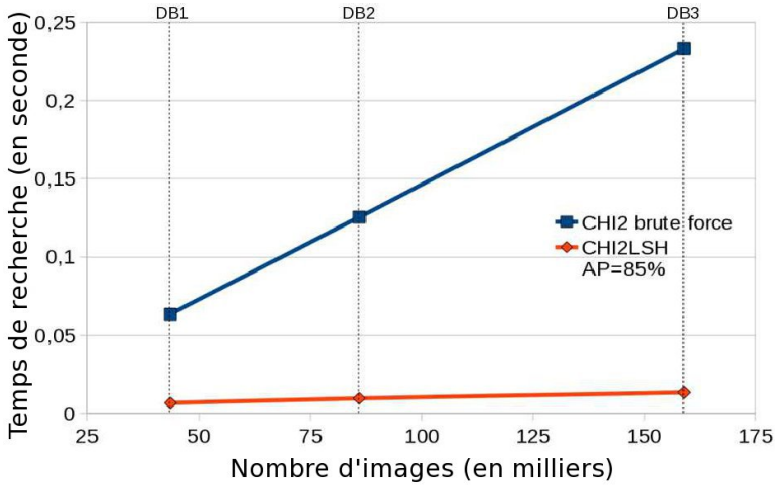


Figure III.4 – Inluence de la taille de la base sur les temps de recherche : comparaison des temps de recherche entre χ^2-*LSH* et la recherche exhaustive en fonction du nombre d'images pour les bases DB1, DB2 et DB3. La précision de la recherche approximé est fixée à 85%.

La complexité de recherche décrit l'évolution des temps de recherche lorsque le volume de données varie. Le second jeu d'expérimentations a pour but de prouver que la complexité de notre schéma est sous-linéaire : $O(n^{\rho})$ avec $\rho < 1$. Il est possible de proposer une méthode d'approximation plus rapide que la recherche exhaustive tout en gardant une complexité linéaire. Pour prouver que notre

méthode a une complexité sous-linéaire, il n'est donc pas suffisant de prouver que la recherche soit plus rapide que la recherche exhaustive. Il faut donc montrer que lorsque la quantité de données augmente, l'augmentation des temps de recherche est moins rapide que celle du volume de données.

Nous proposons donc de mesurer l'évolution du temps de recherche en faisant varier la taille de la base. Le temps de recherche est ainsi présenté sur la Figure III.4 pour les 3 bases, DB1, DB2 et DB3 en fixant la précision de recherche à 85%. Cette précision est obtenue avec W qui vaut 149, 144 et 140 respectivement pour les bases DB1, DB2 et DB3 et en fixant $M = 26$ et $L = 115$ pour les 3 bases. Comme le montre la Figure III.4, la recherche exhaustive a une complexité linéaire : la base DB3 contient $3,64$ fois plus d'images que la base DB1 et le temps de recherche pour DB3 est $3,67$ fois plus lent que pour DB1 (respectivement 233 et 63 millisecondes).

Pour notre schéma rapide, la durée de recherche pour DB3 et seulement $1,98$ fois plus lent que pour DB1 (respectivement $13,44$ et $6,77$ millisecondes). Pour une précision de 85%, χ^2-*LSH* est $9,37$ fois plus rapide que la recherche exhaustive pour une base de 44K images et devient $17,35$ fois plus rapide pour une base de 160K images. La rapidité de notre méthode de recherche augmente donc avec la taille de la base sans dégrader la précision ce qui prouve que notre méthode a une complexité sous-linéaire.

3.1.3 Ressources mémoires : recherche exacte vs χ^2-*MPLSH*

Dans cette partie, nous évaluons l'amélioration de χ^2-*MPLSH* sur χ^2-*LSH* en termes de ressources mémoires. Avant de mesurer la diminution d'occupation mémoire, nous vérifions que l'algorithme de recherche χ^2-*MPLSH* permet d'obtenir la même efficacité que χ^2-*LSH*. L'ensemble des tests sont réalisés sur la base DB3.

Les premiers tests ont pour but de vérifier que χ^2-*MPLSH* est capable d'atteindre une précision de recherche similaire à χ^2-*LSH* tout en utilisant très peu de tables de hachage et donc beaucoup moins de ressources mémoires. Figure III.5, nous mesurons la précision en fonction de la rapidité de recherche de χ^2-*MPLSH* avec les paramètres par défaut de χ^2-*LSH* : $W = 140$ et $M = 26$ pour 2, 4 et 6 tables de hachage. χ^2-*MPLSH* requiert le réglage d'un paramètre supplémentaire, le nombre de probes T (nombre de buckets visités par table de hachage). Nous avons fait varier T entre 25 et 150. Ces valeurs peuvent paraître importantes mais elles sont à rapprocher du nombres de tables de hachage utilisées habituellement avec l'algorithme χ^2-*LSH*. Il peut être noté que ces valeurs sont similaires à celles testées dans [LJW+07].

Comme montré sur la Figure III.5, pour atteindre une précision entre $0,8$ et $0,9$, seules 6 tables sont nécessaire pour χ^2-*MPLSH* contre entre 75 et 150 pour χ^2-*LSH* (voir Fig. III.3). De plus, seules 4 tables de hachage suffisent pour atteindre une précision entre $0,6$ et $0,8$.

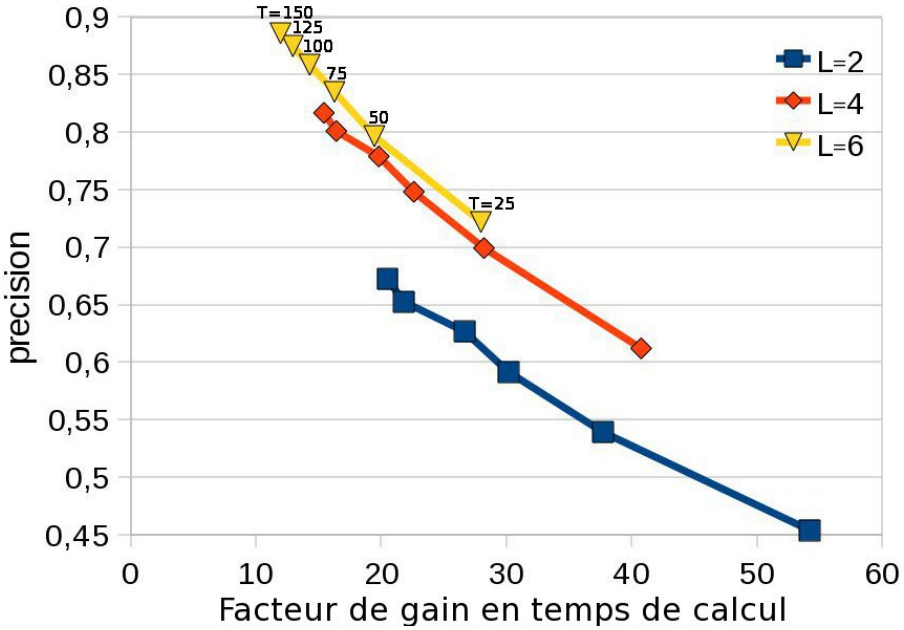


Figure III.5 – Nombre de tables de hachage L nécessaire à χ^2-*MPLSH* pour atteindre une précision importante : courbes de précision en fonction de la rapidité de recherche pour 3 valeurs de L comprises entre 2 et 6. W et M sont fixés en utilisant les paramètres par défaut de χ^2-*LSH*. 6 valeurs de T permettent de contrôler le compromis entre efficacité et rapidité de la recherche.

L'utilisation des ressources mémoires en fonction du nombre de tables de hachage est fournie dans les Tableaux III.1 et III.2 respectivement pour les algorithmes χ^2-*LSH* et χ^2-*MPLSH*.

L	25	50	75	100	125	150
RAM (Mo)	1 000	1 650	2 240	2 860	3 470	4 090

TAB. III.1 – Occupation mémoire de χ^2-*LSH* en fonction du nombre de tables de hachage L pour la base DB3.

L	2	4	6
RAM (Mo)	445	527	546

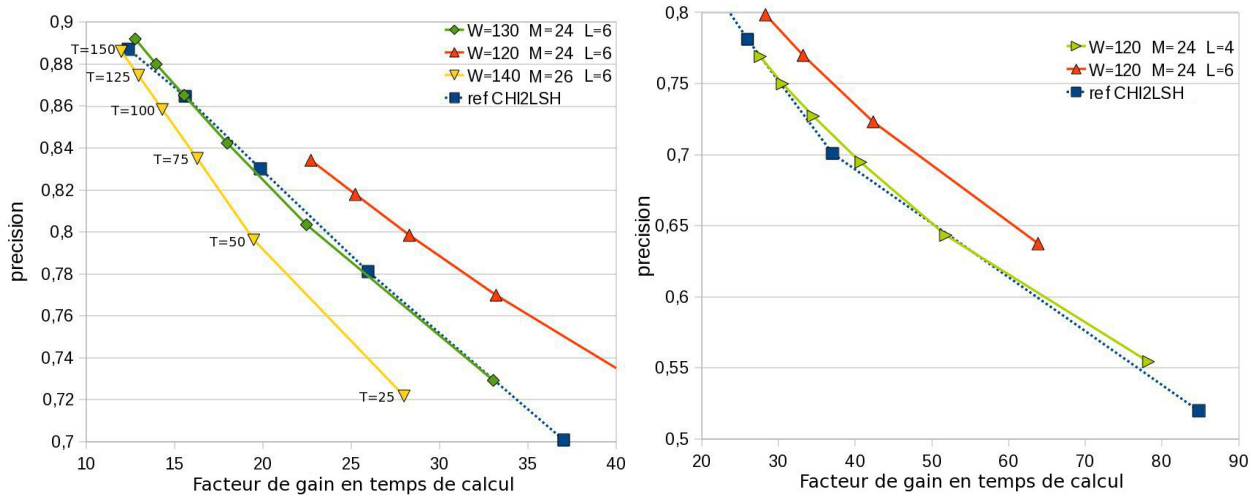
TAB. III.2 – Occupation mémoire de χ^2-*MPLSH* en fonction du nombre de tables de hachage L pour la base DB3.

L'algorithme χ^2-*MPLSH* permet de passer d'une occupation mémoire de l'ordre du Gigaoctet à une occupation mémoire de l'ordre de la centaine de Megaoctets. On observe une diminution d'utilisation des ressources mémoires d'un facteur compris entre 6 et 8 pour une précision de recherche équivalente.

Même si des précisions similaires sont atteintes, la rapidité de recherche est légèrement dégradée. En effet, pour une précision de $0,7$, χ^2-*MPLSH* atteint un facteur de gain en temps de recherche de 30 contre 40 pour χ^2-*LSH*. On peut cependant noter que le paramétrage de χ^2-*MPLSH* a été réalisé

à partir de celui de χ^2-*LSH*. Or les paramètres optimaux des deux méthodes sont peut-être différents. On peut, par exemple, penser que, comme plusieurs buckets autour du bucket requête sont explorés, il est possible de diminuer W sans perte de précision.

Lors du second test, nous étudions l'influence des paramètres W, M, K et L de χ^2-*MPLSH* afin d'optimiser l'efficacité de l'algorithme. Le paramètre T est utilisé pour contrôler le compromis entre rapidité et précision. L'objectif de ce test est de montrer que χ^2-*MPLSH* permet d'atteindre les mêmes performances que χ^2-*LSH*. Nous utilisons donc χ^2-*LSH* comme algorithme de référence (paramètres par défauts : $W = 140$, $M = 26$ et L compris entre 25 et 150). Comme montré sur la Figure III.6,



(a) précision entre 0,9 et 0,7

(b) précision entre 0,8 et 0,5

Figure III.6 – Influence des paramètres de χ^2-*MPLSH* sur l'efficacité de la recherche : courbes précision en fonction de la rapidité de recherche pour W compris entre 120 et 140, M de 24 ou 26, L de 4 ou 6. Pour chaque courbe, le paramètre T varie entre 25 et 150 pour contrôler le compromis entre précision et rapidité. La courbe de référence est donnée par l'algorithme χ^2-*LSH* et ses paramètres par défaut.

paramétrer χ^2-*MPLSH* avec les valeurs W et M par défaut de χ^2-*LSH* (courbe en jaune) n'est pas optimal et ne permet pas d'atteindre l'efficacité de χ^2-*LSH* (courbe en bleu). Cependant, en fixant $M = 24$ et en réduisant W, l'efficacité de χ^2-*MPLSH* s'améliore. Ainsi, fixer $W = 130$, $M = 24$ et $L = 6$ permet à χ^2-*MPLSH* (courbe en vert foncé) d'atteindre les mêmes performances que χ^2-*LSH* pour une précision comprise entre 0,72 et 0,9. De plus, pour $W = 120$, $M = 24$ et $L = 4$, χ^2-*MPLSH* (courbe en vert clair) atteint les mêmes performances que χ^2-*LSH* pour une précision comprise entre 0,55 et 0,76. Il peut même être noté que pour une précision comprise entre 0,65 et 0,80, paramétrer χ^2-*MPLSH* avec $W = 120$, $M = 24$ et $L = 6$ permet d'obtenir une meilleure efficacité de recherche que χ^2-*LSH*.

Finalement, nous venons de montrer que χ^2-*MPLSH* offre la même efficacité de recherche que

χ^2-*LSH* tout en permettant une réduction importante des ressources mémoires. Ainsi, pour une même précision de recherche, χ^2-*MPLSH* permet des gains en occupation mémoire entre 2 et 8 par rapport à χ^2-*LSH*. En considérant une machine de calcul moderne de 32Go de RAM, le système est ainsi capable de traiter une base de 10 millions d'images pour des histogrammes de taille 128.

3.2 Comparaison de la qualité de recherche entre χ^2-*MPLSH* et E^2-*MPLSH*

L'objectif de cette expérimentation est de comparer la qualité de recherche offerte par les deux méthodes : E^2-*MPLSH* et χ^2-*MPLSH* en fonction de la rapidité. Pour évaluer la qualité de recherche des méthodes, on se place dans un contexte simple de classification supervisée. Ce contexte nous permet de montrer que lorsque l'on considère les descripteurs globaux tels que les histogrammes couleurs ou les histogrammes de textures, la distance χ^2 et notre algorithme χ^2-*MPLSH* sont bien plus performants que les structures d'index basées distance l_2.

3.2.1 Protocole expérimental

Comme dans [IT03, CHV99], nous réalisons l'évaluation de notre schéma en construisant une base issue de la base d'images COREL. Cette base, appelée COREL154, contient $c = 154$ classes de 99 images chacunes, ce qui représente une base de $n =$15 246 images.

Nous nous intéressons aux tâches CBIR [GCPF08b, SWS+00] dont l'objectif est de faire des recherches sémantiques. Dans ce contexte, les descripteurs globaux basés couleur et texture sont encore largement utilisés.

Chaque image $\mathbf{p}$ est décrite par une vecteur de 128 dimensions obtenu en concaténant un histogramme couleur de 64 chrominances de l'espace Lab et un histogramme de 64 textures obtenu avec des filtres de Gabor.

Soit $\{\mathbf{p}_i\}_{1,n}$ l'ensemble des n images de la base. L'ensemble d'apprentissage est défini par $\mathcal{A} = \{(\mathbf{p}_i, y_i)_{i=1,n} | y_i \neq 0\}$, où $y_i = c$ si l'image $\mathbf{p}_i$ appartient à la classe c.

Dans cette évaluation, nous considérons un classifieur *KNN* avec pour règle de classification [HTF09] :

$$f_c(\mathbf{p}) = \sum_{N_j(\mathbf{p}) \in KNN(p)} (f(\mathbf{p}, N_j(\mathbf{p})) | y_j = c) \tag{III.19}$$

où $N_j(\mathbf{p})$ est le j^{th} voisin de $\mathbf{p}$ dans l'ensemble $\mathcal{A}$, k le nombre de plus proches voisins considérés et $f(\mathbf{p},\mathbf{q}) = 1 - \frac{D(\mathbf{p},\mathbf{q})}{R}$ avec R le rayon de recherche et $D(\mathbf{p},\mathbf{q})$ la distance l_2 ou χ^2 entre les images $\mathbf{p}$ et $\mathbf{q}$. Nous fixons $k = 20$.

Comme pour le Challenge PASCAL, nous utilisons le score de classification MAP (cf. Annexe A.2). Nous utilisons comme protocole expérimental la stratégie *leave-one-out* : pour chaque

classe c, toutes les images de la classe c sont ajoutées à l'ensemble d'apprentissage comme images pertinentes et toutes les autres images comme images non-pertinentes. A chaque test, une image de la classe pertinente est temporairement retirée de l'ensemble d'apprentissage pour être utilisée comme image requête. Cette image requête permet d'initialiser une recherche. Le résultat de la recherche permet de calculer le score de précision. L'image requête est ensuite remise dans l'ensemble de recherche et une nouvelle image de la classe c est retirée pour effectuer la recherche suivante. Un score AP (*Average Precision*) est calculé pour chaque classe c en moyennant la précision des recherches effectuées pour la classe c. Le score MAP est obtenu en moyennant les scores AP de toutes les classes.

3.2.2 Evaluation

L'évaluation décrite au paragraphe précédent est réalisée avec 4 méthodes de recherche : E^2-*LIN* (recherche exhaustive avec la distance l_2), χ^2-*LIN* (recherche exhaustive avec la distance χ^2), E^2-*MPLSH* et χ^2-*MPLSH*. Ces 4 évaluations ont pour objectif de montrer l'intérêt de la distance χ^2 ainsi que l'intérêt d'une structure d'index dédiée à cette distance.

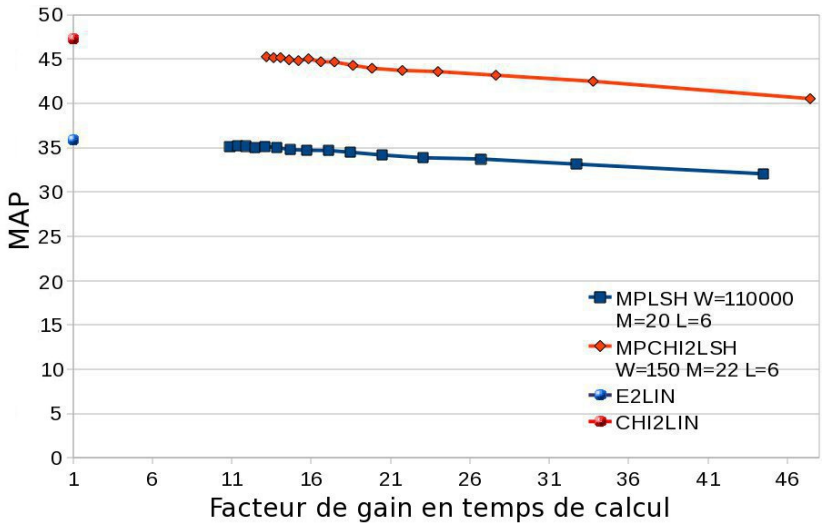


Figure III.7 – Comparaison de la qualité de recherche obtenue avec les distances l_2 et χ^2 : MAP en fonction des gains en temps de calcul pour les 4 méthodes de recherches : E^2-*LIN*, χ^2-*LIN*, E^2-*MPLSH* et χ^2-*MPLSH*.

La Figure III.7 montre le MAP en fonction des gains en temps de calcul des 4 méthodes de recherche comparées. Nous avons fixés expérimentalement les paramètres optimaux des structures d'index E^2-*MPLSH* et χ^2-*MPLSH*. Ainsi, E^2-*MPLSH* est paramétré avec les valeurs : $W = 110000$, $M = 20$ et $L = 6$ et χ^2-*MPLSH* avec :$W = 115$, $M = 22$ et $L = 6$. Nous avons fait varier T entre 150 et 10 pour augmenter la rapidité de recherche au détriment de la précision. Les recherches exhaustives E^2-*LIN* et χ^2-*LIN* permettent de mesurer les gains en temps de recherche ainsi que la précision des

recherches. Nous pouvons ainsi confirmer que dans ce contexte, la distance χ^2 est plus adaptée que la distance l_2. En effet, nous obtenons un MAP de 36 pour la distance l_2 contre 47 en χ^2 soit un gain de $30,56\%$. De plus notre structure d'index permet de diminuer le temps de recherche tout en conservant une qualité de recherche supérieure à E^2-*MPLSH*. Ainsi, pour un gain de temps de calcul de 44, la qualité de recherche de E^2-*MPLSH* est de 32 de MAP contre 41 pour χ^2-*MPLSH* soit un gain de 28%.

4 Conclusion

Nous avons présenté dans ce chapitre notre approche χ^2-*LSH* et son extension χ^2-*MPLSH* pour effectuer des recherches d'images rapides dans les grandes bases. Un grand nombre de tests réalisés sur des bases entre 50 000 et 160 000 images ont validé empiriquement l'efficacité de notre méthode en grande dimension. Pour une base de 160 000 images et une précision de recherche de 85%, χ^2-*LSH* permet d'effectuer des recherches 17 fois plus rapidement qu'une recherche exhaustive (soit un temps de recherche de 13 millisecondes). Enfin, nous avons validé l'intérêt d'une méthode conçue pour la distance χ^2 dans le contexte de recherche d'images basée signature globale.

Conclusion de la Partie I

Dans cette partie, nous avons présenté en détail le principe de LSH, une famille de méthodes de recherche des plus proches voisins basée hachage. Cette famille de méthodes offre un formalisme mathématique qui permet de définir les critères que doivent respecter les méthodes au travers de la définition de fonctions de hachage pour obtenir des algorithmes de recherche efficaces. Une fois le principe de la méthode décrit, nous avons présenté les fonctions de hachage les plus répandues dans la littérature. Nous avons enfin introduit une nouvelle fonction de hachage, χ^2-*LSH*, conçue pour la distance χ^2. Nous avons démontré que cette fonction de hachage respecte le principe de *Locality Sensitive Hashing* et confirmer l'intérêt de notre méthode par de nombreuses expérimentations sur plusieurs bases d'images. Nous avons profité de l'optimisation Multi-Probe pour proposer un schéma de recherche, χ^2-*MPLSH*, utilisant peu de ressources mémoires. Selon nos estimations en terme de temps de calcul et de ressources mémoires, un corpus allant jusqu'à 10 millions d'images peut être traité suivant cette méthodologie.

Deuxième partie

Signature locale et recherche rapide dans les grandes bases d'images.

Dans la partie précédente, nous avons étudié des systèmes de recherche capables de retrouver des images similaires à une image requête en utilisant une signature globale. Cette signature permet de représenter chaque image sous la forme d'un vecteur compact (de l'ordre de 100 dimensions) obtenue en synthétisant les caractéristiques visuelles des images.

Pour améliorer la qualité de recherche, il est possible de concevoir des systèmes de recherche capables d'analyser plus en détail les images. Cette finesse d'analyse doit se retrouver dans toutes les étapes de la recherche, aussi bien lors de l'indexation que lors de la stratégie de recherche en elle-même.

Pour que ces systèmes soient performants, cette finesse de traitement ne doit pas dégrader les performances en temps de recherche et en besoin de ressources mémoires. De plus, ces systèmes doivent être capables de traiter les volumes de données de l'ordre du million d'images.

Toutes ces problématiques font l'objet de cette partie.

Chapitre IV

Similarité entre sacs de caractéristiques locales

Lorsque l'objet d'intérêt ne couvre pas toute la surface de l'image, des descripteurs visuels très précis comme les descripteurs locaux, *i.e.* par zones d'intérêt des images (points, blocs, régions, *etc*) sont plus adaptés que les descripteurs globaux. Chaque image est représentée par un ensemble de caractéristiques locales pertinentes, *e.g.* des points d'intérêt SIFT, pour décrire chaque zone de l'image. Ces descripteurs ont prouvé leur intérêt aussi bien pour les tâches de détection de copies que pour la recherche sémantique.

Dans ce chapitre, nous introduisons le formalisme général dans lequel nous unifions plusieurs approches de l'état de l'art. Puis, nous proposons nos solutions pour optimiser la recherche par similarité dans ce contexte.

Sommaire

1 Formalisme **72**

2 Approches dictionnaires **72**

3 Vector of locally aggregated descriptors (VLAD) **74**

4 Stratégie par vote **75**

5 Approches noyaux **76**

6 Optimisation des fonctions noyaux **78**

7 Conclusion **83**

1 Formalisme

Dans cette partie, une image I_j est représentée par un sac B_j contenant s vecteurs non ordonnés $\mathbf{b}_{sj} \in \mathbb{R}^p$: $B_j = \{\mathbf{b}_{sj}\}_s$. Soit $\mathcal{B}$ la base d'images et $\mathcal{F}$ l'ensemble des caractéristiques visuelles $\mathbf{b}$ de toutes les images de $\mathcal{B}$. Soit I_q l'image requête représentée par son sac $B_q = \{\mathbf{b}_{rq}\}_r$.

La première difficulté, que l'on retrouve aussi bien dans le contexte de la détection de copies que celui de la recherche sémantique, est de définir une fonction de similarité capable de fournir un score de ressemblance $score_j = score(I_q, I_j)$ entre l'image requête I_q et une image I_j respectivement représentées par les sacs B_q et B_j. Une fois cette fonction de similarité définie, on se retrouve dans un contexte de recherche classique où le processus de recherche dans la base $\mathcal{B}$ peut s'écrire :

$$\underset{B_j \in \mathcal{B}}{Sort}\,(score_j) \tag{IV.1}$$

Plusieurs stratégies ont été proposées dans la littérature pour définir le score $score_j$. Nous verrons que la plupart de ces stratégies peuvent se réécrire formellement de la façon suivante :

$$score_j = \sum_{\mathbf{b}_{rq} \in B_q} \sum_{\mathbf{b}_{sj} \in B_j} f(\mathbf{b}_{rq}, \mathbf{b}_{sj}) \tag{IV.2}$$

où f est une fonction de similarité entre deux descripteurs locaux $\mathbf{b}_{rq}$ et $\mathbf{b}_{sj}$. Les principales stratégies sont détaillées dans les sections suivantes.

2 Approches dictionnaires

La première solution pour traiter les sacs de vecteurs est de se ramener à une signature vectorielle (cf. Sec. I.2) à l'aide d'un dictionnaire de mots visuels (ou *codebook* en anglais). Le dictionnaire est généralement construit en utilisant des méthodes de regroupement (ou *clustering*), telles que K-means, qui ont pour rôle de partitionner l'ensemble des vecteurs $\mathbf{b}_{sj}$ de l'espace des caractéristiques $\mathcal{F}$ en cellules (ou *cluster*). Le dictionnaire est alors défini par l'ensemble des mots visuels correspondant au prototype de chaque cellule (généralement défini par son centre de gravité). Une image est ensuite représentée par un vecteur, nommé sac de mots (BoW), de la taille du dictionnaire, dont chaque composante référence la présence ou non d'un des mots visuels. Ce vecteur peut être construit de plusieurs façons. Dans la version la plus basique, la $i^{\text{ème}}$ case du BoW contient le nombre d'occurrences du

mot visuel *i* dans l'image. En d'autres termes, une image est représentée par la distribution de l'occurrence des mots du dictionnaire. D'autres approches plus fines ont été proposées telles que considérer la fréquence d'apparition des mots au lieu de leur occurrence, ou d'ajouter une pondération prenant en compte la rareté des mots (tel que le *tf-idf* [PCI+07, YJHN07]).

On se retrouve donc dans le cas étudié au Chapitre II où le score $score_j$ est défini par le produit scalaire entre deux vecteurs BoW ou tout autre similarité usuelle. Les caractéristiques locales permettent de construire des dictionnaires de très grandes tailles [SZ03, CPIZ07] capables de conserver une finesse de représentation importante. C'est pourquoi les BoW construits à partir des caractéristiques locales sont généralement considérés comme des signatures locales [Fou02].

Jegou *et al.* [JDS08a] puis Bo *et al.*[BS09] ont montré que l'utilisation d'un dictionnaire de mots visuels pour la recherche d'images peut se réécrire en utilisant le formalisme défini à l'éq. (IV.2).

La construction du dictionnaire repose sur un quantifieur Q qui s'écrit formellement comme une fonction :

$$\begin{aligned} Q : \mathbb{R}^d &\rightarrow [1,k] \\ \mathbf{x} &\longmapsto Q(\mathbf{x}) \end{aligned} \tag{IV.3}$$

et qui à chaque descripteur local $\mathbf{x} \in \mathbb{R}^d$ fait correspondre un index entier faisant référence à un mot visuel du dictionnaire. Le quantifieur $Q(\mathbf{x})$ correspond alors à l'index du mot visuel le plus proche du descripteur $\mathbf{x}$.

Introduisons la notation générale :

$$f_{\mathcal{P}}(\mathbf{x},\mathbf{y}) = \mathbb{1}_{\mathcal{P}(\mathbf{x},\mathbf{y})} \tag{IV.4}$$

où $\mathbb{1}_{\mathcal{P}(\mathbf{x},\mathbf{y})}$ est la fonction indicatrice qui prend la valeur 1 quand le prédicat $\mathcal{P}$ est vrai et 0 sinon.

La stratégie dictionnaire (dans sa version basique) peut se réécrire avec le formalisme introduit à l'éq. (IV.2) où f est définie par :

$$f_Q(\mathbf{x},\mathbf{y}) = \mathbb{1}_{Q(\mathbf{x})=Q(\mathbf{y})} \tag{IV.5}$$

Le $score_j$ correspond alors à un produit scalaire entre deux vecteurs BoW [JDS08a]. Cette écriture est facilement généralisable aux extensions des approches BoW (voir [Auc09]).

Le grand intérêt de cette approche est que l'étape qui permet de passer d'une représentation de type BoF à une représentation BoW, bien que de complexité très importante, est réalisée hors ligne (lors de l'indexation). Le processus de recherche n'est donc par ralenti par cette représentation très fine. De plus, le produit scalaire est calculé très efficacement en utilisant des *Inverted Files* (ou fichiers inversés) [SZ03] qui exploitent le fait que les vecteurs BoW (de très grande dimension, plus de 1 000 mots) sont très creux (ils contiennent majoritairement des valeurs nulles) et donc que $f_Q(\mathbf{b}_{sj},\mathbf{b}_{rq}) = 0$

pour la majorité des tuples (s, j, r).

Les approches dictionnaires ont deux inconvénients principaux. Le premier repose sur la complexité calculatoire de la construction du dictionnaire. Pour les grandes bases, le dictionnaire est construit à partir d'un échantillon de la base afin de lever cette limitation. Cependant, il est difficile d'estimer la taille correcte de l'échantillon à considérer pour garantir de bonnes performances. Cela dépend du jeu de données (structure, taille et densité des clusters, dimension) et nécessite donc de faire des hypothèses fortes, et la plupart du temps limitantes, sur les clusters ou leur répartition. De plus, les performances des méthodes de regroupement se dégradent lorsque la dimension des données augmente, ce qui limite la qualité du dictionnaire extrait [SDI06, VCPF08b, BMM09].

Le second inconvénient est lié à la quantification qui attribue un mot du dictionnaire à un descripteur local quel que soit son degré de représentativité. Cette assignation « dure » (*hard assignment*) entraîne deux problèmes : incertitude et plausibilité du mot visuel. L'incertitude du mot visuel fait référence au problème de sélection du bon mot visuel lorsque plusieurs candidats sont pertinents. En d'autres termes, à quel mot assigner un vecteur se trouvant sur la frontière entre plusieurs cellules ? L'approche dictionnaire choisit simplement le « meilleur » mot visuel (le plus proche) ignorant la pertinence des autres candidats. La plausibilité du mot visuel dénote le problème de choisir un mot visuel lorsqu'aucun candidat n'est réellement approprié dans le dictionnaire. Cette approche assigne le mot visuel qui s'ajuste le mieux même si ce dernier n'est pas approprié.

