Représentation compacte des sacs de mots pour l'indexation d'images

Hervé Jégou, Matthijs Douze, Cordelia Schmid

Représentation compacte des sacs de mots pour l'indexation d'images

Hervé Jégou Matthijs Douze Cordelia Schmid

INRIA Grenoble, équipe-projet LEAR

prenom.nom@inria.fr

Résumé

La représentation des images par descripteurs locaux est invariante à de nombreuses transformations. Une de ses principales limitations est la quantité de mémoire requise pour décrire chaque image. Une représentation de type sac de mots est ainsi limitée à quelques millions d'images pour un serveur raisonnablement équipé.
Dans cet article, nous étudions la possibilité de compresser et indexer conjointement les représentations sacs de mots, et introduisons une méthode de recherche approximative sur ce type de vecteurs creux. La comparaison entre deux images consiste à calculer une espérance de distance de Hamming entre leur représentations. Nos expériences montrent que l'approche proposée requiert dix à cent fois moins de mémoire que la représentation initiale et améliore l'efficacité de la recherche, tout en offrant une qualité de recherche comparable.

Mots clef

recherche d'image, sac de mots, grandes bases d'images, compression

Abstract

One of the main limitations of image search based on bag-of-features is the memory usage per image. Only a few million images can be handled on a single machine in reasonable response time. In this paper, we first evaluate how the memory usage is reduced by using lossless index compression. We then propose an approximate representation of bag-of-features obtained by projecting the corresponding histogram onto a set of pre-defined sparse projection functions, producing several image descriptors. Coupled with a proper indexing structure, an image is represented by a few hundred bytes. A distance expectation criterion is then used to rank the images. Our method is at least one order of magnitude faster than standard bag-of-features while providing excellent search quality.

Keywords

large scale image search, bag-of-features, compression

1 Introduction

La recherche d'images à partir du contenu est un domaine en forte expansion [10, 4, 13, 5]. Le problème consiste à trouver, au sein d'une base d'images, celles qui contiennent les mêmes objets que l'image requête, ou qui représentent la même scène. L'approche la plus populaire aujourd'hui, initialement proposée dans [14], repose sur une description des images par sac de mots (SDM). Elle consiste à quantifier les descripteurs locaux [7] calculés sur des régions d'intérêt, qui sont préalablement extraites au moyen d'un détecteur invariant aux transformations affines [9]. Les indices de quantification de ces descripteurs sont appelés *mots visuels*, par analogie avec la recherche de documents textuels. Dans l'esprit du modèle vectoriel utilisé en texte, l'image est représentée par l'histogramme des fréquences d'apparition des mots visuels. En combinaison avec un fichier inversé, cette représentation permet le calcul efficace de similarités entre images. Cette méthode de recherche a été étendue de nombreuses manières pour prendre en compte la spécificité des images. La quantification et le calcul de distance ont été rendus plus efficaces [10, 13]. Une post-vérification spatiale filtre les résultats en fonction de leur cohérence géométrique avec la requête [7, 12]. Des gains en pertinence et en efficacité ont également été obtenus en raffinant la représentation des descripteurs [4] et en intégrant partiellement de l'information géométrique [4, 11].

Les principaux avantages de la représentation SDM sont 1) son occupation mémoire limitée par rapport à une approche par géométrie épipolaire, 2) la rapidité de la recherche, qui résulte du caractère creux des données et de son exploitation par un fichier inversé. Plutôt que stocker les 128 composantes de nombreux descripteurs SIFT, il suffit de traiter une entrée de quelques octets par descripteur. Il devient possible d'indexer de l'ordre d'un million d'images. Cependant, les performances sont encore éloignées des systèmes de recherche texte : l'ordre du milliard d'image reste hors d'atteinte. Des deux contraintes matérielles, efficacité du calcul et complexité en mémoire, la seconde est la plus contraignante et la moins traitée.

Dans cet article, nous analysons dans un premier temps les méthodes existantes qui répondent au mieux à notre ob-

jectif de compacité de la représentation. Nous proposons ensuite un système qui permet de passer à des échelles nettement plus grandes que celles possibles par SDM. Notre approche consiste à produire plusieurs petits descripteurs à partir d'un SDM. Chacun de ces descripteurs, appelé *miniSDM*, fournit une information partielle sur le SDM original. Un calcul d'espérance de distance fusionne les mesures fournies par chacun de ces descripteurs et permet d'ordonner les images par pertinence décroissante. Cette méthode est d'un à deux ordres de grandeur plus efficace que SDM et permet de représenter une image par une centaine d'octets environ (voir en section 5). Contrairement à [15], où les images sont représentées par des vecteurs binaires obtenus à partir de descripteurs globaux, les miniSDM héritent des avantages et inconvénients de la représentation SDM. En particulier, l'utilisation de descripteurs locaux permet de traiter une large classe de transformations, avec un traitement de l'information géométrique qui demeure cependant purement local.

2 Protocole d'évaluation

Nous introduisons dans cette section la représentation SDM utilisée comme référence dans cet article, ainsi que les jeux de données utilisés pour l'évaluation.

