SC-LSH: Une Méthode d'Indexation pour une Recherche de Similarité Approximative dans l'Espace Multidimensionnel

Sanaa Chafik*/, Imane Daoudi***, Mounim A. El Yacoubi* et Hamid El Ouardi****

**Telecom SudParis / Mines-Telecom Institute*
9 rue Charles Fourier, 91011, Evry Cedex France
{sanaa.chafik,mounim.el_yacoubi}@telecom-sudparis.eu
***Hassan II Ain Chock University - ENSEM*
****Hassan II Ain Chock University - ENSEM – LISER/GREENTIC*
Route d'El Jadida, km 7, BP: 8118, Oasis – Casablanca Maroc
{i.daoudi,h.elouardi}@ensem.ac.ma

RÉSUMÉ. *Locality Sensitive Hashing (LSH) est l'une des techniques les plus prometteuses pour la résolution des problèmes de la recherche des plus proches voisins dans l'espace de grande dimension. Euclidien Exact LSH (E2LSH) est la variante la plus populaire du LSH qui a été appliquée avec succès dans de nombreuses applications multimédia. Toutefois, l'E2LSH présente des limitations qui affectent les performances de recherche. La principale limitation de l'E2LSH est l'espace mémoire important utilisé. Afin de parvenir à une bonne qualité de recherche, un grand nombre de tables de hachage est nécessaire. Ce papier propose un nouvel algorithme de hachage pour remédier au problème d'espace de stockage, tout en conservant la bonne qualité de recherche et un meilleur temps de calcul. Les résultats expérimentaux obtenus sur une base de données réelle à grand échelle montrent l'intérêt de notre approche.*

ABSTRACT. *Locality Sensitive Hashing (LSH) is one of the most promising techniques for solving nearest Neighbours search problem in high dimensional space. Euclidean LSH is the most popular variation of LSH that has been successfully applied in many multimedia applications. However, the Euclidean LSH presents limitations that affect search performances. The main limitation of the Euclidean LSH is the large memory consumption. In order to achieve a good accuracy, a large number of hash tables is required. This paper propose a new hashing algorithm to overcome the storage space problem, while keeping a good accuracy and better query time. The Experimental results on a real large-scale dataset show the interest of our approach..*

MOTS-CLÉS : *Recherche des Plus Proches Voisins, Recherche d'Image par le Contenu (CBIR), Malédiction de la Dimension, LSH, Hachage Multidimensionnelle, Passage à l'Echelle.*

KEYWORDS: *Nearest Neighbour Search, Content Based Image Retrieval (CBIR), Curse of Dimensionality, LSH, Multidimensional Indexing, Scalability.*

1. Introduction

Avec l'évolution de la technologie numérique et la prolifération de l'internet, les données multimédia envahissent de nos jours le monde de l'information. Le partage et l'échange des données multimédia ont poussé le monde de la recherche à mettre en place des techniques efficaces permettant la gestion et l'organisation de ce type de données. Ces techniques font appel à des outils d'*indexation* et de *structuration* (connu sous le nom de l'indexation multidimensionnelle), que l'on trouve dans deux domaines de compétences différents : *l'analyse d'images* et *les bases de données.*

Les recherches en analyse d'images se sont concentrées sur la mise au point d'approches de description -appelées aussi *indexation*- du contenu visuel de l'image qui soient performantes en termes de reconnaissance. Les recherches en bases de données quant à elles, ont pour objectif de développer des techniques d'indexation multidimensionnelle permettant de faciliter la recherche dans les grands volumes de données.

Les architectures développées en bases de données sont organisées selon deux grandes familles : les méthodes d'indexation conventionnelle et approximative.

Les méthodes d'indexation conventionnelle permettent l'indexation des vecteurs multidimensionnels selon leur localisation et distribution dans l'espace multidimensionnel tel que le R-tree (Guttman, 1984) et KD-tree (Bently, 1990) et leurs dérivées.

Malheureusement cette famille de méthodes souffre du problème connu sous le nom de la *malédiction de la dimension* « The curse of dimensionnality » (Weber, 1998). En effet à partir de la dimension $16\,(d>16)$, les performances de ces techniques se dégradent et deviennent comparables ou inferieurs à une recherche linéaire. La deuxième famille de méthodes a été proposée pour apporter des solutions au problème de la malédiction de la dimension qui se pose dans les espaces de grandes dimensions. Cette famille repose sur la compression de données et propose des techniques de recherche approximatives fournissant des résultats légèrement moins précis, mais offrant par contre une forte réduction de temps de calcul. Nous trouvons dans cette famille, des méthodes comme le VA-File (Blott et al., 1990) et ses variantes (Ye et al., 2010), (Lu et al., 2006), les familles LSH (Indyk et al., 1998), (Datar, 2004), (Gorisse et al., 2012) et leurs variantes (Wang et al., 2013), (Andoni et al., 2006), (Gan et al., 2013), (Lv et al., 2007), BitMatrix (Calistru et al., 2012), KRA+-Block (Daoudi et al., 2008), multi-dimensional inverted index (Feng et al., 2013), etc. Malheureusement les architectures d'indexation et de recherche disponibles aujourd'hui présentent plusieurs limitations notamment au passage à l'échelle : 1) un temps de calcul assez important 2) Limitation de l'espace mémoire 3) dégradation de la qualité de la recherche.