Pour résoudre ces problèmes et lisser cette assignation dure, plusieurs approches ont été proposées. Dans [vGGVS08], les auteurs proposent de décrire un descripteur local par plusieurs mots visuels. Ainsi, chaque descripteur a un poids unitaire qui est réparti entre ses meilleurs représentants en fonction de leur pertinence. Ces approches sont appelées assignation « douce » (ou *soft-assignment*).

3 Vector of locally aggregated descriptors (VLAD)

Récemment, différentes approches proposent des modélisations plus sophistiquées que celles des dictionnaires ont été considérées dans la communauté de la vision par ordinateur. La modélisation par mélanges de distributions gaussiennes [PD07, ZCL+09] qui offre un compromis intéressant entre exhaustivité et compacité de la représentation, ont prouvé leur efficacité dans les tâches de classification [EVGW+09].

Dans [JDSP10], Jegou *et al.* ont proposé une simplification de la représentation de Fisher [PD07]. Cette signature est plus compacte que l'approche par BoF et plus informative que celle par BoW. Comme pour l'approche BoW, un dictionnaire $\mathcal{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_M\}$ de M mots est appris. Pour chaque mot visuel $\mathbf{w}_m$, les différences $\mathbf{x} - \mathbf{w}_m$ des descripteurs locaux $\mathbf{x}$ assignés au mot $\mathbf{w}_m$ sont accumulées.

La signature produite est un vecteur $\mathbf{v}$ de $d' = M \times d$ dimensions où M correspond au nombre de mots du dictionnaire et d la dimension d'un descripteur local. Dans la suite, nous décrivons le descripteur par $\mathbf{v}_{m,i}$, où les indices $m = 1 \ldots M$ et $i = 1 \ldots d$ font respectivement référence à l'indice du mot visuel et à la composante du descripteur local. Une composante de $\mathbf{v}$ est donc obtenue en sommant tous les descripteurs $\mathbf{x}$ du sac B :

$$\mathbf{v}_{m,i} = \sum_{\mathbf{x} \in B | Q(\mathbf{x}) = m} \mathbf{x}_i - \mathbf{w}_{m,i} \tag{IV.6}$$

où $\mathbf{x}_i$ et $\mathbf{w}_{m,i}$ font respectivement référence à la $i^{\text{ème}}$ composante du descripteur $\mathbf{x}$ et au mot $\mathbf{w}_m$ qui lui est assigné.

4 Stratégie par vote

Les représentations dictionnaires et leurs dérivées entraînent une perte d'information importante lors de l'étape de fusion du sac de vecteurs en un vecteur unique. Des alternatives, telles que l'approche par vote introduite par David Lowe [Low04], proposent de s'affranchir de cette étape de fusion en travaillant directement sur les données brutes, *i.e.* les BoF.

Les stratégies par vote sont essentiellement utilisées en détection de copies où l'on cherche à retrouver les images de la base contenant des scènes ou des objets identiques à l'image requête. Les images cibles peuvent avoir subi des transformations dues à des traitements après prise de vue, tels que la compression, des recadrages, des incrustations, ... ou à des prises de vues différentes.

L'objectif est donc de rechercher des appariements très forts entre des zones de l'image requête et des zones des images de la base. Le principe de base de la méthode par vote est donc de rechercher, dans l'ensemble $\mathcal{F}$, les plus proches voisins (ppv) de chacun des r vecteurs $\mathbf{b}_{rq}$ de l'image requête I_q. La similarité au niveau image est obtenue avec une fusion tardive des appariements locaux entre images. Cette similarité est calculée en utilisant une stratégie par vote qui consiste à incrémenter le $score_j$ de l'image I_j de 1 pour chaque vecteur $\mathbf{b}_{sj}$ appartenant aux ppv d'un des vecteurs $\mathbf{b}_{rq}$ de l'image requête.

Lorsque l'approche par vote est basée sur une recherche de *RNN* (cf. Sec. II.4), on peut se ramener à l'écriture formelle de l'éq. (IV.2) où f est alors définie par :

$$f_{RNN}(\mathbf{x}, \mathbf{y}) = \mathbb{1}_{D(\mathbf{x},\mathbf{y}) \leq R} \tag{IV.7}$$

où $D(.,.)$ est la distance (ou mesure de dissimilarité) définie dans l'espace des descripteurs locaux. Lorsque l'on considère les descripteurs SIFT, vecteurs de 128 dimensions, la distance Euclidienne est communément utilisée.

Une autre stratégie consiste à utiliser une recherche *KNN* (cf. Sec. II.4), f est alors définie par :

$$f_{KNN}(\mathbf{x},\mathbf{y}) = \mathbb{1}_{\mathbf{x}\in KNN(\mathbf{y})} \tag{IV.8}$$

Généralement, un très grand nombre de vecteurs (entre 100 et 10 000 vecteurs SIFT) sont extraits pour représenter une image [Low04]. Même pour une base de 5 000 images, les recherches *ppv* doivent s'effectuer dans un ensemble $\mathcal{F}$ pouvant contenir entre 500 000 et 50 000 000 descripteurs. Les temps de calcul pour une recherche de *ppv* exacte (linéaire) deviennent rapidement prohibitifs. L'utilisation d'une structure d'index pour accélérer la recherche s'impose et des algorithmes de recherche approximée permettent de trouver un compromis entre efficacité et précision.

5 Approches noyaux

Les fonctions noyaux permettent de définir des fonctions de similarité sur sac prenant plus précisément en compte l'information de tous les descripteurs locaux que les approches par vote et les approches dictionnaires.

Pour construire des noyaux sur sacs de vecteurs, il faut trouver une fonction Φ qui à tout sac B_j fait correspondre un vecteur $\Phi(B_j)$ défini dans un espace de Hilbert. La fonction noyau K est alors définie par un produit scalaire dans l'espace induit :

$$K_j = K(B_q,B_j) = \langle \Phi(B_q), \Phi(B_j) \rangle \tag{IV.9}$$

Dans le cas des fonctions noyaux sur sacs, on note $score_j = K_j$.

Beaucoup de travaux dans le domaine de la Vision par Ordinateur (*Computer Vision* en anglais) se sont intéressés au problème [GD07, CW04, STC04, KJ03].

La méthode proposée par Grauman *et al.*, Pyramid Matching Kernel [GD05, GD07] a connu un grand succès. La fonction noyau est obtenue en projetant les données dans un espace induit explicite, *i.e.* où la fonction d'injection Φ s'écrit explicitement. Cette fonction d'injection permet de projeter les descripteurs b_{sj} du sac B_j dans l'espace de Hamming où le sac B_j est représenté par un vecteur binaire de très grande dimension. Cette projection est obtenue en concaténant plusieurs histogrammes d'occurrence des descripteurs b_{sj} dans les cellules de l'espace des caractéristiques $\mathcal{F}$ définies par des grilles régulières à différentes échelles (cf. Fig. IV.1). Les histogrammes sont ensuite pondérés en fonction de la résolution de l'échelle utilisée avant d'être binarisés. La similarité entre les deux sacs B_j et B_q, donnée par $\langle \Phi(B_j), \Phi(B_q) \rangle$, revient à compter le nombre de descripteurs b_{sj} et b_{rq} tombant dans les mêmes cellules des grilles utilisées pour découper l'espace $\mathcal{F}$. Ce comptage est pondéré par

la taille des cellules considérées. Comme illustré sur la Figure IV.1, cette méthode consiste à apparier les descripteurs des deux images se trouvant dans les mêmes cellules de l'espace $\mathcal{F}$.

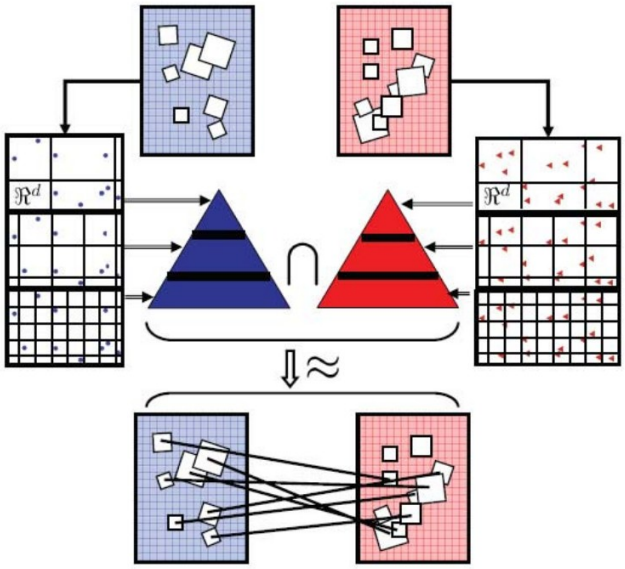


Figure IV.1 – Pyramid Matching Kernel : similarité entre deux images basée sur l'appariement des zones locales des images en passant par un espace induit implicite [GD07].

On retrouve donc la notion de comptage comme définie dans l'approche par vote. Cependant, ce comptage prend en compte de façon plus fine la notion de proximité entre descripteurs. Cette fonction permet de mettre en relation des zones des images qui ne sont par strictement identiques mais qui ont des caractéristiques communes. La similarité entre images est donc moins discriminante que l'approche par vote ce qui permet d'utiliser cette fonction de similarité dans un contexte de recherche sémantique (cf. Sec. I.2.2.2).

Ces approches ne sont pas strictement orientées noyaux dans le sens où la plupart du travail se concentre sur des étapes de prétraitement pour projeter les sacs dans un espace $\mathbb{R}^d$ de dimension finie.

Une classe de fonctions noyaux sur sacs sans évaluation explicite de la fonction de projection Φ est donnée par [STC04, KJ03, WCG03, WS03] :

$$K_{basics}(B_j, B_q) = \sum_{\mathbf{b}_{rj} \in B_j} \sum_{\mathbf{b}_{sq} \in B_j} k(\mathbf{b}_{rj}, \mathbf{b}_{sq}) \tag{IV.10}$$

où k est une fonction noyau mineur sur $\mathbf{b}_{rj}$. On a alors Φ la fonction de projection des sacs dans un espace de Hilbert $\mathscr{H}$ défini par $\Phi(B_j) = \sum_{\mathbf{b}_{rq} \in B_j} \phi(\mathbf{b}_{sj})$ avec ϕ, la fonction de projection des descripteurs dans $\mathscr{H}$.

Cette approche peut être vue comme une approche par vote où le score $score_j$ de l'éq. (IV.2) est

définie par K et où la fonction f est défini par k. En d'autres termes, au lieu d'incrémenter le score $score_j$ pour chaque bon appariement (comme c'est le cas pour l'approche dictionnaire et l'approche par vote), tous les appariements possibles participent au vote à la hauteur de leur importance estimée par k.

[Lyu05, GCPF07] étendent cette classe de fonction noyaux en élevant le noyau mineur k à la puissance p dans le but d'augmenter le fossé entre bons et mauvais appariements :

$$K_{power}(B_j, B_q) = \sum_{\mathbf{b}_{sj} \in B_j} \sum_{\mathbf{b}_{rq} \in B_q} (k(\mathbf{b}_{sj}, \mathbf{b}_{rq}))^p \tag{IV.11}$$

La puissance p permet de contrôler la discrimination de la fonction de similarité. Si on se place dans un contexte de recherche sémantique, on souhaite rapprocher des objets qui peuvent être assez différents mais qui appartiennent à une même classe. Il faut donc avoir une fonction de similarité moins discriminante que dans le contexte de la détection de copie. Dans le premier cas, on peut moyenner les similarités locales en utilisant une puissance p faible alors que dans le second cas, on peut augmenter le pouvoir discriminant du noyau en utilisant une puissance p plus élevée.

Bien que ces familles de noyaux soient très pertinentes pour évaluer la similarité entre deux sacs de vecteurs, elles ont une complexité calculatoire très élevée qui limite leur utilisation à des volumes de données très petits. En effet, la complexité de tels noyaux (en $o(nm^2)$ avec n la taille de la base et m le nombre moyen de descripteurs par image) devient impraticable pour les tâches de recherche d'images par similarité dans les grandes bases, spécialement lorsque les sacs de vecteurs contiennent beaucoup de caractéristiques.

Nous proposons par la suite des optimisations de cette famille de noyaux, pour permettre de réduire sa complexité calculatoire tout en gardant sa pertinence.

6 Optimisation des fonctions noyaux

Dans les applications de recherche orientée précision [CMM+98], le classement de toute la base n'est pas requis. Seul le classement des N images les plus similaires à la requête I_q, appelé TOPN, est considéré (généralement, N est fixé par l'utilisateur). Comme seules les images appartenant au TOPN sont pertinentes, il n'est pas nécessaire de calculer la similarité entre I_q et toutes les images de la base. Nous proposons deux méthodes pour diminuer la complexité des noyaux sur sacs. La première méthode, K_{select}, s'inspire de la stratégie par vote. La seconde méthode, *Ultra fast*, est basée sur l'approximation du noyau mineur.

6.1 K_{select}

Le premier schéma d'approximation proposé s'inspire de la stratégie par vote proposée par Lowe [Low04]. L'approche opère en deux étapes.

Dans un premier temps, on constitue un ensemble $\mathcal{S}$ en sélectionnant rapidement les $S = |\mathcal{S}|$ images de la base qui ont la plus grande « *chance* » d'appartenir à l'ensemble des images recherchées, *i.e.* les images les plus similaires à l'image requête I_q (le TOPN). Cette « *chance* » est mesurée à l'aide de l'heuristique suivante : plus une image I_j a de descripteurs $\mathbf{b}_{sj}$ qui s'apparient (proches) avec les descripteurs $\mathbf{b}_{rq}$ de l'image I_q, plus l'image I_j a de *chance* d'appartenir au TOPN. Si aucun des descripteurs d'une image I_j n'est proche d'au moins un des descripteurs locaux de l'image requête, la similarité entre I_j et I_q est forcément trop faible pour faire partie du TOPN. Dans ce cas, on a $K(B_j, B_q) < |B_j||B_q|e^{-R^2/\sigma^2}$ avec R le rayon de recherche permettant de paramétrer l'approche par vote éq. (IV.7). Inversement, plus une image I_j a de descripteurs locaux proches des descripteurs locaux de l'image requête, plus la similarité entre I_j et I_q peut être importante. En effet, si on a m appariements de descripteurs entre les deux images, $K(B_j, B_q) \geq me^{-R^2/\sigma^2}$.

Plusieurs approches sont possibles pour construire $\mathcal{S}$ tout en respectant le schéma proposé. Une première approche possible est de sélectionner toutes les images ayant obtenues au moins p votes. Cependant, il est alors difficile de choisir le paramètre p. De plus, pour une même valeur de p et suivant la difficulté de l'image requête, le nombre d'images sélectionnées peut être très variable, entraînant une variance importante des temps de recherche. C'est pourquoi nous proposons plutôt de sélectionner les S images ayant le plus de votes. Le paramètre S peut alors être choisi suivant la réduction de complexité de recherche souhaitée $r = \frac{|\mathcal{B}|}{S}$. Cette approche est alors beaucoup moins sensible à la variabilité du nombre d'appariements pour l'image considérée entraînant des temps de recherche plus stables.

Cette première estimation de la recherche n'est pas assez précise pour fournir un résultat pertinent.

Une seconde étape est alors utilisée pour trier plus finement les images ainsi sélectionnées. Nous proposons d'améliorer ce premier classement en calculant la vraie similarité noyau. La similarité noyau peut fortement modifier l'ordre des images définies par l'étape de vote. En effet, il est possible qu'une image ayant peu de descripteurs appariés soit plus similaire à l'image requête qu'une image ayant beaucoup d'appariements, tout dépend de la qualité de ces appariements. C'est ce qui est mesuré par le noyau. Le schéma résultant est une approximation du classement obtenu avec le noyau exact. Le problème d'optimisation de l'éq. (IV.1) devient :

$$\underset{B_j \in \mathcal{S}}{Sort}(K(B_q, B_j)) \tag{IV.12}$$

Ce raffinement du tri est uniquement réalisé sur le sous-ensemble $\mathcal{S}$ obtenu à l'étape précédente car la complexité calculatoire de cette étape peut être très importante.

6.2 Ultra fast

Le second schéma proposé pour réduire la complexité de l'approche noyau repose sur la remarque que lorsque l'on calcule la similarité entre 2 images similaires, un grand nombre d'appariements entre descripteurs locaux ont des similarités k trop faibles pour influencer le score total K des images appartenant au TOPN. Pour réduire la complexité du noyau, on propose donc de prendre uniquement en compte les noyaux mineurs k de scores élevés. Le score $UFast_K$ ainsi obtenu est alors une approximation du vrai noyau K.

6.2.1 Principe

Nous introduisons une nouvelle notation $f_{k,\mathcal{P}}$ qui permet de généraliser la notation $f_{\mathcal{P}}$ de l'éq. (IV.4) en prenant en compte l'approximation du noyau mineur k :

$$f_{k,\mathcal{P}}(\mathbf{x},\mathbf{y}) \triangleq k(\mathbf{x},\mathbf{y}) f_{\mathcal{P}(\mathbf{x},\mathbf{y})} \tag{IV.13}$$

Le noyau *Ultra fast*, est alors défini par :

$$UFast_{K_j} = UFast_K(B_j, B_q) = \sum_{\mathbf{b}_{rq}\in B_q} \sum_{\mathbf{b}_{sj}\in B_j} f_{k,\mathcal{P}}(\mathbf{b}_{rq}, \mathbf{b}_{sj}) \tag{IV.14}$$

On peut remarquer qu'il s'agit d'une généralisation du formalisme proposé par Jegou *et al.* [JDS08a] qui met la fonction $f_{\mathcal{P}}$ dans un cadre plus général incluant les fonctions de similarité noyau sur sacs.

Dans ce cadre, on peut considérer la fonction $f_{k,RNN}$:

$$f_{k,RNN}(\mathbf{x},\mathbf{y}) = k(\mathbf{x},\mathbf{y}) f_{RNN}(\mathbf{x},\mathbf{y}) = k(\mathbf{x},\mathbf{y})\, \mathbb{1}_{D(\mathbf{x},\mathbf{y})\leq R} \tag{IV.15}$$

ou encore la fonction $f_{k,KNN}$:

$$f_{k,KNN}(\mathbf{x},\mathbf{y}) = k(\mathbf{x},\mathbf{y}) f_{KNN}(\mathbf{x},\mathbf{y}) = k(\mathbf{x},\mathbf{y})\, \mathbb{1}_{\mathbf{x}\in KNN(\mathbf{y})} \tag{IV.16}$$

Cette méthode peut être interprétée comme une stratégie par vote qui prend en compte la similarité entre chaque appariement au lieu de leur attribuer un poids unitaire. Comme pour la stratégie par vote, les appariements sont détectés par une recherche de type plus proches voisins. Au lieu de faire cette recherche de façon exhaustive, nous utilisons une structure d'index, LSH, permettant d'obtenir

une complexité de recherche sous-linéaire. Ainsi, le temps de calcul pour retrouver les plus proches voisins est bien plus faible que le temps de calcul du vrai noyau sur sacs.

6.2.2 Algorithme

La famille de noyaux que nous avons considérée est définie par :

$$UFast_K(B_j, B_q) = \sum_{\mathbf{b}_{rq} \in B_q} \sum_{\mathbf{b}_{sj} \in B_j} f_{k,LSH}(\mathbf{b}_{rq}, \mathbf{b}_{sj}) \tag{IV.17}$$

où $f_{k,LSH}(\mathbf{x},\mathbf{y}) = k(\mathbf{x},\mathbf{y}) f_{LSH}(\mathbf{x},\mathbf{y})$ avec $f_{LSH}(\mathbf{x},\mathbf{y})$ qui vaut 1 si $\mathbf{y}$ a été retrouvée comme ppv de $\mathbf{x}$ par l'implémentation de LSH considérée et 0 sinon.

L'algorithme 3 détaille les étapes de la recherche rapide.

Algorithm 2

1: La similarité K_j est initialisée à 0 pour toutes les images de la base.
2: **for all** descripteur de l'image requête $\mathbf{b}_{rq}$ **do**
3: une recherche ppv parmi tous les descripteurs locaux de la base $\mathcal{F}$ en utilisant une structure d'index permettant une recherche rapide
4: **for all** images I_j pour lesquelles un descripteur local $\mathbf{b}_{sj}$ a été trouvé **do**
5: augmenter la similarité :

$$K_j := K_j + k(\mathbf{b}_{sj}, \mathbf{b}_{rq}) \tag{IV.18}$$

6: **end for**
7: **end for**

Cet algorithme permet de calculer efficacement les similarités k_j en utilisant le fait que pour la majorité des descripteurs locaux de $\mathcal{F}$, les valeurs $f_{k,\mathcal{P}}$ sont nulles.

Dans le cas où l'on s'intéresse à l'implémentation de LSH permettant d'approximer la fonction f_{RNN} définie éq. (IV.7), f_{LSH} peut s'écrire :

$$f_{LSH}(\mathbf{x},\mathbf{y}) = f_{RNN}(\mathbf{x},\mathbf{y}) f_{BK}(\mathbf{x},\mathbf{y}) \tag{IV.19}$$

avec $f_{BK}(\mathbf{x},\mathbf{y})$ définie par :

$$f_{BK}(\mathbf{x},\mathbf{y}) = \amalg_{\mathbf{x} \in [key_i(\mathbf{y})]_{i=1\ldots L}} \tag{IV.20}$$

avec $key_i(\mathbf{y})$ la case de la table i contenant le vecteur $\mathbf{y}$.

Il peut être noté que $f_{BK}(\mathbf{x},\mathbf{y})$ est une fonction symétrique. En effet, si un vecteur $\mathbf{x}$ tombe dans la même case que le vecteur $\mathbf{y}$ pour cette table de hachage, le vecteur $\mathbf{y}$ appartient également à la même

case que le vecteur $\mathbf{x}$ pour l'une des tables de hachage. Il en résulte que, $f_{LSH}(\mathbf{x},\mathbf{y})$ est également une fonction symétrique.

Finalement, le noyau *Ultra Fast* défini dans l'éq. (IV.17) peut se réécrire comme :

$$\begin{aligned} UFast_K(B_j,B_q) &= \sum_{\mathbf{b}_{rq}\in B_q}\sum_{\mathbf{b}_{sj}\in B_j} f_{k,LSH}(\mathbf{b}_{rq},\mathbf{b}_{sj}) \\ &= \sum_{\mathbf{b}_{rq}\in B_q}\sum_{\mathbf{b}_{sj}\in B_j} k(\mathbf{b}_{rq},\mathbf{b}_{sj}) f_{LSH}(\mathbf{b}_{rq},\mathbf{b}_{sj}) \\ &= \sum_{\mathbf{b}_{rq}\in B_q}\sum_{\mathbf{b}_{sj}\in B_j} k(\mathbf{b}_{rq},\mathbf{b}_{sj}) f_{RNN}(\mathbf{b}_{rq},\mathbf{b}_{sj}) f_{BK}(\mathbf{b}_{rq},\mathbf{b}_{sj}) \\ &= \sum_{\mathbf{b}_{rq}\in B_q}\sum_{\mathbf{b}_{sj}\in B_j} k(\mathbf{b}_{rq},\mathbf{b}_{sj}) \mathbb{1}_{D(\mathbf{b}_{rq},\mathbf{b}_{sj})\leq R} \mathbb{1}_{\mathbf{b}_{rq}\in[key_i(\mathbf{b}_{sj})]_{i=1\ldots L}} \end{aligned} \tag{IV.21}$$

6.2.3 Propriétés de $UFast$

Une matrice symétrique $\mathbf{K}$ de $n\times n$ réels satisfaisant $\sum_{i,j} c_i c_j K_{ij} \geq 0, \forall c_i \in \mathbb{R}$ est dite *semi-définie positive*. Soit un noyau k et un ensemble de données $\mathbf{x}_1,\ldots,\mathbf{x}_n \in \mathcal{F}$, la matrice $\mathbf{K}$ de $n\times n$ réels définies par $\mathbf{K} := (k(\mathbf{x}_i,\mathbf{x}_j))_{ij}$ est appelée matrice de Gram (ou matrice noyau) de k sur $\mathbf{x}_1,\ldots,\mathbf{x}_n$.

Définition de la Semi-Définie Positivité (sdp) d'un noyau : soit $\mathcal{F}$ un ensemble non vide. Une fonction $k : \mathcal{F}\times\mathcal{F}\rightarrow\mathbb{R}$, qui $\forall n\in N, \mathbf{x}_i\in\mathcal{F}$ permet de construire une matrice de Gram semi-définie positive est appelée noyau semi-défini positif.

Montrons que le noyau $UFast$ est sdp. Nous utilisons les notations fournie par [BHS+07]. Quand la taille de la base devient très grande, l'étape de calcul du f_{RNN} dans le schéma LSH devient trop lente et est donc généralement supprimée. Dans ce cas, $f_{k,LSH}$ se réduit à : $f_{k,LSH}(\mathbf{x},\mathbf{y}) = k(\mathbf{x},\mathbf{y}).f_{BK}(\mathbf{x},\mathbf{y})$. Il en résulte :

$$UFast_K(B_j,B_q) = \sum_{\mathbf{b}_{rq}\in B_q}\sum_{\mathbf{b}_{sj}\in B_j} k(\mathbf{b}_{rq},\mathbf{b}_{sj}) f_{BK}(\mathbf{b}_{rq},\mathbf{b}_{sj}) \tag{IV.22}$$

Comme une somme de noyaux *spd* est toujours un noyau *spd*, pour prouver que $UFast_K(B_j,B_q)$ de l'éq. (IV.22) est un noyau *spd*, il suffit de prouver que $k = f_{k,LSH}$ est un noyau *spd*. Comme un produit de noyaux *spd* est toujours un noyau *spd*, il suffit de prouver que f_{BK} est un noyau *spd* tant que la fonction k considérée est une fonction noyau.

En considérant n-uplets $\mathbf{b}_{kj}\in\mathcal{F}$, nous pouvons les trier en fonction de l'indice de la case de la table de hachage dans laquelle le descripteur local tombe.

La matrice de Gram $\mathbf{K} := (f_{BK}(\mathbf{b}_{ri},\mathbf{b}_{sj}))_{ij}$ de $n\times n$ valeurs réelles est une matrice symétrique bloc diagonale où chaque bloc correspond aux descripteurs locaux tombant dans la même case de la table de hachage.

Comme tous les blocs sont des matrices carrées unitaires, la matrice de Gram est semi-définie positive (les valeurs propres sont données par la taille de chaque bloc). Il s'ensuit que f_{BK} est un noyau *spd* ce qui termine de prouver que $UFast$ (éq. (IV.22)) est un noyau sdp.

7 Conclusion

On a décrit 3 approches pour construire une fonction de similarité à partir de la représentation BoF : l'approche par vote, les dictionnaires et les fonctions noyaux (cf. Sec. 2). Le principal avantage de l'approche dictionnaire est que cette stratégie permet facilement de passer à l'échelle. En effet, l'utilisation d'un dictionnaire pour décrire les images permet de réduire de façon importante la quantité d'information à traiter. D'autre part, des structures d'index, telles que les fichiers inversés, permettent d'obtenir des systèmes de recherche très rapides adaptés aux grands volumes de données. Cependant, cette réduction d'information rend les systèmes de recherche moins précis que ceux utilisant le vote ou les noyaux.

L'approche par vote est « réservée » au contexte de la détection de copies car les fonctions de similarité obtenues sont trop discriminantes pour être utilisées en recherche sémantique.

Les fonctions noyaux ont le grand avantage de permettre une combinaison efficace avec les algorithmes d'apprentissage tout en offrant un compromis intéressant entre les approches dictionnaires et les approches par vote.

Nous avons introduit 2 approches basées noyaux dont l'une, appelée $UFast$, s'appuie directement sur une fonction de similarité entre sacs de vecteurs.

Chapitre V

Evaluation et discussion

Dans ce chapitre, nous présentons les expériences menées pour tester les méthodes K_{select} et $UFast_K$ présentées au chapitre précédent.

La première phase de tests a été réalisée sur VOC06 (cf. Annexe A.1) afin de valider et comparer nos deux méthodes. Nous proposons également une première comparaison avec les méthodes classiques de l'état de l'art. Dans la seconde phase de tests, la base INRIA Holidays (cf. Annexe A.1) et son protocole expérimental, nous ont permis de comparer la méthode $UFast_K$ avec les dernières méthodes de l'état de l'art dans un contexte proche de celui de la détection de copies. Une troisième phase de tests menée sur VOC06, nous a permis d'évaluer notre approche en utilisant différents descripteurs locaux. Nous terminons ce chapitre par une discussion des améliorations possibles de notre système ainsi que de l'intérêt de notre formalisme.

Sommaire

1 Validation sur VOC06 . . . . . **86**

2 Comparaison en utilisant la base INRIA Holidays **94**

3 Evaluation de différents détecteurs et descripteurs BoF sur VOC06 . . . . . **98**

4 Discussions **100**

1 Validation sur VOC06

1.1 Protocole expérimental

Nous avons réalisé l'évaluation des méthodes décrites précédemment en considérant les 10 classes de la base VOC06 (cf. Annexe A.1) contenant 5 304 images. Il est important de noter que les résultats ne sont pas comparables avec ceux fournis lors de la compétition PASCAL (cf. Annexe A.1) pour les deux raisons suivantes :

- nous n'évaluons pas tout le classement de la base mais uniquement les N images du top de la recherche (appelé TOPN),
- nous nous intéressons à un problème de recherche et non à une tâche de classification comme c'est typiquement le cas pour la compétition PASCAL. Contrairement aux tâches de classification, nous ne disposons pas d'ensemble d'apprentissage. Une recherche est alors réalisée en considérant uniquement une image requête.

La base a été indexée en utilisant l'approche combinant l'extraction et la description des points d'intérêt SIFT [Low04]. Chaque image est alors représentée par un sac d'environ 150 PoI. Nous obtenons une base de 750 000 SIFT. Chaque PoI est décrit par un vecteur SIFT de 128 dimensions issu de la concaténation de 16 histogrammes de 8 orientations des gradients d'une région de l'image.

Nous avons utilisé l'implémentation E^2LSH de la structure d'index LSH fournie par A. Andoni [AI05]. Pour tous nos tests, nous avons choisi comme paramètres d'E^2LSH : R compris entre 175 et 300, $L = 100$ et $M = 20$.

Pour évaluer la qualité de la recherche, nous avons comparé le MAP (*Mean Average Precision*) du TOPN (cf. Annexe A.2) du vrai noyau avec celui des méthodes de recherche approximée. L'efficacité de notre méthode est évaluée en mesurant le facteur de gain de temps de calcul, *i.e.* le rapport entre le temps de recherche du vrai noyau et le temps de recherche des méthodes de recherche approximée. Dans nos expérimentations, nous avons fixé $N = 100$.

1.2 $UFast_K$ vs K_{Select}

Nous présentons dans ce paragraphe les résultats de la comparaison des deux méthodes que nous avons proposées pour diminuer la complexité des noyaux sur sacs, K_{Select} et $UFast_K$ pour le noyau K_{power} introduit éq. (IV.11). Nous avons fixé les paramètres p et σ du noyau K_{power} respectivement à 5 et 100.

Les deux paramètres affectant les performances de K_{Select} sont :

- Le rayon de recherche R correspond au paramètre de l'étape de sélection par vote. Ce rayon permet de régler le compromis entre discrimination et généralisation des appariements. Nous

l'avons fixé à $R = 200$.

– La taille S de la sélection $\mathcal{S}$ qui permet de contrôler le degré d'approximation du noyau. Nous avons fait varier S entre 100 et 300.

