Locality-Sensitive Hashing for Chi2 Distance

David Gorisse, Matthieu Cord, and Frederic Precioso

Abstract—In the past ten years, new powerful algorithms based on efficient data structures have been proposed to solve the problem of Nearest Neighbors search (or Approximate Nearest Neighbors search). If the Euclidean Locality Sensitive Hashing algorithm which provides approximate nearest neighbors in a Euclidean space with sub-linear complexity is probably the most popular, the Euclidean metric does not always provide as accurate and as relevant results when considering similarity measure as the Earth-Mover Distance and χ^2-distance. In this paper, we present a new LSH scheme adapted to χ^2-distance for approximate nearest neighbors search in high-dimensional spaces. We define the specific hashing functions, we prove their local-sensitivity and compare, through experiments, our method with the Euclidean Locality Sensitive Hashing algorithm in the context of image retrieval on real image databases. The results prove the relevance of such a new LSH scheme either providing far better accuracy in the context of image retrieval than Euclidean scheme for an equivalent speed, or providing an equivalent accuracy but with a high gain in terms of processing speed.

Index Terms—Sublinear Algorithm, Approximate Nearest Neighbors, Locality Sensitive Hashing, χ^2-distance, image retrieval

1 Introduction

In the recent years, the development of effective methods to retrieve images in large databases (more than 100 000 images) has seen a growing interest both in computer vision and data mining research communities.

Commonly, "Image Similarity Search" refers to the problem of retrieving, from a database, all images sharing perceptual characteristics with a query image q. Describing explicitly these perceptual characteristics requires:

- to extract relevant visual features to represent perceptual image characteristics,
- to define a (dis)-similarity function between the two image representations in order to evaluate the original perceptual similarity of these two images.

Let us notice that these two processes are not independent but related to the proper matching between distance and visual features. When discriminating histograms for instance, some (dis)-similarity functions, or distances, are more accurate than the L_2 distance [1]–[3].

The most simple approach to perform "Image Similarity Search" is then to compute the similarity function between the query image q and all database images in order to find the k best images. In Content Based Image Retrieval (CBIR), more sophisticated systems, like interactive search [4], have been considered in order to perform semantic similarity search and to reduce the semantic gap. However, as the complexity of such approaches grows linearly with the size of the database, this problem is intractable when considering the sizes of current databases. Facing the huge increase of database sizes, a third issue must be considered: the design of a data structure allowing to quickly identify similar images.

The type of index structure depends directly on what visual features (descriptors) are considered to describe the images. Indeed, in the context of copy detection or near duplicate search, images are described with sparse vectors, as it is the case for Bag-of-Word description approaches (BoW), inverted files are then an optimal solution. Indeed, this index structure allows fast exact search fully exploiting vector sparsity.

However, in the context of semantic search, when images are described with dense vectors as for instance color histograms [1], [4], all known techniques to solve the similarity search problem fall prey to "the curse of dimensionality" [5].

Indeed, exact search methods, as kd-tree approaches, are only possible for small size description vectors [6]. Approximate Nearest Neighbors algorithms (ANN) have shown to be interesting approaches to overcome the issue of dimensionality by drastically improving the search speed while maintaining good precision [6].

The effectiveness of such approximate similarity search methods are classically evaluated using two criteria:

- efficiency: speedup factor between exact and approximate search,
- accuracy: fraction of the k nearest images retrieved by exact search which are also retrieved by approximate search.

Several approximate search algorithms have been proposed like VA-files, Best-Bin-First, Space Filling Curves, K-means (see [7] and references therein), NV Tree [8]. One of the most popular ANN algorithms is the Euclidean Locality Sensitive Hashing (E2LSH) proposed by Datar et al. [9]. The authors proved their scheme to integrate into the theoretical framework provided by

- *D. Gorisse is with ETIS, CNRS/ENSEA/UCP, Fr. and Yakaz, Fr. E-mail: david@yakaz.com*
- *F. Precioso is with ETIS, CNRS/ENSEA/UCP, Fr. and I3S-UMR6070-UNS CNRS E-mail: frederic.precioso@unice.fr*
- *M. Cord is with LIP6, UPMC-Sorbonne Universités, France. E-mail: matthieu.cord@lip6.fr*

[10], based on hashing functions with a strong "local-sensitivity" in order to retrieve nearest neighbors in a Euclidean space with a complexity sub-linear in the amount of data. The LSH Scheme has been successfully used in several multimedia applications [11], [12].

Several families of hash functions have recently been proposed to improve the performance of E2LSH [13]. However, all of them are only designed for the L_2 distance while the Earth-Mover Distance (EMD) and the χ^2-distance (refer to 3.1 for the definition of the χ^2 distance considered in this article) proved to often lead to better results than using the Euclidean metric for image and video retrieval task [1]–[3], especially when global descriptors such as color histograms are used to describe the images.

Indyk and Thaper [14] have proposed to embed the EMD metric into the L_2 norm, then to use the original LSH scheme to find the nearest neighbor in the Euclidean space. They thereby obtain an approximate nearest neighbor search algorithm for the EMD metric. To compare different ANN schemes in the same metric space, the intuitive approach is to compare the set of k-Nearest-Neighbors (k-NN) retrieved from a query for both scheme with the exact k-NN (in this specific metric space) of the same query, as in the work of Muja and Lowe [15] in Euclidean space for instance. Such an evaluation remains valid for comparing ANN schemes for two different metrics when one of the metric can be embedded into the other (as with EMD embedded into L_2).

In this paper, we design a new LSH scheme for the χ^2-distance that we call χ^2-LSH. To the best of our knowledge, the χ^2-distance cannot be embedded into the L_2 distance since the χ^2-distance between two vectors intrinsically normalizes the two histogram intra-component variances. We thus directly introduce the χ^2 distance into the definition of the hash function family for LSH. Then, the main issue, when comparing the χ^2-LSH and E2LSH schemes, is that the same query will not provide comparable k-NN sets. We thus consider semantic search to compare the two hashing schemes.

Some works have been focused on designing adaptive data-driven hashing schemes to avoid partitioning in low-density regions [16], [17]. However, our objective is not to obtain a non-uniform tiling of the space but to create a hashing local sensitive within the meaning of the Chi2 distance (two feature vectors, similar within the meaning of the χ^2-distance, fall into the same bucket while two dissimilar feature vectors do not).