Nous nous sommes intéressés dans cet article à l'une des méthodes les plus populaires basée sur le concept du Locality Sensitive Hashing (LSH), appelée Exact Eulidean LSH ou E2LSH, permettant la recherche approximative dans les grands espaces multidimensionnels. Le choix de cette classe de méthode est justifié par la qualité et l'efficacité de cette dernière en comparaison à différentes méthodes d'indexation multidimensionnelles (Liu et al., 2004), (Chafik et al., 2014), Bien que

l'approche E2LSH soit prometteuse pour résoudre le problème de recherche des plus proches voisins *(ppv)*, elle présente des limitations qui affectent la structure et les performances de recherche. La principale limitation de l'E2LSH est l'importance de l'espace mémoire occupé. Afin d'augmenter la probabilité d'attribution d'une même valeur de hachage aux vecteurs similaires, la méthode E2LSH utilise plusieurs fonctions de hachage aléatoires et indépendantes, ce qui conduit à une sévère redondance des valeurs de hachage et une redondance des tables de hachage. En outre, lors de l'utilisation des grands ensembles de données à grande échelle, plusieurs tables de hachage sont nécessaires conduisant ainsi à une forte consommation d'espace mémoire.

Dans cet article, nous proposons une nouvelle approche fondée sur le concept du LSH appelé *Symmetries of the Cube LSH* (SC-LSH) pour une recherche multidimensionnelle efficace des *pp*v. Le nouveau système SC-LSH permet l'utilisation de fonctions de hachage dépendantes et aléatoires, réduisant la redondance des valeurs de hachage et augmente ainsi la probabilité de collision des vecteurs similaires. Le caractère semi aléatoire de la nouvelle approche permet la réduction du nombre de table de hachage et donc moins d'espace mémoire requis, de plus le SC-LSH fournit une bonne qualité de recherche similaire à l'approche E2LSH dans un temps restreint.

Le reste de cet article est organisé comme suit. Nous discutons d'abord les préliminaires dans la section 2, puis nous présentons notre algorithme dans la section 3. La section 4 comporte les résultats expérimentaux obtenus comparant l'approche proposée avec l'E2LSH. Enfin, nos conclusions et perspectives sont présentées dans la section 5.

2. Préliminaires

Dans cette section, nous définissons le concept du Locality Sensitive Hashing, puis, nous passons en revue les principales familles LSH existantes adaptées pour différentes mesures de similarité. Ensuite, nous détaillons la technique E2LSH adaptée à la distance euclidienne.

d	Dimension
n	Nombre de vecteur dans la base de données
$d(.,.)$	Distance
$h_{i,j}$	Fonction de hachage
ppv	Plus Proches voisins
K	Nombre de fonction de hachage
L	Nombre de table de hachage
$B(q,r)$	Boule de rayon r et de centre q
$\|.,.\|$	Distance euclidienne
$P_r[.]$	Probabilité
f_p	Fonction de densité de probabilité

$\Re$	Ensemble des nombres Réels
U	Univers
Z	Ensemble des nombres entiers relatifs
$O(.)$	Complexité

Tableau 1. *Notations*

2.1. *Locality Sensitive Hashing*

Locality Sensitive Hashing (LSH) (Andoni et al., 2006) est l'une des méthodes les plus populaires et les plus efficaces dans le domaine de l'indexation multidimensionnelle approximative. Le concept du LSH repose sur l'utilisation de plusieurs fonctions de hachage, décrivant la répartition des données dans l'espace multidimensionnel. Cette méthode permet d'attribuer la même valeur de hachage aux vecteurs multidimensionnels les plus proches, ainsi les vecteurs ayant une même valeur de hachage ont une forte probabilité d'être similaires. Cette approche est une solution aux problèmes de recherche des plus proches voisins (*ppv*) dans l'espace de grande dimension. Les vecteurs voisins d'une requête donnée sont obtenus à partir de la valeur de hachage de cette dernière et les valeurs de hachage de l'ensemble des vecteurs de l'espace de données. Les valeurs de hachage des vecteurs multidimensionnels sont obtenues à partir des fonctions de hachage appliquées sur tous les vecteurs de l'espace, et sont ensuite placées dans les buckets[1] des tables de hachage. Cette structuration sous forme de tables de hachage permet de faciliter et d'accélérer la recherche des vecteurs multidimensionnels. Nous constatons que le concept du LSH dépend étroitement des fonctions dites *locality sensitive*.
Une famille de fonctions de hachage F est dite une famille *locality sensitive* si elle satisfait les définitions suivantes:

Définition 1. (Indyk et al., 1998) *Une famille LSH* $F = \{\Re^d \to U\}$ *est dite* (r_1, r_2, P_1, P_2) *-sensitive, pour tout* $p, q \in \Re^d$ *, avec* $P_1 > P_2$ *et* $r_1 < r_2$ *:*