La seconde approximation du noyau considérée est le noyau $UFast_K$ avec son implémentation *RNN* définie éq. (IV.15). L'approximation de ce noyau est alors donnée par :

$$UFast_{K_{power}}(B_j, B_q) = \sum_{\mathbf{b}_{sj} \in B_j} \sum_{\mathbf{b}_{rq} \in B_q} (f_{k,LSH}(\mathbf{b}_{sj}, \mathbf{b}_{rq}))^p \tag{V.1}$$

Le paramètre affectant les performances de $UFast_K$ est le rayon de recherche R que nous avons testé pour $R = 225$ et $R = 250$.

Noyau	K_{power}	K_{select}			$UFast_K$	
Paramètre		$\|S\| = 100$	$\|S\| = 200$	$\|S\| = 300$	$R = 225$	$R = 250$
MAP du TOP100	9,45	8,84	9,09	9,14	9,43	9,45
Temps (sec)	59,38	0,96	1,71	2,38	0,34	0,63
Accélération	1	61,67	34,72	24,96	175,68	94,99

TAB. V.1 – Résultat de recherche (MAP, Temps) avec K_{power} pour $p = 5$ et les deux schémas d'approximation $UFast_K$ et K_{Select} (VOC06)

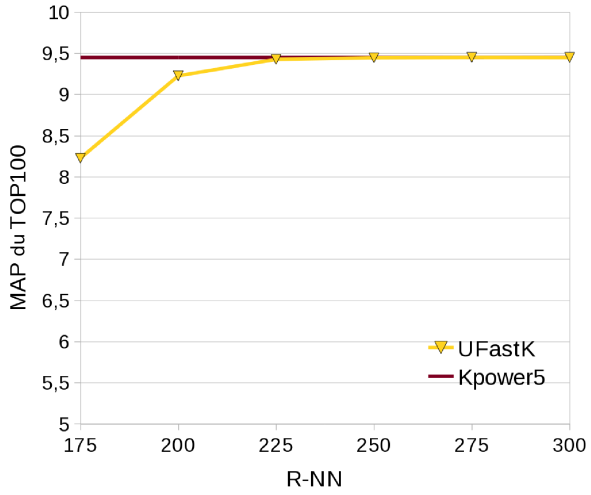
Le Tableau V.1 compare K_{Select} et $UFast_K$ avec le vrai noyau K_{power}. Contrairement à $UFast_K$, K_{Select} ne permet pas d'atteindre la même qualité de recherche que le vrai noyau K_{power}. En effet, pour $|S| = 300$, le MAP de K_{Select} est $3,4\%$ inférieur à celui de $UFast_K$. Ceci montre qu'une image ayant beaucoup de points appariés n'a pas forcément un meilleur score qu'une image qui en a peu. Tout dépend de la qualité des appariements. Pour cette base, le nombre d'appariements entre deux images similaires est très faible (de l'ordre de la dizaine). De ce fait, il est plus rapide de calculer une similarité approximée sur une grande partie de la base que de calculer la vraie similarité K_{power} sur un petit ensemble d'images. C'est pourquoi $UFast_K$ est bien plus rapide que K_{Select}. Si le nombre d'appariements pertinents augmentait, K_{Select} pourrait devenir plus avantageux.

1.3 Evaluation de $UFast_K$ pour K_{power}

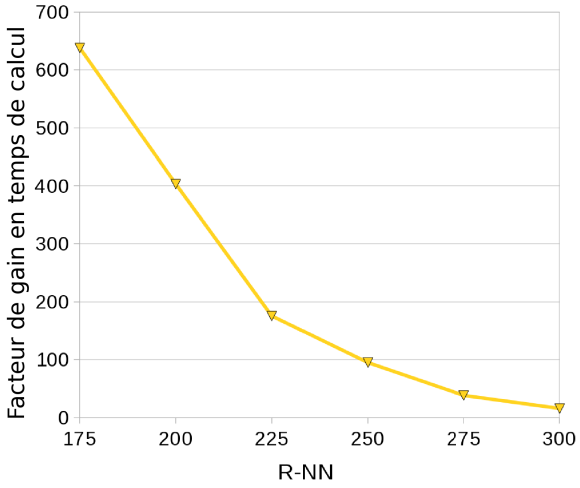
Comme nous l'avons vu dans le paragraphe précédent, la méthode $UFast_K$ fournit, pour la base VOC06, les meilleures performances aussi bien en précision qu'en rapidité. C'est pourquoi nous avons consacré ce paragraphe à une évaluation approfondie de cette méthode toujours pour K_{power} avec $p = 5$ et $\sigma = 100$.

Le degré d'approximation du noyau est contrôlé par le rayon de recherche R. L'évolution de la précision et de l'efficacité de l'approximation est montrée sur la Figure V.1 pour des rayons de

recherche compris entre 175 et 300.



(a) MAP du TOP100 pour les 10 classes de VOC06



(b) Facteur de gain du temps de recherche par rapport au vrai noyau

Figure V.1 – Efficacité et qualité de la recherche de $UFast_K$ pour K_{power5}

Noyau	$UFast_K$		K_{power}
Rayon de recherche	$R = 225$	$R = 250$	$--$
Temps (sec)	0,34	0,63	59,38

TAB. V.2 – Temps moyen de recherche de K_{power} vs $UFast_K$ (VOC06)

On vérifie bien sur la Figure V.1 que la précision de $UFast_K$ s'améliore lorsque l'on augmente le rayon de recherche jusqu'à atteindre la précision du vrai noyau K_{power}. On observe que la précision augmente rapidement entre $R = 175$ et $R = 225$ et que $UFast_K$ atteint la même qualité de la recherche que K_{power} pour $R = 250$. On peut en conclure que les appariements entre vecteurs de SIFT dont la distance est supérieure à 250 ont une similarité trop faible pour influencer le classement parmi les images du TOP100 de la recherche. Comme le montre la Figure V.1, élaguer les appariements pour $R > 250$ permet de réduire les temps de calcul d'un facteur 95. $R = 225$ est un point de fonctionnement très intéressant car il permet d'atteindre une réduction du temps de calcul important, un facteur 175, tout en gardant une qualité de la recherche très proche de celle du vrai noyau. La perte n'est alors que de $0,25\%$ de MAP.

Comme on peut le voir sur Tableau V.2 cette approximation permet de passer d'un temps de recherche de l'ordre de la minute, bien souvent trop long pour un utilisateur, à un temps de recherche inférieur à la seconde et ceci pour une qualité de la recherche similaire.

1.4 Evaluation de $UFast_K$ pour K_{match}

Pour prouver que notre stratégie s'étend bien à d'autres noyaux, nous avons testé la méthode $UFast_K$ avec une seconde famille de noyaux que nous avons appelée K_{match}. Cette famille a été proposée par Eichhorn et Chapelle dans [EC04].

K_{match} est définie de la façon suivante :

$$K_{match}(B_j, B_q) = \frac{1}{2}\left(K_{avgM}(B_j, B_q) + K_{avgM}(B_q, B_j)\right) \tag{V.2}$$

où K_{avgM} est défini par :

$$K_{avgM}(B_j, B_q) = \frac{1}{|B_j|}\sum_{s=0}^{|B_j|} \underset{r=0,\ldots,|B_q|}{max}\left(k(\mathbf{b}_{sj}, \mathbf{b}_{rq})\right) \tag{V.3}$$

En résumé, ce noyau mesure la similarité moyenne des meilleurs appariements possibles entre deux images.

L'approximation $UFast_K$ de ce noyau s'écrit alors :

$$UFast_{K_{match}}(B_j, B_q) = \frac{1}{2}\left(\hat{K}_{avgM}(B_j, B_q) + \hat{K}_{avgM}(B_q, B_j)\right) \tag{V.4}$$

où $\hat{K}_{avgM}$ est défini par :

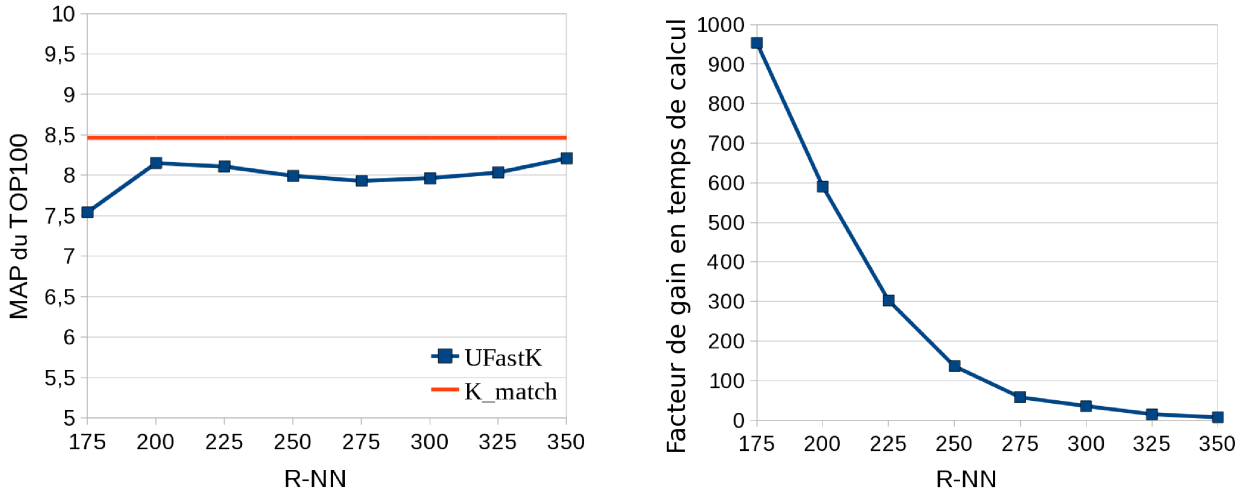
$$\hat{K}_{avgM}(B_j, B_q) = \frac{1}{|B_j|}\sum_{s=0}^{|B_j|} \underset{r=0,\ldots,|B_q|}{max}\left(f_{k,LSH}(\mathbf{b}_{sj}, \mathbf{b}_{rq})\right) \tag{V.5}$$

Les résultats de l'évaluation de la précision et de l'efficacité de notre approximation sont présentés en Figure V.2 pour des rayons de recherche variant entre 175 et 350.

Noyau	$UFast_K$		K_{match}
Rayon de recherche	$R = 200$	$R = 350$	
Temps (sec)	0,15	12,20	86,38

TAB. V.3 – Temps moyen de recherche de K_{match} vs $UFast_K$ (VOC06)

On peut observer sur la Figure V.2 que la précision de $UFast_K$ augmente rapidement pour R entre 175 et 200 jusqu'à atteindre une qualité de la recherche proche de celle de K_{match}. Pour $R = 200$, la différence de la qualité de la recherche avec le vrai noyau n'est que de $3,8\%$. Comme le montre la Figure V.2, ce point de fonctionnement permet d'accélérer les temps de calcul d'un facteur 590. En autorisant une faible détérioration ($3,8\%$ du MAP), on peut ainsi passer d'un temps de recherche



(a) MAP du TOP100 pour les 10 classes de VOC06

(b) Facteur de gain du temps de recherche par rapport au vrai noyau

Figure V.2 – Efficacité et qualité de la recherche de $UFast_K$ pour K_{match}

proche d'une minute et demie à un temps de recherche de 150 millisecondes.

Contrairement à l'approximation de K_{power}, augmenter le rayon de recherche et donc diminuer le degré d'approximation du noyau ne se résume pas directement à augmenter la qualité de la recherche. En effet, pour un rayon entre 200 et 275 la qualité de la recherche diminue légèrement puis elle réaugmente pour un rayon supérieur. Nos investigations pour comprendre ce phénomène ont permis de réaliser que le comportement de l'approximation du noyau est différent pour les classes *vache* et *mouton* de celui pour les 8 autres classes. Nous avons alors évalué la qualité de la recherche séparément pour ces 2 classes (cf. Tab. V.5) et pour les 8 autres (cf. Tab. V.4).

Noyau	$UFast_K$				K_{match}
Rayon de recherche	$R = 200$	$R = 225$	$R = 250$	$R = 275$	
MAP du TOP100	7,52	7,30	7,04	6,84	6,82

TAB. V.4 – Qualité de recherche de $UFast_K$ pour les 8 classes de VOC06

On remarque sur le Tableau V.4 que la qualité de recherche du noyau approximé, pour les 8 classes considérées, est meilleure que celle du vrai noyau K_{match}. Ainsi, pour $R = 200$, le MAP de $UFast_{K_{match}}$ est $10,3\%$ meilleur que celui de K_{match}. On observe également que l'augmentation du rayon de recherche dégrade la qualité de recherche pour rejoindre celle du vrai noyau. Nous expliquons cette observation en remarquant que l'objet d'intérêt ne recouvre pas toute l'image, un certain nombre de SIFT caractérise donc des zones non pertinentes de l'image comme le fond. D'autre part, la forme des objets d'intérêt pouvant être très variable, même les SIFT décrivant des zones de l'objet ne sont

pas tous caractéristiques de la classe d'objets recherchée. Il en résulte que seul un nombre limité de SIFT sont pertinents. Or, le noyau K_{match} s'efforce de rechercher un appariement pour tous les SIFT de l'image même si ces derniers sont mauvais. Ce noyau est donc fortement affecté par des appariements non pertinents. Notre interprétation est qu'un rayon de recherche de 200 permet de sélectionner les appariements pertinents et de filtrer les appariements non pertinents. Comme les fonds des images sont de nature plus variable que les objets d'intérêt, ils nécessitent un rayon de recherche plus grand. C'est pourquoi diminuer le rayon de recherche permet de se concentrer sur les objets d'intérêt. Les SIFT caractérisant le fond de l'image sont alors assimilables à du bruit. Notre approximation permet donc de réduire ce bruit et ainsi d'améliorer la recherche tout en diminuant drastiquement les temps de calcul.

Noyau	$UFast_K$			K_{match}
Rayon de recherche	$R = 200$	$R = 275$	$R = 350$	$--$
MAP du TOP100	10,69	12,30	12,90	15,05

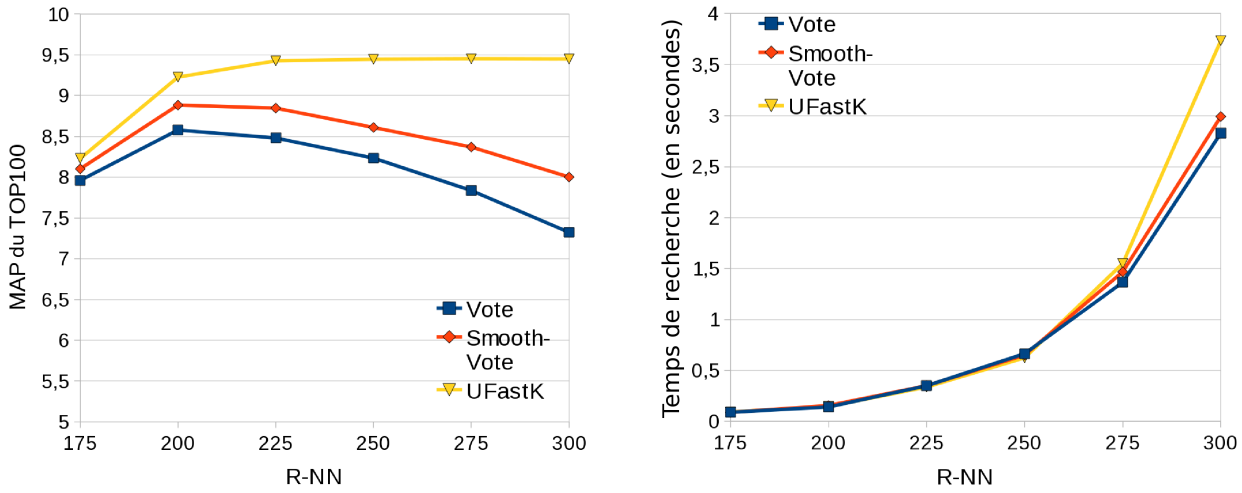
TAB. V.5 – Qualité de recherche de $UFast_K$ pour les 2 classes mouton et vache de VOC06

Pour les catégories *vache* et *mouton*, le constat est différent. En effet, plus on augmente le rayon de recherche, plus la qualité de recherche s'améliore. Notre interprétation est que pour ces deux catégories, le contexte de l'objet est très informatif. En effet, les *vaches* et les *moutons* sont, dans cette base, en majorité dans une prairie. Il en résulte qu'identifier le fond de l'image peut aider à identifier la catégorie d'objets recherchée. Pour ces deux catégories, l'ensemble des appariements a alors son importance. C'est pourquoi, limiter le nombre d'appariements ne permet pas d'atteindre la qualité de recherche du vrai noyau K_{match}. Il s'ensuit que plus on augmente le rayon de recherche plus on améliore la qualité de recherche (cf. Tab. V.5).

1.5 $UFast_K$ vs Vote

Dans ce paragraphe, nous comparons notre méthode $UFast_K$ éq. (V.1), utilisée pour approximer le noyau K_{power5}, avec deux versions de l'approche par vote. La première version consiste en la version classique éq. (IV.7) de l'approche par vote avec une assignation dure. La seconde version consiste en une version lissée où chaque appariement est pondéré par un noyau gaussien. Cette seconde version de l'approche par vote peut également être interprétée par notre approximation du noyau K_{power} avec une puissance $p = 1$. La qualité et l'efficacité de la recherche de ces méthodes sont présentées en Figure V.3.

La Figure V.3 met en évidence que la méthode $UFast_K$ aboutit à une amélioration significative comparée aux deux stratégies par vote. Contrairement aux approches par vote, la qualité de recherche



(a) MAP du TOP100 pour les 10 classes

(b) Temps moyen de recherche

Figure V.3 – Comparaison de l'efficacité et de la qualité de recherche entre l'approche par vote et $UFast_K$ pour K_{power5} (VOC06)

de $UFast_K$ ne diminue par lorsque l'on augmente le rayon de recherche. En effet, lorsque l'on augmente le rayon de recherche des approches par vote, on augmente le nombre d'appariements non pertinents. Pour l'approche par vote classique, vu que tous les appariements ont le même poids, la qualité de recherche est grandement dégradée. Un appariement pertinent est pris en compte de la même façon qu'un appariement non pertinent. L'approche par vote lissé prenant en compte la qualité des appariements permet de limiter la détérioration mais dans une mesure moindre que l'approche $UFast_K$. En effet, $UFast_K$ augmentant fortement le fossé entre bons et mauvais appariements est capable de s'affranchir des mauvais appariements alors que l'approche par vote lissé ne permet pas de filtrer le bruit de façon suffisante.

La Figure V.3 compare les temps de calcul des 3 approches. On peut observer que pour $R \leq 250$, les approches par vote et l'approximation du noyau ont des temps de recherche comparables. Pour ces trois méthodes, plus de 90% des temps de recherche sont consacrés à l'étape de recherche RNN et les 10% restant sont dévoués à l'étape de tri. Le temps d'évaluation des noyaux mineurs k est alors négligeable. A partir d'un rayon de recherche de 300, le temps d'évaluation de k n'est plus négligeable. L'approche par vote lissé et $UFast_K$ deviennent respectivement 6% et 32% plus lents que l'approche par vote classique. Rappelons que le point de fonction optimal de l'approximation du noyau se trouve pour R entre 200 et 250, nous pouvons donc considérer que notre méthode à une efficacité comparable aux approches par vote.

1.6 Noyau sur sacs (BoF) vs Dictionnaire (BoW)

Dans ce paragraphe, nous mettons en évidence l'intérêt de l'approche noyau en comparant le noyau K_{power5} avec une implémentation classique de l'approche dictionnaire. Pour l'approche dictionnaire, toute image de la base ainsi que l'image requête est représentée par un seul vecteur. Ce vecteur est un histogramme d'occurrences des mots visuels du dictionnaire dans l'image. Les mots visuels sont obtenus par un clustering de type K-means des SIFT de la base. Nous avons utilisé les mêmes descripteurs locaux que ceux utilisés pour l'approche noyau. Nous avons testé les performances de la méthode pour différentes tailles de dictionnaire. Chaque valeur de l'histogramme est ensuite pondérée par le schéma classique tf-idf [PCI+07], qui diminue le poids des mots souvent présents dans la base et donc moins discriminants. Communément, la distance entre deux vecteurs ainsi construits est mesurée avec une distance l_2. La recherche par similarité revient donc à effectuer une recherche des plus proches voisins au sens de la distance l_2 entre l'image requête et les images de la base. Ne cherchant pas à comparer les temps de recherche entre les deux méthodes, nous n'avons pas utilisé de structure d'index, nous nous sommes donc contentés d'une recherche linéaire. On rappelle que les vecteurs de données étant creux (la majorité des composantes sont de valeurs nulles), la structure d'index souvent privilégiée dans ce contexte est *les fichiers inversés*. Les résultats sont rassemblés dans le Tableau V.6 pour des dictionnaires entre 100 et 1 000 mots.

Méthode	BoW					K_{power5}
Taille du dictionnaire	100	200	300	500	1000	−−
MAP du TOP100	6,50	6,53	6,10	5,73	5,3	9,45

TAB. V.6 – Comparaison de la qualité de recherche entre l'approche BoW et l'approche BoF avec K_{power5} pour les 10 classes de VOC06

Nous pouvons observer sur le Tableau V.6 que l'approche dictionnaire ne permet pas d'atteindre la même qualité de recherche que l'approche noyau. En effet, K_{power5} avec un MAP de 9,45 obtient une précision au moins 45% meilleure que l'approche dictionnaire qui obtient au mieux un MAP de 6,53. La taille optimale du dictionnaire, 200 « mots », peut paraître faible mais elle est à rapprocher du nombre de descripteurs locaux extraits par image (150 PoI/image). De plus, la taille du dictionnaire permet de contrôler le compromis entre généricité et discrimination. Même s'il est communément admis que pour les tâches de détection de copies où la discrimination est très importante, la taille du dictionnaire doit être importante, ce n'est pas si évident pour les tâches de recherche par similarité sémantique.

2 Comparaison en utilisant la base INRIA Holidays

2.1 Protocol expérimental

La base d'images INRIA Holidays (cf. Annexe A.1) est une collection de 1 491 images de vacances dont 500 d'entre elles sont utilisées comme images requêtes. Cette base offre un cadre expérimental pour la recherche par similarité facilitant la comparaison entre notre approche et les méthodes de l'état de l'art. Nous avons en effet utilisé le protocole expérimental de [JDS08b] où la qualité de recherche est mesurée par le MAP. Nous avons également utilisé les descripteurs locaux fournis par [JDS08b]. Ils ont été extraits en utilisant le détecteur Hessian affine [MTS+05] et le descripteur SIFT [Low04]. La base de caractéristiques extraites comprend 4,5 millions de points d'intérêt. Chaque image est ainsi décrite par 3 000 PoI en moyenne.

Pour l'ensemble des tests, nous avons utilisé la structure d'index E^2-*MPLSH* [LJW+07] une extension de E^2-*LSH* qui permet de réduire considérablement l'occupation mémoire. Au lieu de stocker 100 tables de hachage en mémoire vive, seules 4 sont suffisantes.

Nous avons considéré le même noyau que celui utilisé pour VOC06, c'est-à-dire, K_{power} éq. (IV.11) avec $p = 5$. Comme cette base se rapproche plus d'une recherche de détection de copies, nous avons choisi l'implémentation éq. (IV.16) de notre approximation du noyau.

$UFast_{K_{power5}}$ est alors défini par :

$$UFast_{K_{power}}(B_j, B_q) = \sum_{\mathbf{b}_{rq} \in B_q} \sum_{\mathbf{b}_{sj} \in B_j} k(\mathbf{b}_{rq}, \mathbf{b}_{sj})^p f_{KNN}(\mathbf{b}_{rq}, \mathbf{b}_{sj}) f_{BK}(\mathbf{b}_{rq}, \mathbf{b}_{sj})$$

Comme, en moyenne dans cette base, seules deux images cibles sont recherchées dans toutes la base, nous avons paramétré *LSH* avec une recherche $2NN$. Cette implémentation de $UFast_K$ est alors beaucoup plus discriminante que l'implémentation utilisée pour VOC06.

2.2 Evaluation de la qualité de recherche de $UFast_K$

Dans ce paragraphe, nous comparons la qualité de recherche de $UFast_K$ avec les méthodes de l'état de l'art.

Le Tableau V.7 présente les résultats des méthodes de l'état de l'art publiées dans [JDS09, JDSP10, JDS08b] en utilisant le même protocol expérimental et les mêmes données (base et descripteurs locaux).

Les deux premières approches, BoW et VLAD sont respectivement présentées dans les Sections 2 et 3. L'approche HE est une approche par vote qui utilise un K-means pour approximer la recherche RNN puis un vote. Chaque descripteur local est encodé d'une manière spécifique pour décrire plus

Méthode	paramètres	MAP
BoW [JDS08a]	$1K$ mots	41,4
	$20K$ mots	44,6
	$200K$ mots	54,9
VLAD [JDS10]	16 mots, D=2048	49,6
	64 mots, D=8192	52,6
HE [JDS09]	$20K$ mots	72,7
$UFast_{K_{power5}}$	$R = 175$ $M = 24$ $L = 4$ $T = 50$	76,0

TAB. V.7 – Comparaison de la qualité de recherche entre $UFast_K$ et les méthodes de l'état de l'art (Holidays)

finement sa localisation dans la cellule du K-means lui correspondant. Ceci permet de supprimer les appariements entre mauvais descripteurs tombant tout de même dans les mêmes cellules.

Ces résultats montrent bien que plus l'on garde d'information, meilleure est la qualité de la recherche. En effet, notre méthode est plus précise que l'approche HE qui comprime les descripteurs locaux. De plus, ces deux méthodes sont bien plus précises que les approches BoW qui quantifient les descripteurs locaux et qui sont alors incapables de différencier deux descripteurs se trouvant dans une même cellule K-means.

Il peut être noté, comme montré dans [JDS08b], qu'un post-traitement tel que la vérification de la cohérence géométrique peut être utilisé pour améliorer les résultats sur cette base. Cependant, ce post-traitement n'a de sens que pour les tâches de détection de copies. Or, dans cette thèse, nous nous intéressons à la recherche par similarité dans un cadre plus général et nous ne disposons pas d'a priori stipulant que les objets recherchés sont identiques. C'est pourquoi nous n'avons pas considéré de tels procédés ici.

2.3 Compromis qualité / temps de recherche de $UFast_K$

Nous évaluons maintenant le compromis entre la qualité et le temps de recherche de $UFast_K$ en fonction des paramètres de la recherche *KNN* ainsi que du nombre de descripteurs par image.

La Figure V.4 présente la qualité de recherche mesurée par le critère du MAP en fonction du temps moyen de recherche. Nous avons fait évoluer le compromis entre qualité et temps de recherche en jouant sur 2 paramètres de E^2-*MPLSH* : le nombre de tables de hachage (L) et le nombre de *probe* (T), *i.e.* le nombre de cellules (*Buckets*) voisines visitées. Comme on peut l'observer sur la Figure V.4, augmenter L et T permet d'augmenter la qualité de recherche mais au détriment du temps de recherche. Ainsi, choisir $L = 4$ et $T = 50$ permet d'atteindre un MAP important (76) mais demande un temps de recherche de $7,46$ secondes. On observe également que passer de $T = 20$ à $T = 50$ double le temps de calcul pour une augmentation assez faible de la qualité de recherche. C'est pourquoi fixer

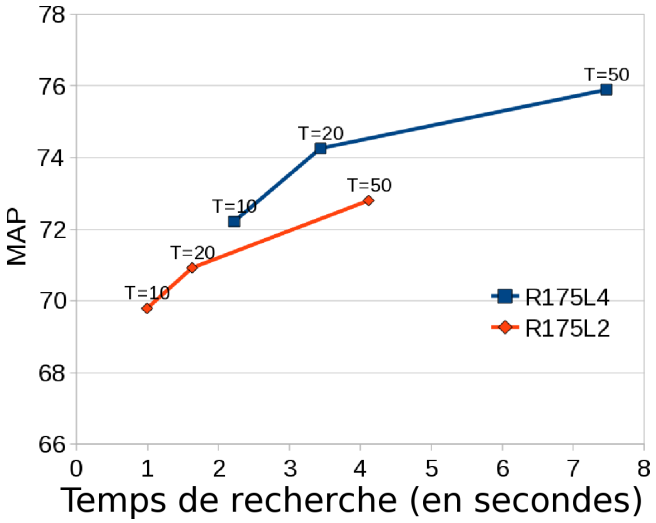


Figure V.4 – MAP du TOP100 en fonction des temps de recherche (Holidays) : courbe rouge pour $L = 2$, bleue pour $L = 4$.

$L = 4$ et $T = 20$ fournit un meilleur compromis puisqu'il permet d'atteindre 74 de MAP pour un temps de recherche de $3,44$ secondes.

On peut remarquer que la base Holidays bien que contenant 5 fois moins d'images que VOC06 demande des temps de recherche plus importants. En effet, notre méthode requiert une recherche *ppv* pour chaque descripteur de l'image requête. La complexité de la recherche est donc linéaire en fonction du nombre de descripteurs utilisés pour décrire l'image requête.

Augmenter l'approximation des recherches *ppv* n'est pas la seule solution pour diminuer les temps de recherche. On pourrait également envisager de réduire le nombre de descripteurs de l'image requête. D'autant plus que tous les descripteurs de l'image ne sont pas pertinents, ou du moins, n'ont pas la même importance. Cependant, il est assez difficile de mesurer a priori l'importance des descripteurs.

Foo *et al.* montrent dans [FS07] qu'il est possible de fortement diminuer le nombre de descripteurs utilisés tout en conservant une bonne qualité de recherche. Leur méthode de sélection des SIFT est basée sur un critère de contraste (amplitude de la différence de gaussienne ou *DoG*). Dans l'algorithme initial des SIFT, un seuil sur ce contraste permet de décider si le point d'intérêt détecté est pertinent ou non. Dans leur méthode, les auteurs trient les régions d'intérêt selon ce contraste et ne conservent que les k régions de plus forts contrastes. C'est cette méthode facile à mettre en œuvre que nous avons considérée. Cependant, comme expliqué dans [LÁJA06], cette méthode a l'inconvénient de supprimer les régions correspondant aux basses fréquences qui ont un contraste naturellement plus faible. D'autres solutions ont été proposées comme de supprimer les descripteurs appartenant aux régions les plus denses de l'espace des caractéristiques [SZ03]. Dans [YJHN07], Yang *et al.* ont également

utilisé des tests statistiques (test du χ^2 et l'information mutuelle) pour supprimer les descripteurs les moins informatifs.

k	∞	2000	1000	500	200
Nb PoI	4,4M	2,3M	1,3M	700K	290K
MAP	75,9	73,8	69,2	63,3	52,9
Temps (sec)	7,46	2,66	0,98	0,36	0,08

TAB. V.8 – Evolution de la qualité et du temps de recherche en fonction du nombre de SIFT par image (Holidays)

Le Tableau V.8 présente les résultats obtenus en diminuant le nombre maximal k de SIFT conservés par image. Ces résultats ont été obtenus en utilisant comme paramètre du E^2-*MPLSH* : $R = 175$ $M = 24$ $L = 4$ et $T = 50$. Limiter à 2 000 le nombre de SIFT par image permet ainsi de passer d'un temps de recherche de $7,46$ secondes à $2,66$ secondes avec une faible perte de qualité (2% de MAP). Nous obtenons donc un meilleur compromis que celui obtenu en augmentant l'approximation de la recherche (avec $T = 20$). En ce limitant à 1 000 SIFT, le compromis est équivalent à l'approximation de E^2-*MPLSH* obtenue pour $L = 2$ et $T = 10$, puisque dans les deux cas, on passe à un temps de recherche proche de la seconde avec une qualité de recherche légèrement inférieure à 70 de MAP.