In this paper, we design a complete χ^2-LSH scheme, then present its extension to Multi-Probe LSH (MPLSH) [18] which allows to reduce the memory usage while preserving, in the speedup/approximation scheme, the better accuracy of the χ^2-based similarity over the L_2-based similarity. Our method retrieves thereby data similar to a given query with a higher search accuracy than L_2 distance approaches and with a complexity sub-linear in the amount of data. In one of our recent works, we have encompassed this χ^2 similarity search scheme in a learning framework in order to design a scalable classification system, based on the fast approximation of Gaussian-χ^2 kernel and compliant with active learning strategies [19].

With the detailed description of χ^2-LSH scheme, we also provide theoretical proofs of both validity and locality-sensitive properties [10] of our hash functions.

We first evaluate the accuracy and the efficiency of our ANN search algorithm (χ^2-LSH vs. linear search) providing extensive experiments for several databases. We then focus on evaluating our method in a CBIR framework, with semantic similarity search experiments for color and texture based features.

2 Locality Sensitive Hashing

In this section we present an overview of the LSH scheme. The intuition behind LSH is to use hash functions to map points into buckets, such that nearby objects are more likely to map into the same buckets than objects that are farther away. A similarity search consists in finding the bucket B the query q hashes into, selecting candidates, i.e., objects in B, and ranking the candidates according to their exact distance to q.

2.1 Background

LSH was first introduced by Indyk and Motwani in [10] for the Hamming metric. They defined the requirements on hash function families to be considered as locality-sensitive (Locality-Sensitive Hashing functions). Let S be the domain of the objects and D the distance measure between objects.

Definition D1: A function family $\mathcal{H} = \{h : S \rightarrow U\}$ is called (r_1, r_2, p_1, p_2)-sensitive, with $r_1 < r_2$ and $p_1 > p_2$, for D if for any $p, q \in S$

- if $D(q,p) \leq r_1$ then $P_{\mathcal{H}}[h(q) = h(p)] \geq p_1$,
- if $D(q,p) > r_2$ then $P_{\mathcal{H}}[h(q) = h(p)] \leq p_2$.

Intuitively, the definition states that nearby objects (those within distance r_1) are more likely to collide ($p_1 > p_2$) than objects that are far apart (those with a distance greater than r_2).

To decrease the probability of false detection p_2, several functions are concatenated: for a given integer M, let us define a new function family $\mathcal{G} = \{g :\rightarrow U^M\}$ such that $g(p) = (h_1(p), \ldots, h_M(p))$, where $h_i \in \mathcal{H}$. As a result, the probability of good detection p_1 decreases too. To compensate the decrease in p_1, several functions g are used. For a given integer L, choose $g_1, \ldots, g_L$ from $\mathcal{G}$, independently and uniformly at random, each one defining a new hash table, in order to get L hash tables.

2.2 Euclidean metric hashing

Datar et al. [9] proposed a method to build a LSH family for the Euclidean metric, named E2LSH.

Their hashing function works on tuples of random projections of the form: $h_{\mathbf{a},b}(\mathbf{p}) = \left\lfloor \frac{\mathbf{a}.\mathbf{p}+b}{W} \right\rfloor$ where W specifies a bin width, $\mathbf{a}$ is a random vector whose each entry is chosen independently from a Gaussian distribution, b

is a real number picked up uniformly in the range $[0, W]$ representing an offset.

Each projection splits the space by a random set of parallel hyperplanes; the hash function indicates in what slice of each hyperplane the vector has fallen.

The main drawback of this approach is a computational limitation since each hash table must be stored in main memory. Moreover, a large number of hash tables is required to reach high accuracy.

2.3 Multi-Probe LSH

Lv et al. in [18] proposed a new indexing scheme for the L_2 metric called Multi-Probe LSH (MPLSH) that overcomes this drawback. MPLSH is based on the E2LSH principle. Indeed, this approach also uses hash function to map points into buckets and the pre-process (hash table construction) is therefore identical. However, the exploration stage is quite different: instead of exploring only one bucket by hash table, success probabilities are computed for several buckets and buckets which are most likely to contain relevant data are examined. Given the property of locality sensitive hashing, we know that if an object is close to a query object but does not hash into the same bucket, it is likely to be in a neighboring bucket. The authors defined a *hash perturbation vector* $\Delta = (\delta_1, \ldots, \delta_M)$ where $\delta_i \in \{-1, 0, 1\}$. For a query q and a hash function $g(x) = (h_1(x), \ldots, h_M(x))$, the success probabilities for $g(q) + \Delta$ are computed and the T most likely buckets of each hash tables are visited. As a result, the authors reduce the number of hash tables by a factor of 14 to 18 for similar search accuracy and query time as E2LSH.

3 χ^2-LSH HASHING

In this section we detail how we adapt the algorithm of E2LSH then MPLSH to the χ^2 distance.

3.1 Basics

Definition D2: For any 2 d-sized vectors $\mathbf{x}$ and $\mathbf{y}$ with strictly positive components, the χ^2-distance between $\mathbf{x}$ and $\mathbf{y}$ is defined as $\chi^2(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{d} \frac{(x_i - y_i)^2}{x_i + y_i}}$.[1]

As mentioned previously, the χ^2-distance is more appropriate than the L_2 distance to compare histograms. We thus want to work with the χ^2 distance keeping the same efficiency as Datar's E2LSH algorithm. Therefore, we map all the points into a space of smaller dimension and cluster this sub-space. The clusterization must ensure that the probability for two points to fall into the same bucket (a cluster cell) is higher when the distance between these two points is smaller than when this distance is greater than a certain threshold. In our case, the sub-space is the line $l_{\mathbf{a}}$. This line is obtained by projecting all points on a random vector $\mathbf{a}$ where each entry is chosen independently from a Gaussian distribution. We uniformly partition this line with respect to the χ^2 distance, i.e. each partition interval $[X_i, X_{i+1}]$ has the same length, W (see Fig. 1):

$$\forall i \in \mathbb{N},\ \chi^2(X_i, X_{i+1}) \stackrel{\text{def}}{=} \sqrt{\frac{(X_i - X_{i+1})^2}{X_i + X_{i+1}}} = W\ . \quad (1)$$

On the other hand, as shown on Fig.1, the length of the interval on the line $l_{\mathbf{a}}$ is not constant anymore if we consider the same bounds X_i and X_{i+1} but with the L_2 distance: $\forall i,\ L_2(X_i, X_{i+1}) \neq W$. The partition, in the sense of the χ^2-distance, ensures that when two points are at a distance less than W after mapping to $l_{\mathbf{a}}$, the probability of being in the same cluster is higher.