2.1 Représentation par sac de mots

Un descripteur SDM représentant une image est calculé en deux étapes. Dans un premier temps, des régions d'intérêt sont extraites par un détecteur Hessien affine [9] et des descripteurs locaux SIFT [7] sont calculés sur les zones identifiées. Ces descripteurs sont alors quantifiés par un quantificateur k-moyennes, dont les centroides ont été préalablement appris sur un ensemble indépendant. Ce quantificateur vectoriel est usuellement dénommé *vocabulaire visuel*, et un élément de son dictionnaire est appelé *un mot visuel*. La quantification consiste à assigner chaque descripteur SIFT à son plus proche voisin euclidien, ce qui revient à l'assimiler au mot visuel le plus proche. La représentation vectorielle de l'image est calculée comme l'histogramme des fréquences d'apparition des mots visuels. Ses composantes sont pondérées et normalisées comme indiqué dans [14], ce qui, pour l'image i, produit un vecteur f_i à k composantes.

2.2 Jeux de données et évaluation

Nous utilisons les bases d'images annotées *INRIA Holidays* et *Kentucky* [10] pour évaluer notre approche dans les mêmes conditions que celles de la littérature. Les descripteurs d'images sont produits selon la procédure décrite dans [4]. Ils ont été calculés sur les jeux de données suivants :

jeu de données	nombre d'images	nombre de requêtes
Holidays	1,491	500
Kentucky	10,200	10,200
Flickr1M	1,000,000	0
Flickr1M*	1,000,000	0

Les bases *Flickr1M* et *Flickr1M**, téléchargées depuis Flickr, permettent d'évaluer la performance à grande échelle. La première est un ensemble de "distracteurs". Elle est utilisée en conjonction avec l'une des bases annotées. La base *Flickr1M** est utilisée à des fins d'apprentissage.

Nous utilisons les mesures de pertinence usuelles pour ces bases, à savoir mAP (*mean average precision*, voir [12]) pour Holidays, et le nombre moyen d'images correctes renvoyés dans les quatre premières positions pour Kentucky. Puisqu'il y a 4 images pertinentes par requête, cette mesure correspond à 4 fois la valeur du rappel lorsque 4 images sont renvoyées par le système ($4 \times$recall@4).

Finalement, dans la mesure où notre objectif est de dépasser les limitations actuelles sur le nombre d'images, nous accordons une importance particulière à la quantité de mémoire occupée par la structure d'indexation, et celle parcourue pour effectuer une recherche, ainsi qu'aux nombre de "hits", c'est-à-dire le nombre d'images effectivement traitées par le système.

3 Discussion préliminaire

Rappelons que nous cherchons à obtenir une représentation d'image invariante à une large classe de transformations (rotations, recadrage, changement de point de vue, etc) et satisfaisant les contraintes suivantes : 1) faible occupation mémoire et 2) faible quantité de mémoire à parcourir lors de la requête. Ce dernier critère est capital lorsque les données sont stockées sur un support lent, tel un disque dur. C'est également un bon indicateur de la complexité lorsque les données sont en mémoire centrale. Dans la suite, nous avons identifié les méthodes de la littérature qui, de notre point de vue, répondent au mieux à ces objectifs. En particulier, nous avons évalué l'intérêt de la compression d'index [17, 18], une approche établie en recherche textuelle.

3.1 SDM binaire

Une manière immédiate de compacter un vecteur SDM consiste à encoder uniquement l'absence ou la présence d'un mot visuel dans une image, en ignorant son nombre exact d'occurrences. On obtient une représentation binaire du SDM (dite *SDM binaire*). Cette approche est proposée dans [14], où les auteurs concluent qu'il réduit légèrement la pertinence pour des vocabulaires visuels d'environ $k = 10000$ mots visuels. Cependant, cette représentation n'a pas, à notre connaissance, été évaluée pour des vocabulaires visuels plus petits ($k < 2000$) ou plus grands (> 20000). La figure 1 compare la pertinence du SDM binaire à la référence sur la base Holidays en fonction de la taille du vocabulaire. Il apparaît que le SDM binaire est une représentation pauvre pour les petits vocabulaires, mais obtient des résultats comparables à la représentation SDM pour des vocabulaires plus grands (> 10000).

Le nombre de bits nécessaire à la représentation d'un vecteur binaire dépend de la méthode d'encodage. La méthode naïve (dite *SDM binaire naïve*), consiste à encoder le vecteur binaire en utilisant 1 bit par dimension. L'utilisation

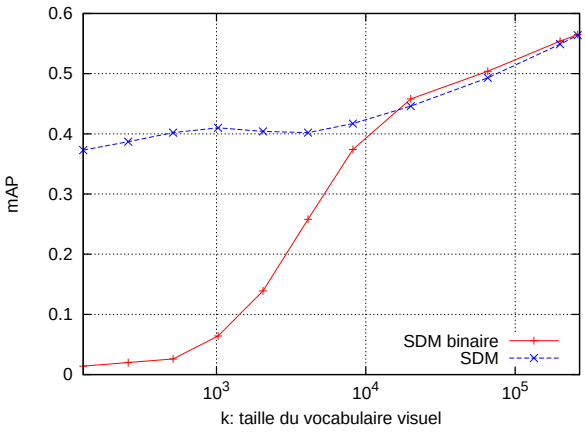

FIG. 1 – Pertinence de la recherche : SDM *vs* SDM binaire

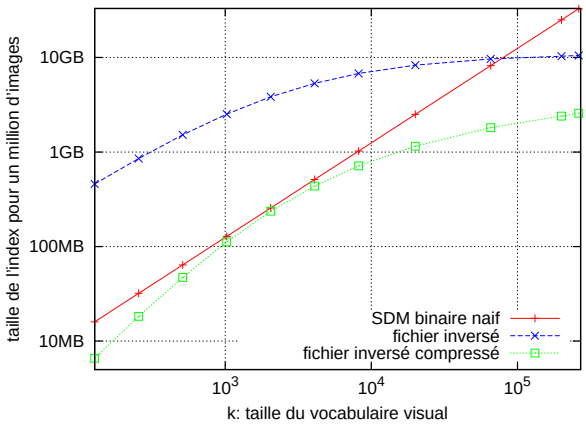

FIG. 2 – SDM binaires : utilisation mémoire de différentes structures d'indexation pour un million d'images.