Ces résultats montrent que réduire le nombre de descripteurs est une piste intéressante pour réduire les temps de recherche.

Notre travail peut également tirer profit des optimisations proposées par Auclair *et al.* dans [ACV07] qui améliorent les temps de recherche de *LSH* lorsque la base considérée est une base de vecteurs SIFT. En effet, les auteurs introduisent une modification de l'algorithme *LSH* prenant en compte la distribution particulière des SIFT. Cette modification permet de diminuer d'un facteur 2 le temps de recherche par rapport au schéma LSH classique.

Comme dans [JDSP10], nous avons également testé notre système sur des bases de 10K et 100K images, en augmentant la taille de la base Holidays par l'ajout de distracteurs issus de la base Flickr. Pour la base de 10K images, contenant 20 millions de SIFT, nous avons obtenu un MAP de 56 pour un temps de recherche moyen de $6,08$ secondes. Ces résultats ont été obtenus en utilisant comme paramètre de E^2-*MPLSH* : $R = 150$ $M = 24$ $L = 2$ et $T = 10$. Pour la base de 100K images, contenant 208 millions de SIFT, nous avons obtenu un MAP de 42 pour un temps de recherche moyen de 44 secondes. Le moteur de recherche utilise alors 31Go de RAM. Ces résultats ont été obtenus en utilisant comme paramètre de E^2-*MPLSH* : $R = 100$ $M = 26$ $L = 2$ et $T = 10$. Ce temp de recherche peut paraître important mais il pourrait être fortement diminuer en réduisant le nombre de PoI des images requêtes. En effet, les images requêtes contiennent en moyenne 3 950 SIFT. Le temps de recherche ramené à la recherche KNN d'un PoI ne prend alors plus que 16 ms. En limitant le nombre de PoI des images requêtes à 200 SIFT, on pourrait alors atteindre des temps de recherche de l'odre

de $3,2$ secondes.

3 Evaluation de différents détecteurs et descripteurs BoF sur VOC06

Comme nous l'avons vu en introduction, un grand nombre de descripteurs BoF ont été proposés dans la littérature. Dans ce paragraphe, nous proposons de comparer 3 détecteurs : SIFT, MSER et les grilles denses (GRILL) couplées avec le descripteur SIFT. Nous avons également testé le descripteur SIFT couleur sur une grille dense (CGRILL). Pour les détecteurs SIFT et MSER, nous avons fait varier le seuil du contraste des DoG pour modifier le nombre de PoI extraits par image. Le Tableau V.9 détaille les différents descripteurs évalués.

Nom	SIFT100	SIFT500	MSER100	MSER300	GRILL	CGRILL
Détecteur	SIFT		MSER		Grille dense	
Descripteur	SIFT					SIFT Couleur RGB
Dimension	128					384
Nb PoI	713 333	2 826 691	359 250	1 582 222	1 363 618	1 363 618
Nb PoI / im	135	533	68	300	257	257

TAB. V.9 – Différents types de descripteurs évalués (VOC06)

Le Tableau V.10 présente l'évaluation des qualités de recherche obtenues avec les 6 descripteurs testés. La qualité de recherche est mesurée par le critère du MAP du TOP100 pour chaque catégorie de la base VOC06. Ces résultats ont été obtenus avec l'approximation $UFast_K$ du noyau K_{power} éq. (IV.11) avec $p = 1$. Nous avons choisi l'implémentation *KNN* (éq. (IV.16)) de notre approximation du noyau avec une recherche $200-NN$. Nous avons utilisé la structure d'index E^2-*MPLSH* avec comme paramètres : $R = 200$, $M = 24$, $L = 4$ et $T = 50$ pour les descripteurs SIFT et $R = 600$, $M = 32$, $L = 4$ et $T = 50$ pour le descripteur SIFT couleur RBG fixés de façon empirique.

Comme on peut l'observer sur le Tableau V.10, la qualité du descripteur dépend énormément de la catégorie considérée. Ainsi, pour la classe *vélo*, ce sont les descripteurs MSER300 et SIFT500 qui donnent les meilleurs résultats. Par contre, pour la classe *voiture*, les meilleurs résultats ont été obtenus avec les grilles denses. Pour les classes *bus* et *personne*, le descripteur MSER100 donne de bien meilleurs résultats que MSER300 et SIFT500. Chaque descripteur va identifier des régions différentes et ne contient pas la même information. C'est pour cette raison qu'un grand nombre de descripteurs sont combinés dans les tâches de catégorisation.

On peut tout de même remarquer qu'en règle générale, plus on détecte de PoI, meilleure est la recherche. Ainsi, SIFT500 donne en moyenne les meilleurs résultats. Pour certaines catégories, comme *bus*, *voiture* et *mouton*, la couleur aide beaucoup à la reconnaissance et permet au descripteur CGRILL

Catégories	SIFT100	SIFT500	MSER100	MSER300	GRILL	CGRILL
Vélo	11,69	39,25	15,85	**39,37**	10,40	16,28
Bus	9,62	7,72	13,46	8,22	10,20	**14.90**
Voiture	14,23	37,51	22,81	31,94	39,28	**41.32**
Chat	6,55	3,72	6,05	4,77	8,40	**8,65**
Vache	11,53	**17,68**	6,26	9,32	11,86	15,03
Chien	6,59	7,11	**9,49**	8,17	7,38	8,72
Cheval	8,57	**10,56**	9,16	9,64	9,41	10,02
Moto	7,34	**11,70**	8,13	9,81	10,12	10,64
Personne	8,87	6,75	**11,85**	6,01	9,39	11,29
Mouton	11,21	17,99	5,91	13,58	13,01	**23,52**
Moyenne	9,62	**16,00**	10,90	14,08	12,95	**16,04**

TAB. V.10 – MAP du TOP100 de $UFast_K$ pour différents descripteurs BoF (VOC06)

de fournir d'aussi bonnes performances moyennes que SIFT500 avec un nombre de descripteurs deux fois moins important. Cependant, cette information a un coût de stockage important, vu que les descripteurs sont 3 fois plus volumineux (384 dimensions).

Le Tableau V.11 présente les temps de recherche moyens pour chaque descripteur.

	SIFT100	SIFT500	MSER100	MSER300	GRILL	CGRILL
Temps (sec)	0,23	1,06	0,12	0,95	0,31	1,84

TAB. V.11 – Temps de recherche de $UFast_K$ pour différents descripteurs BoF (VOC06)

Comme on peut le vérifier sur le Tableau V.11, plus on augmente le nombre de descripteurs par image, plus le temps de recherche est important. D'autre part, la taille du descripteur influe également sur les temps de recherche. Ainsi, CGRILL demande un temps de recherche 6 fois plus long que la grille de SIFT classique ($1,84$ secondes pour CGRILL contre $0,31$ secondes pour GRILL) (on pourrait sûrement améliorer ce temps de recherche en jouant sur les paramètres E^2-*MPLSH*).

De ces résultats, on peut conclure que MSER est mieux adapté que SIFT pour décrire les images lorsque l'on a une contrainte sur la quantité de PoI à extraire. En effet, avec seulement 68 PoI par image, MSER100 obtient un meilleur MAP avec un temps de recherche presque 2 fois plus rapide que SIFT100. Par contre, lorsque l'on augmente le nombre de descripteurs, SIFT devient plus intéressant. Les grilles denses de SIFT ne sont pas à négliger car elles offrent un très bon compromis temps/qualité de recherche. Elles sont 3 fois plus rapides que MSER300 ou SIFT500 pour une perte minime de 3% de MAP.

4 Discussions

4.1 Ressources mémoires

Les noyaux sur sac permettent d'atteindre une meilleure qualité de recherche que les approches classiques de types BoW pour les tâches de recherche par similarité. Cependant, la représentation BoW a l'avantage d'être plus rapide et de nécessiter moins de ressources mémoires (pour des dictionnaires de taille raisonnable). En effet, contrairement à la représentation BoF qui nécessite de stocker en mémoire primaire l'ensemble des descripteurs locaux pour chaque image, la représentation BoW nécessite juste de stocker une entrée pour chaque mot visuel du dictionnaire. La concision de cette représentation est rendue possible grâce à une quantification des descripteurs locaux qui entraîne une perte d'information et donc une diminution de la qualité de la recherche. Plus on garde d'information, plus on est précis mais moins on est efficace (aussi bien en terme de rapidité que d'occupation mémoire). Le choix de la méthode dépend donc du compromis entre précision et efficacité.

Quand une précision importante est demandée, l'approche noyau sera alors préférée. Dans ce cas, l'occupation mémoire pourra tout de même être optimisée sans trop de perte de précision. Plusieurs travaux ont déjà proposé des solutions qui permettent de réduire les ressources mémoires. Dans [FS07], les auteurs ont réduit le nombre de descripteurs locaux par image et donc l'occupation mémoire d'un facteur 10 pour une qualité de recherche presque similaire. Dans [KS04], les auteurs utilisent une ACP qui permet de réduire la dimension des SIFT de 128 à 20 entraînant une réduction mémoire d'un facteur 6.

Prenons le descripteur SIFT de 128 dimensions encodées en char. Stocker tous les descripteurs locaux de la base en mémoire vive nécessite : $|B| \times n \times d$ octets de RAM, avec $|B|$, le nombre de descripteurs locaux par image, n, la taille de la base et d, la dimension d'un descripteur local.

Pour une machine disposant de $32Go$ de RAM, avec 200 descripteurs locaux par image (en utilisant [FS07]), le système est capable de travailler avec les bases contenant jusqu'à $1,3M$ d'images.

Pour de très grandes bases (plus de $10M$ d'images), le nombre de descripteurs locaux à stocker devient alors prohibitif et seules les approches vectorielles (BoW, GIST [TFW08]) sont capables de traiter une aussi grande quantité de données si l'on souhaite travailler en mémoire primaire.

4.2 Intérêt du formalisme : BoW et noyau explicite

Lorsque l'on considère des noyaux explicites, notre formalisme permet de réécrire la représentation BoF en un vecteur unique permettant ainsi de conserver la précision de l'approche noyau tout en utilisant la concision de la représentation par BoW. Nous pouvons alors tirer profit des améliorations de la réécriture des BoW, comme par exemple, la représentation BoW basée sur l'encodage des SIFT

proposée par Jegou *et al.* dans [JDSP10].

Réécrivons l'approche VLAD (cf. Sec. 3 Chapitre IV) comme un noyau sur BoF avec un noyau explicite. Soit U_i, un vecteur composé de $M-1$ zéros et d'une valeur 1 à la i^{th} composante. Soit ϕ une fonction d'injection de $\mathbb{R}^d$ vers $\mathbb{R}^{d\times M}$ telle que :

$$\phi(\mathbf{x}) = \mathbf{x} \otimes U_m - \mathbf{w}_m \otimes U_m \tag{V.6}$$

où $m = Q(\mathbf{x})$ comme définie éq. (IV.3) et $\otimes$ le produit tensoriel.

Le vecteur VLAD $\mathbf{v}_j$ du sac B_j (défini éq. (IV.6)) peut maintenant se réécrire :

$$\mathbf{v}_j = \sum_{\mathbf{b}_{sj}\in B_j} \phi(\mathbf{b}_{sj}) = \Phi(B_j) \tag{V.7}$$

Le produit scalaire utilisé pour calculer la similarité entre 2 vecteurs peut se développer :

$$\begin{aligned} \langle \mathbf{v}_q, \mathbf{v} \rangle &= \langle \Phi(B_q), \Phi(B_j) \rangle = K_{vlad}(B_q, B_j) \\ &= \left\langle \sum_{\mathbf{b}_{rq}\in B_q} \phi(\mathbf{b}_{rq}), \sum_{\mathbf{b}_{sj}\in B_j} \phi(\mathbf{b}_{sj}) \right\rangle \\ &= \sum_{\mathbf{b}_{rq}\in B_q} \sum_{\mathbf{b}_{sj}\in B_j} k_{vlad}(\mathbf{b}_{rq}, \mathbf{b}_{sj}) \end{aligned} \tag{V.8}$$

Nous pouvons voir que l'éq. (IV.6) permet bien de revenir à l'éq. (IV.10) qui définit un noyau sur sac :

$$\begin{aligned} k_{vlad}(\mathbf{x}, \mathbf{y}) &= \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle \\ &= \langle \mathbf{x} - \mathbf{w}_i, \mathbf{y} - \mathbf{w}_i \rangle \, \mathbb{1}_{Q(x)=Q(y)=i} \\ &= \langle \widetilde{\mathbf{x}}, \widetilde{\mathbf{y}} \rangle \, \mathbb{1}_{\mathcal{P}(x,y)} \end{aligned} \tag{V.9}$$

On reconnaît la formulation basique éq. (IV.13) de notre méthode d'approximation des noyaux sur sacs où $f_{\mathcal{P}}$ est alors définie par un quantificateur éq. (IV.5). K_{vlad} est alors un cas spécifique de notre formalisme $UFast_K$ éq. (IV.21) où k est un noyau linéaire shifté et f_{LSH} remplacée par f_Q. On peut noter que notre formalisme $UFast_K$ éq. (IV.21) est plus général car d'avantages de noyaux sont pris en compte.

En reconsidérant les résultats de précision obtenus sur la base Holidays, nous pouvons supposer que l'injection non linéaire utilisée dans K_{power} est plus pertinente que celle linéaire utilisée dans K_{vlad} du moins pour cette base.

Conclusion de la Partie II

Les méthodes de recherche d'images basées sur l'utilisation de sacs de points d'intérêt (BoF) suscitent actuellement un engouement important. Cependant, cette représentation complexe des données nécessite des traitements coûteux incompatibles avec l'accroissement des données.

Dans un premier temps, nous avons présenté les principales solutions proposées dans la littérature pour définir une similarité à partir de cette représentation et proposer des algorithmes de recherche efficaces. Nous avons proposé un cadre unificateur entre les approches basées dictionnaire, par vote et similarité noyau. Nous nous sommes ensuite concentrés sur la conception d'une approche, $UFast_K$, permettant d'approximer les méthodes basées noyaux. La méthode proposée permet d'utiliser pleinement la description BoF. Notre approximation, exploitant le schéma de recherche sous-linéaire LSH, permet une réduction importante des temps de recherche rendant possible le traitement de bases conséquentes (entre 100 000 et 1 000 000 d'images). Un grand nombre de tests, sur plusieurs bases, ont validé l'efficacité de notre méthode.

Troisième partie

Recherche interactive rapide dans les grandes bases d'images

Dans les parties précédentes, nous avons étudié des systèmes de recherche automatique capables de retrouver rapidement des images similaires à une image requête. Cependant, ces méthodes sont connues pour souffrir du "fossé sémantique/numérique" entre les descriptions de bas niveau et les critères de recherche de plus haut niveau sémantique des utilisateurs.

En introduction, nous avons présenté le principe du bouclage de pertinence qui vise à combler ce "fossé sémantique" en faisant intervenir l'utilisateur dans la boucle de recherche. Cependant, la complexité de cette approche étant linéaire par rapport à la taille de la base, ce principe n'est pas adapté à la recherche dans les grandes bases d'images.

Dans cette partie, nous nous intéressons aux approches interactives passant à l'échelle. Nous considérons tout particulièrement les méthodes proposant de n'examiner qu'une portion de la base pour rechercher les images d'intérêt : les images les plus pertinentes (le cœur de classe) et les images les plus incertaines (celles à présenter à l'utilisateur pour l'étape d'annotation).

Chapitre VI

Méthodes d'apprentissage interactif rapides

Nous nous intéressons aux systèmes de recherche interactive basés classification qui améliorent itérativement la recherche en demandant à l'utilisateur d'annoter des images sélectionnées automatiquement. Comme vu en Introduction I.2.2.2, à chaque interaction entre l'utilisateur et le système, deux problèmes d'optimisation se posent : retrouver le cœur de classe et sélectionner les images à annoter.

Ces optimisations peuvent être accélérées en travaillant directement sur la fonction de similarité, soit en l'approximant, soit en la bornant. Cependant, en apprentissage interactif, l'objectif n'est pas de trouver la solution optimale sur l'espace des caractéristiques, mais sur l'ensemble des images $\mathbf{x}$ d'une base $\mathcal{B}$.

Dans un premier temps, nous présentons les méthodes de la littérature pour la recherche rapide du cœur de classe. Ensuite, nous détaillons les méthodes permettant de retrouver rapidement les images à annoter. Enfin, nous introduisons notre stratégie, nommée SALSAS qui propose de réduire la complexité de l'apprentissage interactif en sélectionnant rapidement un pool d'images. Ce pool est à la fois utilisé pour la recherche du cœur de classe et pour la recherche des images à annoter.

Sommaire

1 Recherche rapide du cœur de classe 110

2 Recherche des images les plus proches de la frontière 116

3 Stratégie d'apprentissage interactif sous-lineaire basée sur la recherche *ppv* approximée 118

1 Recherche rapide du cœur de classe

1.1 Fonctions de similarité basiques

Lorsque l'on est capable de réexprimer la fonction de similarité $f_{\mathcal{A}}(\mathbf{x})$ comme une distance entre un point d'ancrage $\mathbf{x}^\star$ de l'espace des caractéristiques et les données $\mathbf{x}$ de la base, on se ramène au cas étudié en première partie. Il est alors possible d'utiliser une structure d'index capable de retrouver rapidement les images les plus similaires à $\mathbf{x}^\star$.

Cette solution est tout à fait adaptée aux méthodes de recherche interactive basées *repondération de la requête* (*query reweighting* (QR) en anglais) [RH00]. En effet, ces techniques consistent à calculer une nouvelle requête à chaque bouclage de pertinence en moyennant les signatures de l'ensemble des images pertinentes et la signature initiale.

Cependant, lorsque l'on travaille dans des espaces induits implicites, comme c'est le cas pour les SVM combinés avec des similarités noyaux, cette réécriture devient moins triviale. En effet, le barycentre est alors calculé dans l'espace induit $\phi(\mathbf{x}^\star) = \sum_{i \in I} \alpha_i \phi(\mathbf{x}_i)$ et non dans l'espace des caractéristiques.

1.2 Recherche par sous échantillonnage

Dans les cas où le calcul d'une nouvelle requête n'est pas trivial, il est possible de localiser un optimum approximatif de $f_{\mathcal{A}}$ en construisant un petit ensemble $\mathcal{M}$ d'images candidates de taille fixe.

On peut sélectionner ce sous-ensemble $\mathcal{M}$ en échantillonnant aléatoirement des images de la base $\mathcal{B}$ [SS02, DW00]. En examinant la pertinence des images de cet échantillon avec la fonction de similarité $f_{\mathcal{A}}$, il est possible de retrouver le meilleur élément de cet ensemble $\mathcal{M}$, noté $\mathbf{x}^\star$, avec une complexité de recherche sous-linéaire.

Pour garantir que l'élément $\mathbf{x}^\star$ est bien une image pertinente, la taille m de l'échantillon $\mathcal{M}$ est fixée en utilisant le théorème 6.33 (*Ranks on Random Subsets*) de la section 6.5 de [SS02] : Soit $\mathcal{B} := \{x_1, \ldots, x_n\}$, une base contenant n images, et $\mathcal{M} \subset \mathcal{B}$ un sous-ensemble aléatoire de taille m. Alors, la probabilité que le meilleur élément de $\mathcal{M}$ appartienne au TOPN de $\mathcal{B}$ vaut au moins $1 - \left(\frac{n-N}{n}\right)^m$.

Il est donc suffisant de considérer $\mathcal{M}$ en sélectionnant aléatoirement m éléments de $\mathcal{B}$ pour garantir, avec une probabilité η, que le meilleur élément de $\mathcal{M}$ appartienne au TOPN de $\mathcal{B}$, avec :

$$m = \frac{ln(1-\eta)}{ln(\frac{n-N}{n})} \tag{VI.1}$$

A titre d'exemple, pour une base de n =100 000 images, il est alors suffisant de sélectionner aléatoirement m =3 000 images pour s'assurer avec une probabilité de 95% que le meilleur élément de $\mathcal{M}$ appartiennent au TOP100 de la base. La recherche est alors 33 fois plus rapide qu'une recherche linéaire.

Cependant, lors des premières itérations du bouclage de pertinence, la fonction de similarité n'est pas assez fiable pour estimer un TOP100. Seules les toutes premières images sont réellement pertinentes. Il est alors nécessaire de considérer $N = 10$ (voire moins). La taille de l'échantillon devient alors trop importante pour que cette méthode soit efficace. A titre d'exemple, pour n =100 000, $N = 10$ et $\eta = 0,95$, il est nécessaire de sélectionner aléatoirement m =30 000 images. Il faut donc examiner un tiers de la base. Le gain devient faible par rapport à une recherche exhaustive.

1.3 Recherche hiérarchique

Dans [PGC06], Panda *et al.* proposent de construire le sous-ensemble d'images candidates de façon déterministe. L'échantillonnage n'est plus aléatoire mais il est guidé par des heuristiques ou des a priori. Ce principe général est très utilisé pour passer à l'échelle les systèmes de recherche.

Les auteurs partent du postulat que si une image $\mathbf{x}_p$ est proche de la requête dans l'espace induit, les images proches de $\mathbf{x}_p$ dans l'espace des caractéristiques ont de fortes « chances » d'être pertinentes aussi car elles partagent les mêmes informations.

Les données sont partitionnées, en prétraitement, à l'aide d'une méthode de clustering telle que Clindex [LCGM+00]. Après cette étape, les données sont classées en S cellules C_i et chaque cellule est représentée par son centre de classe c_i.

L'hypothèse sous-jascente est que la fonction de similarité $f_{\mathcal{A}}$ est localement régulière, *i.e.* $\forall \mathbf{x} \in \{C_i\}, \forall \mathbf{y} \notin \{C_i\}, f_{\mathcal{A}}(\mathbf{x}) > f_{\mathcal{A}}(\mathbf{y})$ si $c_i = \max_{j\in[1,\ldots,S]} f_{\mathcal{A}}(c_j)$.

L'étape de recherche du cœur de classe s'opère en deux temps. Dans un premier temps, la fonction de similarité est évaluée uniquement sur les C centres de classe afin d'évaluer la pertinence de chaque cellule. Dans un second temps, seules les cellules les plus pertinentes sont explorées. La fonction de similarité est alors évaluée sur toutes les images des cellules sélectionnées.

Il est cependant difficile de déterminer, a priori, le nombre S de cellules à utiliser pour partitionner les données de la base ainsi que le nombre de cellules à explorer lors de l'étape de recherche.

1.4 K-VA files

La méthode K-VA files, proposée par Heisterkamp *et al.* [HP05], est une méthode de recherche hiérarchique où le partitionnement des données est réalisé dans l'espace induit. De plus, un critère

d'arrêt permet de déterminer le nombre de cellules à explorer.

Cette technique est obtenue en étendant la méthode de recherche rapide VA-file [WSB98] aux fonctions noyaux. Les données $\phi(\mathbf{x})$ sont décomposées sur une sous-base orthonormée incomplète ν dans l'espace induit en utilisant l'orthonormalisation de Gram-Schmidt [BJ03, CSTL02]. Un vecteur $\phi(\mathbf{x})$ s'exprime alors :

$$\phi(\mathbf{x}) = \sum_{t}^{d-1} \gamma_t \nu_t + \phi^{\perp}(\mathbf{x}) = \gamma\nu + \phi^{\perp}(\mathbf{x})$$

avec ν la sous-base composée de d vecteurs et $\phi^{\perp}(\mathbf{x_i})$ le reste de la décomposition (la base est incomplète).

Il est important de noter que dans les espaces induits implicites, les vecteurs $\phi(\mathbf{x})$, ν et $\phi^{\perp}(\mathbf{x})$ ne sont pas accessibles directement. Cependant, on peut représenter chaque donnée de la base par le vecteur γ et la norme du résidu $\mathbf{G_x} = \langle \phi^{\perp}(\mathbf{x}), \phi^{\perp}(\mathbf{x}) \rangle$. La seule perte d'information réside dans l'indétermination de la direction de $\phi^{\perp}(\mathbf{x})$.

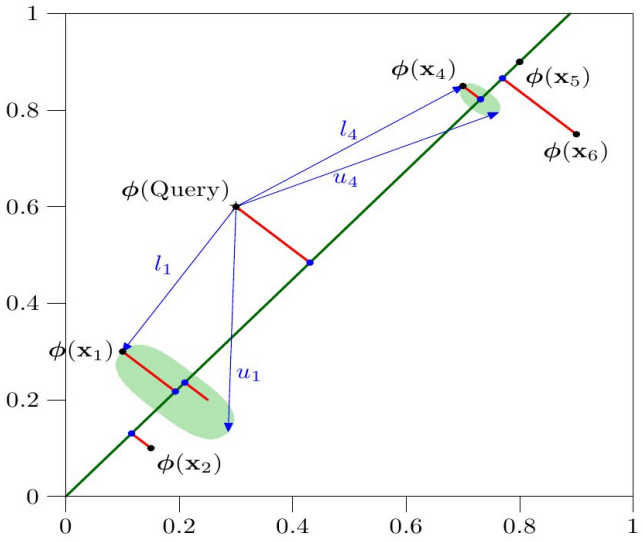
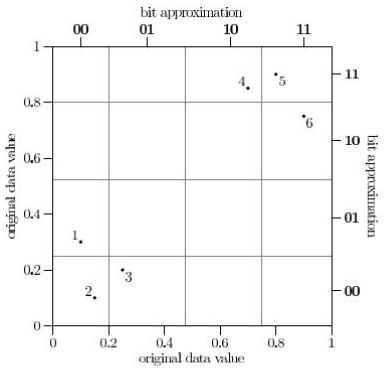


Figure VI.1 – Bornes inférieures l_i et suppérieures u_i de la similarité entre Q et $\mathbf{x}_i$ dans un espace induit par ϕ [HP05].

Après décomposition, il n'est plus possible de calculer la similarité entre une requête $\mathbf{x}_q$ et une image $\mathbf{x}_p$ de la base. Cependant, en considérant $\phi^{\perp}(\mathbf{x}_p)$ et $\phi^{\perp}(\mathbf{x}_q)$ de même direction ou de direction opposée, il est possible de borner la similarité entre la requête et une image de la base (cf. Fig. VI.1).

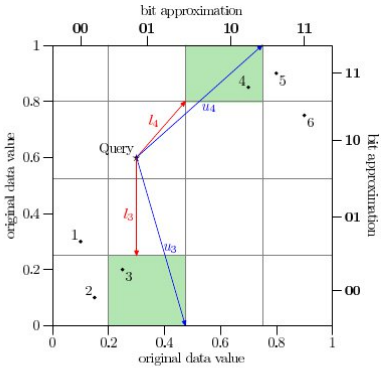
Maintenant que l'on est capable de réexprimer toute donnée $\phi(\mathbf{x})$, on peut partitionner l'espace induit en utilisant une quantification vectorielle. Les données sont en effet réparties dans des cellules représentées par un mot de bits obtenu en échantillonnant régulièrement les composantes du vecteur γ et le résidu G (cf. Fig. VI.2).



(a)

	Vectors	VA-File
1	(0.1,0.3)	00 01
2	(0.15,0.1)	00 00
3	(0.25,0.2)	01 00
4	(0.7,0.85)	10 11
5	(0.8,0.9)	11 11
6	(0.9,0.75)	11 10

(b)



(c)

Figure VI.2 – Approximation du VA-File [HP05].

Lors de l'étape de recherche, la requête est décomposée sur la base ν. Les bornes inférieures et supérieures de la fonction de similarité sont calculées pour chaque cellule du K-VA file. Les cellules sont ensuite triées en fonction de leur borne inférieure. Toutes les cellules dont la borne inférieure est plus grande que la borne supérieure de la première cellule sont éliminées. Les pertinences des images contenues dans la première cellule sont ensuite calculées. Tant que la borne inférieure de la cellule suivante est plus petite que le score de l'image la plus pertinente, la cellule suivante est explorée.

Cette méthode de recherche a uniquement été utilisée avec des SVM à 1 classe, *i.e.* des fonctions de similarité définies par des sommes pondérées (par des coefficients positifs) de vecteurs de l'espace induit.

1.5 Kernel Indexer : KDX

Dans [PC05, PGC06], Panda *et al.* ont proposé une méthode de recherche, KDX, permettant d'effectuer des recherches rapides dans l'espace induit à partir de requêtes s'exprimant sous la forme de sommes pondérées (par des coefficients à valeur dans $\mathbb{R}$) de vecteurs de l'espace induit. Cette structure d'index est donc compatible avec les classifieurs SVM à 2 classes.

Cette méthode utilise les propriétés géométriques des espaces induits de norme unitaire pour partitionner les données $\phi(\mathbf{x})$. L'image $\mathbf{x}_c$, la plus proche du centre des données dans l'espace induit, est utilisée comme référence pour partitionner les données en hyper-anneaux centrées sur $\phi(\mathbf{x}_c)$ (cf. Fig. VI.3). Les cellules regroupent les images en fonction de leur angle avec $\phi(\mathbf{x}_c)$.

Lors de la recherche, l'angle entre la requête $w = \sum_i y_i \alpha_i \phi(\mathbf{x}_i)$ et la donnée de référence $\phi(\mathbf{x}_c)$ permet de sélectionner une cellule ainsi que les images associées (cf. Fig. VI.4). La pertinence des ces images est évaluée avec la fonction de similarité et la meilleure image $\mathbf{x}^\star$ est sélectionnée. Les anneaux voisins sont ensuite explorés à la recherche d'une meilleure image. En effet, tant que l'angle

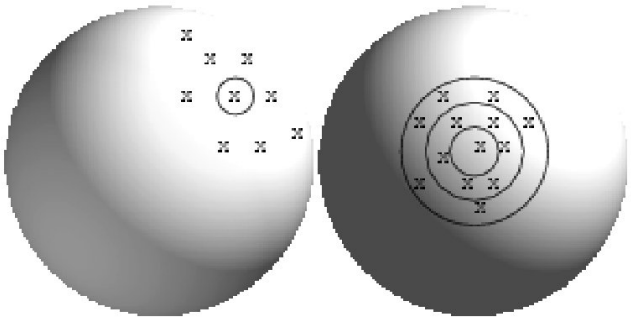
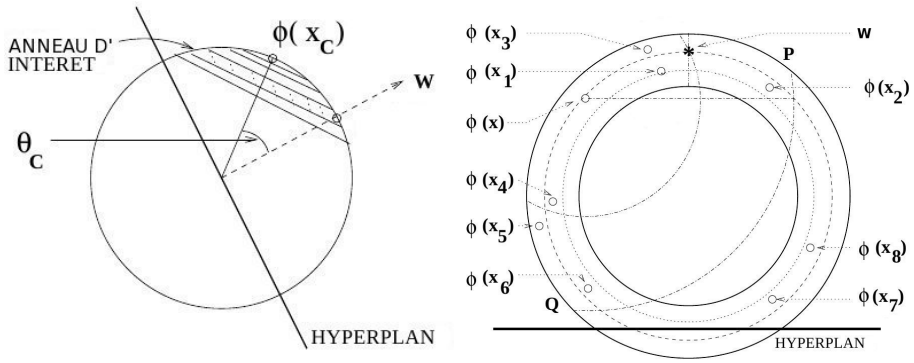


Figure VI.3 – Approximation du l'élément central et des anneaux [PC05].



(a) Anneau d'intérêt A (b) Sélection du meilleur candidat de A

Figure VI.4 – Etapes de recherche [PC05].

entre w et $\phi(\mathbf{x}_c)$ est supérieure à celui entre w et $\mathbf{x}^\star$, il est possible qu'un anneau voisin contienne de meilleurs éléments. Il vient que plus l'angle entre le point d'ancrage $\phi(\mathbf{x}_c)$ et la requête w est important, moins l'anneau sélectionné est représentatif des données recherchées et moins on a de chance de sélectionner une bonne image.