3.2 Hash function definition

Given the sequence $(X_n)_{n\in\mathbb{N}}$ which satisfies eq.(1) with initial value set to zero: $X_0 = 0$ and $\mathbf{a}$ a random vector where each entry is chosen independently from a Gaussian distribution with positive value $\mathcal{N}^+(0, 1)$, we want to define hash functions $h_{\mathbf{a}}$ such that:

$$\forall \mathbf{p} \in \mathbb{R}^{+d},\ h_{\mathbf{a}}(\mathbf{p}) = n \text{ iff } X_{n-1} \leq \mathbf{a}.\mathbf{p} < X_n\ . \quad (2)$$

We can rewrite eq.(1):

$$X_n = X_{n-1} + W^2 \frac{\sqrt{8.\frac{X_{n-1}}{W^2} + 1} + 1}{2}\ . \quad (3)$$

By fixing $X_0 = 0$, we get:

$$X_n = \frac{n(n+1)}{2} W^2\ . \quad (4)$$

Let us denote f, the function such that $f(n) = X_n = \frac{n(n+1)}{2} W^2$. We have now to determine, for any $x = \mathbf{a}.\mathbf{p}$, the integer n such that $n \leq \lfloor f^{-1}(x) \rfloor < n + 1$. [2] In the next section we exhibit the definition of f^{-1}, that we call y_W, and prove the validity of the hash function family $h_{\mathbf{a}}(\mathbf{p})$ hence defined.

3.3 Efficient hashing scheme: χ^2-LSH

Definition D3: For any $W \in \mathbb{R}^+$, let y_W be a function defined such that:

$$\forall x \in \mathbb{R}^+\ , \quad y_W(x) = \frac{\sqrt{\frac{8x}{W^2} + 1} - 1}{2} \quad (5)$$

Proposition P1: Given a point $\mathbf{p} \in (\mathbb{R}^+)^d$, $\mathbf{a}$ a random vector where each entry is chosen independently from a Gaussian distribution with positive value $\mathcal{N}^+(0, 1)$, hash functions according to χ^2-hashing can be built as follows:

$$h_{\mathbf{a}}(\mathbf{p}) = \lfloor y_W(\mathbf{a}.\mathbf{p}) \rfloor\ . \quad (6)$$

Proof: To prove *P1*, we first define the sequence $(Y_n)_{n\in\mathbb{N}}$:

1. The χ^2-distance is a *weighted standardized Euclidean* distance thereby a true metric [20]. Furthermore, even if there is not a firm consensus on the definition of this distance, all our developments still hold for a definition without the square-root by substituting W to W^2 in all equations.

2. Since f is a strictly increasing function, f^{-1} is strictly increasing too. Thus, knowing the interval $[X_n, X_{n+1}[$ into which x is falling, is sufficient to determine n (when f^{-1} cannot be explicitly computed for instance). A tree structure, based on $(X_n)_{n\in\mathbb{N}}$ values, can then be considered to efficiently find the right interval (and thus the hash value n) for any given $x = \mathbf{a}.\mathbf{p}$.

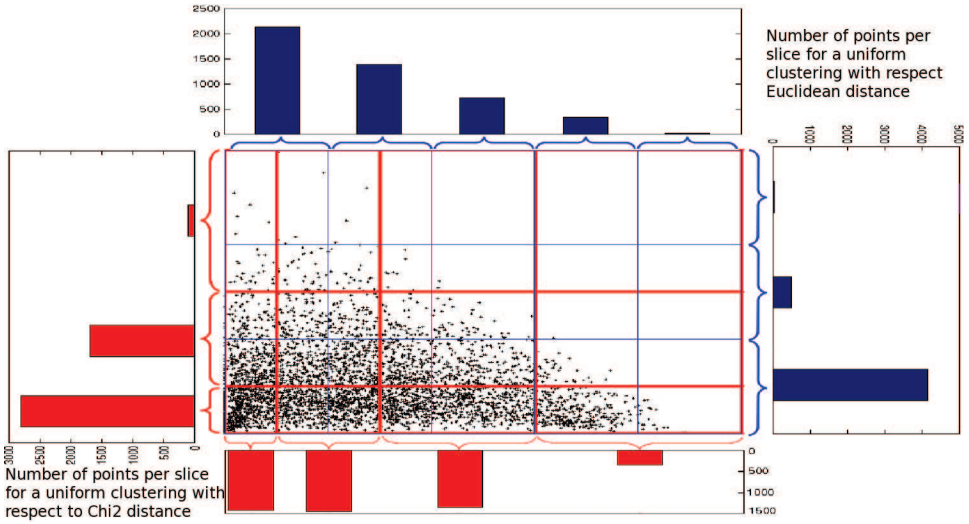


Fig. 1. χ^2 and L_2 space tilings. 4500 image feature vectors have been randomly selected from the COREL database. Space grids for 2 features components with respect to χ^2 (in red) and to L_2 (in blue) distances are reported. Points falling into the same bucket in χ^2 hashing scheme may fall into seperate buckets in L_2 hashing scheme.

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

We prove that for all $n \in \mathbb{N}$, $(H_n) : Y_n = n$ by induction on n.
Since by definition $Y_0 = 0$, then (H_0) is true.
Rewriting eq.(7) as $Y_n^2 + Y_n = 2\frac{X_n}{W^2}$ and combining this with eq.(1), we get:

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

Let us now assume that (H_{n-1}) true, i.e. $Y_{n-1} = n-1$.
From eq.(7) we deduce that: $2\frac{X_{n-1}}{W^2} = n(n-1)$.
Introducing this result in eq.(8), we then get:
$Y_n(Y_n+1) = n(n+1)$.
The only positive solution of this equation is $Y_n = n$.

We thus show that the hash function defined in eq.(6) satisfies:

$$\forall \mathbf{p}' \in (\mathbb{R}^+)^d \quad s.t. \quad X_n = \mathbf{a}.\mathbf{p}', \quad h_{\mathbf{a}}(\mathbf{p}') = \lfloor y_W(X_n) \rfloor = n . \tag{9}$$

It is then straightforward, using the strict monotony of y_W, to see that:

$$\forall \mathbf{p} \in (\mathbb{R}^+)^d \quad s.t. \quad X_n \leq \mathbf{a}.\mathbf{p} < X_{n+1}, \quad h_{\mathbf{a},b}(\mathbf{p}) = n . \tag{10}$$

This completes the proof of proposition *P1*.