mémoire résultante est donc de $\lceil k/8 \rceil$ octets par image. Comme la représentation SDM binaire n'est pertinente que pour des gros vocabulaires, l'usage mémoire résultant est typiquement de 10 ko par image (figure 2).

Une meilleure stratégie consiste à exploiter le caractère creux des vecteurs binaires de grande taille, qui peuvent être représentés par une liste de couples (position non nulle, valeur). Dans le cas binaire, cela revient à ne coder que les positions non nulles, par exemple au moyen d'un codage par plage. Cette stratégie se combine bien avec une structure de fichier inversé [18], et permet de calculer efficacement les distances entre le vecteur requête et l'ensemble des vecteurs SDM binaires compressés de la base.

3.2 Compression d'index

La représentation par fichier inversé peut être couplée avec un codeur entropique pour produire une structure compressée *et* indexable. Cette compression d'index est une méthode établie en recherche textuelle [18]. La motivation est d'exploiter, via un algorithme de (dé-)compression, la distribution des mots visuels pour rendre la structure plus compacte. Dans la suite, nous nous intéressons à cette méthode pour la compression des vecteurs SDM binaires.

Ces vecteurs peuvent théoriquement être compressés jusqu'à l'entropie du vecteur. Celle-ci est cependant difficile à mesurer et à exploiter, en raison du trop grand nombre d'éléments dans l'hypercube de Hamming (2^k). Une borne supérieure, équivalente à l'entropie vectorielle lorsque les composantes sont indépendantes, est obtenue en additionnant les entropies marginales des variables aléatoires binaires associées à chacune des dimensions. Cette borne peut être quasiment atteinte en couplant, dans le fichier inversé, un codage par plage avec un codeur arithmétique [16]. Différentes méthodes de compression d'index ont été considérées dans la littérature de recherche textuelle. Certaines approchent de près la borne entropique, voir [17] pour un tour d'horizon récent.

La figure 2 donne l'usage mémoire, calculé à partir des entropies binaires marginales, associé à un fichier inversé indexant un million d'images. Ces mesures, extrapolées de la base Holidays, montrent clairement l'intérêt de la compression d'index face à un fichier inversé classique. La quantité de mémoire à lire lors de la requête est réduite dans les mêmes proportions, ce qui peut compenser le surcoût de calcul induit par le décodage entropique si l'élément limitant est le temps de lecture.

3.3 Discussion

La compression d'index couplée avec une représentation SDM binaire est une méthode prometteuse (figure 1). Elle requiert une utilisation mémoire de l'ordre de 1 à 2 ko par image (figure 2), soit de cinq à dix fois moins que la représentation SDM usuelle. Les besoins en ressource mémoire sont similaires à ceux de l'algorithme min-Hash. Celui-ci avait été initialement proposé dans un contexte de détection de copies [1], où 768 octets suffisaient à indexer une image. Cependant ces performances étaient obtenues dans des conditions avantageuses (détection d'images quasiment identiques). Pour la reconnaissance d'objet ou de scènes [2], la méthode requiert de l'ordre de 6 ko par image pour offrir des résultats acceptables.

Dans un contexte de recherche de copies, le min-Hash demeure néanmoins compétitif par rapport à la compression d'index. En effet, le nombre de "hits", c'est-à-dire le nombre de documents effectivement classés, est plus faible pour cette méthode (seulement 5% des images sont considérées), ce qui est nettement meilleur que l'approche par fichier inversé, binaire ou pas, compressé ou non.

La méthode introduite dans la section suivante vise à dépasser les limites de la compression d'index et du min-Hash : (1) elle offre une pertinence comparable à une représentation par sac de mots avec quelques centaines d'octets ; (2) elle ne parcourt qu'une faible fraction de la base ; (3) elle retourne un nombre limité d'images.

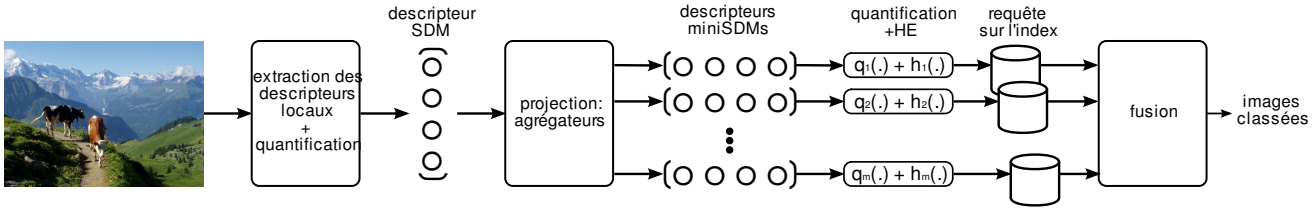

FIG. 3 – Principe de notre approche.