Aucune garantie n'est donnée quant à la convergence de l'algorithme vers le meilleur élément. Il est en effet possible de ne jamais sélectionner l'anneau contenant le meilleur élément. C'est pourquoi les auteurs proposent d'arrêter la recherche après l'examen d'un nombre fixe de candidats.

L'idée de cette méthode reste cependant très intéressante et la méthode pourrait facilement être améliorée. On pourrait construire plusieurs partitions avec des points d'ancrage différents. Lors de la recherche, seule la partition dont le point d'ancrage est le plus représentatif pourrait être utilisée. On pourrait également utiliser cette méthode pour construire une table de hachage. Chaque point d'ancrage pourrait permettre de construire une fonction de hachage. La partition obtenue serait issue de plusieurs découpages de l'espace induit.

1.6 Recherche dans l'espace des caractéristiques

Dans [PC06], Panda et Chang proposent une solution pour approximer la fonction $\phi^{-1}(w)$ lorsque w est défini par un classifieur SVM à 2 classes. Cette approximation permet de passer de l'espace induit implicite à l'espace des caractéristiques. Il est alors possible d'utiliser un R-Tree [Man06] pour rechercher le cœur de classe dans l'espace des caractéristiques.

Le passage de l'espace induit à l'espace des caractéristiques est rendu possible par l'utilisation de l'algorithme SVC (Support Vector Clustering) [BHHSV02]. Cette méthode permet de regrouper les données d'apprentissage appartenant aux mêmes régions de l'espace des caractéristiques. Ces régions d'intérêt, centrées sur les différents modes du cœur de classe, sont ensuite approximées par des boîtes englobantes. Ces boîtes sont ensuite utilisées comme requête du R-Tree.

La principale limitation de cette méthode vient de l'utilisation du R-Tree qui passe difficilement à l'échelle lorsque la dimension de l'espace des caractéristiques devient importante. Ce problème est aggravé par la particularité des requêtes représentées par des boîtes (hyper-pavés) qui ont une grande chance d'appartenir à plusieurs cellules du R-Tree.

1.7 Synthèse

Les algorithmes de recherche rapide du cœur de classe sélectionnent un pool d'images afin d'évaluer la fonction de similarité uniquement sur un sous-ensemble de la base. Ce pool d'images peut être construit aléatoirement, en partitionnant la base ou en utilisant des structures d'index.

Certaines approches basées partitionnement sont dites pessimistes contrairement aux approches basées structures d'index qui sont dites optimistes. Les méthodes optimistes tentent de retrouver directement la meilleure cellule, celle qui contient les données d'intérêt. Cependant, comme nous l'avons illustré avec KDX [PGC06], elles ne parviennent pas toujours à obtenir directement la cellule d'intérêt. Un grand nombre de cellules doivent alors être explorées avant de trouver les images d'intérêt. Les approches pessimistes reposent sur le constat que n'étant pas capables de trouver directement la meilleure cellule, il peut être plus intéressant de tester toutes les cellules afin d'en déduire un ordre de parcours. Pour que cette stratégie soit efficace, il faut que la fonction de test soit cependant rapidement calculable. Lorsque la fonction de test a le même coût calculatoire que la fonction de similarité, les approches pessimistes ont une complexité optimale en $O(\sqrt{n})$ avec n la taille de la base. Cet optimum est atteint pour $\sqrt{n}$ cellules contenant chacune $\sqrt{n}$ images. Les approches optimistes peuvent quant à elles espérer atteindre une complexité plus faible.

Nous avons également vu qu'il est possible de travailler dans l'espace des caractéristiques ou directement dans l'espace induit. La répartition des images en cellules dans l'espace induit est rendue possible par l'utilisation de certaines données $\phi(\mathbf{x})$ permettant de construire une référence dans cet

espace induit. Cette référence, définie par une base de décomposition ou un point d'ancrage, est certes incomplète (elle ne permet pas de représenter complètement les données) mais elle est suffisante pour découper l'espace.

2 Recherche des images les plus proches de la frontière

Nous allons maintenant présenter l'état de l'art des méthodes traitant du passage à l'échelle pour la sélection des images les plus incertaines.

2.1 Extension des méthodes de recherche du cœur de classe

Comme pour la recherche du cœur de classe, il est possible d'utiliser la recherche aléatoire. La seule différence est qu'au lieu de garder l'image du pool qui maximise la fonction de similarité, on conserve l'image dont le score de pertinence est le plus proche de 0.

De même, il est également possible d'utiliser la recherche hiérarchique en explorant la cellule dont l'élément représentant est celui dont le score de pertinence est le plus proche de 0.

La méthode proposée par Panda et Chang dans [PC06] permet également de retrouver les images les plus incertaines. En effet, par construction, les boîtes englobantes contiennent à la fois les images les plus pertinentes mais également les images les plus incertaines.

2.2 M-Tree

Dans [CEOT07], Crucianu *et al.* proposent une méthode dédiée à la recherche rapide des images les plus incertaines.

Cette méthode est obtenue en adaptant un M-Tree à la recherche des éléments les plus proches d'un hyperplan. Cet arbre de recherche étant basé métrique (ou similarité), il s'étend naturellement à la recherche dans des espaces induits.

Le principe du M-Tree est que chaque nœud de l'arbre regroupe les éléments de la base dont la distance avec le représentant du nœud est inférieure à un seuil. Ce seuil correspond au rayon du nœud. Le principe des M-Tree repose sur les deux résultats suivants issus de l'inégalité triangulaire :

- Si la distance entre la requête Q et le nœud courant O_r est supérieure à la somme du rayon de recherche $r(Q)$ et du rayon du nœud $r(O_r)$: $D(Q,O_r) > r(Q) + r(O_r)$, alors il n'est pas utile d'explorer le nœud O_r puisque $D(Q,X_j) > r(Q)$, $\forall X_j \in O_r$.
- Si $|D(Q,O_p) - D(O_p,O_r)| > r(Q) + r(O_r)$, il n'est pas utile de calculer $D(Q,O_r)$. En effet, $D(Q,O_r) \geq |D(Q,O_p) - D(O_p,O_r)|$ d'où $D(Q,O_r) > r(Q) + r(O_r)$.

En remarquant que la distance euclidienne dans l'espace induit entre deux données $\phi(\mathbf{x}_i)$ et $\phi(\mathbf{x}_j)$ est explicitable en utilisant l'« astuce du noyau » :

$$||\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)||_2^2 = k(\mathbf{x}_i, \mathbf{x}_i) + k(\mathbf{x}_j, \mathbf{x}_j) - 2k(\mathbf{x}_i, \mathbf{x}_j)$$

le M-Tree est directement utilisable quel que soit le noyau.

En considérant que la recherche des images les plus incertaines consiste à retrouver les images satisfaisant :

$$\mathbf{x}_{i^\star} = \arg\min_{\mathbf{x}_i}(|\sum_a \alpha_a y_a k(\mathbf{x}_i, \mathbf{x}_a) + b|)$$

c'est-à-dire retrouver les images les plus proches de la frontière $\mathbf{H}$:

$$D(\mathbf{H}, \mathbf{x}_i) = (|\sum_a \alpha_a y_a k(\mathbf{x}_i, \mathbf{x}_a) + b|)$$

On peut utiliser cette nouvelle métrique dans le M-Tree et redéfinir les deux principes décrits précédemment :

- Si $D(\mathbf{H}, O_r) > r(Q) + r(O_r)$, alors il n'est pas utile d'explorer le nœud O_r puisque $D(\mathbf{H}, X_j) > r(Q), \forall X_j \in O_r$.
- Cependant la règle : $D(\mathbf{H}, O_r) \geq |D(\mathbf{H}, O_p) - D(O_r, O_p)|$ n'est pas vraie dans tous les cas. Elle tient toujours uniquement lorsque $D(\mathbf{H}, O_r) \geq D(\mathbf{H}, O_p) - D(O_r, O_p)$. Lorsque $D(\mathbf{H}, O_p) - D(O_r, O_p) \leq 0$, il est alors nécessaire de calculer $D(\mathbf{H}, O_r)$.

Cette méthode fait partie de la famille de partitionnement des données dans l'espace induit, ce qui la rend compatible avec tous les noyaux de norme unitaire. D'autre part, elle est compatible avec les classifieurs SVM à deux classes.

La limitation de cette méthode repose sur le fait que l'hyperplan séparateur peut intersecter un grand nombre de nœuds obligeant à explorer une partie très importante des données.

2.3 Synthèse

Le principe général des algorithmes de recherche rapide des images les plus incertaines est identique à celui de la recherche rapide du cœur de classe. L'objectif est de sélectionner rapidement un pool d'images afin d'évaluer la fonction de similarité sur un sous-ensemble de la base.

Cependant la construction de ce pool d'images est plus complexe. En effet, l'optimalité de la fonction objective : $\{\mathbf{x}\} \in \mathbb{R}^d | \sum_i \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) = 0$ n'est plus définie, dans l'espace induit, par un point (comme c'est la cas pour la recherche du cœur de classe) mais par un hyperplan (et un ensemble de modes dans l'espace des caractéristiques).

Les techniques pessimistes, telle que la recherche hiérarchique, qui permettent d'utiliser des fonctions de tests complexes sont encore utilisables. Cependant, les structures d'index, comme le M-Tree [CEOT07], exigent des adaptations importantes.

Une autre solution, précédemment présentée par Panda *et al.*dans [PC06], montre qu'il est possible de retrouver les images les plus incertaines en utilisant un rayon de recherche suffisamment grand autour du cœur de classe.

3 Stratégie d'apprentissage interactif sous-lineaire basée sur la recherche ppv approximée

Dans cette section, nous présentons notre stratégie, nommée SALSAS (pour *Sub-linear Active Learning Strategy with Approximate KNN Search*), permettant de passer à l'échelle un système de recherche interactif. Notre méthode consiste à sélectionner un pool d'images en approximant la fonction de similarité définie par un classifieur SVM à 2 classes. Cette approximation permet de travailler dans l'espace des caractéristiques. Le passage à l'échelle est alors géré via une structure d'index tel que χ^2-*LSH* qui a déjà prouvé son efficacité sur les grandes bases d'images et qui est compatible avec des descripteurs de grande dimension. Tout comme [PC06], nous proposons une stratégie permettant à la fois la recherche rapide du cœur de classe ainsi que la sélection des images les plus incertaines.

3.1 Justification de la construction du pool d'images

Rappelons que dans le contexte de la recherche d'image par similarité, on peut noter que l'utilisateur n'est pas intéressé par le tri de toute la base mais uniquement par les N images les plus pertinentes. Seul le tri des N images du TOP du classement, appelé TOPN, est alors utile (habituellement, N est fixé par l'utilisateur). Nous voulons utiliser cette spécificité pour résoudre le problème du passage à l'échelle (relatif à la taille de la base) des méthodes d'apprentissage actif en ligne. Le tri de toute la base implique « d'examiner » au moins une fois chaque image de la base. Notre idée est de réduire ce processus de tri en sélectionnant un pool d'images, appelé $\mathbb{S}$. Notre méthode s'inspire donc du principe général des méthodes de recherche rapide présenté dans ce chapitre en Section 1.3. Ce pool est obtenu grâce à une heuristique qui favorise la sélection d'images du TOPN. Dans le but d'être réellement sous-linéaire et de ne pas examiner (même brièvement) toutes les images de la base, une structure d'index doit être utilisée pour récupérer le pool $\mathbb{S}$ rapidement. Nous commençons par détailler notre stratégie de sélection du pool $\mathbb{S}$ avant de présenter la stratégie d'apprentissage actif limité à $\mathbb{S}$.

3.2 Sélection rapide d'un sous-ensemble d'images pertinentes

Rappelons que la pertinence d'une image $\mathbf{x}$ est estimée par la fonction de pertinence $f_{\mathcal{A}}$ (cf. Sec. I.2.2.2) : $f_{\mathcal{A}}(\mathbf{x}) = \sum_{i=1}^{|\mathcal{A}|} y_i \alpha_i k(\mathbf{x}_i, \mathbf{x})$ (sans perte de généralité, on omet ici le b ; les α_i sont positifs).

Le TOPN est constitué des N images de la base $\mathcal{B}$ qui maximisent $f_{\mathcal{A}}$. Cette fonction peut être dissociée de la façon suivante :

$$f_{\mathcal{A}}(\mathbf{x}) \quad = f_{\mathcal{A}^+}(\mathbf{x}) - f_{\mathcal{A}^-}(\mathbf{x}) \quad = \sum_{p=1}^{|\mathcal{A}^+|} \alpha_p k(\mathbf{x}_p, \mathbf{x}) - \sum_{n=1}^{|\mathcal{A}^-|} \alpha_n k(\mathbf{x}_n, \mathbf{x}) \tag{VI.2}$$

où $\mathcal{A}^+$ représente les images annotées positivement : $(\mathbf{x}_p, y_p = +1) \in \mathcal{A}$ et $\mathcal{A}^-$, celles annotées négativement : $(\mathbf{x}_n, y_n = -1) \in \mathcal{A}$.

Notre stratégie consiste, dans un premier temps, à approximer la maximisation de $f_{\mathcal{A}}$ en ne considérant que $f_{\mathcal{A}^+}$ (qui majore $f_{\mathcal{A}}$). Nous supposons ici que si une image $\mathbf{x}$ est proche d'un des exemples positifs de l'ensemble d'apprentissage ($\mathbf{x}_p \in \mathcal{A}^+$), $\mathbf{x}$ a de bonnes « chances » d'avoir une valeur $f_{\mathcal{A}^+}(\mathbf{x})$ importante (pas pertinent pour toute fonction k mais pour les noyaux vérifiant : $\forall \mathbf{x}$, $k(\mathbf{x}_p, \mathbf{x}_p) \geq k(\mathbf{x}_p, \mathbf{x})$). Or, si le score $f_{\mathcal{A}^+}(\mathbf{x})$ est important, le score $f_{\mathcal{A}}(\mathbf{x})$ peut également être important (car $\forall i$, $\alpha_i \geq 0$ et $\forall \mathbf{x}, \mathbf{y} \in \mathcal{B}$, $k(\mathbf{x}, \mathbf{y}) \geq 0$). Cette hypothèse n'est pas vraie pour toutes les images $\mathbf{x}$, puisque la fonction $f_{\mathcal{A}^-}$ influe également sur le score $f_{\mathcal{A}}$ en pénalisant certaines images. Cependant, ce n'est pas une limitation de l'approche car $f_{\mathcal{A}}$ étant disponible, il est aisé d'ajouter une étape pour filtrer les images non pertinentes.

La première étape de notre processus, appelée *sélection*, consiste donc à sélectionner parmi l'ensemble des images non annotées $\mathcal{U}$, les images qui sont les plus proches d'au moins un des exemples positifs de l'ensemble d'apprentissage $\mathcal{A}^+$. Cette étape permet de construire le pool de candidats noté $\mathcal{C}$ en effectuant une recherche *ppv* pour les données de $\mathcal{A}^+$. Les recherches *ppv* peuvent être réglées en fonction des valeurs α des données de $\mathcal{A}^+$. Il est alors possible de sélectionner davantage de candidats dans le voisinage des données ayant un poids α important que dans le voisinage des données ayant un poids faible. Le grand intérêt de la méthode est que la recherche des *ppv* peut être réalisée avec une complexité sous-linéaire par rapport à la taille de la base (ou $\mathcal{U}$). C'est la principale motivation de notre stratégie de « sélection ». De cette façon, nous pouvons récupérer des candidats très rapidement. Même si un grand nombre d'entre eux ne sont pas pertinents, cette stratégie est toujours efficace puisque la pertinence des candidats peut être vérifiée.

La seconde étape de notre stratégie, appelé « élagage », a pour but de filtrer la sortie de l'algorithme de *ppv*. Comme nous venons de le noter, un candidat $\mathbf{x} \in \mathcal{C}$ (sélectionné par l'étape de « sélection ») peut finalement avoir un score $f_{\mathcal{A}}$ faible. L'étape d'« élagage » a donc pour but de supprimer les

éléments les moins bons de $\mathcal{C}$. Cette opération est réalisée en créant le pool $\mathcal{S}$ contenant les p meilleurs éléments du pool $\mathcal{C}$. Comme $|\mathcal{C}| \ll |\mathcal{U}|$, évaluer $f_{\mathcal{A}}$ sur chacune des images de $\mathcal{C}$ n'est pas pénalisant. Lorsque le pool $\mathcal{S}$ de taille $p = |\mathcal{S}|$ est assez grand ($p \geq N$), l'approximation du TOPN est simplement extraite de $\mathcal{S}$.

Pour que notre schéma soit efficace, la recherche *ppv* doit être rapide. Nous utilisons une structure d'index telle que *LSH* (présentée au Chap. II). Comme *LSH* est principalement conçue pour les distances l_1 et l_2, pour rester cohérent avec la fonction noyau utilisée par la fonction de pertinence, notre système se limite aux noyaux construits à partir de ces distances. Cependant, il a été observé que la qualité de recherche obtenue en utilisant ces distances n'est pas toujours satisfaisante. La distance χ^2 s'est souvent montrée plus performante pour les tâches de recherche d'images et de vidéos [GCPF08b, CHV99, SvdSdR+08, SvdSdR+09]. Grâce à notre schéma χ^2-*LSH* (cf. Chap. III), notre système s'étend également aux noyaux construits à partir de la distance χ^2.

3.3 Stratégie active

Nous traitons maintenant du second verrou concernant le problème de scalabilité de l'apprentissage actif. Ce problème de scalabilité s'opère lors de l'étape de sélection en ligne des images à présenter à l'utilisateur pour être annotées. L'objectif de cette étape est de définir une stratégie qui permet de sélectionner les meilleures images à annoter parmi l'ensemble des images non annotées $\mathcal{U}$. Ces images annotées sont ajoutées à l'ensemble d'apprentissage $\mathcal{A}$ et la fonction de pertinence $f_{\mathcal{A}}$ est mise à jour. L'algorithme passe ensuite à l'itération suivante du bouclage de pertinence.

Dans le but de passer à l'échelle, nous proposons de tirer profit de l'étape précédente permettant d'approximer le classement du TOPN. En effet, notre stratégie est basée sur l'utilisation de l'ensemble $\mathcal{S}$ au lieu de $\mathcal{U}$ pour sélectionner les meilleurs images à annoter.

En plus d'avoir déjà à disposition ce sous-ensemble, d'autres raisons motivent la recherche des images à annoter dans le pool $\mathcal{S}$:

- De façon analogue à [PC06], où les auteurs proposent de construire des boîtes englobantes centrées sur les images les plus pertinentes et suffisamment grandes pour contenir les images les plus incertaines, il est possible d'utiliser un pool $\mathcal{S}$ de taille assez importante ($|\mathcal{S}| > N$) pour s'assurer qu'il contienne également des images incertaines.
- Le problème de classification considéré est excessivement déséquilibré. En effet, lorsque l'on considère les très grandes bases de données, la taille de l'ensemble des images pertinentes est toujours bien plus petite que la taille de l'ensemble des images non pertinentes. Il s'ensuit qu'une image annotée positivement a plus de chance de se trouver près du centre de la classe d'intérêt qu'une image annotée négativement. En se concentrant pas trop loin des images an-

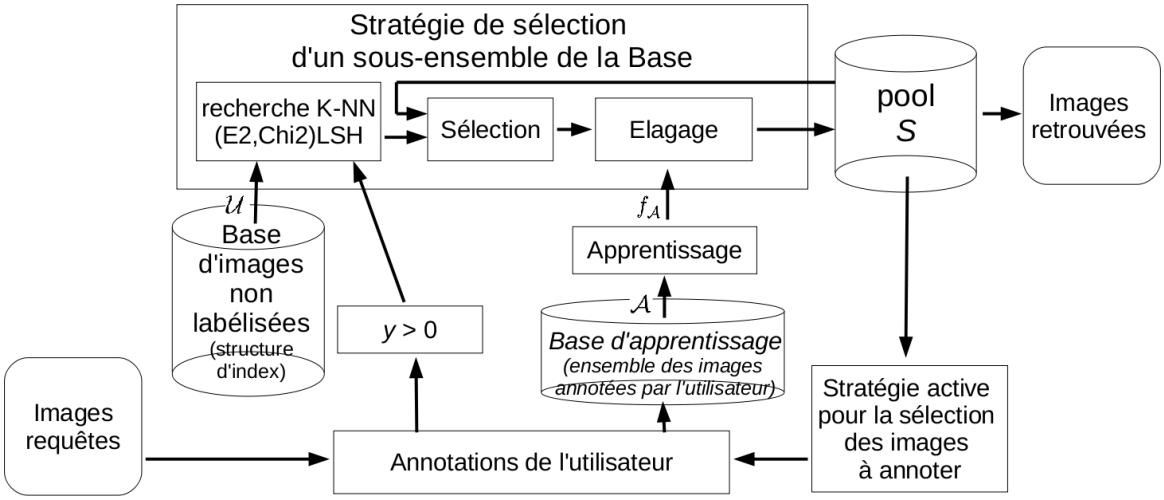


Figure VI.5 – schéma du système d'apprentissage interactif de complexité sous-linéaire par rapport à la taille de la base

notées positivement, soit dans $\mathcal{S}$, nous pouvons espérer accroître les chances de retrouver des images pertinentes et ainsi rééquilibrer le problème d'apprentissage.

– Tant que l'utilisateur n'est pas satisfait des résultats, cela signifie que la fonction de similarité $f_{\mathcal{A}}$ utilisée pour estimer la pertinence des images n'est pas complètement fiable. Des images considérées comme pertinentes par $f_{\mathcal{A}}$ peuvent donc se révéler non pertinentes par l'utilisateur. Il s'ensuit que le pool $\mathcal{S}$ contient plusieurs images qui peuvent remettre en cause la fonction $f_{\mathcal{A}}$ et ainsi améliorer l'estimation de la fonction de pertinence.

En définitive, nous proposons d'utiliser la méthode de la diversité des angles [Bri03] définie à l'équation (I.2), en considérant $\mathcal{S}$ au lieu de $\mathcal{U}$, pour sélectionner les images à annoter :

$$i^{\star} = \arg\min_{\mathbf{x}_i \in \mathcal{S}} (\lambda |f_{\mathcal{A}}(\mathbf{x}_i)| + (1-\lambda)(\max_{\mathbf{x}_j \in \mathcal{A}} \frac{|k(\mathbf{x}_i, \mathbf{x}_j)|}{\sqrt{k(\mathbf{x}_i, \mathbf{x}_i) k(\mathbf{x}_j, \mathbf{x}_j)}})) \tag{VI.3}$$

λ est un paramètre qui permet de favoriser le degré de diversité ou celui d'incertitude des candidats. Nous n'avons pas étudié l'impact de ce paramètre qui a été fixé à $1/2$ dans toutes nos expériences.

3.4 Système

Le schéma de notre système est résumé Figure VI.5. Une nouvelle recherche est démarrée lorsque l'utilisateur fournit des *images requêtes* annotées positivement et/ou négativement au système. Au moins une image annotée positivement est requise. Ces images sont utilisées pour initialiser l'ensemble d'apprentissage $\mathcal{A}$ (*données labélisées*). Cet ensemble permet d'apprendre la fonction de pertinence $f_{\mathcal{A}}$ obtenue en entraînant un classifieur SVM (*apprentissage*). En parallèle, les images anno-

tées positivement ($y > 0$) sont utilisées pour effectuer des recherches *ppv* dans l'*ensemble des données non labélisées* $\mathcal{U}$. Les images ainsi retrouvées permettent d'initialiser le pool $\mathcal{S}$. Les recherches *ppv* sont réalisées par un algorithme de recherche rapide tel que E^2-*LSH* ou χ^2-*LSH* suivant le noyau utilisé par la fonction de pertinence $f_{\mathcal{A}}$. $f_{\mathcal{A}}$ est ensuite utilisée, lors de l'étape d'*élagage*, pour trier le pool $\mathcal{S}$. Seules les p meilleures images sont alors conservées. L'étape d'*élagage* permet ainsi de s'assurer que la taille du pool $\mathcal{S}$ reste constante en supprimant les moins bons éléments. Le TOPN est ensuite présenté à l'utilisateur (*Images retrouvées*). L'étape de *stratégie active* sélectionne les images les plus incertaines dans $\mathcal{S}$. Ces images sont ensuite présentées à l'utilisateur pour être annotées.

Au cours de l'itération suivante, les images annotées par l'utilisateur sont ajoutées dans l'ensemble d'apprentissage $\mathcal{A}$. La fonction de pertinence $f_{\mathcal{A}}$ est mise à jour. Les images annotées positivement par l'utilisateur ($y > 0$) au cours de cette itération sont utilisées comme requête de l'algorithme *LSH*. Les images retrouvées sont ajoutées au pool $\mathcal{S}$. L'étape d'*élagage* filtre le pool $\mathcal{S}$ pour que sa taille reste constante. Le TOPN est ensuite présenté à l'utilisateur (*Images retrouvées*). L'étape de *stratégie d'apprentissage interactif* sélectionne les images les plus incertaines dans $\mathcal{S}$. Le système boucle tant que l'utilisateur n'est pas satisfait des résultats de sa recherche.

Comme nous venons de le voir, après chaque itération, la fonction de pertinence $f_{\mathcal{A}}$ est réapprise à partir du nouvel ensemble d'apprentissage $\mathcal{A}$. Le pool $\mathcal{S}$ doit donc être remis à jour. La mise à jour de $\mathcal{S}$ est réalisée par le bloc *sélection* en approximant $f_{\mathcal{A}^+}$ de l'équation (VI.2). Les images de l'ensemble $\mathcal{U}$ proche d'au moins un des exemples positifs de l'ensemble $\mathcal{A}^+$ doivent alors être retrouvées. Cependant, en notant que $\mathcal{S}$ contient déjà les images sélectionnées aux itérations précédentes, cette étape est accélérée en ajoutant uniquement les *ppv* des nouvelles images annotées positivement.

3.5 Algorithme et complexité

L'implémentation de notre méthode est donnée dans l'algorithme 3. Nous considérons ici que la recherche est initialisée avec une image requête I_q annotée positivement. A la ligne 2, le pool $\mathcal{S}$ est initialisé avec les p images retrouvées par la recherche *ppv* rapide de l'image I_q dans $\mathcal{U}$. La complexité de cette étape est en $O(n^{\rho})$ avec $\rho < 1$ qui dépend du paramétrage de la structure d'index utilisée. A chaque bouclage de pertinence, de la ligne 4 à 8, suivant la présence (ou non) d'annotation négative, la fonction de pertinence $f_{\mathcal{A}}$ est mise à jour avec un SVM à 2 classes ou un SVM à 1 classe [CZH01]. La complexité de l'étape d'apprentissage est en $O(\mathcal{A}^2)$. Tant que le nombre d'annotations par itération et/ou le nombre d'itérations restent raisonnables, on a $|\mathcal{A}| << n$, la complexité de cette étape est négligeable. A la ligne 9, le pool $\mathcal{S}$ est trié avec $f_{\mathcal{A}}$. La complexité de cette étape en $O(|S|.log(|S|))$. A cette étape, S contient au plus $(p + b.k) << n$ images. A la ligne 11, dans le but de garantir une taille de $\mathcal{S}$ constante, seules les p images les plus pertinentes sont conservées. A la ligne 13, le TOPN de $\mathcal{S}$

Algorithm 3

Require: I_q, $\mathcal{U}$, K, p, b ; /* Image requête, données non labélisées, nombre de ppv, taille du pool, nombre d'annotations par itération */

$\mathcal{A} \leftarrow (I_q, +1)$
$\mathcal{S} \leftarrow p$-NN(I_q)
loop
  if $\forall(\mathbf{x}_s, y_s) \in \mathcal{A}, \exists s \mid y_s < 0$ **then**
    $f_{\mathcal{A}} \leftarrow$ 2-class SVM($\mathcal{A}$)
  else
    $f_{\mathcal{A}} \leftarrow$ 1-class SVM($\mathcal{A}$)
  end if
  trier($\mathcal{S}$) (en calculant $f_{\mathcal{A}}(\mathbf{x}_i)$ $\forall \mathbf{x}_i \in \mathcal{S}$)
  if $|\mathcal{S}| > p$ **then**
    retirer $\{\mathcal{S}_r\}_{r \in [p,\ldots,|\mathcal{S}|]}$
  end if
  montrer TOPN of $\mathcal{S}$
  for $a = 0$ to b **do**
    $\mathbf{x}_s \leftarrow \arg\min_{\mathbf{x}_i \in \mathcal{S}} (\frac{1}{2} * |f_{\mathcal{A}}(\mathbf{x}_i)| + \frac{1}{2}(\max_{\mathbf{x}_j \in \mathcal{A}} \frac{|k(\mathbf{x}_i, \mathbf{x}_j)|}{\sqrt{k(\mathbf{x}_i, \mathbf{x}_i) k(\mathbf{x}_j, \mathbf{x}_j)}}))$
    $y_s \leftarrow$ annotation de l'utilisateur $\{-1, +1\}$
    $\mathcal{A} \leftarrow \mathcal{A} \cup (\mathbf{x}_s, y_s)$
    if $y_s > 0$ **then**
      $\mathcal{S} \leftarrow \mathcal{S} \cup KNN(\mathbf{x}_s)$
    end if
  end for
end loop

est présenté à l'utilisateur comme résultat intermédiaire. A la ligne 15, la stratégie active sélectionne les b images du pool $\mathcal{S}$ à présenter à l'utilisateur pour être annotées. La complexité de cette étape est en $O(b.(|S|+|A|).log(|S|+|A|)) = O(|S|.log(|S|))$ avec $|S| = p$. A la ligne 16, les images sélectionnées par la stratégie active sont annotées par l'utilisateur avant d'être ajoutées à l'ensemble d'apprentissage $\mathcal{A}$. A la ligne 19, le pool $\mathcal{S}$ est mis à jour en ajoutant les K images ppv de chaque image positivement annotée par l'utilisateur à l'itération courante. La complexité de cette étape est en $O(i_p.n^\rho)$ avec i_p, le nombre d'images annotées positivement ($0 \leq i_p \leq b$). Le système reboucle à la ligne 4 tant que l'utilisateur n'est pas satisfait de sa recherche. La complexité de notre schéma est donc en $O(n^\rho)$ et ne dépend que du paramétrage de la structure d'index utilisée pour réaliser les recherches ppv.

Chapitre VII

Evaluation de SALSAS

Dans ce chapitre, nous présentons une évaluation de la stratégie SALSAS proposée au chapitre précédent. Nous détaillons dans un premier temps les différents protocoles expérimentaux utilisés. Nous observons ensuite les gains en temps de recherche de notre méthode par rapport au système classique (cf. Sec. I.2.2.2). Enfin, nous mesurons la robustesse de notre stratégie en faisant varier les paramètres de la recherche interactive.

Sommaire

1 **Plan expérimental** 125
2 **Performances de SALSAS** 130
3 **Performance du système avec d'autres paramètres** 136
4 **Conclusion** 138

1 Plan expérimental

L'objectif principal de ces tests est double : il s'agit tout d'abord de vérifier que notre système est bien capable de retrouver les images recherchées par un utilisateur et ensuite, de vérifier que notre solution a bien une complexité de recherche sous-linéaire par rapport à la taille de la base. Nous détaillons d'abord les bases qui ont servi à effectuer nos expérimentations puis les critères d'évaluation. Enfin, nous détaillons les paramètres du système et les valeurs que nous avons utilisées pour l'obtention de nos résultats.

1.1 Bases

Le premier jeu de test utilisé pour évaluer les performances de notre méthode est la base VOC06 [EZWVG06] contenant 5 304 images réparties en 10 catégories (cf. Annexe A.1).