□

To avoid boundary effects, we introduce, as in [9], an offset in the hash functions, *i.e.* a real number b picked randomly following the uniform distribution $\mathcal{U}([0,1[)$:

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

Since all points are shifted by b, the previous construction of the hash functions holds by setting $X_0 = b$.
Let $\mathcal{H}$ be the family of such hash functions.

3.4 χ^2-LSH: a locality sensitive function

We now demonstrate that the original LSH scheme (*Definition D1*) still holds for this family $\mathcal{H}$.

Theorem 1 (χ^2-LSH sensitivity): The χ^2 hash function family $\mathcal{H}$, defined in eq.(11), is (r_1, r_2, p_1, p_2)-sensitive when input vectors get positive components.

Proof: Let us define P as the probability of the hash functions to be locality-sensitive:

$$\begin{aligned} P &= P_{\mathcal{H}}[h_{\mathbf{a},b}(\mathbf{q}) = h_{\mathbf{a},b}(\mathbf{p})] \\ &= P_{\mathbf{a},b}\left[\begin{array}{c} \exists n, n \leqslant y_W(\mathbf{a}.\mathbf{p}) + b < n+1, \\ n \leqslant y_W(\mathbf{a}.\mathbf{q}) + b < n+1 \end{array}\right] \end{aligned}$$

For all $\mathbf{a}$, as defined in the previous section, we can consider either $\mathbf{a}.\mathbf{p} \leq \mathbf{a}.\mathbf{q}$ or $\mathbf{a}.\mathbf{p} \geq \mathbf{a}.\mathbf{q}$ without loss of generality. For the sake of demonstration clarity, let us consider $\mathbf{a}.\mathbf{p} \leq \mathbf{a}.\mathbf{q}$. a and b are independent and both admit density probabilities $p(\mathbf{a})$, $p(b)$. Then P may be computed using marginalization over b with the following integral bounds. From the 2 previous inequalities we have: $n \leqslant y_W(\mathbf{a}.\mathbf{p}) + b \leqslant y_W(\mathbf{a}.\mathbf{q}) + b < n+1$, so that bounds on b are:

$$n - y_W(\mathbf{a}.\mathbf{p}) \leqslant b < n + 1 - y_W(\mathbf{a}.\mathbf{q}) \tag{12}$$

and we also have:

$$0 \leqslant y_W(\mathbf{a}.\mathbf{q}) - y_W(\mathbf{a}.\mathbf{p}) \leqslant 1 \tag{13}$$

Integrating on the random variable b leads to:

$$\int_{n-y_W(\mathbf{a}.\mathbf{p})}^{n+1-y_W(\mathbf{a}.\mathbf{q})} db \; = 1 \; - \; (y_W(\mathbf{a}.\mathbf{q}) - y_W(\mathbf{a}.\mathbf{p})) \tag{14}$$

P may then be rewrited as:

$$P = P_{\mathbf{a}}\left[0 \leqslant 1 - (y_W(\mathbf{a}.\mathbf{q}) - y_W(\mathbf{a}.\mathbf{p})) \leqslant 1\right] \tag{15}$$

Combining the relation between hash function $h_{\mathbf{a},b}$ and interval bounds X_n in the sense of χ^2 hashing of eq.(2) with the expression eq.(4),

we can then rewrite,

$$P = P_{\mathbf{a}}\left[0 \leqslant 1 - \frac{1}{(n+1)W^2}\mathbf{a}(\mathbf{p}-\mathbf{q}) \leqslant 1\right] \quad (16)$$

Following the reasonning in [9], we use the 2-stable distribution property: for two vectors $\mathbf{p}$ and $\mathbf{q}$, a random variable $\mathbf{a}$ where each entry is drawn from a 2-stable distribution, $\mathbf{a}.(\mathbf{q}-\mathbf{p})$ is distributed as cX where $c = \|\mathbf{p}-\mathbf{q}\|_{L_2}$ and X is a random variable drawn from a 2-stable distribution. It follows that:

$$P = p(c) = \int_0^{(n+1)W^2} \frac{1}{c} f(\frac{t}{c})(1 - \frac{t}{(n+1)W^2})dt\ , \quad (17)$$

where $f(t)$ denotes the probability density function of the absolute value of the 2-stable distribution.

Let us define $c' = \chi^2(\mathbf{p},\mathbf{q})$. Since $\mathbf{p}_i$ and $\mathbf{q}_i \in \mathbb{R}^+$ and since on $(R^+)^d \times (R^+)^d$, the scalar product between the gradients of the χ^2-distance and the L_2-distance is always positive: the two distances vary similarly. Therefore, p decreases monotonically with respect to c, and p decreases also monotonically with respect to c'. Reminding that $r_1 < r_2$, if we set $p_1 = p(r_1)$ and $p_2 = p(r_2)$, then $p_2 < p_1$. This concludes the proof of *Theorem 1*: Our $\mathcal{H}$ family is (r_1, r_2, p_1, p_2)-sensitive. □

3.5 Multi-probe hashing scheme: χ^2-MPLSH

To adapt the multi-probe scheme to the χ^2-metric we have to compute the success probability of finding a vector $\mathbf{p}$ that is close to $\mathbf{q}$ after it has been perturbed by a *hash perturbation vector*, as presented in section 2.3 $Pr\left[g(\mathbf{p}) = g(\mathbf{q}) + \Delta\right]$. As all coordinates h_i in the hash functions g are independent, we have:

$$Pr\left[g(\mathbf{p}) = g(\mathbf{q}) + \Delta\right] = \prod_{i=1}^{M} Pr\left[h_i(\mathbf{p}) = h_i(\mathbf{q}) + \delta_i\right] \quad (18)$$

Note that each hash function h_i first maps $\mathbf{q}$ to a line and divides the line into slots of length W. We remind that the length of a slot, according to the χ^2-metric, is W, i.e., the χ^2-distance between two consecutive bounds X_i and X_{i+1} is equal to W. Each slot is numbered and the hash value is the number of the slot $\mathbf{q}$ falls into. A point $\mathbf{p}$ close to $\mathbf{q}$ is likely to fall in the same slot as $\mathbf{q}$ but the probability it falls in one of the adjacent slots $h_i(\mathbf{q}) - 1$ or $h_i(\mathbf{q}) + 1$ is high too. In fact, the closer $\mathbf{q}$ to the right boundary X_{i+1}, the higher the success probability that $\mathbf{p}$ falls into $h_i(\mathbf{q}) + 1$. Thus, the position of $\mathbf{q}$ in the slot allows to compute the success probability of $h_i(\mathbf{q}) + \delta_i$. For $\delta \in \{-1, +1\}$, let $x_i(\delta)$ be the distance of $\mathbf{q}$ from the boundary between $h_i(\mathbf{q})$ and $h_i(\mathbf{q}) + \delta$, then $x_i(\delta) = |k_i(\mathbf{q}) - h_i(\mathbf{q}) - \delta|$ with $k_i(\mathbf{q}) = y(\mathbf{a}.\mathbf{p}) + b$.