4 Approche proposée

Le problème central de l'indexation des images par une représentation SDM est qu'il n'existe pas, à notre connaissance, de technique efficace de recherche approximative de plus proches voisins dans un ensemble de vecteurs creux. En effet, utiliser un algorithme comme le *Locality-Sensitive-Hashing* [3] est moins efficace que de calculer les distances exactes par un système de listes inversées, qui ne parcourt que les positions non nulles. L'approche proposée dans cette section, et illustrée par la figure 3, peut effectuer une telle recherche. La requête est effectuée en 1) produisant un ensemble de descripteurs à partir d'un vecteur SDM, qui sont 2) indexés séparément et 3) utilisés pour estimer une distance avec les images indexées. Ces étapes sont détaillées ci-dessous.

4.1 Agrégations du vocabulaire visuel

La première étape de notre approche produit un ensemble de descripteurs d'images à partir d'un SDM. Chacun de ces descripteurs est obtenu en agrégeant plusieurs composantes du SDM en entrée. Pour cela, nous introduisons un ensemble de matrices creuses $\mathcal{A} = \{A_1, \ldots, A_m\}$ de taille $d \times k$, où d est la dimension du descripteur en sortie et k la dimension du SDM d'entrée. La matrice A_j est appelée un *agrégateur*, car elle groupe plusieurs cellules du quantificateur initial. Par exemple, pour $k = 12$ et $d = 3$, nous définissons le premier agrégateur de la manière suivante :

$$A_1 = \underbrace{\left[\begin{array}{cccccccccccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right]}_{k} \Bigg\} d \quad (1)$$

Le nombre de composantes non nulles dans chaque ligne d'un agrégateur est $n_z = k/d$. Ce nombre est choisi de manière à produire un vecteur de dimension raisonnable, c'est-à-dire pour lequel une stratégie de recherche approximative dense peut fonctionner. Si nous choisissons, par exemple, $n_z = 8$ pour $k = 1000$, nous obtenons des descripteurs de dimension $d = 125$. Remarquez que ce choix de A_1 dans (1) est effectué sans perte de généralité car il n'y a pas d'ordre particulier entre les mots visuels.

Les autres agrégateurs sont obtenus en permutant le vecteur SDM d'entrée. Par exemple, pour $k = 12$ et $d = 3$, la permutation aléatoire (11, 2, 12, 8, 9, 4, 10, 1, 7, 5, 6, 3), produit la matrice d'agrégation

$$A_2 = \left[\begin{array}{cccccccccccc} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \quad (2)$$

Multiplier l'agrégateur A_j par le vecteur SDM f_i produit le vecteur

$$\omega_{i,j} = A_j \times f_i \quad (3)$$

de dimension d. Il peut être vu comme le vecteur SDM qui résulte de l'agrégation, dans l'espace de description initial, des cellules de Voronoï correspondant aux mots visuels groupés par la matrice d'agrégation. Le vecteur obtenu est appelé *miniSDM*. À ce stade, une image i est décrite par les m miniSDM $\omega_{i,j}$, $1 \leq j \leq m$, obtenus par la multiplication du SDM d'entrée par les agrégateurs $A_1, \ldots, A_m$.

4.2 Structure d'index

Nous bâtissons notre structure d'index à partir de l'approche de Hamming Embedding (HE), introduite dans [4] pour indexer les descripteurs locaux. Cette structure représente un vecteur par un index de quantification, qui localise grossièrement le descripteur, et un vecteur binaire qui raffine sa position au sein de la cellule de Voronoï identifiée par la quantification. Une structure d'indexation est construite pour chaque agrégateur A_j, soit m fichiers inversés au total. Chaque fichier inversé contient une entrée par image de la base.

Quantification. Le miniSDM $\omega_{i,j}$ est quantifié par le quantificateur q_j associé au $j^{\text{ème}}$ agrégateur, ce qui produit un index de quantification $c_{i,j} = q_j(\omega_{i,j}) \in \{1, .., k'\}$, où k' est le nombre de centroïdes du quantificateur.
L'ensemble des quantificateurs vectoriels $q_j(.), 1 \leq j \leq m$ est préalablement appris sur un grand nombre de miniSDM provenant de la base d'images Flickr1M*. Notons que la taille du dictionnaire k' est indépendante de celle utilisée pour quantifier les descripteurs SIFT : nous pouvons choisir $k \neq k'$.

Génération de la signature binaire. L'objectif de cette étape est de calculer une signature binaire $b_{i,j}$ de longueur d bits qui raffine la localisation du miniSDM au sein de la cellule identifiée par la quantification. Nous utilisons le même procédé de génération de signature que dans [4] :

1. Le miniSDM $\omega_{i,j}$ est "blanchi" par une matrice de rotation aléatoire R de taille $d \times d$.

2. Chaque bit du vecteur $b_{i,j}$ est obtenu en comparant chaque composante du vecteur $R\omega_{i,j}$ à une valeur seuil. Ce seuil est préalablement calculé comme la médiane de la composante considérée sur le jeu d'apprentissage.