L'utilisation d'une vérité terrain est requise puisque les interactions humaines avec le système sont simulées pendant les sessions d'apprentissage : l'appartenance d'une image à une classe de la vérité terrain doit être connue par l'*utilisateur virtuel* afin de simuler fiablement le bouclage de pertinence. De plus, la vérité terrain est également requise pour évaluer la qualité de recherche.

Dans le but de mesurer l'évolution de la complexité des algorithmes en fonction de la taille de la base, des distracteurs sont ajoutés.

Les expérimentations ont été réalisées sur des bases de 5 000, 20 000, 60 000, 100 000 et 180 000 images. La base de 20 000 images a été obtenue en ajoutant les images de VOC07 et VOC08. La vérité terrain a été obtenue en fusionnant les catégories de même nom que VOC06. Les trois autres bases ont été obtenues en ajoutant, comme distracteurs, les images keyframe des bases TrecVid 2007, 2008 et 2009. Nous avons considéré toutes ces images comme non pertinentes (n'appartenant à aucune des 10 classes considérées).

Les signatures utilisées sont constituées de distributions de 64 couleurs et 64 textures (telles que décrites au Chap. I et utilisées pour les évaluations du Chap. III).

1.2 Critères d'évaluation

L'efficacité de notre méthode est évaluée, comme détaillé en Section I.4.1, suivant deux critères : qualité et vitesse de recherche.

La qualité de recherche est évaluée suivant la mesure du MAP (*Mean Average Precision*) des N premières images retournées par le système (cf. Annexe A.2). N est un paramètre qui doit être choisi par l'utilisateur. Dans nos évaluations nous avons fixé $N = 200$ en considérant qu'un utilisateur n'est jamais intéressé par plus de 200 images. Choisir une valeur de N plus petite aurait pour effet de diminuer les temps de recherche de notre système tout en augmentant la qualité de la recherche.

Nous évaluons la vitesse de recherche en mesurant le ratio du temps de réponse de l'approche linéaire par rapport à celui de l'approche rapide. Cette mesure permet de s'affranchir des performances de la machine utilisée.

Dans toutes nos expérimentations, les résultats ont été fournis en considérant 100 sessions de recherche par catégorie. Les expérimentations ont tout d'abord été menées en considérant la base de 5 000 images où chaque session a été initialisée en sélectionnant aléatoirement une image de la classe considérée. Les même images requêtes ont été utilisées pour toutes les autres bases.

1.3 Paramétrage

Dans ce paragraphe, nous donnons des précisions quant aux réglages des paramètres des méthodes d'apprentissage interactif. Ces systèmes possèdent un grand nombre de paramètres.

1.3.1 Sessions de recherche

Les paramètres des sessions sont les suivants :
- b annotations par itération du bouclage de pertinence.
- i itération du bouclage de pertinence par session.
- l sessions effectuées pour chaque catégorie testée.
- *pos* images annotées positivement dans la requête initiale d'une session.
- *neg* images annotées négativement dans la requête initiale d'une session.

Pour tous nos tests, le paramètre l a été fixé à $l = 100$ de manière à disposer d'un grand nombre d'échantillons pour obtenir une mesure fiable.

Nous avons testé deux jeux de paramètres. Dans le premier jeu, i et b ont été fixés à $i = 50$ et $b = 1$ afin d'observer avec précision l'évolution des comportements des systèmes. La requête initiale correspond à $pos = 1$ (et $neg = 0$) image tirée au hasard dans la catégorie recherchée.

Dans le second jeu, nous fixons $b = 5$ et $i = 25$ afin de vérifier que notre système est multi-requête. La requête initiale correspond à $pos = 3$ (respectivement $neg = 2$) images tirées au hasard dans la catégorie recherchée (respectivement dans le reste des images disponibles).

1.3.2 Paramètres internes des méthodes

Les méthodes d'apprentissage *SVM* nécessite de définir la fonction de similarité noyau k utilisée par la fonction de pertinence.

Nous avons testé deux noyaux *RBF* ($k(\mathbf{x}_i, \mathbf{x}_j) = exp(-D(\mathbf{x}_i, \mathbf{x}_j)^2/2\sigma^2)$) : le noyau l_2-*RBF*, souvent utilisé par défaut, où $D(\mathbf{x}_i, \mathbf{x}_j) = ||\mathbf{x}_i - \mathbf{x}_j||_2$ et le noyau χ^2-*RBF*, où $D(\mathbf{x}_i, \mathbf{x}_j) = \chi^2(\mathbf{x}_i, \mathbf{x}_j)$. SALSAS est définie avec le second noyau.

Ces deux noyaux requièrent de régler un paramètre supplémentaire, la valeur de σ qui permet de contrôler la largeur de la « cloche » de la gaussienne. σ est déterminée avec des tests préliminaires réalisés sur la base VOC06 en utilisant l'heuristique suivante :
- calcul du point central $\mathbf{x}_c$ de l'espace des caractéristiques,
- calcul de la distance moyenne D_m entre $\mathbf{x}_c$ et toutes les images de la base,
- σ est fixé pour que D_m soit égale à la moitié de la valeur maximale de K :

$$\sigma = D_m/2\sqrt{2\log(2)} \simeq D_m/2,35$$

.

On peut noter que l'on peut obtenir une bonne estimation de $\mathbf{x}_c$ et D_m sans utiliser toutes les données de la base. Nous avons estimé la valeur de σ à partir de 10 réalisations de 500 échantillons. On a obtenu $\sigma = 213$ pour la distance χ^2 et σ =97 503 pour la distance l_2. Les valeurs de σ ont été arrondies à 200 pour le noyau χ^2-*RBF* et à 10^5 pour le noyau l_2-*RBF*. Les heuristiques sont préférées à la *validation croisée*, en apprentissage interactif, car la taille de l'ensemble d'apprentissage est toujours très faible.

1.3.3 Paramètres propres à SALSAS

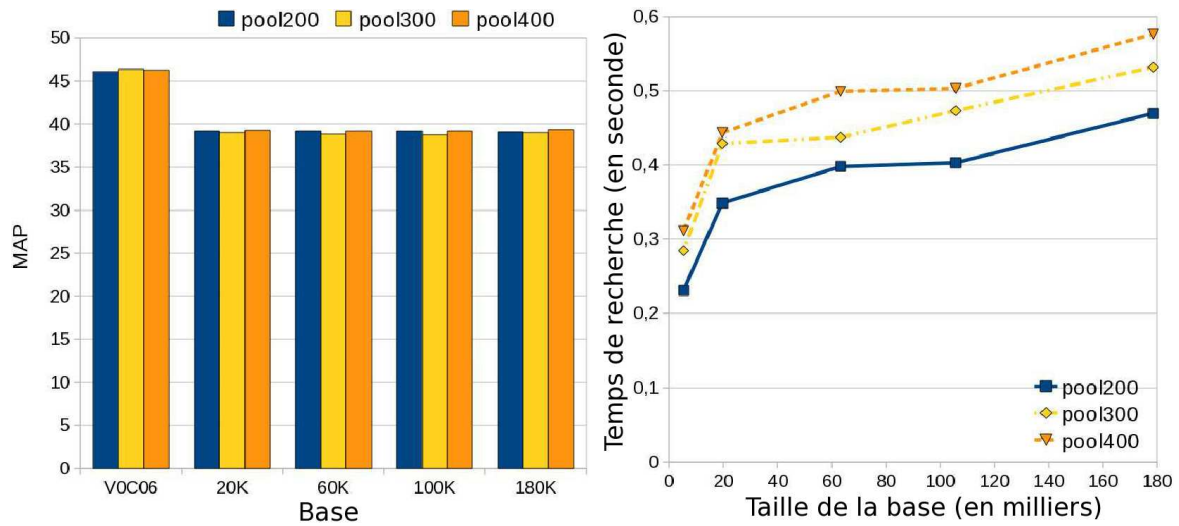
Notre algorithme de recherche rapide requiert de fixer deux autres paramètres :

– p, la taille du pool $\mathcal{S}$.

– K, le nombre de *KNN* recherchés à chaque mise à jour du pool.

Le paramètre K permet de régler le taux de renouvellement du pool $\mathcal{S}$ entre chaque itération. Nous avons fixé $K = p/2$. Nous n'avons pas testé l'influence de ce paramètre.

La taille du pool a été fixée en réalisant des tests préliminaires avec 3 valeurs de p : 200, 300 et 400. Pour ces 3 tests, nous utilisons la même structure d'index, LSH, avec son paramétrage par défaut que nous détaillons au paragraphe suivant.



(a) MAP du TOP200 à la 50$^{\text{ième}}$ itération vs taille de la base

(b) Temps de recherche à la 50$^{\text{ième}}$ itération vs taille de la base

Figure VII.1 – Influence de la taille du pool : évaluation de la qualité et de la rapidité de recherche pour 50 itérations avec 1 annotation par itération pour un pool de 200, 300 et 400 images.

Nous pouvons voir sur la Figure VII.1.(a) que doubler la taille du pool ne permet pas d'améliorer la qualité de la recherche. Ces résultats tendent donc à montrer que les motivations données en Sec-

tion VI.3.3 pour justifier de l'utilisation du pool d'images $\mathcal{S}$ pour la recherche des images à annoter sont fondées. Ainsi, un pool de petite taille peut également contenir des images incertaines capables de remettre en cause la fonction de pertinence afin de l'améliorer.

De plus, en augmentant la taille du pool, on augmente également les temps de recherche (cf. Fig. VII.1.(b)) diminuant l'efficacité de la méthode. On peut remarquer que les temps de recherche pour $p = 200$ et $p = 400$ ne sont pas doublés alors que le système évalue la pertinence de deux fois plus d'images. En effet, l'étape de plus grande complexité est la recherche des *KNN*. Or, l'algorithme *LSH* est paramétré de la même façon pour ces 3 jeux de tests. Seule la valeur de K change mais ce paramètre n'influe quasiment pas sur les temps de recherche.

D'autre part, on peut noter, à partir de la Figure VII.1.(a), que même lorsque la taille de la base augmente, augmenter la taille du pool ne permet pas d'augmenter la qualité de le recherche. Ainsi, la taille p du pool $\mathcal{S}$ et le nombre d'images $p/2$ sélectionnées pour la mise à jour de $\mathcal{S}$ peuvent rester constant quelle que soit la base considérée.

C'est pourquoi, nous avons fixé $p = 200$ et $K = 100$ dans le reste de nos expérimentations.

1.3.4 Paramètres de la recherche *KNN*

Les derniers réglages proviennent de l'utilisation de la structure d'index LSH utilisée pour effectuer les recherches *KNN* rapides (avec $K = 100$). Le paramètre principal est la largeur des tranches W utilisées pour découper l'espace. W est réglé pour correspondre au rayon de recherche minimal contenant les K images les plus proches de chaque image requête. Comme il est impossible de calculer ce rayon pour toutes les images de la base, nous utilisons la méthode de sous échantillonnage statistique présentée en Section VI.1.2 pour l'estimer.

D'après l'équation (VI.1), pour la base VOC06 contenant $n = 5304$ images, il est suffisant de sélectionner aléatoirement un ensemble $\mathcal{M}$ de taille $m = 158$ pour s'assurer avec une probabilité de 95% que l'élément de $\mathcal{M}$ le plus proche de l'image requête I_q appartienne aux 100 plus proches voisins de I_q.

Nous avons utilisé ce test pour estimer la distance minimale à une requête quelconque de la base. Cette estimation est obtenue avec 100 réalisations. Nous avons utilisé la distance au 95 percentile pour fixer W. Nous avons obtenu 387 pour la distance χ^2 et 173 520 pour la distance l_2. Nous avons alors fixé $W = 400$ pour χ^2-*LSH* et $W = 1,75 \times 10^5$ pour E^2-*LSH*.

Pour les 3 autres paramètres de LSH, nous nous sommes inspirés des valeurs par défaut données dans [LJW+07]. Nous avons fixé $L = 4$ tables de hachage, $M = 24$ projections par fonction de hachage et $T = 100$ probes.

1.4 Notations

Nous utilisons les notations suivantes pour identifier les différents algorithmes testés :

- LIN_L2 représente la méthode linéaire paramétrée avec le noyau l_2-*RBF*,
- LIN_CHI2 représente la méthode linéaire paramétrée avec le noyau χ^2-*RBF*,
- SALSAS représente notre méthode rapide paramétrée avec le noyau χ^2-*RBF* et la structure d'index χ^2-*MPLSH*,
- V1 représente notre méthode rapide paramétrée avec le noyau $ł_2$-*RBF* et la structure d'index E^2-*MPLSH*,
- V2 représente notre méthode rapide paramétrée avec le noyau χ^2-*RBF* et la structure d'index E^2-*MPLSH*,

Nous avons utilisé comme méthode linéaire l'approche de référence, SVM_{active} avec son extension : Diversité des Angles, présentée en Section I.2.2.2.

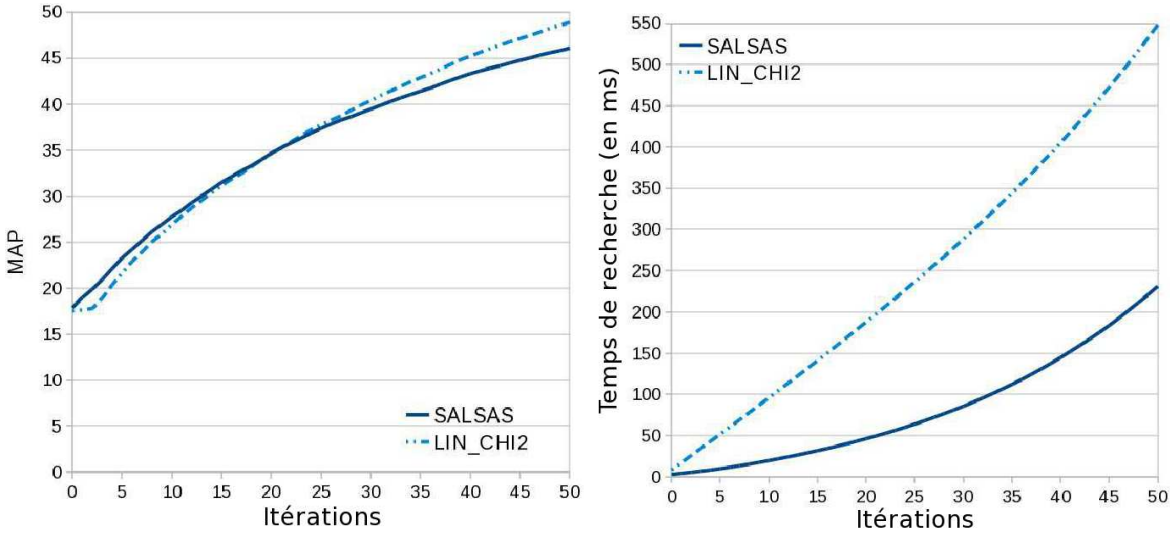
2 Performances de SALSAS

Dans ces tests, nous voulons vérifier que SALSAS offre la même qualité de recherche que les schémas classiques tout en permettant de réduire les temps de recherche.

2.1 SALSAS vs LIN_CHI2

SALSAS est tout d'abord comparée au schéma LIN_CHI2 sur la base VOC06. Comme nous pouvons le constater en Figure VII.2.(a), SALSAS fournit de meilleurs résultats que LIN_CHI2 pour les 20 premières itérations. A partir de l'itération 21, la qualité de recherche obtenue par SALSAS devient légèrement moins bonne que celle obtenue avec LIN_CHI2. La différence reste faible puisqu'à l'itération 50, SALSAS atteint une précision de 46,06 contre 48,84 de MAP pour LIN_CHI2. Il y a donc moins de 3% d'écart entre les deux méthodes. En revanche, SALSAS est plus de deux fois plus rapide que l'approche linéaire sur cette base de seulement $5K$ images. En effet, comme montré en Figure VII.2.(b), une recherche de 50 itérations prend 230 ms avec notre méthode contre 550 ms pour la méthode linéaire.

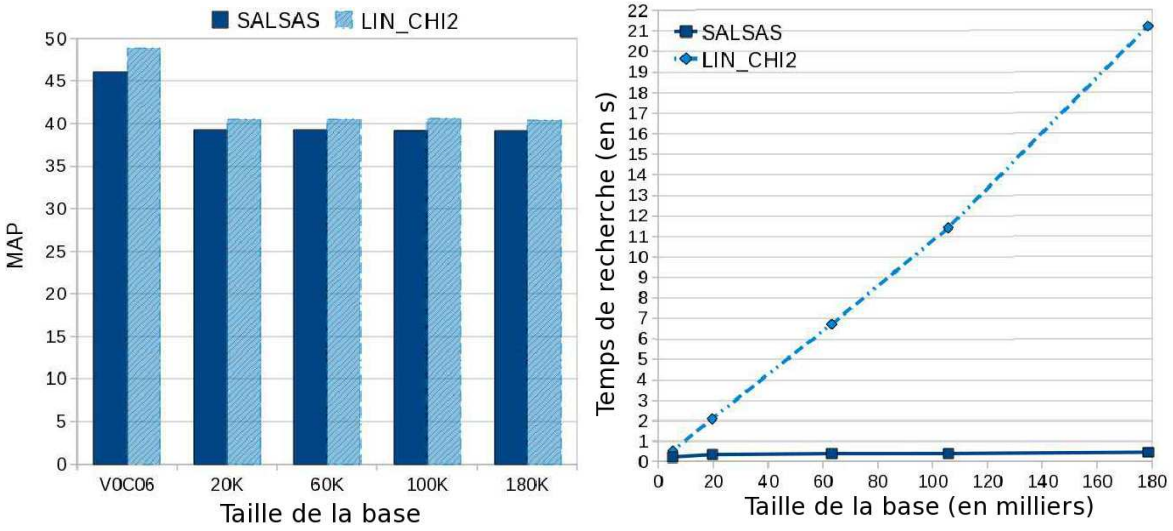
Examinons l'évolution de la qualité et de la vitesse de recherche de SALSAS lorsque la taille de la base augmente. Les résultats de MAP et de temps de recherche sur les 5 bases (cf. Sec. VII.1.1) sont reproduits sur la Figure VII.3. Les MAP de SALSAS et de LIN_CHI2 sont très proches pour l'ensemble des bases testées. Ainsi, pour ces deux méthodes, la qualité de recherche est légèrement meilleure avec la base VOC06 qu'avec la base 20K. Les images ajoutées à VOC06 pour construire



(a) MAP du TOP200 en fonction du nombre d'itérations pour la base VOC06

(b) Temps de recherche en fonction du nombre d'itérations pour la base VOC06

Figure VII.2 – Evaluation de la qualité et de la vitesse de recherche de SALSAS pour la base VOC06 en fonction du nombre d'itérations.



(a) Qualité de recherche : MAP du TOP200

(b) Rapidité de recherche : Temps de recherche

Figure VII.3 – Evolution de l'efficacité de recherche de SALSAS en fonction de la taille de la base : la mesure d'efficacité est obtenue en comparant la qualité et la rapidité de recherche entre SALSAS et LIN_CHI2. Les résultats sont obtenus avec une annotation par itération et 50 bouclages de pertinence. Les tailles de bases considérées sont comprises entre 5K et 180K images.

la base de 20K, de nature très proche des images requêtes utilisées augmentent la difficulté des recherches. Par contre, le MAP entre 20K et les 3 autres bases n'évoluent plus. Les distracteurs ajoutés

pour construire les 3 bases 60K, 100K et 180K sont de nature assez différentes avec les images requêtes utilisées n'entraînant pas une augmentation de la difficulté des recherches.

Dans un second temps, on peut observer que la diminution de précision entre la base VOC06 et 20K est plus important pour la méthode LIN_CHI2 que pour la méthode SALSAS. En effet, la différence de précision entre les deux méthodes est de 3% pour VOC06 et elle n'est plus que de $1,3\%$ pour les 4 autres bases.

Notre dispositif permet ainsi d'obtenir une qualité de recherche quasiment équivalente à la recherche linéaire. La mise à jour du pool $\mathbb{S}$ ainsi que la sélection des images à annoter dans $\mathbb{S}$ (de taille fixe) permettent donc d'obtenir un TOPN quasiment aussi précis que la recherche linéaire.

On peut observer sur la Figure VII.3.(b) que l'augmentation de la taille de la base influe très peu sur les temps de recherche de SALSAS comparativement aux temps de recherche de LIN_CHI2. En effet, pour la base de $180K$ images, SALSAS ne requiert que $0,47$ sec contre $21,21$ sec pour la méthode LIN_CHI2. SALSAS est alors 45 fois plus rapide que la recherche linéaire. Les temps de recherche de la méthode LIN_CHI2 pour la base de 180K images sont 38 fois plus lents que pour la base 5K (respectivement $0,55$ et $21,21$ sec pour $5K$ et $180K$ images) alors que les temps de recherche de la méthode SALSAS ne font que doubler (respectivement $0,23$ et $0,47$ sec pour $5K$ et $180K$ images). D'après ces tests, notre méthode présente effectivement une complexité de recherche sous-linéaire par rapport à la taille de la base.

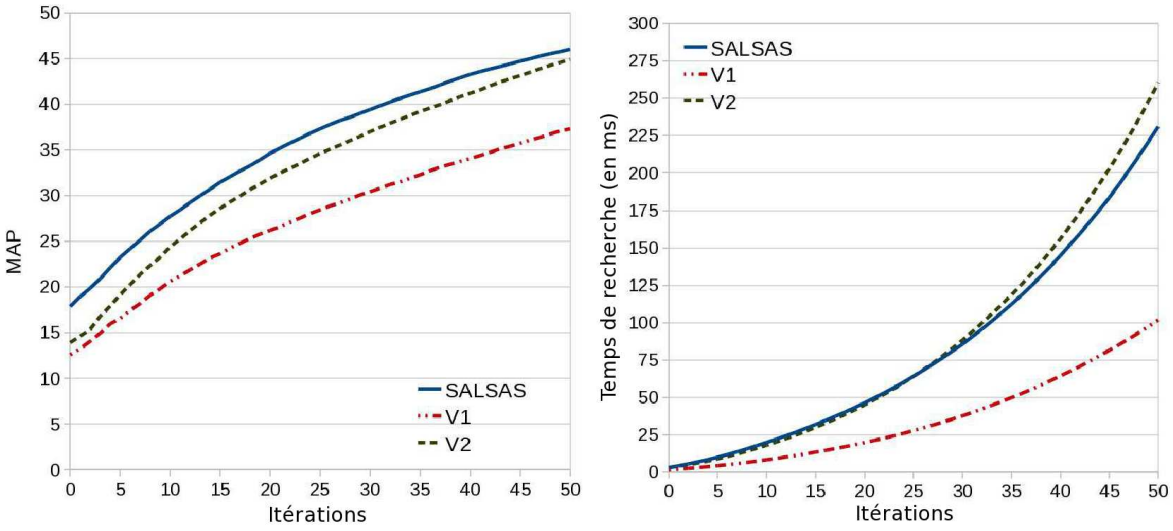
L'utilisation d'un pool $\mathbb{S}$ de taille fixe quelle que soit la taille de la base permet de rendre la complexité des étapes du calcul du TOPN et de la sélection des images à annoter de complexité indépendante de la taille de la base. La complexité de la recherche de SALSAS est donc uniquement liée à l'algorithme utilisé pour réaliser les recherches *KNN* rapides.

Nous avons donc montré que SALSAS est capable de diminuer drastiquement les temps de calcul de l'apprentissage interactif tout en conservant une qualité de recherche similaire (un exemple de session de recherche est donné en Annexe Fig. A.1).

2.2 Intérêt de χ^2-*LSH* : Optimisation de la structure d'index

Dans ces tests, nous évaluons la contribution de la structure d'index χ^2-*LSH* optimisée pour la distance χ^2. Pour ce faire, nous comparons SALSAS avec deux autres implémentations de notre schéma rapide, V1 et V2. Rappelons que, contrairement à SALSAS, V1 et V2 utilisent, pour la mise à jour du pool $\mathbb{S}$, la structure d'index E^2-*LSH* réalisant des recherches *ppv* rapides avec une distance l_2. La différence entre V1 et V2 concernent la fonction noyau utilisée par la fonction de pertinence. V1 et V2 sont paramétrées respectivement avec un noyau l_2-*RBF* et un noyau χ^2-*RBF*.

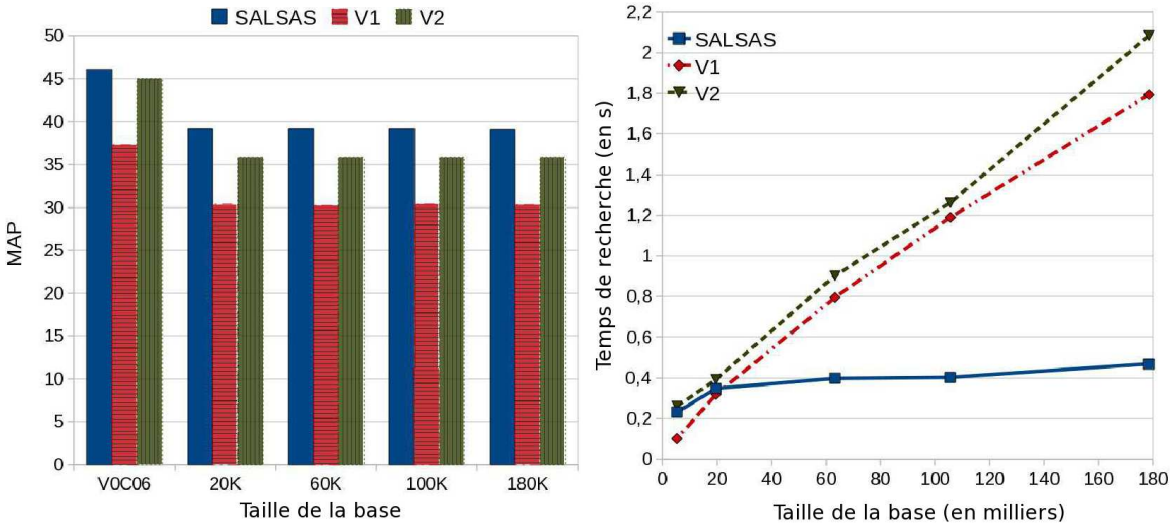
Les résultats sont présentés en Figure VII.4 et en Figure VII.5.



(a) MAP du TOP200 en fonction du nombre d'itérations du bouclage de pertinence pour la base VOC06

(b) Temps de recherche en fonction du nombre d'itérations pour la base VOC06

Figure VII.4 – Comparaison de la qualité et de la vitesse de recherche pour la base VOC06 en fonction du nombre d'itérations avec différentes implémentations de notre méthode de recherche rapide.



(a) MAP du TOP200 à la 50$^{\text{ième}}$ itération VS taille de la base

(b) Durée de recherche à la 50$^{\text{ième}}$ itération VS taille de la base

Figure VII.5 – Evolution du la qualité et de la vitesse de recherche en fonction de la taille de la base pour 50 itérations avec différentes implémentations de notre méthode de recherche rapide.

On peut observer (cf. Fig. VII.4.(a)) que la méthode V2 est plus précise que la méthode V1. La combinaison d'une recherche *ppv* avec une distance l_2 pour la mise à jour du pool et un noyau χ^2-*RBF* n'est donc pas absurde. Cependant, elle ne permet pas d'atteindre les même performances que la

méthode SALSAS. La structure d'index χ^2-*LSH* présentée au Chapitre III reste donc très pertinente dans ce contexte de recherche interactive.

Concernant les temps de recherche, la Figure VII.4.(b) montre un avantage pour la méthode V2. Les méhtodes SALSAS et V2 sont en effet environs 2 fois plus lentes que V1 pour la base de plus petite taille, VOC06. En effet, le calcul de la distance χ^2 est plus complexe que celui de la distance l_2, ralentissant la structure d'index χ^2-*LSH* et donc la méthode SALSAS. Une des raisons pour laquelle la méthode V2 est plus lente que V1, alors qu'elles utilisent les même structures d'index, est que V2, contrairement à V1, ne peut pas utiliser la distance l_2 fournie à la sortie de la structure d'index E^2-*LSH* pour calculer le noyau χ^2-*RBF* car les distances sont différentes. C'est également pour cette raison que V2 est plus lente que SALSAS.

Cependant, en augmentant la taille de la base (cf. Fig. VII.5), l'avantage du point de vue rapidité de la méthode V1 disparaît. SALSAS devient aussi rapide que V1 dès la base de $20K$ images et elle devient même plus rapide que V1 pour les bases de plus grandes tailles. Pour la base de $180K$ images, les 50 itérations prennent 0.47 sec pour SALSAS contre 1.80 sec pour V1. Ceci tend à montrer que le découpage de l'espace utilisé par la structure d'index χ^2-*LSH* est mieux adapté à la distribution des données que celui du schéma E^2-*LSH*.

2.3 Analyse détaillée des résultats

categories	LIN CHI2	SALSAS
vélo	44,38	45,28
bus	30,33	29,74
voiture	81,59	86,73
chat	20,51	18,38
vache	42,66	36,02
chien	17,40	17,83
cheval	19.87	20,74
moto	31,12	30,36
personne	62,04	64.9
mouton	54,34	40,98
Moyenne	40,42	39,10

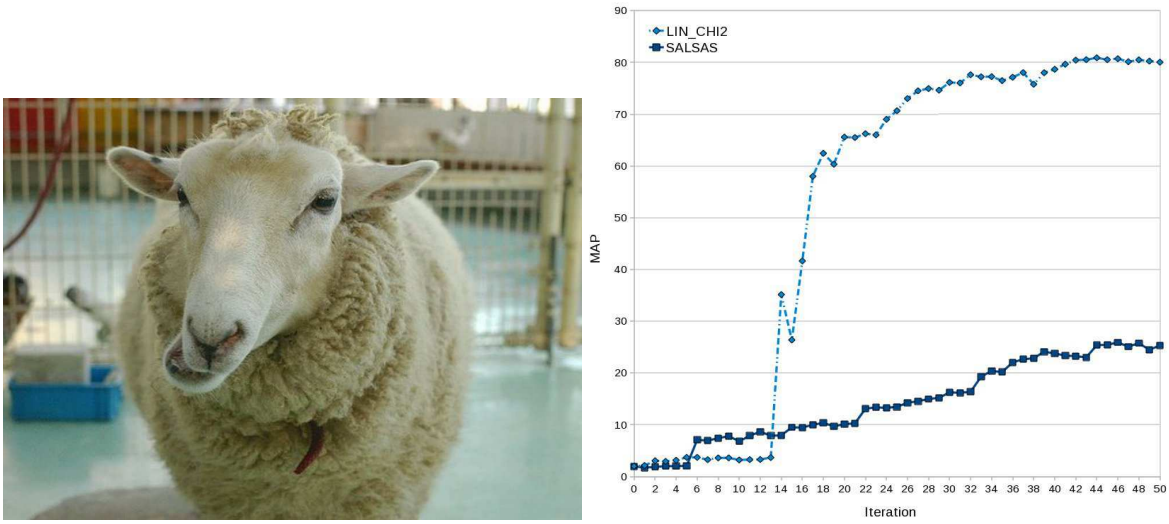
TAB. VII.1 – Détail du MAP du TOP200 des 10 catégories considérées à l'itération 50 du bouclage de pertinence sur la base de $180K$ images pour la méthode SALSAS.

Dans le Tableau VII.1, nous détaillons le MAP pour chacune des 10 catégories testées sur la base $180K$. Pour sept des dix catégories, la précision de SALSAS est similaire à celle de la méthode linéaire. En effet, la différence de MAP n'excède pas 2%. Cependant, pour les catégories *voiture*,

vache et *mouton*, on observe un écart de MAP de plus de 5%. Notre méthode atteint une meilleure précision de recherche pour la classe *voiture* mais les performances sont assez dégradées pour les classes *vache* et *mouton*.