- Si $p \in B(q, r_1)$ alors $P_{rF}[h(p) = h(q)] \geq P_1$
- Si $p \notin B(q, r_2)$ alors $P_{rF}[h(p) = h(q)] \leq P_2$

Définition 2. (Charikar, 2002) *Une famille LSH* $F = \{\Re^d \to U\}$ *est dite locality sensitive si on a la propriété suivante ;*

$$P_{rF}[h(p) = h(q)] = sim(p, q)$$

Avec $sim(p, q)$ *une fonction de mesure de similarité.*

Afin d'augmenter l'écart entre la forte probabilité P_1 et la faible probabilité P_2, et d'augmenter ainsi le fossé entre une bonne et une fausse détection, il est nécessaire d'utiliser plusieurs fonctions de hachage g_j définie par : $g_j : \Re^d \longrightarrow Z^k$,

1. Bucket: Type de données qui regroupe les objets entre eux, le terme est utilisé dans les algorithmes de hachage, où différents objets qui ont le même code de hachage (valeur de hachage) vont dans le même "seau".

avec $g_j(q) = (h_{1,j}(q),\ldots,h_{K,j}(q))$, où $h_{t,j}(1 \le t \le K, 1 \le j \le L)$ sont des fonctions sélectionnées indépendamment et aléatoirement à partir de F. Ces dernières vont permettre le hachage de l'ensemble des vecteurs de données et de construire ainsi la structure d'index, en plaçant chaque vecteur p dans le bucket correspondant à une fonction $g_j(p)$. Notant que les buckets stockent uniquement les références des vecteurs afin de faciliter l'accès aux vecteurs de données.
L'algorithme de recherche d'une requête q commence par le calcul des valeurs de hachage $g_1(q),\ldots,g_L(q)$. Les distances entre la requête et les vecteurs du même bucket sont ensuite calculées. Enfin, les vecteurs les plus proches de la requête sont retenus, ils représentent les résultats de la recherche.

2.2. *Les Familles LSH*

Au cours de ces dernières années, plusieurs familles LSH ont été proposées, caractérisée chacune par une fonction de hachage particulière. Dans la suite, nous passons en revue les principales familles LSH existantes.

— ***Min-hash (Broder et al., 1998) :*** est une famille de fonction de hachage, se basant sur l'estimation de la similarité entre deux ensembles. Elle consiste à sélectionner aléatoirement une permutation π de l`espace U. Pour un ensemble A de données, la fonction de hachage est définie par :

$$h(x) = \min\{\pi(a) \mid a \in A\}$$

La probabilité de collision entre deux ensembles A et B est définie par $P_r[h(A) = h(B)] = s(A,B)$, où $s(A,B)$ est le coefficient de Jaccard [19], défini par :

$$s(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Cette technique est souvent utilisée par les moteurs de recherche pour la détection des pages web redondantes. Elle est utilisée aussi comme un outil pour l'apprentissage des règles d'association : Clustering en grande échelle.

— ***Hachage pour la distance de Hamming (Indyk et al., 1998) :*** c'est une famille de fonction de hachage adaptée à la distance de Hamming, dans un espace binaire de dimension d. Pour un vecteur binaire appartenant à $\{0,1\}^d$, la famille F de fonctions de hachage est l'ensemble des projections sur une des d coordonnées définie comme suit :

$$F = \{h : \{0,1\}^d \longrightarrow \{0,1\} \mid h(x) = x_i, i = 1,\ldots,d\}$$

Où x_i est la i^e coordonnée de x. La fonction h est sélectionnée aléatoirement à partir de F, elle permet de sélectionner un bit au hasard à partir des coordonnées du vecteur x.

— ***Hachage pour la similarité du cosinus (Charikar, 2002) :*** c'est une famille LSH constituée des fonctions de hachage suivantes :

$$h_{\vec{r}}(\vec{p}) = \begin{cases} 0 \ \ si \ \ \vec{p}.\vec{r} < 0 \\ 1 \ \ si \ \vec{p}.\vec{r} \geq 0 \end{cases}$$

Ces fonctions permettent le partitionnement de l'espace en deux sous espaces à travers un hyperplan choisi aléatoirement, et défini par le vecteur normal unitaire $\vec{r}$. La probabilité de collision entre deux vecteurs $\vec{p}$ *et* $\vec{q}$ est calculée comme suit :

$$P_r[h(\vec{p}) = h(\vec{q})] = 1 - \frac{\theta(\vec{p},\vec{q})}{\pi}$$

La projection aléatoire est une technique de hachage multidimensionnel permettant l'estimation de la similarité entre vecteurs en se basant sur la valeur binaire des fonctions de hachage.