As, in [18], we then estimate the probability that $\mathbf{p}$ falls into the slot $h_i(\mathbf{q}) + \delta$ by:

$$Pr\left[h_i(\mathbf{p}) = h_i(\mathbf{q}) + \delta\right] \approx e^{-Cx_i(\delta)^2} \quad (19)$$

where C is a constant depending on the data.

Introducing the eq.(19) in eq.(18), we deduce the score representing the likelihood of finding a point close to q in the bucket defined by Δ: $score(\Delta) = \sum_{i=1}^{M} x_i(\delta_i)^2$. The smaller the score, the higher the probability of finding a point in the bucket perturbed by Δ.

For each query, we compute all possible perturbed vectors Δ and use their scores to select the T buckets likely to hold a nearest neighbor.

4 EFFICIENCY AND ACCURACY ANALYSIS

In this section, we present the empirical performance evaluation of our hash function. We first evaluate the efficiency for different accuracy values of the χ^2-LSH scheme. Next, we assess the memory factor gain by the χ^2-MPLSH scheme.

4.1 Experimental Setup

No consensus among the community is currently achieved on a standardized experimental setup for the evaluation of high-dimensional indexing methods. Therefore, among the main challenges for researchers focusing on this topic, we should point out the choice of databases, of queries, and of ground truth. Though early works tried to use synthetic data, following a uniform random distribution, it is now usually accepted that this is an unrealistic context of evaluation. Recent works are usually assessed on real data. We performed evaluation of our scheme on the 158,929 keyframes of Trecvid 2009 database. Each keyframe is represented by a high dimensional histogram of 128 bins obtained by the concatenation of 2 histograms, one of 64 chrominance from CIE L*a*b and one of 64 textures from Gabor filters.

To assess the influence of the size of the database, we have built 3 datasets, one of 43,616 images corresponding to Trecvid 2007 database, called DB1, a second of 86,077 images corresponding to Trecvid 2008 database, called DB2, and the whole database corresponding to Trecvid 2009, called DB3. For each dataset, we created an evaluation benchmark by randomly picking 25,000 images as query images.

The ground truth is built by brute force search of each query image's exact k Nearest Neighbors, using the χ^2 distance and not including the query image itself. The ground truth is used both to compute the speedup factor and to estimate the approximation of χ^2-LSH. In our experiments, we set $k = 20$.

Performance is evaluated on two aspects: accuracy (the capability of the method to return the correct results using precision metric) and efficiency (the capability of the method to use as few resources as possible using speed up ratio with respect to brute force search). Therefore, this measure does not depend neither on the machine nor on the operating system.

4.2 Speedup factor: exact search vs χ^2-LSH

We ran two sets of experiments to assess the speedup performance of χ^2-LSH: we evaluate the impact of both the LSH parameters and the size of the dataset.

Three parameters impact the performance of the LSH algorithm: the number of hash tables (L), the number of projections per hash value (M) and the size of the

search window (W). The first set of experiments allows us to study the influence of the two last parameters (M and W). We used the first parameter L to control the trade-off between accuracy and efficiency. Firstly,

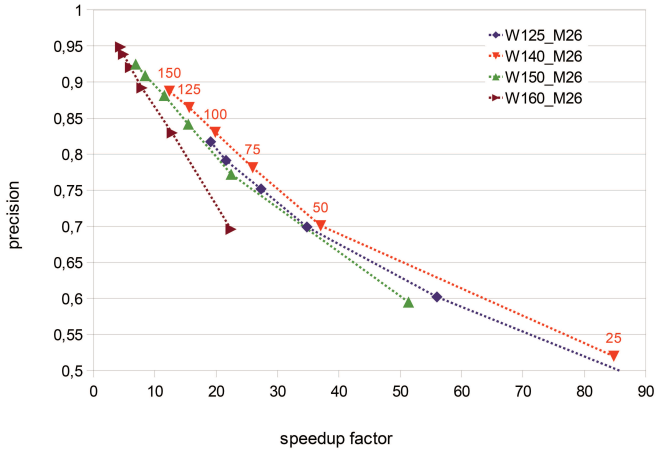


Fig. 2. Influence of W: curves for 4 values of W, M fixed to 26 and 6 different values of L

to study the influence of W, we fix M to 26 and we make W and L vary. Fig. 2 reports the evaluation on DB3 of the influence of the two parameters W and L. For each pair of parameters, the efficiency and the accuracy are computed. W spans the range from 125 to 160 and L from 25 to 150. We observe that for a same L, increasing the value of W decreases the speedup. Indeed, as we increase the value of W, we increase the number of false positives per hash tables, i.e. points that are not near neighbors but hash into the same bucket. As a result, the number of candidates to visit increases. However, at the same time the precision increases too. Indeed, the number of false negatives, i.e. points that are near neighbors but do not hash into the same bucket, decreases.

As we can see in the Fig. 2, $W = 140$ gives the best trade-off between false negatives and false positives. Secondly, to study the influence of M, we hold fixed the

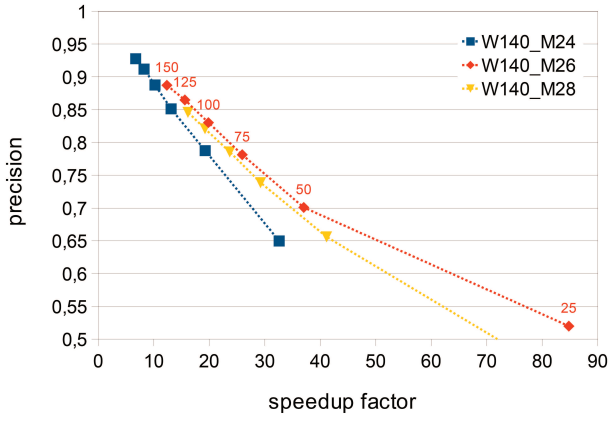


Fig. 3. Influence of M: curves for 3 values of M, W fixed to 140 and 6 different values of L

parameter W to 140 and we make M and L vary. Fig. 3 reports the evaluation on DB3 of the influence of the two other parameters M and L. For each pair of parameters, the efficiency and the accuracy are computed. M spans the range from 24 to 28, and L from 25 to 150. We notice that for a same L, increasing the value of M increases the speedup. Indeed, as we increase the value of M, we decrease the number of false positives per hash tables. As a result, the number of candidates to visit decreases. However, at the same time the precision decreases too. Indeed, the number of false negatives increases. To keep low the probability of false negatives, we have to increase the number of tables L. However, it takes more time to compute two tables rather than one table. Therefore a good trade-off is to be found between the number of candidates to check and the computation time for hashing. This explains why we obtain better performance for $M = 26$ than for $M = 28$.