Pour une grande proportion d'images d'une de ces deux classes, un mode est très bien localisé dans l'espace des caractéristiques. En revanche, le reste des images de la classe est dispersé dans l'espace. Retrouver le cœur de classe à partir d'une des images du mode est donc plus facile qu'à partir d'une image isolée.

Comme l'illustre la Figure VII.6, quand une image isolée est utilisée pour initialiser la recherche, le système d'apprentissage interactif demande plusieurs itérations avant de présenter à l'utilisateur des images permettant d'améliorer la recherche (10 itérations sont nécessaires pour l'exemple de la Fig. VII.6).



(a) Exemple d'image requête très spécifique

(b) MAP du TOP200 en fonction du nombre d'itération de l'image exemple avec la base 180K

Figure VII.6 – Illustration de la limitation de SALSAS lorsque les images requêtes représentent une modalité très spécifique de la catégorie d'objets recherchée

Pendant ces premières itérations, le système explore une grande partie de l'espace avant de trouver une image appartenant au mode principal de l'espace des caractéristiques (*mouton* dans un pré). Une fois cette image trouvée, l'accès à un grand nombre d'image pertinentes est facilité. Le MAP augmente alors très significativement.

Notre méthode, en se concentrant uniquement autour des images labélisées positivement, contraint l'exploration. Il est alors possible que lorsque le système est initialisé avec une image requête trop difficile (représentant une modalité très spécifique de la classe d'objet recherchée), il ne puisse pas explorer suffisamment pour trouver les autres modalités.

3 Performance du système avec d'autres paramètres

Dans ces tests, nous évaluons la robustesse de notre stratégie en modifiant les paramètres du système. Dans un premier temps, nous faisons varier les paramètres des sessions de recherche. Dans un second temps, nous vérifions que notre stratégie est toujours valide avec d'autres fonctions noyaux.

3.1 Multi-requête

Nous avons testé notre système en faisant varier aussi bien le nombre d'images requêtes que le nombre d'annotations par itération dans le but d'évaluer l'influence de ces paramètres. Les résultats avec 5 images requêtes (3 images labélisées positivement et 2 négativement) et 5 annotations par itération sont présentés sur les Figures VII.7 et VII.8.

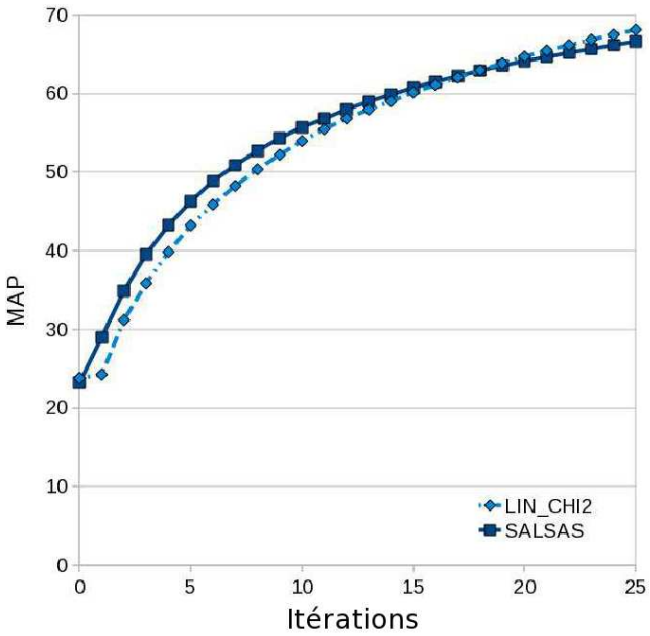
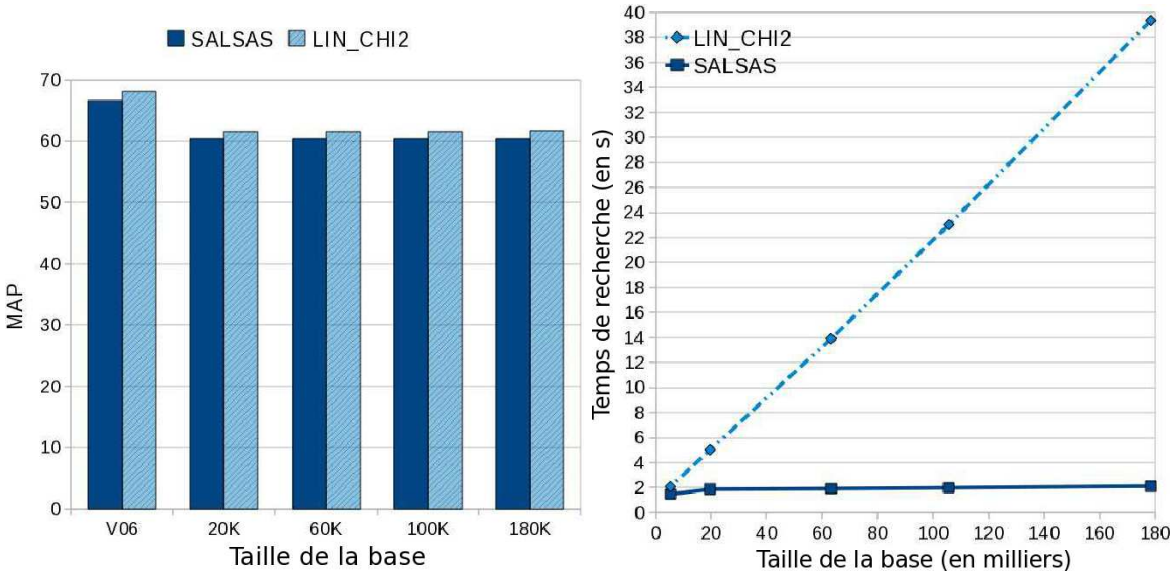


Figure VII.7 – Système multi-requêtes : évolution du MAP du TOP200 en fonction des itérations sur la base VOC06 avec 5 annotations par itération et 5 images requêtes.

Comme nous pouvons le voir sur ces figures, la précision de SALSAS est similaire à l'approche linéaire. En effet, en augmentant le nombre d'images requêtes, le cas présenté en Figure VII.6 a moins de chance de se produire, ce qui rapproche la précision de SALSAS de celle de la méthode linéaire.

D'autre part, ces résultats montrent que notre système est capable de sélectionner, parmi les images du pool $\mathcal{S}$, des images aussi pertinentes à faire annoter par l'utilisateur que la méthode linéaire qui explore toutes les données. Enfin, la mise à jour du pool $\mathcal{S}$ reste également pertinente malgré une



(a) MAP du TOP200 à la 25$^{\text{ième}}$ itération vs taille de la base

(b) Temps de recherche à la 25$^{\text{ième}}$ itération vs taille de la base

Figure VII.8 – Système multi-requêtes : évolution de la précision et de la rapidité de recherche en fonction de la taille de la base pour 25 itérations avec 5 images requêtes et 5 annotations par itération.

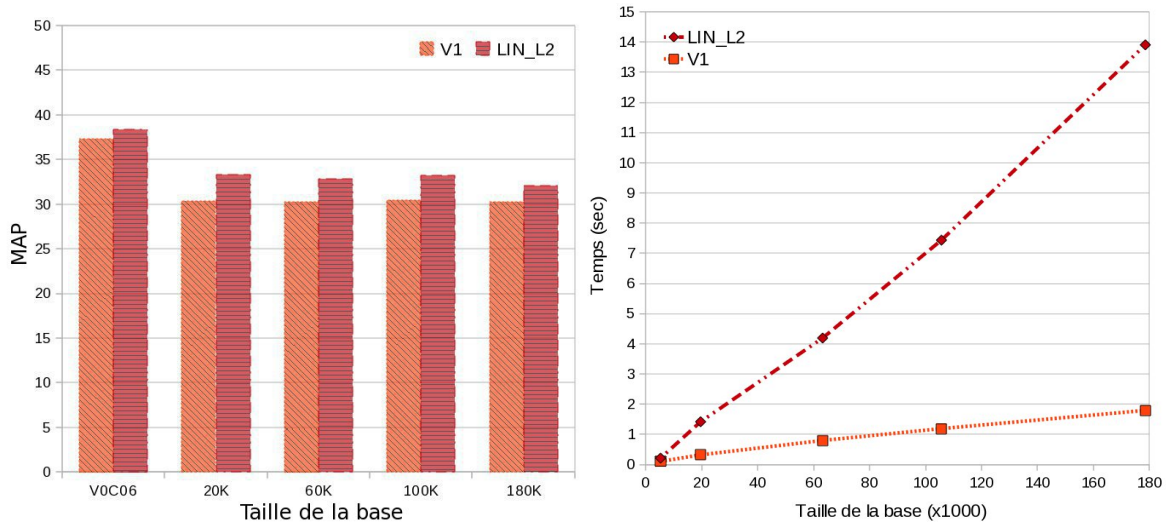
modification plus importante de la fonction de décision $f_{\mathcal{A}}$ due à l'augmentation plus importante de l'ensemble d'apprentissage.

Comme montré sur la Figure VII.8.(b), bien que les temps de recherche de SALSAS augmentent avec l'accroissement du nombre d'annotations par itération, notre système reste 20 fois plus rapide que l'approche linéaire pour 5 annotations par itération. D'autre part, la complexité de la recherche reste sous-linéaire par rapport à la taille de la base puisque les temps de recherche évoluent très peu entre les bases de 20K et 180K images.

3.2 Système avec un noyau l_2-*RBF*

Dans ces tests, nous comparons la méthode V1 avec LIN_L2.

La Figure VII.9 compare les MAP et les temps de recherche de V1 et LIN_L2 en fonction de la taille de la base. Comme on peut le constater, notre méthode obtient une qualité de recherche proche de la recherche linéaire tout en diminuant les temps de recherche. Cependant, on observe une dégradation de la qualité de recherche plus importante entre V1 et LIN_L2 (entre 3 et 10%) qu'entre SALSAS et LIN_CHI2 (entre 3 et $1,3\%$). Notre méthode a toujours une complexité de recherche sous-linéaire. En effet, les gains en temps de recherche augmentent avec la taille de la base. V1 est 2 fois plus rapide pour la base de $5K$ images et 7,8 fois plus rapide pour la base de $180K$ images.



(a) MAP du TOP200 à la 50$^{\text{ième}}$ itération VS taille de la base

(b) Durée de recherche à la 50$^{\text{ième}}$ itération VS taille de la base

Figure VII.9 – Evolution du la qualité et de la vitesse de recherche de la méthode V1 et de LIN_L2 en fonction de la taille de la base pour 50 itérations.

Cependant, ces gains sont bien moindres que pour SALSAS.

Notre stratégie est donc validée sur un autre noyau. Ce test montre également que la structure d'index χ^2-*LSH* est mieux adaptée que E^2-*LSH* dans notre contexte. En effet, les recherches sont alors plus rapides et les images retrouvées correspondent au mieux aux critères de recherche. Le pool d'images est alors remis à jour plus rapidement et avec des images plus pertinentes.

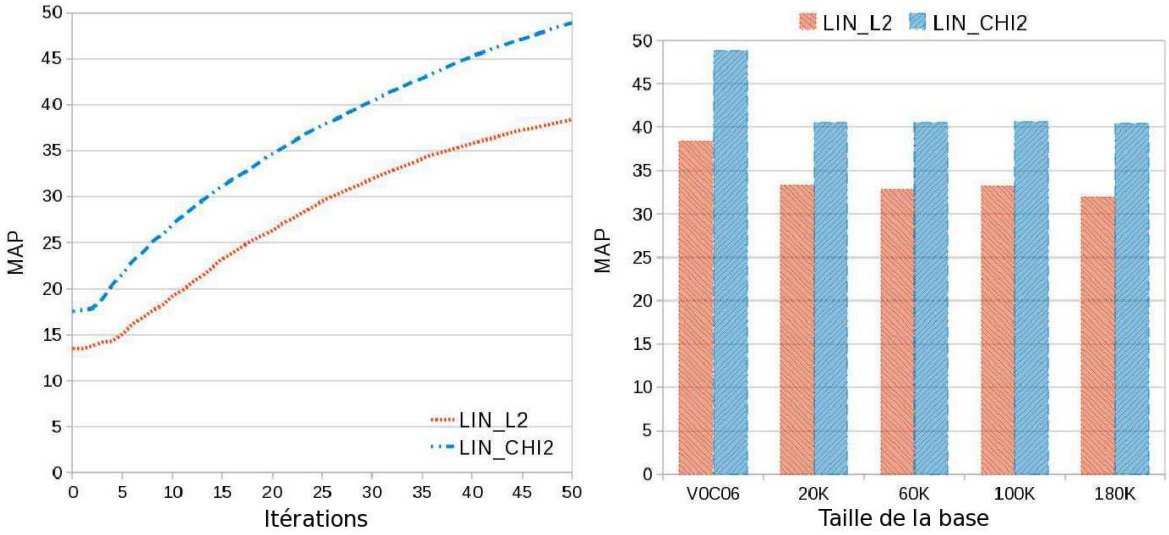
3.3 Intérêt du χ^2 : comparaison des méthodes LIN_L2 et LIN_CHI2

Comme nous venons de le voir, notre stratégie est légèrement moins efficace avec le noyau l_2-*RBF* qu'avec le noyau χ^2-*RBF*. C'est pourquoi nous proposons une comparaison de ces deux noyaux.

Comme le montre la Figure VII.10, la distance χ^2 améliore grandement la recherche pour chaque itération et sur l'ensemble des bases considérées. Au bout de la 50$^{\text{ième}}$ itération, on observe une différence de MAP entre 7% et 10%. Dans ce contexte, la distance χ^2 permet d'obtenir une fonction de pertinence plus robuste et elle facilite également l'apprentissage.

4 Conclusion

Les expériences présentées dans ce chapitre ont permis de valider la stratégie proposée au chapitre précédent sur plusieurs points. Nous avons montré que SALSAS offre une qualité de recherche simi-



(a) MAP du TOP200 en fonction du numbre d'itérations pour la base VOC06

(b) MAP of TOP200 à la $50^{\text{ième}}$ iteration pour les bases 5K, 20K, 80K, 100K et 180K

Figure VII.10 – Comparaison de la quatité de recherche de l'approche linéaire entre le noyau χ^2-*RBF* et le noyau l_2-*RBF*

laire à l'approche de référence [TC01] tout en permettant de réduire de façon importante les temps de calcul de la recherche interactive. Ensuite, nous avons confirmé l'intérêt de la distance χ^2 et de notre structure d'index χ^2-*LSH*. Enfin, nous avons montré que SALSAS gère efficacement les requêtes multiples.

Conclusion de la Partie III

L'apprentissage interactif est une solution très pertinente pour combler le fossé sémantique/numérique. Cependant, il n'est pas bien adapté aux grandes bases d'images car les étapes de sélection des images à annoter et de tri de la base ont généralement une complexité importante par rapport à la taille de la base (linéaire ou plus).

Après avoir présenté un état de l'art des méthodes s'attaquant à ce problème, nous avons proposé une stratégie nommée SALSAS. Elle est basée sur l'utilisation d'une structure index efficace, LSH, qui permet de sélectionner rapidement un pool d'images pertinentes afin d'accélérer aussi bien la sélection des images à annoter que le tri de la base. Une stratégie a également été proposée pour mettre à jour rapidement ce pool d'images à chaque bouclage de pertinence.

Les résultats des expérimentations menées sur des grandes bases d'images ont montré que notre algorithme permet d'atteindre la même précision que la méthode de référence, l'approche de Tong [TC01] avec la diversité des angles, tout en réduisant les temps de recherche d'un facteur 45. Une recherche en ligne de 50 itérations pour une base de 180 000 images demande $0,47$ pour notre méthode contre 21 *sec* classiquement.

Chapitre VIII

Conclusion générale et perspectives

La recherche d'images par le contenu est une problématique majeure pour la gestion des grandes bases d'images actuelles. En effet, avec l'augmentation de la taille des bases d'images, l'intégration de structures d'index permettant d'obtenir des systèmes de recherche de complexité sous-linéaire par rapport à la taille des bases s'impose. Dans cette thèse, nous nous sommes intéressés au passage à l'échelle des méthodes de recherche automatique et interactive de catégories.

1 Bilan

Nous avons traité dans une première partie le passage à l'échelle de la recherche automatique de catégories lorsque les images sont indexées avec des signatures globales de type Bag-of-Words.

Nous nous sommes intéressés aux structures d'index de type hachage et plus particulièrement à la famille de méthodes LSH. Nous avons proposé une nouvelle fonction de hachage qui permet d'effectuer des recherches rapides avec des fonctions de similarité utilisant la distance χ^2. Cette distance offre une meilleure qualité de recherche pour les signatures Bag-of-Words que les distances usuelles telles que la distance l_2. Les structures d'index de type LSH nécessitent beaucoup de ressources mémoires pour être efficaces ce qui restreint leur utilisation aux bases de l'ordre de 100 000 images. Nous avons cependant proposé un schéma de recherche utilisant peu de ressources mémoires en s'inscrivant dans le schéma d'optimisation Multi-Probe. Notre méthode permet d'envisager la recherche automatique d'images dans des bases de l'ordre de 1 à 10 millions d'images.

Dans une seconde partie, nous avons traité le passage à l'échelle de la recherche automatique de catégories lorsque les images sont indexées avec des signatures locales de types sacs de signatures,

Bag-of-Features. Les méthodes de recherche d'images basées sur l'utilisation de Bag-of-Features suscitent actuellement un engouement important car elles considèrent des descriptions locales ou partielles des images et sont donc plus adaptées à la recherche de catégories d'objets. Cependant, cette représentation complexe des données nécessite, pour comparer les signatures, des traitements coûteux qui sont incompatibles avec l'accroissement de la taille des bases de données. Alors que les solutions actuelles compressent l'information contenue dans ces signatures, nous avons défini de nouvelles fonctions de similarité basées sur des noyaux sur sacs qui permettent d'exploiter toute l'information fournie par cette représentation. Afin de réduire la complexité de cette approche, nous avons proposé d'approximer les fonctions noyaux. Cette approximation est obtenue en considérant uniquement les signatures les plus pertinentes. Cette sélection est réalisée très rapidement grâce à une structure d'index telle que LSH. Bien que cette solution réduise grandement les temps de recherche, le coût en ressources mémoires des signatures rend son utilisation limitée aux bases allant jusqu'à 1 million d'images. Un effort tout particulier a également été fait sur le formalisme qui permet d'unifier les approches Bag-of-Features et Bag-of-Words.

Le dernier point que nous avons traité en partie III concerne le passage à l'échelle de la recherche interactive de catégories. Nous avons considéré une approche de classification active qui combine la classification binaire et l'apprentissage actif. Le rôle de la classification binaire est d'apprendre, à partir d'un ensemble d'exemples, une fonction de pertinence capable de discriminer la classe des images recherchées de la classe des autres images. Pour retrouver les images recherchées, il est ensuite nécessaire d'évaluer la pertinence de chaque image de la base à l'aide de cette fonction. Cette étape a alors une complexité linéaire par rapport à la taille de la base ce qui en fait la première étape critique dont il faut diminuer le coût pour obtenir un algorithme de recherche de complexité sous-linéaire. Le but de l'apprentissage actif est de construire itérativement l'ensemble d'apprentissage (*i.e.* bouclage de pertinence) en sélectionnant les images à ajouter à cet ensemble, c'est à dire en sélectionnant les image à faire annoter par l'utilisateur de façon à optimiser la classification à chaque itération. Cette sélection demande également d'examiner toutes les images de la base, ce qui en fait la seconde étape critique. Afin de définir une méthode de recherche de complexité sous-linéaire, nous avons proposé une heuristique utilisant un algorithme de recherche des K-plus proches voisins (K-ppv) pour créer rapidement un pool d'images de taille fixée. On peut alors restreindre l'examen des images à ce pool aussi bien pour l'étape de détection des images recherchées que pour l'étape de sélection des images à annoter sans dégrader la qualité de la recherche. Une stratégie est également proposée pour mettre à jour rapidement ce pool d'images à chaque bouclage de pertinence. Le système de recherche proposé a une complexité qui dépend uniquement de l'algorithme K-ppv. En utilisant une structure d'index de type LSH capable d'effectuer des recherches K-ppv avec une complexité sous-linéaire, on s'assure donc d'obtenir un système de recherche interactif qui passe à l'échelle. Ce système compatible avec

le schéma χ^2-*LSH* a permis de montrer une nouvelle fois l'intérêt de notre approche.

2 Perspectives

Dans cette thèse, nous avons développé des approches combinant la richesse des fonctions noyaux et l'efficacité de structures d'index telles que Locality-Sensitive Hashing (LSH) pour permettre le passage à l'échelle des méthodes de recherche d'images par le contenu. Ces travaux ouvrent de nombreuses perspectives dont :

- à court terme :
  - Accélérer la recherche interactive en utilisation le schéma LSH proposé par Kulis *et al.* [KG09] qui permet d'effectuer des recherches rapides directement dans l'espace induit.
  - L'extension de notre système d'apprentissage interactif rapide aux signatures de type BoF en nous inspirant de notre approximation des noyaux sur sac $UFast_K$.
  - L'intégration d'une représentation compressée des données (type VLAD de Jégou *et al.* [JDSP10]) qui nous permettra d'étendre notre approche à la recherche interactive dans des bases de plusieurs dizaines de millions d'images. L'enjeu étant ici d'analyser finement la perte issue de cette représentation compressée et d'étudier des méthodes de compression concurrentes pour ne pas perdre le bénéfice, que nous avons mis en lumière dans nos travaux, d'élaborer des fonctions de similarités noyaux adaptées aux grands volumes de données.
  - La réduction du nombre de PoI utilisé pour la description BoF afin d'étendre l'utilisation de $UFast_K$ à des bases de plus grandes tailles.
- à moyen terme :
  - L'extension à la distance l_2 du schéma LSH proposé par Kulis *et al.* [KG09] qui permet de réaliser des recherches rapides directement dans les espaces induits implicites est une piste intéressante. En effet, si ce schéma repose sur l'introduction d'une fonction noyau à la place du produit scalaire dans l'approche de Charikar *et al.* [Cha02], la principale difficulté réside en la construction d'un vecteur aléatoire de l'espace induit, utilisé pour partitionner l'espace. En utilisant la solution proposé dans [KG09] et la relation $k(\mathbf{x},\mathbf{x})+k(\mathbf{y},\mathbf{y})-2k(\mathbf{x},\mathbf{y}) = ||\phi(\mathbf{x})-\phi(\mathbf{y})||_2^2$, on pourrait envisager d'étendre la structure d'index E^2-*LSH* à la recherche rapide dans les espaces induits implicites lorsque les noyaux sont de norme unitaire.
  - Une seconde solution pour réaliser des recherches rapides directement dans les espaces induits implicites pourrait être proposée en se basant sur la méthode KDX [PC05]. Rappelons que l'idée fondatrice de cette méthode consiste à créer une partition de l'espace induit à partir d'un point d'ancrage. On pourrait améliorer cette technique en multipliant le nombre de par-

titions (avec des points d'ancrage différents). Lors de la recherche, seule la partition dont le point d'ancrage est le plus représentatif pourrait être utilisée. On pourrait également utiliser cette méthode pour construire une table de hachage. Chaque point d'ancrage pourrait permettre de construire une fonction de hachage. La partition obtenue serait issue de plusieurs découpages de l'espace induit.

- à plus long terme, même si notre stratégie active a prouvé son efficacité dans nos expérimentations, elle fait intervenir un certain nombre d'heuristiques que nous pourrions analyser conjointement du point de vue du cadre combinatoire proposés dans Scholokopf *et al.* [SS02] et du point de vue du cadre probabiliste de LSH afin de démontrer théoriquement la pertinence de notre méthode.

Les systèmes de recherche d'images par similarité proposés aujourd'hui sont capables de traiter de l'ordre de 10 millions d'images. Or, certaines des bases d'images évoquées en introduction contiennent plusieurs milliards d'images et continuent de croître rapidement. L'effort de compréhension et de recherche pour traiter ces bases reste un problème ouvert.

Chapitre IX

Annexes

A.1 Bases d'évaluation

Nous présentons dans ce paragraphe les bases de références utilisées pour évaluer les systèmes CBIR. On distingue deux types de bases, les bases de détection de copies et les bases de catégories.

Les bases de détection de copies sont composées d'un ensemble d'images requêtes et d'une base de test contenant les images à retrouver. La vérité terrain est définie par l'ensemble des images de la base test à retrouver pour chaque image requête.

Les bases de catégories sont composées d'un ensemble de classes, d'une base d'apprentissage et d'une base de test. La base d'apprentissage est utilisée pour construire un détecteur pour chaque classe. Les détecteurs sont évalués sur la base de test en utilisant la vérité terrain qui définit la classe à laquelle appartient chaque image.

Parmi les bases de références utilisées par les systèmes proches de la détection de copies, on distingue la base Holidays [JDS08b] contenant 1 491 images et la base Oxford5K [PCI+07] contenant 5 062 images.

Holidays [JDS08b] : La base Holidays (cf. Fig. A.1) contient des photos de vacances de 500 lieux touristiques prises sous différents angles de vues. Une image de chaque lieu est utilisée comme image requête. Le système de recherche doit retrouver parmi les 991 images tests de la base, celles qui ont été pris au même endroit. Pour la majeure partie des requêtes, seules une à deux images sont pertinentes.

Oxford5K [PCI+07] : La base Oxford5K est un ensemble de 5 062 images représentant 11 mo-

Figure A.1 – Exemple d'images de la base Holidays

numents historiques de la ville d'Oxford. Ces images ont été extraites du site de partage de photo Flickr[1]. 55 images sont définies comme images requêtes. La base contient 100 images pertinentes pour chaque image requête. Le protocole évalue la capacité du système à retrouver 10 bonnes images par requête.

Parmi les bases de référence utilisées par les systèmes de catégorisation, on distingue la base COREL (utilisée dans [CHV99]) et les bases utilisées pour les challenges PASCAL[2] et TrecVid[3].

COREL : COREL (cf. Fig. A.2) est une base commerciale de 20 000 images réparties en 200 catégories. Un grand nombre de catégories étant trop connectées, l'utilisation de toute la base ne permet pas d'évaluer de façon pertinente les systèmes de recherche. C'est pourquoi, chaque utilisateur s'est construit sa propre base en ne sélectionnant que certaines catégories. Ainsi dans [CHV99], les auteurs ont construit une base de 1 400 images en considérant uniquement 14 catégories. Nous avons utilisé COREL pour construire une base, nommée COREL154, de 154 catégories contenant 15 246 images.

VOC [EZWVG06] : Le challenge PASCAL est une compétition annuelle internationale de catégorisation d'images qui a commencé en 2005. Chaque année, une nouvelle base est proposée aux participants. Nous avons utilisé les bases de 2006, 2007 et 2008 que nous avons respectivement appelées VOC06, VOC07 et VOC08. La base VOC06 (cf. Fig. A.3) contient 5 304 images réparties en

[1] http ://www.flickr.com/
[2] http ://pascallin.ecs.soton.ac.uk/challenges/VOC/
[3] http ://trecvid.nist.gov/

Figure A.2 – Exemple d'images de la base COREL

10 catégories (voiture, mouton, vache, personne, moto, ...). VOC07 et VOC08 contiennent respectivement 9 963 et 4 340 images réparties en 20 catégories (10 catégories ont été ajoutées à celles de VOC06).

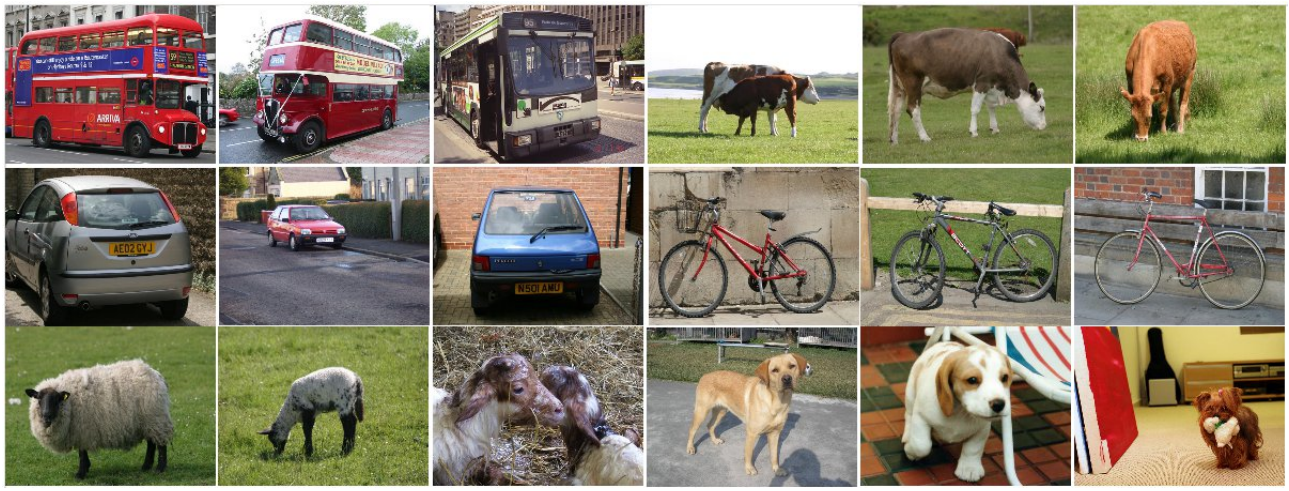


Figure A.3 – Exemple d'images de la base VOC06

TrecVid [SvdSdR+09] : Le challenge TrecVid est une compétition annuelle internationale de catégorisation de vidéos qui a commencé en 2001. Chaque vidéo peut être représentée par une image caractéristique appelée keyframe. Ces bases vidéos sont alors traitées comme des bases d'images. Nous avons utilisé les bases TrecVid 2007, TrecVid 2008 et TrecVid 2009 qui sont issues de 400 heures d'archives vidéos fournies par « The Netherlands Institute for Sound and Vision ». TrecVid

2007 et TrecVid 2008 sont représentées par 40 000 images chacune et TrecVid 2009 par 80 000 images. Nous avons compilé ces trois bases pour former une base de 160 000 images que nous avons appelée TrecVid.

A.2 Mesure de performances

Pour évaluer les systèmes de recherche, on utilise les mesures de précision, de rapidité et de qualité définies en Introduction I.4.1.