— ***Hachage pour la distance*** l_p ***(Datar, 2004)*** : c'est une famille LSH regroupant l'ensemble des fonctions $h_{a,b}$: $\Re^d \longrightarrow Z$, définies par

$$h_{a,b}(v) = \left\lfloor \frac{a.v + b}{w} \right\rfloor$$

Où a est un vecteur aléatoire de la distribution p-stable, et b un nombre réel choisi uniformément de $[0, w]$, où $w \in \Re$. La probabilité de collision entre deux vecteurs $\vec{p}$ *et* $\vec{q}$ est définie par :

$$p(c) = P_r[h(p) = h(q)] = \int_0^w \frac{1}{c} f_p(\frac{t}{c})(1 - \frac{t}{w})dt$$

Cette technique de hachage se base sur la distribution p-stable $p \in [1,2]$ [26]. Les distributions stables les plus connues sont la distribution gaussienne (ou normale) pour l_2 et la distribution de Cauchy pour l_1 . Cette famille de fonction a prouvé son efficacité pour la mesure de similarité et la recherche dans les grandes dimensions [17].

— ***Hachage pour la distance*** χ^2 ***(Gorisse et al, 2012)*** : Une nouvelle famille LSH adaptée à la distance χ^2, sa fonction de hachage est définie par:

$$h_{a,b}(p) = \left\lfloor \frac{\sqrt{(8.\frac{a.p}{w^2} + 1)} - 1}{2} + b \right\rfloor$$

Où a est un vecteur aléatoire de la distribution p-stable, et b un nombre réel choisi uniformément de $[0,1]$, et w est un scalaire pour la longueur de quantification, calculé en utilisant la distance χ^2 [7]. Cette famille LSH a été appliquée avec succès dans le cadre de la recherche par le contenu des images et vidéos, lorsque les données sont représentées par des histogrammes.

Dans ce qui suit, nous décrivons l'une des plus célèbres techniques LSH, se basant sur la distance l_p, appelé Euclidian Exact LSH (E2LSH). Considérée comme une solution aux problèmes de recherche des *ppv* dans les grands espaces multidimensionnels,

2.3. *La Méthode E2LSH*

L'Exact Euclidean Locality Sensitive Hashing ou E2LSH[2] est une technique de hachage multidimensionnel pour la résolution du problème de la recherche des $(R,1-\delta)$ -*ppv* dans l'espace euclidien l_2. Pour une requête donnée q, l'E2LSH permet de retourner tous les vecteurs $p \in \Re^d$ de la boule $B(q,r)$ avec une probabilité égale au moins à $1-\delta$ (avec δ est la probabilité qu'un plus proche voisin p n'est pas retourné). La construction de la structure d'index du l'E2LSH s'effectue en calculant la valeur de hachage de chaque vecteur de l'ensemble de données, en utilisant la fonction de hachage $g_j(p) = (h_{1,j}(p),...,h_{K,j}(p))$ avec $(1 \le j \le L)$, ou $h_{a,b}(v)$: $\Re^d \longrightarrow Z$, est définie par :

$$h_{a,b}(v) = \left\lfloor \frac{a.v+b}{w} \right\rfloor$$

Où a est un vecteur aléatoire de la distribution p-stable, et b un nombre réel choisi uniformément de $[0,w]$, où $w \in \Re^d$. Les vecteurs p de l'ensemble de données ayant les mêmes valeurs de hachage sont placés dans les mêmes buckets $g_j(p)$ avec $(1 \le j \le L)$.

Pour s'assurer de la pertinence des résultats retournés, la méthode E2LSH utilise plusieurs tables et fonctions de hachage afin d'augmenter le fossé entre bonne et mauvaise détection. A partir de l'ensemble de données, les valeurs optimales du nombre de tables et fonctions de hachage sont calculées pour minimiser le temps de calcul de l'algorithme de recherche. Après la construction de la structure d'index l'E2LSH procède à la recherche de la requête qui consiste à calculer la valeur de hachage de la requête, et placer cette dernière dans le bucket adéquat. Ensuite, la distance euclidienne entre la requête et les vecteurs se trouvant dans le même bucket est calculée. Enfin, les vecteurs les plus similaires en termes de distance sont retournés.

Bien que l'E2LSH est une approche prometteuse pour résoudre le problème de recherche des *ppv*, mais présente une dégradation de performance en espace mémoire. En effet, pour assurer une bonne qualité de recherche l'approche E2LSH utilise plusieurs tables de hachage, qui nécessite un espace mémoire très important. La section suivante présente une nouvelle approche LSH pour remédier au problème

2. http://www.mit.edu/~andoni/LSH/.

de stockage de l'approche E2LSH.

3. Approche Proposée

Dans cette section, nous présentons la structure d'index et l'algorithme de recherche de notre nouvelle approche SC-LSH

3.1. *Structure d'Index*

Nous proposons un nouveau schéma basé sur le concept LSH que nous avons nommé *Symmetries of the Cube Locality Sensitive Hashing* (SC-LSH), pour la bonne gestion de l'espace mémoire. Le principe de cette nouvelle technique est la structuration de l'espace multidimensionnel en un ensemble de cubes disjoints. Pour ce faire, le SC-LSH propose la projection des vecteurs de l'espace multidimensionnel sur les axes de symétries du cube (Figure 1), en utilisant la fonction de hachage $h_{a,b}(v):\Re^d \longrightarrow Z$, définie par :

$$h_{a,b}(v) = \left\lfloor \frac{a.v+b}{w} \right\rfloor$$

Où a est un vecteur multidimensionnel représentant la direction des symétries du cube et b un nombre réel choisi uniformément de $[0,w]$, où w est nombre réel positive dépendant de la longueur de l'arête du cube e. Pour les diagonales de l'espace (ligne de couleur rose, Figure 1), la valeur de w est égale à $e\sqrt{3}$, et pour les axes de rotation (ligne de couleur bleu, Figure 1), la valeur de w est égale à e (Figure 1).