To sum up, $W = 140 <$ with $M = 26$ are considered as default parameters because they give a good trade-off between accuracy and efficiency.

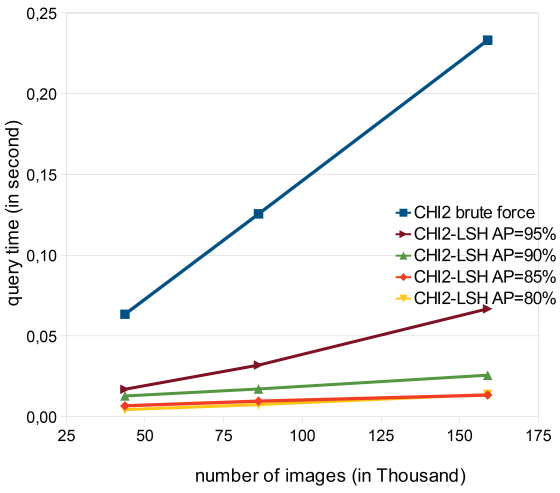


Fig. 4. query time vs number of images for DB1, DB2 and DB3 and different Average Precision rates

The second set of experiments aims at evaluating the complexity of the search, according to the database size in order to prove the sub-linearity of our algorithm complexity, and the stability of the method with respect to W parameter value. In Fig. 4, we evaluate the search time for the 3 datasets and for Average Precision (AP) of 80%, 85%, 90% and 95%. M and L are respectively set to 26 and 115. As we can see in Fig. 4, the brute force search is indeed linear: DB3 is 3.64 times higher than DB1 and the search time for DB3 is 3.67 times slower than for DB1 (respectively 233 and 63 msecs).

For our χ^2-LSH scheme, the search time is of course connected to the target AP. We can notice that for an AP of 85%, our χ^2-LSH scheme provides a very good trade-off between AP and query time which is the same as for an AP rate of 80%. For an accuracy of 85%, the query time is only 1.98 times higher for DB3 than for DB1 (respectively 13.44 and 6.77 msecs), which illustrates the sub-linear complexity of our scheme. Our χ^2-LSH is

Average Precision		0.8	0.85	0.9	0.95
W	DB1	142	149	161	188
	DB2	132	144	155	176
	DB3	128	140	150	169

TABLE 1

then 9.37 times faster for the database of 44K images (DB1) and becomes 17.35 times faster for the database of 160K images (DB3), than the exhaustive search. It follows that the efficiency of our fast scheme increases with the size of the database for an AP of 85%. This remains true for an AP of 90% but the gain over exhaustive search decreases since our χ^2-LSH is 4.92 times faster for DB1 and becomes 9.07 times faster for DB3 than the exhaustive search. For the highest AP rate of 95%, the gain over the exhaustive search is almost linear with a gain of about 3.5 times faster for the 3 databases.

The value of the parameter W is directly connected to both the target value for the AP and the size of the database. In the following table (Tab.1), we present the variations of W during the previous experiment. The value of W is set to 142, 132 and 128, respectively for DB1, DB2 and DB3, to maintain an AP of 80%. These W values evolve, with the target AP, homogeneously for the 3 databases. One can also see that, as expected, the larger the database, the more dense the feature space is, and thus the smaller the LSH bin width W is. Of course, in order to reach an AP of 95%, our scheme has to be less approximating and thus the value of W increases to account for feature vectors farther away from the query, considering hence more of the feature space. This table illustrates also the stability of W with respect to both the increase of the size of the database and the increase of AP. Indeed, we can notice that whatever the rate for the target AP is, W decreases within a range of less than 15 to get adapted to the database size. Equivalently, whatever the size of the database considered, W increases within a range of less than 40 to get adapted to the increase of the AP rate.

4.3 Memory usage: χ^2-LSH vs χ^2-MPLSH

In this part, we assess the improvement of χ^2-MPLSH on χ^2-LSH in terms of space requirements and effectiveness for various accuracy values on the DB3 dataset.

First, we evaluate the accuracy according to the efficiency of χ^2-MPLSH to ensure that χ^2-MPLSH is able to reach the same precision as χ^2-LSH with few hash tables, thus less memory usage. Results are reported in Fig. 5. In this experiment, we set $W = 140$ and $M = 26$ and we make the parameter L span the range from 2 to 6 and the parameter T from 25 to 150. To reach a precision between 0.8 and 0.9 (see Fig. 5), only 6 hash tables are necessary for χ^2-MPLSH against between 75 and 150 for χ^2-LSH (see Fig. 3). Moreover, only 4 tables are required for a precision between 0.6 and 0.8.

On Tab.2 and Tab.3, we compare memory usage for χ^2-LSH and χ^2-MPLSH. By using the second scheme, the memory requirement falls down from Gb to hundreds of Mb.

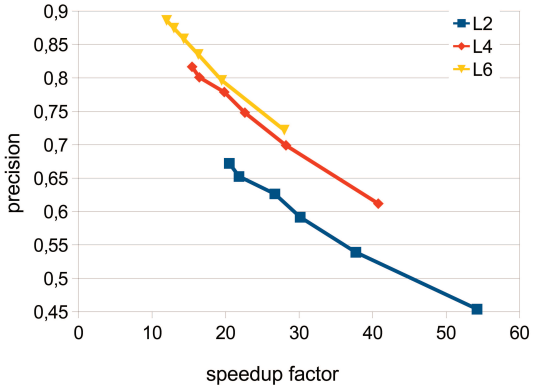


Fig. 5. χ^2-MPLSH, influence of L: curves for 3 values of L, W and M fixed to 26 and 140 and 6 values for T