Dans nos évaluations nous avons utilisé la mesure du MAP (*Mean Average Precision*) des N premières images retournées par le système pour évaluer la qualité de la recherche. Le MAP du TOPN est calculé de la façon suivante : Soit le TOPN défini par $R^N = \{r_1, r_2, \ldots, r_N\}$, l'ensemble des N images triées renvoyé par le système et C, l'ensemble des images pertinentes de la base pour la classe considérée. Pour chaque image j de R^N, soit $|C \cup R^j|$ le nombre d'images pertinentes parmi les j premières images renvoyées par le système. Pour chaque image requête, la précision des N premières images retournées (AP_N) est évaluée :

$$AP_N = \frac{1}{N} \sum_{j=1}^{N} \frac{|C \cup R^j|}{j} \Delta(r_j)$$

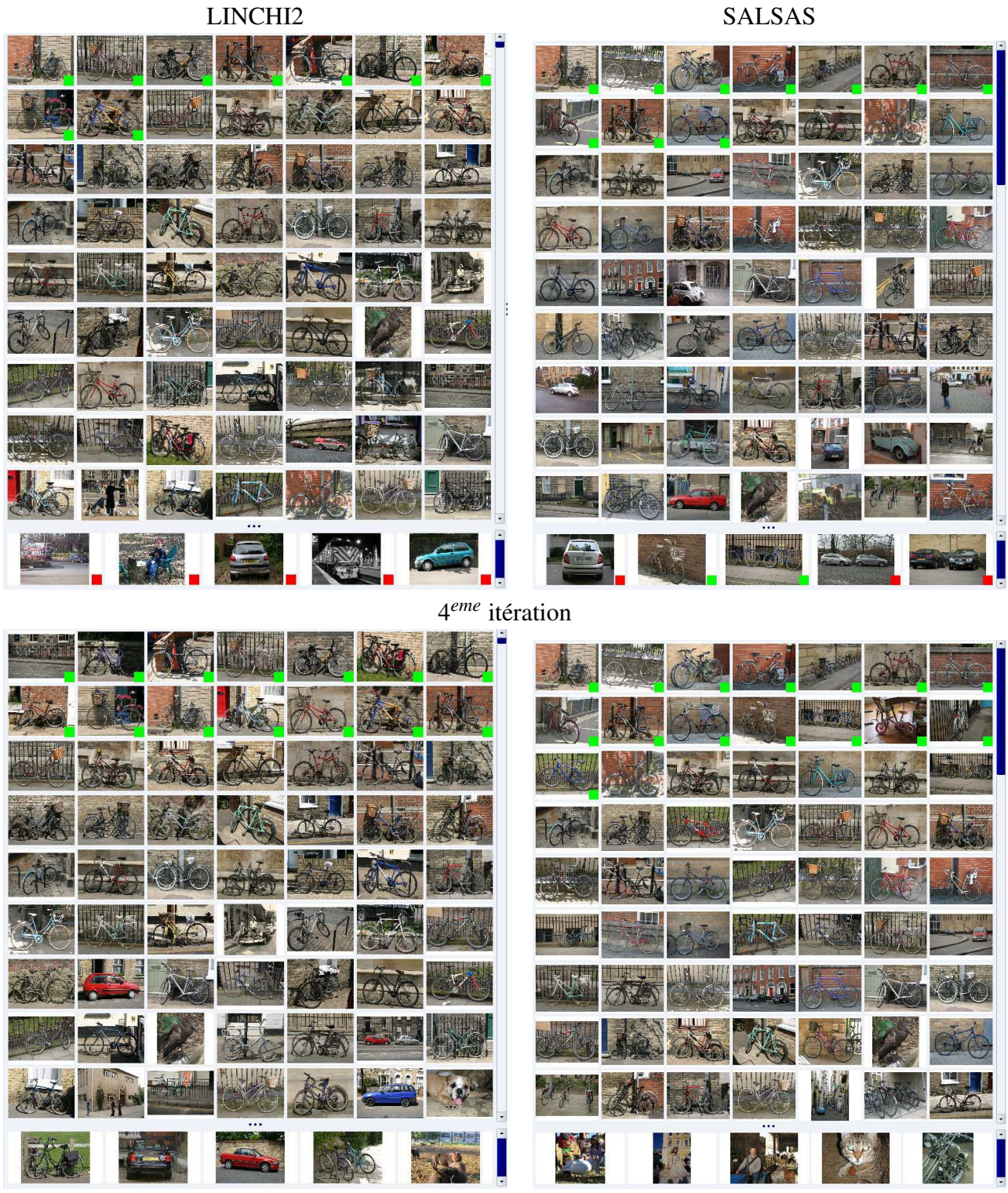
avec $\Delta(r_j) = 1$ si $r_j \in C$ et 0 autrement. Le MAP est ensuite obtenu en moyennant les valeurs de précision pour toutes les requêtes. Lorsque la base est divisée en catégories, un MAP est calculé par catégorie puis un MAP global est obtenu en moyennant les MAP de chaque catégorie. Dans le manuscrit, le MAP est présenté en pourcentage.

A.3 Exemple de session de recherche interactive

LINCHI2 SALSAS

Initialisation avec 1 image requête de vélo

Première itération



TAB. A.1 – Comparaison entre LINCHI2 et SALSAS pour une session de recherche d'une bicyclette. Les sessions sont initialisées avec une image requête. 5 annotations par itération de recherche. Partie supérieure : résultat de la recherche. Partie inférieure : images sélectionnées par l'apprentissage actif. Carré vert : image annotée positivement. Carré rouge : image annotée négativement.

Publications de l'auteur

JOURNAUX INTERNATIONAUX

1. D. Gorisse, M. Cord, F. Precioso, *SALSAS : Sub-linear Active Learning Strategy with Approximate k-NN Search*, Pattern Recognition, *accepted*, 2010.

2. E. Valle, M. Cord, S. Philipp-Foliguet, D. Gorisse, *Indexing personal image collections : A flexible, scalable solution*, Information Sciences, *in press*, 2010.

CONFÉRENCES INTERNATIONALES

1. D. Gorisse, M. Cord, F. Precioso, *Scalble active learning strategy for object category retrieval*, In Proceedings of IEEE International Conference on Image Processing (ICIP 10), Hong Kong, Chine, Septembre 2010.

2. D. Gorisse, M. Cord, F. Precioso, *Optimization on active learning strategy for object category retrieval*, In Proceedings of IEEE International Conference on Image Processing (ICIP 09), Le Caire, Egypte, novembre 2009.

3. B. Delezoide, H. Le Borgne, P.A. Moëllic, D. Gorisse, F. Precioso, F. Wang, B. Merialdo, P. Gosselin, L. Pellerin and others, *IRIM at TRECVID2009 : High Level Feature Extraction*, TREC Video Retrieval Evaluation Online Proceedings (TRECVID), 2009.

4. D. Gorisse, M. Cord, F. Precioso, S. Philipp-Foliguet, *Fast Approximate Kernel-Based Similarity Search for Image Retrieval Task*, In Proceedings of the 19th International Conference on Pattern Recognition (ICPR 08), Tampa, Florida, USA, aout 2008.

5. D. Gorisse, F. Precioso, M. Cord, S. Philipp-Foliguet, *Summarization scheme based on near-duplicate analysis*, In Proceedings of the 2nd ACM TRECVid Video Summarization Workshop (TVS 08), Vancouver, British Columbia, Canada, Octobre 2008.

6. G. Quénot, J. Benois-Pineau, B. Mansencal, E. Rossi, M. Cord, F. Precioso, D. Gorisse, P. Lambert, B. Augereau, L. Granjon and others, *Rushes summarization by IRIM consortium : redundancy removal and multi-feature fusion* In Proceedings of the 2nd ACM TRECVid Video Summarization Workshop (TVS 08), Vancouver, British Columbia, Canada, Octobre 2008.

7. D. Gorisse, M. Jordan, S. Philipp-Foliguet, F. Precioso, *3D Content-Based Retrieval in Artwork Databases*, In Proceedings of the IEEE 3DTV-Conference, Kos Island, Greece, Mai 2007.

JOURNAUX INTERNATIONAUX soumis

1. D. Gorisse, M. Cord, F. Precioso, *Locality-Sensitive Hashing Scheme for Chi2 distance*, IEEE transactions on pattern analysis and machine intelligence, 2010.

Références bibliographiques

[ACV07] A. Auclair, L. Cohen, and N. Vincent. How to Use SIFT Vectors to Analyze an Image with Database Templates. *Adaptive Multimedial Retrieval : Retrieval, User, and Semantics*, pages 224–236, 2007. 97

[AI05] A. Andoni and P. Indyk. E2lsh. http ://www.mit.edu/∼andoni/LSH/, 2005. 35, 86

[AIM06] A. Andoni, P. Indyk, and C. MIT. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *47th Annual IEEE Symposium on Foundations of Computer Science*, pages 459–468, 2006. 43

[Auc09] A. Auclair. *Méthodes rapides pour la recherche des plus proches voisins SIFT : application à la recherche d'images et Contributions à la reconstruction 3D multi-vues*. PhD thesis, Université Paris Descartes, 2009. 73

[BAG03] S.-A. Berrani, L. Amsaleg, and P. Gros. Recherche approximative de plus proches voisins : Application à la reconnaissance d'images par descripteurs locaux. In *Technique et Science Informatiques*, volume 22, pages 1201–1230, 2003. 16

[BBK01] C. Bohm, S. Berchtold, and D.A. Keim. Searching in high-dimensional spaces : Index structures for improving the performance of multimedia databases. *ACM Computing Surveys (CSUR)*, 33(3) :322–373, 2001. 19

[Ben75] J.L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9) :517, 1975. 19

[BEWB05] Antoine Bordes, Seyda Ertekin, Jason Weston, and Léon Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6 :1579–1619, September 2005. 18

[BG09] G.J. Burghouts and J.M. Geusebroek. Performance evaluation of local colour invariants. *Computer Vision and Image Understanding*, 113(1) :48–62, 2009. 9

[BHHSV02] A. Ben-Hur, D. Horn, H.T. Siegelmann, and V. Vapnik. Support vector clustering. *The Journal of Machine Learning Research*, 2 :125–137, 2002. 115

[BHS+07] G. BakIr, T. Hofmann, B. Scholkopf, A.J. Smola, and B. Taskar. *Predicting structured data*. The MIT Press, 2007. 82

[BJ03] F.R. Bach and M.I. Jordan. Kernel independent component analysis. *The Journal of Machine Learning Research*, 3 :1–48, 2003. 112

[BK59] R. Bellman and R. Kalaba. On adaptive control processes. *Automatic Control, IRE Transactions on*, 4(2) :1–9, 1959. 19

[BL02] J.S. Beis and D.G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 1000–1006. IEEE, 2002. 28

[BMM09] Eric Bruno and Stephane Marchand-Maillet. Multimodal preference aggregation for multimedia information retrieval. *Journal of Multimedia*, 4(5) :321–329, 2009. 74

[Bor10] Antoine Bordes. *New Algorithms for Large-Scale Support Vector Machines*. PhD thesis, Université Pierre et Marie Curie, Computer Science Laboratory of Paris 6 (LIP6), February 2010. 8

[Bri03] K. Brinker. Incorporating diversity in active learning with support vector machines. *Machine Learning International Workshop then Conference*, pages 59–66, 2003. 16, 17, 121

[BS09] L. Bo and C. Sminchisescu. Efficient Match Kernel between Sets of Features for Visual Recognition. In *Advances in Neural Information Processing Systems*, December 2009. 73

[Bur87] Stevenson Burton. *The Home Book of Proverbs, Maxims, and Familiar Phrases*. MacMillan Publishing Company, April 1987. 6

[Car99] A.C Carrilero. Les espacesc de représentation de la couleur. *Technical Report 99D006*, 1999. 9

[CC08] M. Cord and P. Cunningham. *Machine Learning Techniques for Multimedia*. Springer, 2008. 16

[CEOT07] M. Crucianu, D. Estevez, V. Oria, and J.P. Tarel. Hyperplane Queries in a Feature-Space M-tree for Speeding up Active Learning. *Bases de Donnees Avancees, Marseille, France, October*, 2007. 116, 118

[CGJ96] DA Cohn, Z. Ghahramani, and MI Jordan. Active learning with statistical models. *Journal of Artifical Intelligence REsearch*, 4 :129–145, 1996. 15, 16

[Cha02] M.S. Charikar. Similarity estimation techniques from rounding algorithms. *ACM*, pages 380–388, 2002. 31, 38, 145

[CHV99] O. Chapelle, P. Haffner, and VN Vapnik. Support vector machines for histogram-based image classification. *IEEE Transactions on Neural Networks*, 10(5) :1055–1064, 1999. 11, 21, 44, 61, 120, 148

[CMM+98] Y. Chiaramella, P. Mulhem, M. Mechkour, I. Ounis, and M. Pasca. Towards a fast precision-oriented image retrieval system. *ACM*, pages 383–384, 1998. 78

[CPIZ07] Ondřej Chum, James Philbin, Michael Isard, and Andrew Zisserman. Scalable near identical image and shot detection. In *CIVR '07 : Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 549–556, New York, NY, USA, 2007. ACM. 73

[CPZ97] P. Ciaccia, M. Patella, and P. Zezula. M-tree : An efficient access method for similarity search in metric spaces. In *Proceedings of the International Conference on Very Large Data Bases*, pages 426–435. Citeseer, 1997. 28

[CPZ08] O. Chum, J. Philbin, and A. Zisserman. Near duplicate image detection : min-hash and tf-idf weighting. In *British Machine Vision Conference*, 2008. 13

[CSB99] J.H. Conway, N.J.A. Sloane, and E. Bannai. *Sphere packings, lattices, and groups*. Springer Verlag, 1999. 43

[CSTL02] N. Cristianini, J. Shawe-Taylor, and H. Lodhi. Latent semantic kernels. *Journal of Intelligent Information Systems*, 18(2) :127–152, 2002. 112

[CW04] Y. Chen and J.Z. Wang. Image categorization by learning and reasoning with regions. *The Journal of Machine Learning Research*, 5 :913–939, 2004. 76

[CZH01] Y. Chen, X. Zhou, and T.S. Huang. One-class SVM for learning in image retrieval. *Urbana*, 51 :61820, 2001. 122

[DDS+09] J. Deng, W. Dong, R. Socher, L.J. Li, K. Li, and L. Fei-Fei. Imagenet : A large-scale hierarchical image database. 2009. 20

[DIIM04] M. Datar, N. Immorlica, P. Indyk, and V.S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. *SCG*, pages 253–262, 2004. 34, 41, 44, 45, 46, 49, 50, 51

[DW00] C. Domingo and O. Watanabe. MadaBoost : A modification of AdaBoost. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pages 180–189. Citeseer, 2000. 110

[EC04] J. Eichhorn and O. Chapelle. Object categorization with svm : kernels for local features. *Advances in Neural Information Processing Systems (NIPS)*, 2004. 89

[Esc78] B. Escofier. Analyse factorielle et distances répondant au principe d'équivalence distributionnelle. *Revue de Statistique Appliquée*, 26(4) :29–37, 1978. 11

[EVGW+09] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2009 (VOC2009) Results, 2009. 74

[EZWVG06] M. Everingham, A. Zisserman, C. K. I. Williams, and L. Van Gool. Pascal voc2006, 2006. 126, 148

[FB81] M.A. Fischler and R.C. Bolles. Random sample consensus : A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6) :381–395, 1981. 13

[FBS06] JH Friedman, F. Baskett, and LJ Shustek. An algorithm for finding nearest neighbors. *Computers, IEEE Transactions on*, 100(10) :1000–1006, 2006. 28

[FCPF01] J. Fournier, M. Cord, and S. Philipp-Foliguet. Retin : A content-based image indexing and retrieval system. *Pattern Analysis & Applications*, 4(2) :153–173, 2001. 10

[FKS03] R. Fagin, R. Kumar, and D. Sivakumar. Efficient similarity search and classification via rank aggregation. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, 2003. 28

[Fou02] J. Fournier. *Indexation d'images par le contenu et recherche interactive dans les bases généralistes*. PhD thesis, Université de de Cergy-Pontoise, octobre 2002. 9, 73

[FS07] J.J. Foo and R. Sinha. Pruning SIFT for scalable near-duplicate image matching. In *Proceedings of the eighteenth conference on Australasian database*, volume 63, page 71. Australian Computer Society, Inc., 2007. 96, 100

[GCP09] D. Gorisse, M. Cord, and F. Precioso. Optimization on active learning strategy for object category retrieval. In *Proceedings of the 16th IEEE International Conference on Image Processing (ICIP)*, pages 1873–1876. IEEE, nov 2009. 22

[GCP10a] D. Gorisse, M. Cord, and F. Precioso. Locality-Sensitive Hashing Scheme for Chi2 distance. *IEEE transactions on pattern analysis and machine intelligence*, 2010. 21

[GCP10b] D. Gorisse, M. Cord, and F. Precioso. SALSAS : Sub-linear Active Learning Strategy with Approximate k-NN Search. *Pattern Recognition*, 2010. 21, 22

[GCP10c] D. Gorisse, M. Cord, and F. Precioso. Scalble active learning strategy for object category retrieval. In *Proceedings of the 17th IEEE International Conference on Image Processing (ICIP)*. IEEE, sep 2010. 21, 22

[GCPF07] P.H. Gosselin, M. Cord, and S. Philipp-Foliguet. Kernel on bags for multi-object database retrieval. *CIVR*, 2007. 12, 78

[GCPF08a] P.H. Gosselin, M. Cord, and S. Philipp-Foliguet. Active learning methods for interactive image retrieval. *IEEE Transactions on Image Processing*, 17(7) :1200–1211, 2008. 16

[GCPF08b] P.H. Gosselin, M. Cord, and S. Philipp-Foliguet. Combining visual dictionary, kernel-based similarity and learning strategy for image category retrieval. *CVIU*, 2008. 10, 11, 21, 44, 61, 120

[GCPPF08] D. Gorisse, M. Cord, F. Precioso, and S. Philipp-Foliguet. Fast Approximate Kernel-Based Similarity Search for Image Retrieval Task. In *Proceedings of the 19th International Conference on Pattern Recognition (ICPR)*, pages 1–4, 2008. 21

[GD05] K. Grauman and T. Darrell. The pyramid match kernel : Discriminative classification with sets of image features. 2005. 76

[GD07] K. Grauman and T. Darrell. Pyramid match hashing : Sub-linear time indexing over partial correspondences. *CVPR*, pages 1–8, 2007. 40, 76, 77

[GIM99] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, pages 518–529. Morgan Kaufmann Publishers Inc., 1999. 34, 37, 55

[Gos05] P.H. Gosselin. *Méhodes d'apprentissage pour la recherche de catégories dans des bases d'images*. PhD thesis, Université de Cergy-Pontoise, 2005. 16

[GPCPF08] D. Gorisse, F. Precioso, M. Cord, and S. Philipp-Foliguet. Summarization scheme based on near-duplicate analysis. In *TVS '08 : Proceedings of the 2nd ACM TRECVid Video Summarization Workshop*. ACM, oct 2008. 21

[Gut84] A. Guttman. R-trees : a dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, pages 47–57. ACM, 1984. 28

[GZ+01] M. Griebel, G. Zumbusch, et al. Hash based adaptive parallel multilevel methods with space-filling curves. In *NIC Symposium*, volume 9, pages 3–00. Citeseer, 2001. 29

[HKLO+00] A. Heinrichs, D. Koubaroulis, B. Levienaise-Obadia, P. Rovida, and J. Jolion. Image indexing and content based search using pre-attentive similarities. In *IEEE Conference on Image Processing*, volume 138. Citeseer, 2000. 9

[HP05] D.R. Heisterkamp and J. Peng. Kernel vector approximation files for relevance feedback retrieval in large image databases. *Multimedia Tools and Applications*, 26(2) :175–189, 2005. 111, 112, 113

[HTF09] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2009. 61

[HZ01] TS Huang and X.S. Zhou. Image retrieval with relevance feedback : from heuristic weightadjustment to optimal learning methods. In *Image Processing, 2001. Proceedings. 2001 International Conference on*, volume 3, 2001. 15

[IM98] P. Indyk and R. Motwani. Approximate nearest neighbors : towards removing the curse of dimensionality. *ACM*, pages 604–613, 1998. 19, 21, 29, 31, 32, 34, 37

[IT03] P. Indyk and N. Thaper. Fast image retrieval via embeddings. In *Int. Workshop Statistical and Computational Theories of Vision*, 2003. 44, 45, 61

[JASG08] H. Jegou, L. Amsaleg, C. Schmid, and P. Gros. Query-adaptative locality sensitive hashing. In *International Conference on Acoustics, Speech, and Signal Processing*, 2008. 43

[JDS08a] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *European conference on computer vision*, pages 304–317. Springer, 2008. 73, 80

[JDS08b] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In Andrew Zisserman David Forsyth, Philip Torr, editor, *European Conference on Computer Vision*, volume I of *LNCS*, pages 304–317. Springer, oct 2008. 94, 95, 147

[JDS09] H. Jégou, M. Douze, and C. Schmid. Packing bag-of-features. In *Proceedings of the International Conference on Computer Vision*, 2009. 94

[JDSP10] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *IEEE Conference on Computer Vision & Pattern Recognition*, jun 2010. 74, 94, 97, 101, 145

[KG09] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. *ICCV, October*, 1 :3, 2009. 40, 41, 145

[KJ03] R. Kondor and T. Jebara. A kernel between sets of vectors. In *Machine Learning-International Workshop then conference*, volume 20, page 361, 2003. 76, 77

[KS04] Y. Ke and R. Sukthankar. PCA-SIFT : A more distinctive representation for local image descriptors. In *Proceedings of Computer Vision and Pattern Recognition*. IEEE Computer Society, 2004. 100

[KSH04] Y. Ke, R. Sukthankar, and L. Huston. Efficient near-duplicate detection and sub-image retrieval. In *ACM Multimedia*, 2004. 21

[LÁJA06] H. Lejsek, F.H. Ásmundsson, B.T. Jónsson, and L. Amsaleg. Scalability of local image descriptors : a comparative study. In *Proceedings of the 14th annual ACM international conference on Multimedia*, page 598. ACM, 2006. 96

[LCGM+00] C. Li, E. Chang, H. Garcia-Molina, J.Z. Wang, and G. Wiederhold. Clindex : Clustering for similarity queries in high-dimensional spaces. *Dept. Comput. Sci., Stanford Univ., Stanford, CA, Tech. Rep*, 2000. 28, 111

[LJW+07] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Multi-probe LSH : efficient indexing for high-dimensional similarity search. In *Proceedings of the 33rd international conference on Very large data bases*, pages 950–961. VLDB Endowment, 2007. 42, 45, 53, 55, 58, 94, 129

[Low04] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2) :91–110, 2004. 9, 13, 75, 76, 79, 86, 94

[LRB09] MJ Lesot, M. Rifqi, and H. Benhadda. Similarity measures for binary and numerical data : a survey. *International Journal of Knowledge Engineering and Soft Data Paradigms*, 1(1) :63–84, 2009. 10

[Lyu05] S. Lyu. Mercer kernels for object recognition with local features. *CVPR*, 2 :223–229, 2005. 78

[Man06] Y. Manolopoulos. *R-trees : Theory and Applications*. Springer Verlag, 2006. 115

[MM99] W.Y. Ma and B.S. Manjunath. Netra : A toolbox for navigating large image databases. *Multimedia Systems*, 7(3) :184–198, 1999. 10

[MMMB07] Donn Morrison, Stéphane Marchand-Maillet, and Eric Bruno. Automatic image annotation with relevance feedback and latent semantic analysis. In *Proceedings 5th International Workshop on Adaptive Multimedia Retrieval*, Paris, France, 2007. 15

[MTS+05] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L.V. Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1) :43–72, 2005. 94

[OT06] A. Oliva and A. Torralba. Building the gist of a scene : The role of global image features in recognition. *Progress in brain research*, 155 :23–36, 2006. 9

[Pan06] R. Panigrahy. Entropy based nearest neighbor search in high dimensions. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, page 1195. ACM, 2006. 42

[PC05] N. Panda and E.Y. Chang. Exploiting Geometry for Support Vector Machine Indexing. In *Proceedings of the 5th International Conference on Data Mining*. Society for Industrial Mathematics, 2005. 113, 114, 145

[PC06] N. Panda and E.Y. Chang. Efficient top-k hyperplane query processing for multimedia information retrieval. In *Proceedings of the 14th annual ACM international conference on Multimedia*, pages 317–326. ACM Press New York, NY, USA, 2006. 115, 116, 118, 120

[PCI+07] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007. 73, 93, 147

[PD07] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR'07*, pages 1–8, 2007. 74

[PGC06] N. Panda, K.S. Goh, and E.Y. Chang. Active learning in very large databases. *Multimedia Tools and Applications*, 31(3) :249–267, 2006. 111, 113, 115

[RH00] Y. Rui and T. Huang. Optimizing learning in image retrieval. In *cvpr*, page 1236. Published by the IEEE Computer Society, 2000. 110

[RHMO97] Y. Rui, T.S. Huang, S. Mehrotra, and M. Ortega. A relevance feedback architecture for content-based multimedia information retrieval systems. *caivl*, page 82, 1997. 15

[RLJ09] J. Ros, C. Laurent, and J.M. Jolion. A Bag of Strings Representation for Image Categorization. *Journal of Mathematical Imaging and Vision*, 35(1) :51–67, 2009. 9

[RTG00] Y. Rubner, C. Tomasi, and L.J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2) :99–121, 2000. 11

[SC67] G.W. Snedecor and W.G. Cochran. Statistical Methods. *Iowa State University Press*, 5(6) :9, 1967. 11

[SC96] B. Schiele and J.L. Crowley. Object recognition using multidimensional receptive field histograms. *Lecture Notes in Computer Science*, pages 610–619, 1996. 11, 21, 44

[SDI06] G. Shakhnarovich, T. Darrell, and P. Indyk. *Nearest-Neighbor Methods in Learning and Vision : Theory and Practice (Neural Information Processing)*. The MIT Press, 2006. 74

[SMMP00] D.M.G. Squire, W. Müller, H. Müller, and T. Pun. Content-based query of image databases : inspirations from text retrieval. *Pattern Recognition Letters*, 21(13-14) :1193–1198, 2000. 19

[SS02] B. Scholkopf and AJ Smola. Learning with Kernels. *MIT Press*, 2002. 11, 110, 146

[STC04] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004. 76, 77

[SvdSdR+08] C.G.M. Snoek, K.E.A. van de Sande, O. de Rooij, B. Huurnink, J.C. van Gemert, J.R.R. Uijlings, J. He, X. Li, I. Everts, V. Nedovic, et al. The MediaMill TRECVID 2008 semantic video search engine. In *Proceedings of the 6th TRECVID Workshop*. Citeseer, 2008. 120

[SvdSdR+09] C.G.M. Snoek, K.E.A van de Sande, O. de Rooij, B. Huurnink, J.R.R Uijlings, M. van Liempt, M. Bugalho, I. Trancoso, F. Yan, M.A. Tahir, K. Mikolajczyk, J. Kittler, M. de Rijke, J.M. Geusebroek, T. Gevers, M. Worring, D.C. Koelma, and A.W.M. Smeulders. The MediaMill TRECVID 2009 semantic video search engine. In *Proceedings of the 7th TRECVID Workshop*, November 2009. 120, 149

[SWS+00] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on pattern analysis and machine intelligence*, 22(12) :1349–1380, 2000. 7, 9, 61

[SZ03] J. Sivic and A. Zisserman. Video Google : A text retrieval approach to object matching in videos. In *Ninth IEEE international conference on computer vision, 2003. Proceedings*, volume 2, pages 1470–1477, 2003. 10, 73, 96

[TC01] S. Tong and E. Chang. Support vector machine active learning for image retrieval. *Proceedings of the ninth ACM international conference on Multimedia*, pages 107–118, 2001. 15, 16, 139, 141

[TFF08] A. Torralba, R. Fergus, and W.T. Freeman. 80 million tiny images : A large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1958–1970, 2008. 20

[TFW08] A. Torralba, R. Fergus, and Y. Weiss. Small codes and large databases for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR'08*, pages 1–8, 2008. 100

[TK02] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2 :45–66, 2002. 16

[TV04] K. Tieu and P. Viola. Boosting image retrieval. *International Journal of Computer Vision*, 56(1) :17–36, 2004. 16

[Val08] E. Valle. *Local-descriptor matching for image identification systems*. PhD thesis, Université de Cergy-Pontoise, 2008. 14, 19, 27

[Vas00] N. Vasconcelos. *Bayesian models for visual information retrieval*. PhD thesis, Massachusetts Institute of Technology, School of Architecture and Planning, Program in Media Arts and Sciences, 2000. 16

[VCPF07] E. Valle, M. Cord, and S. Philipp-Foliguet. 3-Way-Trees : A Similarity Search Method for High-Dimensional Descriptor Matching. In *IEEE International Conference on Image Processing, 2007. ICIP 2007*, volume 1, 2007. 55

[VCPF08a] E. Valle, M. Cord, and S. Philipp-Foliguet. High-dimensional descriptor indexing for large multimedia databases. In *Proceeding of the 17th ACM conference on Information and knowledge management*, pages 739–748. ACM, 2008. 35

[VCPF08b] E. Valle, M. Cord, and S. Philipp-Foliguet. High-Dimensional Descriptor Indexing for Large Multimedia Databases. In *CIKM '08 : Proceedings of the 17th ACM Conference on Information and Knowledge Management*, page 739–748, Napa Valley – CA, USA, 2008. ACM. 74

[vdSGS10] K.E.A. van de Sande, T. Gevers, and C.G.M. Snoek. Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010. 9

[vGGVS08] J.C. van Gemert, J.M. Geusebroek, C.J. Veenman, and A.W.M. Smeulders. Kernel codebooks for scene categorization. In *Proc. ECCV*, volume 5. Springer, 2008. 74

[VV98] V.N. Vapnik and V. Vapnik. *Statistical learning theory*. Wiley New York, 1998. 16

[WCG03] C. Wallraven, B. Caputo, and A. Graf. Recognition with local features : the kernel recipe. In *Proc. ICCV*, pages 1–2, 2003. 77

[WS03] L. Wolf and A. Shashua. Learning over sets using kernel principal angles. *The Journal of Machine Learning Research*, 4 :913–931, 2003. 77

[WSB98] R. Weber, H.J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of the International Conference on Very Large Data Bases*, pages 194–205. Citeseer, 1998. 28, 112

[YJHN07] J. Yang, Y.G. Jiang, A.G. Hauptmann, and C.W. Ngo. Evaluating bag-of-visual-words representations in scene classification. In *Proceedings of the international workshop on Workshop on multimedia information retrieval*, page 206. ACM, 2007. 73, 96

[ZCL+09] Xi Zhou, Na Cui, Zhen Li, Feng Liang, and Thomas S. Huang. Hierarchical Gaussianization for Image Classification. In *Twelfth IEEE International Conference on Computer Vision, 2009. ICCV 2009*, 2009. 74

Avec la révolution numérique de cette dernière décennie, la quantité de photos numériques mise à disposition de chacun augmente plus rapidement que la capacité de traitement des ordinateurs. Les outils de recherche actuels ont été conçus pour traiter de faibles volumes de données. Leur complexité ne permet généralement pas d'effectuer des recherches dans des corpus de grande taille avec des temps de calculs acceptables pour les utilisateurs. Dans cette thèse, nous proposons des solutions pour passer à l'échelle les moteurs de recherche d'images par le contenu. Dans un premier temps, nous avons considéré les moteurs de recherche automatique traitant des images indexées sous la forme d'histogrammes globaux. Le passage à l'échelle de ces systèmes est obtenu avec l'introduction d'une nouvelle structure d'index adaptée à ce contexte qui nous permet d'effectuer des recherches de plus proches voisins approximées mais plus efficaces. Dans un second temps, nous nous sommes intéressés à des moteurs plus sophistiqués permettant d'améliorer la qualité de recherche en travaillant avec des index locaux tels que les points d'intérêt. Dans un dernier temps, nous avons proposé une stratégie pour réduire la complexité de calcul des moteurs de recherche interactifs. Ces moteurs permettent d'améliorer les résultats en utilisant des annotations que les utilisateurs fournissent au système lors des sessions de recherche. Notre stratégie permet de sélectionner rapidement les images les plus pertinentes à annoter en optimisant une méthode d'apprentissage actif.

Mots clés : LSH, recherche sémantique, grandes bases d'images, passage à l'échelle, apprentissage interactif, apprentissage actif

In this last decade, the digital revolution results in a massive increase of digital picture quantities. The database sizes grow much faster than the processing capacity of computers. The current search engines which have been conceived for smaller data volumes do not any more allow to perform retrieval in these new corpuses with acceptable response times for users. In this thesis, we developed scalable content-based image search engines. At first, we considered automatic search engines where images are indexed with global histograms. Secondly, we focused on more sophisticated engines allowing to improve the search quality by working with bag of features. At last, we proposed a strategy to reduce the complexity of interactive search engines which allow to iteratively improve results by using labels that the users supply to the system during search sessions.

Keywords : LSH, Content Based Image Retrieval, huge image database, large scale, interactive search, active learning