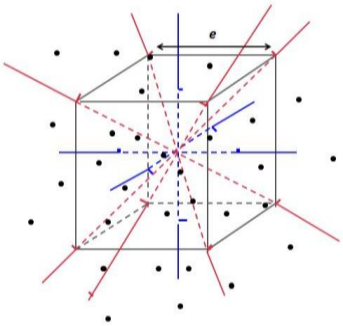

Figure 1. *Hachage avec les symétries du cube*

Après une analyse expérimentale des performances de l'approche proposée, nous avons restreint le nombre d'axes de symétries à sept. Le premier axe de symétrie est choisi aléatoirement suivant une distribution gaussienne et les autres axes sont générés à partir du premier axe en utilisant les propriétés mathématiques du cube. L'utilisation des symétries du cube pour le hachage multidimensionnel des vecteurs de l'espace permet la structuration de l'espace en un ensemble de cubes disjoints, ou

les vecteurs similaires seront placés dans le même cube de longueur d'arête e défini par la fonction : $g(p)=(h_1(p),...,h_7(p))$, (Figure 2). Afin d'assurer de bonne performance en qualité de recherche il est nécessaire d'utiliser plusieurs fonctions de hachage. Une étude expérimentale a été réalisée pour choisir le nombre de table de hachage approprié.

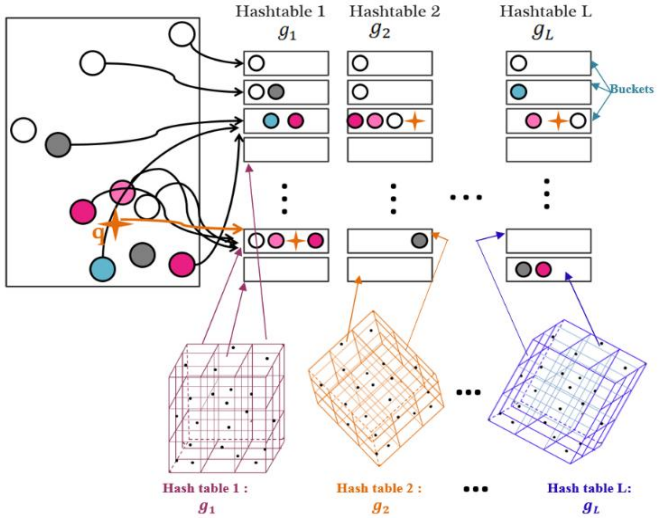

Figure 2. *Structure d'Index du SC-LSH*

3.2. *Algorithme de Recherche des PPV*

Comme dans l'approche E2LSH, notre algorithme de recherche commence d'abord par le calcul des valeurs de hachage du vecteur requête q, $g_j(q)=(h_{1,j}(q),...,h_{7,j}(q))$ avec $(1\le j\le L)$, ce qui permet d'indexer le vecteur requête dans le bucket approprié. Ensuite la distance euclidienne entre les vecteurs appartenant au même bucket et le vecteur requête est calculée déterminant ainsi les *ppv*.

4. Résultats Expérimentaux

Dans cette section, nous évaluons les performances de l'approche proposée pour la recherche des *ppv* sur une base de données réelle et synthétique. Nous comparons les performances du SC-LSH avec l'approche E2LSH.

4.1. *Bases de Données et Mesure d'Evaluation*

Nous avons réalisé nos expériences sur deux grandes bases de données :

SIFT1M[3]: une grande base de données d'image constituée d'un million de descripteurs SIFT extraits à partir d'image aléatoire, chaque descripteur est représenté par un vecteur multidimensionnel de 128 dimensions.
SynGIST: Une base de données synthétique crée à partir de la base d'image *GIST1M*[4].
Nous choisissons au hasard 100 vecteurs de la base de données pour former le fichier requête et nous utilisons les vecteurs restants pour former la base de données de test. Nous utilisions le même critère d'évaluation utilisé dans (Gan et al., 2013) pour évaluer la qualité de recherche.
Pour une requête donnée, soient $p_1, p_2,...,p_m$ les m*-ppv* retournés, triés dans l'ordre croissant de leurs distances de la requête q, et soient $p_1^*, p_2^*,...,p_m^*$ les vrais m*-ppv*, le *rank-i approximation ratio* est défini par :

$$R_i(q) = \frac{\|p_i, q\|}{\|p_i^*, q\|}$$

L'*overall ratio* (OR) est défini par:

$$\text{OR} = \frac{1}{m}\sum_{i=1}^{m} R_i(q)$$

Cette mesure permet de définir la qualité des résultats retournés, plus l'*overall ratio* est proche de un, plus les résultats sont plus précis, si l'*overall ratio* est égale à un, les résultats sont exacts.
Pour un ensemble de requête Q, l'*average overall ratio* est défini, représentant la qualité générale de tous les m*-ppv*. la qualité des voisins pour chaque rang est la moyenne du *rank-i approximation ratio* pour toutes les requêtes en Q :

$$\frac{1}{|Q|}\sum_{\forall q \in Q} R_i(q) .$$

Dans ce papier, nous utilisons l'*average rank-i ratio* et le *mean average rank-i ratio* (*MARR*), pour évaluer la qualité de la similarité de recherche. En outre, nous fournissons également le temps de calcul et l'espace mémoire occupé (taille de la structure d'index $O(nL)$).
Nos tests ont été réalisés sur une machine de distribution Linux Ubuntu 12,04 de 1.86GHz avec 6 Go de RAM et 146 Go de disque local.

4.2. *Sélection des Paramètres*

L'objectif principal de la sélection des paramètres optimaux est de choisir les paramètres adéquats pour permettre une recherche rapide et de bonne qualité, tout en utilisant le minimum de temps de calcul. L'approche proposée dépend

3. http://corpus-texmex.irisa.fr
4. http://corpus-texmex.irisa.fr

essentiellement du nombre de tables de hachage L qui définit la qualité de la recherche: un grand nombre de tables de hachage augmente la probabilité de bonne collision, par conséquent, il nécessite un grand espace mémoire.
L'approche SC-LSH dépend aussi de la longueur de l'arête du cube e, qui définit le volume du cube et le nombre de vecteurs indexé à l'intérieur du cube. Une grande longueur d'arête permet d'augmenter la probabilité de collision, par conséquent le temps de calcul augmente. Pour bien choisir la longueur optimale du côté e nous effectuons plusieurs tests sur différentes valeur de e . Nous avons remarqué que la valeur $e = 4$ fournit de bonnes performances en qualité et temps de calcul. Pour trouver le nombre optimal de tables de hachage L , nous effectuons plusieurs tests pour différentes valeurs de L sur la base de données *SIFT1M*. Le Tableau 2 résume les performances de la méthode de SC-LSH pour différentes valeurs de L .

L	**10**	**20**	**30**	**40**	**50**	**60**
MARR	0.725	0.741	0.747	0.749	0.750	0.750
Temps de Calcul (s)	0.1062	0.1536	0.1842	0.2086	0.2189	0.2358
Espace Mémoire (GB)	0 .1117	0.2235	0.3352	0.4470	0.5587	0.6705

Tableau 2. *Performances du SC-LSH pour différentes valeurs de L sur la base SIFT1M*

Dans la suite, nous avons fixé expérimentalement la valeur de l'arête du cube à $e = 4$, et nous avons choisi $L = 30$, le nombre approprié de table de hachage pour l'indexation de l'ensemble de données *SIFT1M.* Nous avons effectué les mêmes tests de performance sur les bases de données synthétiques *SynGIST* afin de définir le nombre approprié de tables de hachage.

4.3. *Performances sur une Base de Données Réelle*

Dans cette section, nous présentons les résultats de l'évaluation de l'approche proposée SC-LSH en utilisant la base de données *SIFT1M*, nous comparons les résultats obtenus avec la méthode E2LSH.

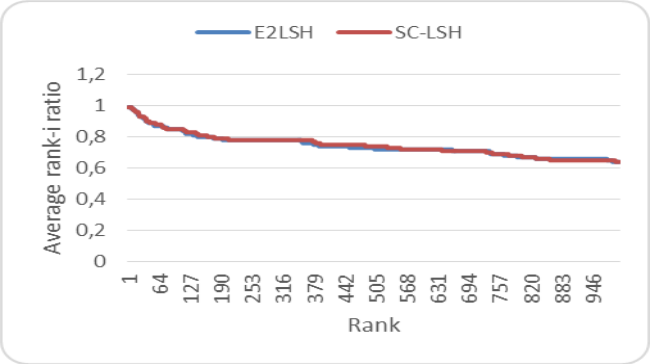

Figure 3. *L'Average rank-i ration sur la base de données SIFT1M pour 1000-ppv*

La Figure 3 montre le résultat du test de qualité de recherche des deux approches SC-LSH et E2LSH effectué sur l'ensemble des données *SIFT1M*. Nous remarquons que les deux méthodes fournissent presque une même qualité de recherche. L'approche proposée surpasse légèrement la méthode E2LSH. En fait, le *mean average rank-i ratio* de la SC-LSH est égale à 0.747, tandis que le *mean average rank-i ratio* du LSH euclidien est égale à 0.744. De plus, l'approche proposée nécessite moins de temps de calcul et d'espace mémoire en comparaison à l'approche E2LSH (voir Tableau 3).