Le $j^{\text{ème}}$ miniSDM associé à l'image i est représenté par le couple $(c_{i,j}, b_{i,j})$, qui est stocké dans un fichier inversé [18, 19]. Cette structure est composée de k' listes inversées, une pour chaque valeur possible du quantificateur q_i. Le couple est stocké dans la liste n° $c_{i,j}$, sous la forme d'une entrée $(i, b_{i,j})$. L'occupation mémoire d'un miniSDM est ainsi donné par

- 4 octets pour l'identifiant d'image i ;
- $\lceil d/8 \rceil$ octets pour le vecteur binaire $b_{i,j}$,

où l'arrondi garantit que les entrées sont alignées en mémoire, afin d'éviter la manipulation de bits. Étant donné que m fichiers inversés sont utilisés (un par agrégateur), l'usage mémoire total par image est de $C_i = m\ (4 + \lceil d/8 \rceil)$ octets.

Lors de la recherche, le couple $(c_{q,j}, b_{q,j})$, qui représente le $j^{\text{ème}}$ miniSDM associé avec l'image requête, est comparé à l'ensemble des éléments de la liste associée à l'index de quantification $c_{q,j}$. Contrairement à [4], le rôle de la signature binaire n'est pas de filtrer les éléments dont la distance de Hamming est en dessous d'un seuil, mais de fournir une distance entre le miniSDM requête et ceux indexées dans la liste inversée. Cette mesure sera utilisée par l'algorithme de fusion détaillé en 4.3.

Assignement multiple. Côté requête, ne considérer que la seule liste associée à l'index de quantification $c_{i,j}$, comme effectué dans [4], n'est pas suffisant pour notre approche, car la probabilité empirique de retourner les plus proches voisins est moins bonne que pour les descripteurs SIFTs. Pour surmonter ce problème, nous adoptons une stratégie similaire à celle proposée dans [8] et [6] : l'assignement multiple. Dans notre cas, cette stratégie consiste à parcourir les listes inversées associées aux t mots visuels les plus proches du miniSDM requête, au lieu de la seule liste associée à l'index de quantification $c_{i,j}$.

Cette stratégie ne modifie pas l'usage mémoire de la structure. En revanche, elle multiplie par un facteur t le nombre de listes inversées visitées et donc le nombre moyen d'entrées parcourues.

4.3 Fusion : espérance de distance

En sortie de la structure d'indexation, nous avons obtenu, pour chaque agrégateur j, une liste d'images potentiellement intéressantes et les distances de Hamming associées. Nous indiquons ici comment ces mesures sont utilisées pour classer les images.

Critère d'espérance de distance. Pour le $j^{\text{ième}}$ agrégateur, et l'image i de la base, nous nous plaçons dans la cellule de quantification associée à son mot visuel c_{ij}. Nous notons par $b_{q,j}$ la signature associée à l'image requête dans cette cellule, et par $b_{i,j}$ celle associée à l'image i. La concaténation $b_q = [b_{q,1}, \ldots, b_{q,m}]$ des signatures binaires provenant de tous les agrégateurs est une représentation de l'image requête. De la même manière, $b_i = [b_{i,1}, \ldots, b_{i,m}]$ représente l'image i de la base indexée. La distance entre ces deux vecteurs peut s'écrire

$$h(b_q, b_i) = \sum_{1 \le j \le m} h(b_{q,j}, b_{i,j}), \tag{4}$$

où $h(x, y)$ est la distance de Hamming entre deux vecteurs binaires x et y. Pour la plupart des images de la base et des agrégateurs, les distances $h(b_{q,j}, b_{i,j})$ n'ont pas été mesurées, car seule une faible proportion des entrées du fichier inversé a été parcourue. Néanmoins, la méthode de génération de la signature binaire (voir 4.2) garantit que l'espérance de cette distance est $\hat{h}(b_{q,j}, b_{i,j}) = d/2$. Nous pouvons donc calculer l'espérance de la distance dans (4) de la manière suivante :

$$\hat{h}(b_q, b_i) = \sum_{1 \le j \le m} \hat{h}(b_{q,j}, b_{i,j}) \tag{5}$$

où

$$\hat{h}(b_{q,j}, b_{i,j}) = \begin{cases} h(b_{q,j}, b_{i,j}) & \text{si } b_{i,j} \text{ est connu,} \\ d/2 & \text{sinon.} \end{cases} \tag{6}$$

Remarquons que, dans (4), l'index de quantification n'apparaît pas. Cet index est seulement utilisé pour sélectionner les vecteurs miniSDM qui sont susceptibles d'être proche du miniSDM requête, et pour lequel il est intéressant d'effectuer une mesure. La figure 4 montre la distribution empirique des distances de Hamming au sein d'une cellule de quantification. Il apparaît clairement que la probabilité d'avoir une distance petite entre deux images sans rapport est plus faible qu'entre deux images similaires. La combinaison de plusieurs miniSDM amplifie cette séparation.

Détails d'implantation. Pour la plupart des images, nous n'avons mesuré aucune distance dans (5), ce qui signifie que la distance par défaut est $m \times d/2$. Le score associé à une image peut donc être obtenu comme une somme sur les distances *observées* :

$$\text{score}_q(i) = \sum_{j / b_{i,j} \text{ est observé}} d/2 - h(b_{q,j}, b_{i,j}), \tag{7}$$

qui est égale à 0 pour les images pour lesquelles aucune signature binaire n'a été observée, et à $d \times m/2$ si l'image de la base est la requête elle-même. Ce score donne un classement des images identique à celui de (5), et est calculé lors du parcours de la structure d'indexation, par accumulation des scores dans une table de hachage.