L	25	50	75	100	125	150
Mem. χ^2-LSH (Gb)	1.00	1.65	2.24	2.86	3.47	4.09

TABLE 2

L	2	4	6
Mem. χ^2-MPLSH (Mb)	445	527	546

TABLE 3

Secondly, we evaluate the effectiveness of χ^2-MPLSH. We use χ^2-LSH has a baseline setting $W = 140$ and $M = 26$ and we make L vary from 25 to 150 to change the accuracy. As shown in Fig. 6, choosing the same parameters $W = 140$ and $M = 26$ for χ^2-MPLSH as for χ^2-LSH and $L = 6$ is not optimal. Indeed, for a similar accuracy, the χ^2-MPLSH does not reach same speedup factor. However, with $W = 130$, $M = 24$ and $L = 6$, χ^2-MPLSH reaches the same efficiency as χ^2-LSH for an accuracy between 0.72 and 0.9. Furthermore, with $W = 120$, $M = 24$ and $L = 4$, χ^2-MPLSH reaches the same efficiency as χ^2-LSH for an accuracy between 0.55 and 0.76. It is worth noting that for accuracy between 0.65 and 0.8, χ^2-MPLSH with $W = 120$, $M = 24$ and $L = 6$ reaches higher efficiency than χ^2-LSH.

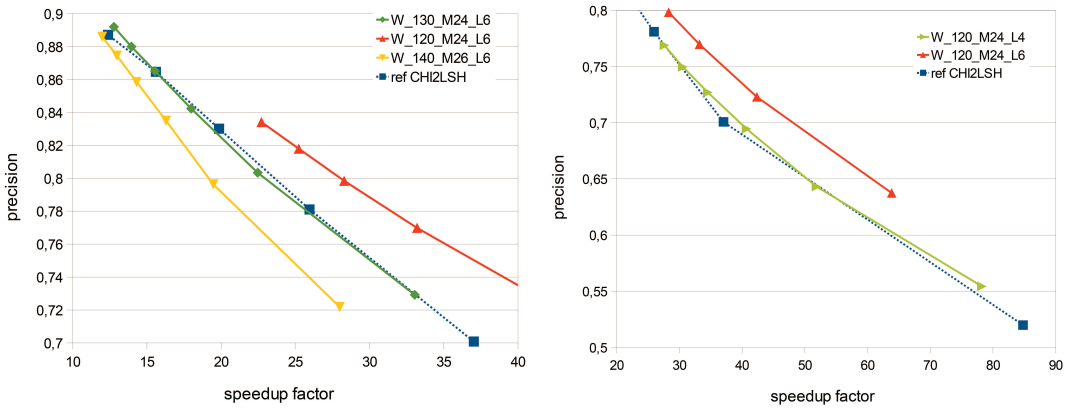
Finally, χ^2-MPLSH achieves the same performance as χ^2-LSH with little memory requirements: for instance, if we consider $W = 120$, $L = 4$, $M = 24$ for χ^2-MPLSH, the memory usage gain is about 8 with the χ^2-LSH default scheme. Considering that the system is installed on a recent computer with 32 Gb of RAM, the system deals with a 10 Million image dataset.

5 Search Quality for χ^2 vs L_2

The aim of this experiment is to compare the search quality of the two methods: $MPLSH$ and χ^2-MPLSH for different parametrizations. To compare the search quality, we consider a context of supervised classification of image database.

5.1 Experimental protocol

We do not perform evaluation on near copy detection database (like INRIA Holidays used in [7]) because, as aforementioned, for this task BoW and inverted file are optimal solutions. Moreover, we do not use SIFT descriptors because they are designed for the L_2-distance.



(a) accuracy between 0.9 and 0.7 (b) accuracy between 0.8 and 0.5

Fig. 6. χ^2-MPLSH: speedup vs precision for several parameters

Unlike the copy detection tasks, we are rather interested in CBIR and interactive CBIR tasks that aim to retrieve semantic classes [2], [4], [21]. In this context, color and texture descriptors are widely used.

Let $\{\mathbf{p}_i\}_{1,n}$ be the n image indexes of the database. A training set is expressed as $\mathcal{A} = \{(\mathbf{p}_i, y_i)_{i=1,n} | y_i \neq 0\}$, where $y_i = c$ if the image $\mathbf{p}_i$ belongs to the class c.

Since we are interested in comparing L_2- and χ^2-distances, we use simple classification rule based on [5]:

$$f_c(\mathbf{p}) = \sum_{N_j(\mathbf{p}) \in k\text{-}NN(p)} (K(\mathbf{p}, N_j(\mathbf{p})) | y_j = c) \qquad (20)$$

where $N_j(\mathbf{p})$ is the j^{th} neighbor point of $\mathbf{p}$ in $\mathcal{A}$, k the number of nearest neighbors considered and $K(\mathbf{p}, \mathbf{q}) = 1 - \frac{d(\mathbf{p},\mathbf{q})}{R}$ with R a search radius and $d(\mathbf{p}, \mathbf{q})$ the L_2 or the χ^2 distance between $\mathbf{p}$ and $\mathbf{q}$. We set $k = 20$.

As [1], [14], we perform evaluation of our scheme on the well-known COREL database. The dataset contains $c = 154$ classes of 99 images per class or $n = 15246$ images. Each image $\mathbf{p}$ is represented by a 128-dimension vector obtained by concatenating 2 histograms, one of 64 chrominances value from CIE L*a*b and one of 64 textures from Gabor filters.

As in PASCAL Challenges, we use the Mean Average Precision (MAP) to assess the classification: To compute the MAP of a class c, each image belonging to this class is successively removed from the training set and used as query image (leave-one-out strategy). A Precision score is computed for each query, then averaged (AP) on the whole class c. The MAP is then computed by averaging the AP of each class.