SC-LSH	**Temps de calcul (s)**	0.1842
	Espace Mémoire (GB)	0.3352
E2LSH	**Temps de calcul (s)**	1.0343
	Espace Mémoire (GB)	4.2244

Tableau 3. *Performances du SC-LSH et de l'E2LSH sur la base SIFT1M*

4.4. *Performances sur une Base de Données Synthétique*

Pour évaluer le comportement de l'approche SC-LSH au passage à l'échelle, nous avons effectué plusieurs tests sur différentes bases de données synthétiques de différentes tailles et dimensions.

4.4.1. *Qualité de recherche*

Le Tableau 4 et 5 montrent la qualité de recherche de l'approche proposée et du E2LSH pour différentes tailles de bases de données n et dimensions d. Nous remarquons que quand la dimension augmente, la qualité de la recherche des deux approches diminue. Nous notons que les performances des deux approches SC-LSH et E2LSH se dégradent en termes de qualité de la recherche pour les grandes dimensions, mais gardent une bonne qualité de recherche pour les grandes bases de données.

Dimension	100	300	500	700
SC-LSH	0.990713	0.514426	0.172545	0.06109
E2LSH	0.990120	0.514006	0.170722	0.06053

Tableau 4. *Le Mean Average rank-i ratio pour 1000-ppv vs. Dimension pour* $n = 250000$

Taille de la base	250000	500000	1000000
SC-LSH	0.990713	0.980763	0.963412
E2LSH	0.990120	0.98	0.963245

Tableau 5. *Le Mean Average rank-i ratio pour 1000-ppv vs. taille de la base pour* $d = 100$

4.4.2. *Temps de Calcul*

Pour évaluer les performances de l'approche proposée en temps de calcul au passage à échelle, nous avons calculé le temps moyen d'exécution sur différentes bases de données synthétiques de différentes tailles et dimensions.

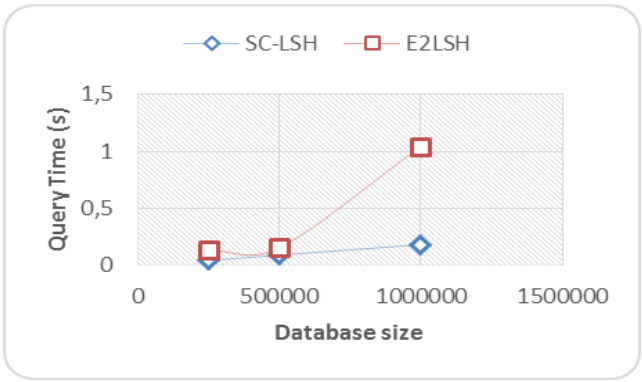

Figure 4. *Temps de calcul vs. Taille de la base pour* $d = 100$

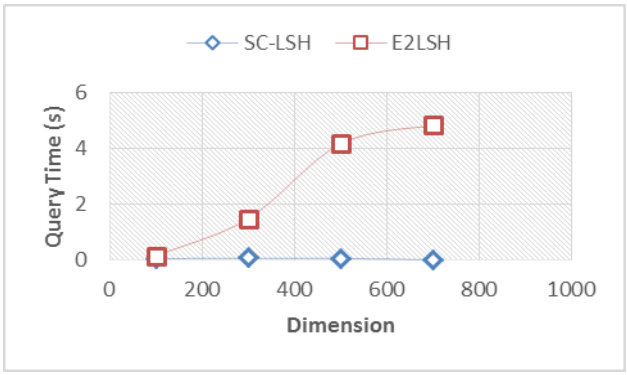

Figure 5. *Temps de calcul vs. Dimension pour* $n = 250000$

Figure 4 et 5 montrent l'évolution du temps de calcul des approches SC-LSH et l'E2LSH par rapport à la taille des bases de données et la dimension. Nous constatons que l'approche proposée est plus rapide que l'E2LSH : le SC-LSH est quatre fois plus rapide que l'E2LSH. Nous constatons donc que l'approche proposée permet d'avoir la même qualité de recherche avec un temps de requête réduit.

4.4.3. *Espace Mémoire*

La méthode LSH est basée sur l'utilisation de plusieurs tables de hachage, pour augmenter la probabilité du hachage des vecteurs proches dans l'espace multidimensionnel dans les mêmes buckets et les vecteurs éloignés dans des buckets différents, une grande capacité d'espace mémoire est nécessaire. Les Tableaux 6 et 7 montrent l'espace mémoire requis par les deux approches pour le stockage de la structure d'index pour différentes base de données de différentes tailles et dimensions. Nous constatons que l'espace mémoire occupé par l'approche proposée est faible et montre une légère hausse lorsque la taille et la dimension augmente en comparaison avec l'approche E2LSH qui nécessite un espace mémoire important plus de quatre fois l'espace requis par l'approche proposée, de plus l'E2LSH garde le même nombre de table de hachage pour différentes dimensions (pour $n = 250000$ on trouve $L = 1485$ équivalent à 4.149 GB pour différentes dimensions).