Le temps de requête peut être amélioré en utilisant un seuil τ sur la distance de Hamming [4]. Le gain provient d'une part de la diminution du nombre de mises à jour des scores, qui génèrent des défauts de cache, et de la diminution du nombre de "hits", c'est-à-dire le nombre d'images ayant un

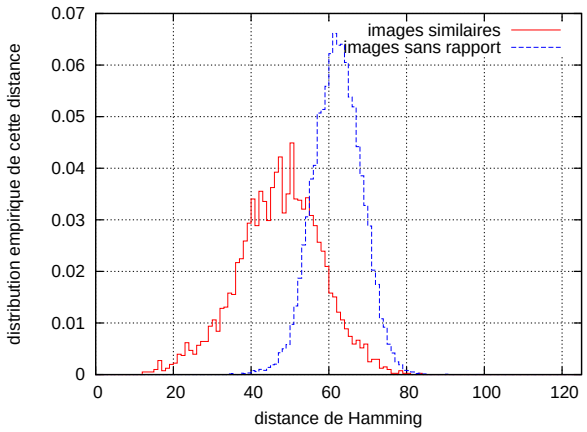

FIG. 4 – Distribution empirique des distances de Hamming entre les signatures binaires indexées dans la même liste inversée pour 1) des images similaires et 2) des images sans rapport. Ces mesures ont été effectué sur la base Holidays.

score. Dans la suite, nous avons fixé $\tau = d/2$, ce qui divise approximativement par 2 le nombre de mise à jour des scores. Ce choix revient à donner la même pénalisation aux images qui sont éloignées au sein de la même cellule et aux images pour lesquelles aucune mesure n'a été effectuée.

5 Évaluation

Nous évaluons notre approche selon le protocole introduit au 2.2. Les résultats sont présentés dans les tables 1 et 2. Les expériences effectuées sur Holidays+Flickr1M sont fournies dans la table 3 et la figure 5.

Sauf contre-indication, nous avons utilisé les paramètres suivants pour notre approche dans toutes les expériences :

$k =$	1000	dictionnaire associé aux SIFT
$n_z =$	8	agrégateur : positions non nulles/ligne
$d = k/n_z =$	125	dimension d'un miniSDM
$k' =$	20000	dictionnaires associés aux miniSDM
$t =$	100	assignement multiple

La meilleure pertinence est obtenue en utilisant une valeur de n_z comprise entre 8 et 12. Afin de limiter l'utilisation mémoire, qui dépend fortement du paramètre $d = k/n_z$, nous utilisons un petit dictionnaire pour les SIFTs : k=1000. Les signatures associées aux miniSDM sont donc de longueur 125.

SDM *vs* SDM binaires et compressés. Les tables 1 à 3 montrent l'excellent comportement des SDM binaires qui, pour les tailles de vocabulaires considérées, offrent une pertinence même supérieure à l'approche SDM de référence, et sont moins gourmands en mémoire. Le nombre de *hits* est le même en binaire, car celui-ci ne dépend que du nombre d'images dont le vecteur SDM partage au moins une position non nulle avec celui de l'image requête. Une

méthode	k	mAP	usage mémoire	*hits*
SDM	1k	0.414	3,087	1,484
SDM	20k	0.446	10,364	1,471
SDM	200k	0.549	12,886	1,412
SDM binaire	20k	0.458	8,291	1,471
SDM binaire	200k	0.554	10,309	1,412
SDM binaire compressé*	20k	0.458	1,174	1,471
SDM binaire compressé*	200k	0.554	1,830	1,412
miniSDM, m=1	1k	0.255	20	19
miniSDM, m=4	1k	0.368	80	48
miniSDM, m=8	1k	0.403	160	68
miniSDM, m=16	1k	0.426	320	93
miniSDM, m=32	1k	0.452	640	120

TAB. 1 – Comparaison de différentes approches à base de SDM sur la base Holidays : pertinence (mAP), ressources mémoires (octets par images de la base) et nombre moyen de *hits* par image requête. (ce nombre doit être comparé au nombre total d'images : 1491). Le paramètre m est le nombre de miniSDM en parallèle ; *estimation à partir de l'entropie marginale du SDM binaire.

méthode	k	score	usage mémoire	*hits*
SDM	20k	2.92	6,662	9,928
SDM binaire	20k	3.02	5,329	9,928
miniSDM, m=1	1k	2.07	20	94
miniSDM, m=8	1k	2.72	160	383
miniSDM, m=16	1k	2.83	320	567
miniSDM, m=64	1k	2.93	1,280	1,078

TAB. 2 – Évaluation sur la base Kentucky : score (4× rappel à 4), nombre moyen de *hits* et utilisation mémoire par image.

entrée d'une liste inversée occupe 4 octets (pour l'identifiant de l'image) dans le cas du SDM binaire, et 5 dans le cas classique en stockant le nombre d'occurrences (et non la fréquence). Cette optimisation nécessite d'effectuer la normalisation à la volée, lors du parcours des listes inversées.

Les performances du SDM binaire compressé sont indiquées dans la table 1. Comme l'information exploitée est la même que dans le cas non compressé, la pertinence est identique et obtenue pour un coût mémoire moindre.