5.2 Quality assessment

In this part, we evaluate the classification as explained in the previous experimental protocol section with four nearest neighbor methods: E2LIN, χ^2-LIN for linear scan using respectively the L_2- and χ^2- distance, $MPLSH$ and χ^2-MPLSH. Fig. 7 shows the AP in function of speedup factors. As in [18], we have experimented with different parameter values for the LSH methods and picked the ones that provide best performance. We set $W = 110000$, $M = 20$, $L = 6$ for $MPLSH$ and $W = 115$, $M = 22$, $L = 6$ for χ^2-MPLSH. Let us notice that the bin width parameter W is squared in our χ^2-LSH 11 and not in Datar's E2LSH algorithm [9]. Thus, W parameter values are comparable (differing from a factor of less than 3). To modify both accuracy and efficiency, T varies from 10 to 150. Linear scanning searches allow

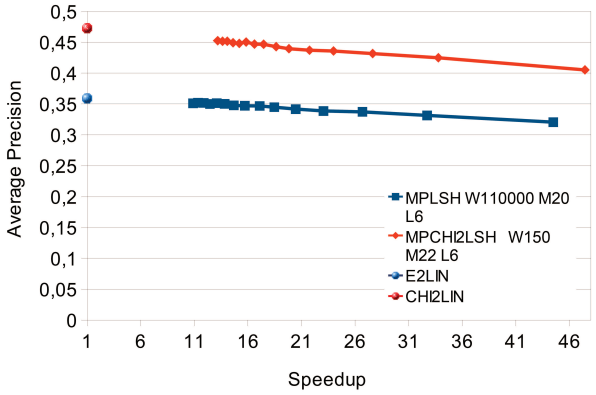


Fig. 7. Average Precision vs speedup

us to compute the speedup factor and to estimate the approximation of LSH. We thus confirm that χ^2 reaches better accuracy than L_2. The AP is 0.36 for L_2 and 0.47 for χ^2 or a gain of 30.56%.

We reduce the approximation of search quality from 10 to 5% (compared with linear scan) by making the parameter T spanning the range from 150 to 10. For all these parametrizations, χ^2-MPLSH is at least 28% more accurate than MPLSH for the same efficiency: for a speedup factor of 44, the precision of $MPLSH$ is 0.32 and the precision of χ^2-MPLSH is 0.41.

6 Conclusion

In this paper, we have introduced a new LSH scheme fitted to the χ^2-distance for approximate nearest neighbor search in high-dimensional spaces. Our method outperforms the original E2LSH algorithm in the context of image and video retrieval when data are represented by histograms. This approach makes the most of LSH efficiency since it does not require any embedding process. The experiments we lead showed the relevance of such a new LSH scheme in the context of image retrieval.

ACKNOWLEDGMENTS

We sincerely thank A. Andoni for providing us with the package E2LSH [**?**]. We are grateful to the reviewers whose valuable comments helped greatly improve the paper.

REFERENCES

[1] O. Chapelle, P. Haffner, and V. Vapnik, "Support vector machines for histogram-based image classification," IEEE trans. Neural Networks, pp. 1055–1064, 1999.
[2] P. Gosselin, M. Cord, and S. Philipp-Foliguet, "Combining visual dictionary, kernel-based similarity and learning strategy for image category retrieval," CVIU, vol. 110, no. 3, pp. 403–417, 2008.
[3] Y. Rubner, C. Tomasi, and L. Guibas, "The earth mover's distance as a metric for image retrieval," Int. J. Comput. Vision, vol. 40, no. 2, pp. 99–121, 2000.
[4] A. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years," IEEE Trans. Pattern Anal. Mach. Intell, pp. 1349–1380, 2000.
[5] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning. Springer, 2009.
[6] H. Samet, Found. of multidim. metric data struct. MK, 2006.
[7] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," IEEE Transactions on Pattern Analysis & Machine Intelligence, vol. 33, no. 1, pp. 117–128, jan 2011.
[8] H. Lejsek, F. Ásmundsson, B. Jónsson, and L. Amsaleg, "NV-Tree: An efficient disk-based index for approximate search in very large high-dimensional collections," IEEE Trans. Pattern Anal. Mach. Intell., vol. 31, no. 5, pp. 869–883, 2009.
[9] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in SOCG. ACM, 2004, pp. 253–262.
[10] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in STOC. ACM, 1998, pp. 604–613.
[11] Y. Ke, R. Sukthankar, and L. Huston, "Efficient near-duplicate detection and sub-image retrieval," in Multimedia. ACM, 2004, pp. 869–876.
[12] D. Gorisse, M. Cord, F. Precioso, and S. Philipp-Foliguet, "Fast Approximate Kernel-Based Similarity Search for Image Retrieval Task," in ICPR. IAPR, 2008, pp. 1873–1876.
[13] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in FOCS. IEEE, 2006, pp. 459–468.
[14] P. Indyk and N. Thaper, "Fast image retrieval via embeddings," in Int. Workshop SC Theories of Vision, 2003.
[15] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in VISSAPP, 2009.
[16] B. Georgescu, I. Shimshoni, and P. Meer, "Mean shift based clustering in high dimensions: A texture classification example," in International Conference on Computer Vision, 2003.
[17] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashng," in NIPS, 2008.
[18] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li, "Multi-probe lsh: Efficient indexing for high-dimensional similarity search," 2007, pp. 950–961.
[19] D. Gorisse, M. Cord, and F. Precioso, "Scalable active learning strategy for object category retrieval," in ICIP. IEEE, 2010.
[20] M. Greenacre, Correspondence Analysis in Practice, Second Edition. Chapman & HallCRC, 2007, iSBN 1-584-88616-1.
[21] E. Chang, S. Tong, K. Goh, and C. Chang, "Support vector machine concept-dependent active learning for image retrieval," IEEE Trans. on Multimedia, vol. 2, 2005.

David Gorisse is now a research engineer at Yakaz. He received an M.Sc. in electrical engineering and telecommunications by ISEN, France, in 2006 and an M.Sc. degrees in computer science from the University of Cergy-Pontoise, France, in 2007. He obtained his PhD degree in Image Processing in 2011 from the University of Cergy-Pontoise as member of ETIS joint Laboratory of CNRS/ENSEA/Univ Cergy-Pontoise, France.

Matthieu Cord is Professor of Image Processing at the Computer Science Department, University of UPMC PARIS 6 since 2006. He received the PhD degree in 1998 from the University of Cergy, France. In 1999, he was post-doc at the Katholieke Universiteit Leuven, Belgium, and he joined the Image Indexing team of the ETIS lab., France. In 2009, he is nominated at the IUF (French Research Institute) for 5 years. His current research interests include Computer Vision, Pattern Recognition, Information Retrieval, Machine Learning for Multimedia Processing, especially interactive learning image and video retrieval systems. He has published over 100 papers and is involved in several International research projects and networks.

Frederic Precioso is Professor at I3S laboratory, University of Nice-Sophia Antipolis since 2011. He has a PhD in Signal and Image Processing, obtained from Univeristy of Nice-Sophia Antipolis,France, in 2004. After a year as Marie-Curie Fellow at CERTH-Informatics and Telematics Institute, (Thessaloniki, Greece), where he worked on semantic methods for object extraction and retrieval, he has become Associate professor at ENSEA since 2005. He used to work on video and image segmentation, active contours and his current main research topics concern video object detection and classification, content-based video indexing and retrieval systems, scalability of such systems.