Dimension	**100**	**300**	**500**	**700**
SC-LSH	0.0279	0.0838	0.1117	0.1396

Tableau 6. *Espace Mémoire vs Dimension pour* $n = 250000$

Taille de la base	**250000**	**500000**	**1000000**
SC-LSH	0.0279	0.0558	0.1117
E2LSH	4.1490	3.3248	4.2244

Tableau 7. *Espace Mémoire vs Taille de la base pour* $d = 100$

5. Conclusion

Dans cet article, nous avons présenté une nouvelle approche basée sur le concept du Locality Sensitive Hashing. L'idée est projeter les vecteurs multidimensionnels sur les axes de symétries du cube ce qui permet la structuration et de regroupement des vecteurs similaires dans le même cube.

Pour évaluer le comportement de l'approche proposée nous avons effectué plusieurs tests sur des bases de données réelles et synthétiques, Nous avons constaté que le SC-LSH dépasse le LSH euclidien en terme de temps de calcul et espace mémoire, et fournit une même qualité de recherche que l'approche E2LSH. Dans

les perspectives de nos travaux futurs, nous avons l'intention d'améliorer la qualité de recherche en particulier pour les ensembles de données de grande dimension.

6. Bibliographie

Guttman A., R-trees: A dynamic index structure for spatial searching. In SIGMOD, 1984, pp. 47-57.

Bentley J. L., K-d trees for semi dynamic vecteur sets. InSoCG, 1990, pp. 187-197.

Weber R., Schek H.J. and Blott S., A quantitative analysis and performance, study for similarity-search methods in high-dimensional spaces. In Proceedings of 24rd International Conference on Very Large Data Bases, New York, USA, 1998, pp.194-205.

Blott S. and Weber R., A Simple Vector Approximation File for Similarity Search in High-Dimensional Vector Spaces. Technical Report 19, ESPRIT project HERMES, March 1997.

Ye L. and Hua Y., The CMVAI-File: An Efficient Approximation-Based High-Dimensional Index Structure. Multimedia Information Networking and Security (MINES), 2010.

Lu H., Ooi B.C., Shen H.T. and Xue X., Hierarchical Indexing Structure for Efficient Similarity Search in Video Retrieval. IEEE Transactions on Knowledge and Data Engineering, November 2006.

Gorisse D., Cord M. and Precioso F., Locality-sensitive hashing for chi2-Distance, IEEE Transactions on Pattern Analysis and Machine Intelligence 34, 2012, pp.402–409.

Wang H ., Cao J., L. Shu, and Rafiei D., Locality sensitive hashing revisited: Filling the gap between theory and algorithm analysis. In Proceedings of the 22nd ACM international conference on Conference on information & knowledge management, 2013, pp. 1969-1978.

Andoni A. and Indyk P., Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. In 47th Annual IEEE Symposium on foundations of Computer Science, 2006, pp. 459-468.

Calistru C., Ribeiro C. and David G., High Dimensional Indexing for Video Retrieval. Multimedia - A Multidisciplinary Approach to Complex Issues, 2012.

Daoudi I., Idrissi K. and Ouatik S.E.A., Kernel region approximation blocks for indexing heterogonous databases. Multimedia and Expo, 2008 IEEE International Conference, 2008, pp. 1237-1240.

Liu T., Moore A. W., Gray A. G. and Yang K., An Investigation of Practical Approximate Nearest Neighbor Algorithm. In proceeding of: Advances in Neural Information Processing Systems 17. Vancouver, British Columbia, Canada, 2004.

Gan J., Feng J., Fang Q. and Ng W., Locality-sensitive hashing scheme based on dynamic collision counting. In proceeding SIGMOD '12 Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. New York, USA, 2012, pp. 541-552.

Lv Q., Josephson W., Wang Z., Charikar M. and Li K., Multi-probe LSH: efficient indexing for high-dimensional similarity search. In VLDB, 2007, pp.950-96.

Indyk P., and Motwani R., Approximate nearest neighbor: Towards removing the curse of dimensionality. In Proceeding STOC '98 Proceedings of the thirtieth annual ACM symposium on Theory of computing, 1998, pp.604-613.

Charikar M.S.. Similarity estimation techniques from rounding algorithms. In Proceedings of the Symposium on Theory of Computing, 2002.

Datar M., Immorlica N., Indyk P. and Mirrokni V., Locality sensitive hashing scheme based on p-stable distributions. In Proceedings of the ACM Symposium on Computational Geometry, 2004.

Broder A. Z., Charikar M., Frieze A. M., and Mitzenmacher M., Min-wise independent permutations. In STOC, 1998, pp. 327–336.

Chafik S., Daoudi I., EL Yacoubi M.A, H. El Ouardi and B. Dorizzi. Locality sensitive hashing for content-based image retrieval: A comparative experimental study. The Fifth International Conference on Next Generation Networks and services (NGNs), 2014.(unpublished)

Haveliwala T. H., Gionis A. and Indyk P., Scalable Techniques for Clustering the Web (Extended Abstract). In Third International Workshop on the Web and Databases (WebDB 2000), Dallas, Texas, 2000.

Feng D. and Yang J., Liu C., An efficient indexing method for content-based image retrieval. Neurocomputing, April 2013, pp.103-114.