MiniSDM. Le comportement de notre approche miniSDM sur les deux bases de référence est donné dans les tables 1 et 2. Avec une utilisation mémoire divisée par au minimum 10, les résultats obtenus sont similaires à la référence SDM. Sur Holidays, la mesure mAP est de 0.452 pour $m = 16$ agrégateurs, ce qui est comparable au 0.446 obtenu pour un SDM associé à un vocabulaire visuel de

$k = 20$k mots visuels. De plus, cette performance est obtenue avec seulement 320 octets par image, soit 32 fois moins que pour l'approche classique. Ces résultats sont confirmés par les mesures sur la base Kentucky, où pour $m = 16$ le score obtenu est de 2.83, à comparer avec le score de 2.85 obtenu dans [2] pour 512 "sketches", c'est-à-dire avec un ordre de grandeur plus de mémoire.

De manière surprenante, pour $m \geq 16$ la pertinence de notre approche est supérieure à celle obtenue par une recherche exacte du SDM d'origine (avec 1000 mots visuels). Cela pourrait être dû à la sous-optimalité de la distance Euclidienne pour la représentation SDM. En effet, pour la base Kentucky, il est connu [10] que cette distance est significativement moins bonne que la distance L1. L'autre propriété intéressante est le faible nombre de *hits*, qui montre l'excellente sélectivité de notre approche : elle ne retourne que 6% des images pour $m = 8$.

La pertinence de notre méthode est inférieure à la méthode de [4], qui obtient mAP=0.751 dans sa meilleure configuration. Rappelons, cependant, que cette approche nécessite 35 ko par image pour Holidays, c'est-à-dire plus de 100 fois plus de mémoire que les miniSDM.

Holidays+Flickr1M : expériences à grande échelle. La figure 5 donne le rappel@N, qui reflète la capacité du système à filtrer un grand nombre d'images, en vue de sélectionner une liste d'images à comparer plus finement dans une étape de post-classement [7, 12]. Nos résultats sont comparables à la référence SDM, ce qui est excellent étant donné ses moindres besoins en ressources (usage mémoire et temps de requête). Notre approche requiert 160 Mo pour $m = 8$ et la requête est effectuée en 132 ms, à comparer avec les 8 Go et 3 s de SDM. Des mesures complémentaires données dans la table 3 confirment les observations précédentes. Pour conclure, soulignons que pour $m = 8$, nous pouvons indexer environ 350 million d'images en mémoire sur une machine bien équipée en RAM (64 Go).

Des résultats typiques de requêtes sont illustrés dans la figure 6. Ces résultats, bien qu'incorrects pour certains au sens de la vérité terrain, sont visuellement ressemblants.

6 Conclusion

Nous avons introduit dans cet article une manière de compacter la représentation par sac de mots : les miniSDM. Cette représentation est obtenue par agrégation de plusieurs cellules du vocabulaire visuel et est compactée. L'utilisation de plusieurs miniSDM donne une représentation redondante du sac de mots initial. Une structure d'indexation efficace combinée avec une fusion par espérances de distances rend la recherche très efficace. Cette approche réduit l'usage mémoire et le temps de recherche d'un à deux ordres de grandeur. De plus, elle réduit significativement la quantité de mémoire parcourue. Les expériences montrent la pertinence de la recherche obtenue avec des représentations extrêmement compactes.

méthode	k	mAP	usage mémoire	mémoire parcourue	temps de recherche
SDM	20k	0.227	7,322	860	22163
SDM	200k	0.315	8,885	148	2827
SDM binaire	20k	0.307	5,858	688	14073
SDM binaire	200k	0.381	7,108	117	2562
miniSDM, m=1	1k	0.066	20	0.19	71
miniSDM, m=8	1k	0.196	160	1.54	132
miniSDM, m=32	1k	0.244	640	6.14	352

TAB. 3 – Expériences sur Holidays+ Flickr1M : temps de requête par image (en ms, pour un cœur processeur), quantité de mémoire utilisée par la structure d'indexation, quantité de mémoire parcourue pendant la recherche, et pertinence de la recherche (mAP).

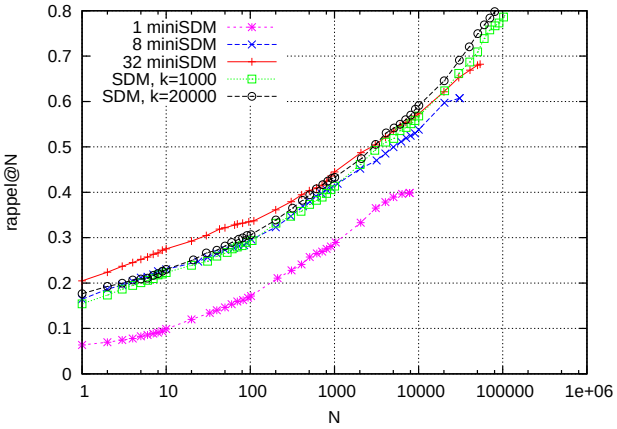

FIG. 5 – Holidays+Flickr1M : répartition des rangs des vrais positifs (rappel@N) pour un million d'images.

Remerciements

Nous remercions les projets ANR RAFFUT et GAIA, ainsi que le projet QUAERO, pour leur support financier.

Références

[1] O. Chum, J. Philbin, M. Isard, and A. Zisserman. Scalable near identical image and shot detection. In *CIVR*, 2007.

[2] O. Chum, J. Philbin, and A. Zisserman. Near duplicate image detection : min-hash and tf-idf weighting. In *BMVC*, 2008.

[3] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the Symposium on Computational Geometry*, pages 253–262, 2004.

[4] H. Jégou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *Proceedings of the European Conference on Computer Vision*, October 2008.

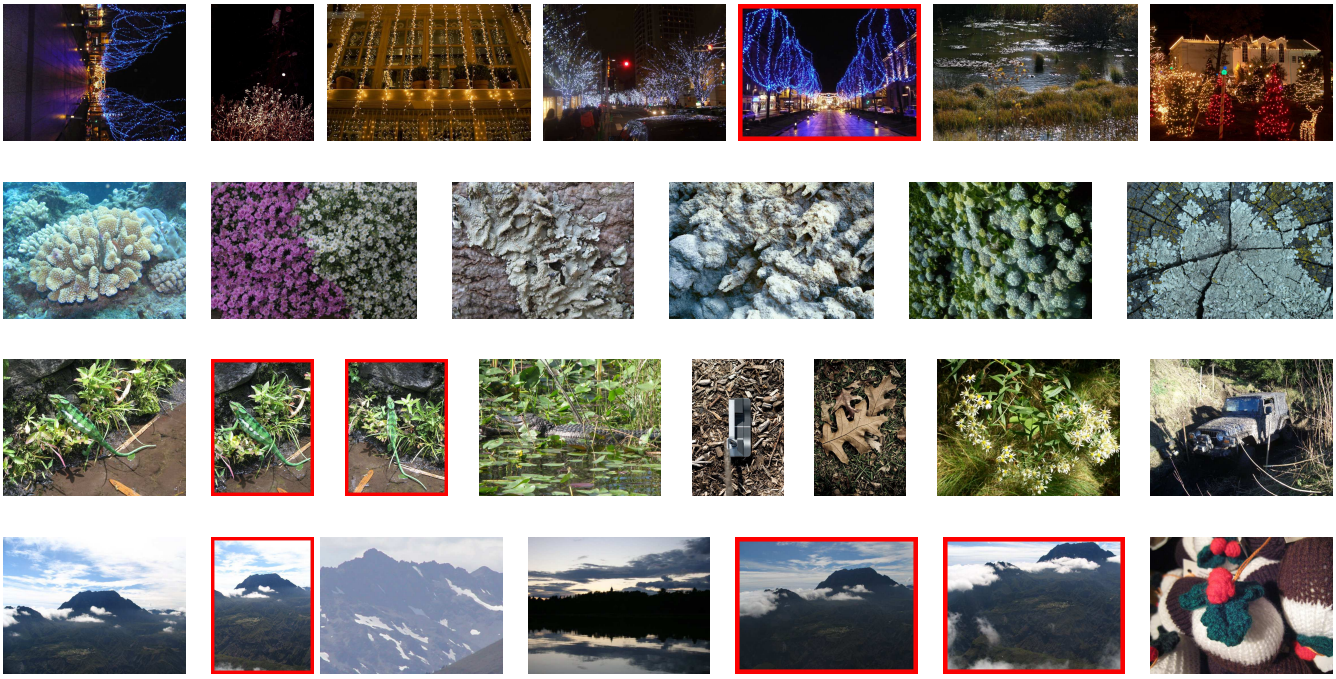

FIG. 6 – Exemples de requêtes images dans la base Holidays+Flickr1M. L'image requête se trouve à gauche, et les résultats renvoyés par l'approche miniSDM sont présentés par ordre décroissant de pertinence de gauche à droite. Notez que nous n'utilisons pas la couleur. Les vrais positifs, au sens de la vérité terrain fournie avec la base Holidays, sont encadrés en rouge.

[5] H. Jégou, M. Douze, and C. Schmid. On the burstiness of visual elements. In *Conference on Computer Vision & Pattern Recognition*, June 2009.

[6] H. Jégou, H. Harzallah, and C. Schmid. A contextual dissimilarity measure for accurate and efficient image search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[7] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2) :91–110, 2004.

[8] 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 International Conference on Very Large DataBases*, pages 950–961, 2007.

[9] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1) :63–86, 2004.

[10] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2161–2168, 2006.

[11] M. Perdoch, O. Chum, and J. Matas. Efficient representation of local geometry for large scale object retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2009.

[12] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[13] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization : Improving particular object retrieval in large scale image databases. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[14] J. Sivic and A. Zisserman. Video Google : A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision*, pages 1470–1477, 2003.

[15] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Advances in Neural Information Processing Systems*, 2008.

[16] I. H. Witten, R. M. Neal, and J. G. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6) :520–540, 1987.

[17] J. Zhang, X. Long, and T. Suel. Performance of compressed inverted list caching in search engines. In *WWW '08 : Proceeding of the 17th international conference on World Wide Web*, pages 387–396, 2008.

[18] J. Zobel and A. Moffat. Inverted files for text search engines. *ACM Computing Surveys*, 38(2) :6, 2006.

[19] J. Zobel, A. Moffat, and K. Ramamohanarao. Inverted files versus signature files for text indexing. *ACM Transactions on Database Systems*, 23(4) :453–490, 1998.