Nearest neighbor search for arbitrary kernels with explicit embeddings

Anthony Bourrier, Florent Perronnin, Rémi Gribonval, Patrick Pérez, Hervé Jégou

Nearest neighbor search for arbitrary kernels with explicit embeddings

Anthony Bourrier , Florent Perronnin , Rémi Gribonval, Patrick Pérez, Hervé Jégou

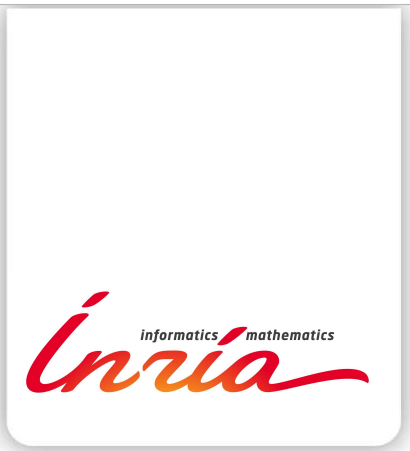
RESEARCH REPORT

N° 8040

August 2012

Project-Teams Metiss and Texmex

ISSN 0249-6399 ISRN INRIA/RR--8040--FR+ENG



Nearest neighbor search for arbitrary kernels with explicit embeddings

Anthony Bourrier* †, Florent Perronnin ‡, Rémi Gribonval†,
Patrick Pérez*, Hervé Jégou†

Project-Teams Metiss and Texmex

Research Report n° 8040 — August 2012 — 14 pages

Abstract: Many algorithms have been proposed to handle efficient search in large databases for simple metrics such as the Euclidean distance. However, few approaches apply to more sophisticated Positive Semi-Definite (PSD) kernels. In this document, we propose for such kernels to use the concept of explicit embedding and to cast the search problem into a Euclidean space. We first describe an exact nearest neighbor search technique which relies on bounds on the approximation of the kernel. We show that, in the case of SIFT descriptors, one can retrieve the nearest neighbor with probability 1 by computing only a fraction of the costly kernels between the query and the database vectors. We then propose to combine explicit embedding with a recent Euclidean approximate nearest neighbor search method and show that it leads to significant improvements with respect to the state-of-the-art methods which rely on an implicit embedding. The database vectors being indexed by short codes, the approach is shown to scale to a dataset comprising 200 million vectors on a commodity server.

Key-words: nearest neighbor search, explicit embedding, product quantization

* Technicolor
† INRIA Rennes-Bretagne Atlantique
‡ Xerox Research Center Europe

RESEARCH CENTRE
RENNES – BRETAGNE ATLANTIQUE

Campus universitaire de Beaulieu
35042 Rennes Cedex

Plongements explicites pour la recherche de plus proche voisin pour un noyau quelconque

Résumé : De nombreuses méthodes ont été proposées pour la recherche efficace de plus proche voisin au sens de la distance euclidienne dans de grandes bases de données. Cependant, peu d'approches s'intéressent au cas plus général où la distance considérée dérive d'un noyau positif. Dans ce document, nous proposons d'exploiter le concept d'*embedding explicite* permettant de plonger les données dans un espace Euclidien et d'effectuer la recherche dans cet espace. Nous décrivons tout d'abord une méthode de recherche exacte de plus proche voisin s'appuyant sur une borne de précision de l'approximation du noyau entre la requête et les éléments de la base. Appliquée à des descripteurs SIFT, cette méthode permet de retrouver le plus proche voisin d'une requête avec probabilité 1 tout en ne calculant qu'une fraction des valeurs du noyau entre la requête et les éléments de la base. Nous proposons ensuite de combiner un *embedding explicite* avec une méthode approchée de recherche de plus proche voisin pour la distance euclidienne, ce qui apporte amélioration substantielle par rapport à des techniques de l'état de l'art s'appuyant sur une approximation implicite. La mise à l'échelle de la méthode est illustrée sur une base de données de 200 millions de vecteurs.

Mots-clés : recherche de plus proche voisin, embedding explicite, product quantization

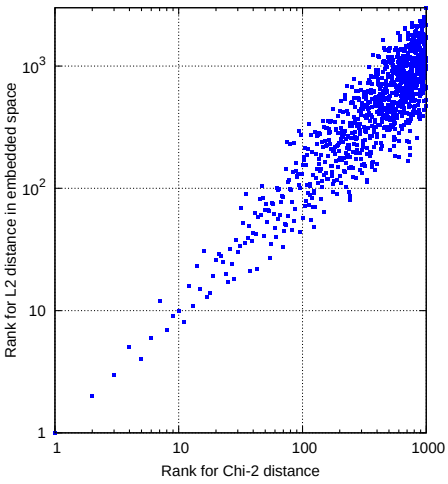


Figure 1: Illustration of the relation between ranking with respect to a given kernel in input space and ranking with respect to Euclidean distance in explicitly embedded space associated with this kernel (Imagenet dataset: 1.261 million images, bag-of-word representation). For a given query, it shows how the ranks obtained for the χ^2 kernel are related to the ranks obtained by the L2 distance in the 512-D embedded space obtained by Kernel PCA. As one can observe, the nearest neighbors are well shared by both metrics.

1 Introduction

The problem of efficient matching on a large scale is especially useful in applications such as large scale image search [24, 25] and classification [9, 1, 5, 19]. This problem is generally treated as an approximate nearest neighbors (ANN) search problem, where one has to find, in a large vector database, the vectors that are closest to a given query with respect to a given distance. Some significant progress has been achieved on this topic in the last decade, in particular with the popular Locality Sensitive Hashing (LSH) algorithm [4]. Efficient and self-tuned implementations of this technique are now available [6], as well as variants such as the FLANN package [18] which considers random trees instead of random projections for better data adaptation. More recently, binarization schemes [28] and compressed-domain approaches [11] have received a particular attention, thanks to the small memory footprint required by these methods. This allows systems to scale to datasets comprising up to one billion elements.

Most of the literature on this problem considers simple metrics, in particular the Euclidean [4] or the Hamming distances [7]. Among the few works addressing the ANN problem for arbitrary kernels, one should notice the Kernelized Locality Sensitive Hashing [14] and the Random Maximum Margin Hashing [13]. Both methods propose a binarization scheme similar to LSH.

The method proposed in this paper addresses the same problem as these techniques, *i.e.* efficiently returning the set of nearest neighbors of a given query vector for a given non-linear PSD kernel. Such kernels are used either because they offer better classification results than the Euclidean distance (for instance the χ^2 distance is known to yield better results on histograms than the Euclidean distance) or to compare objects represented by more complex structures, such as graphs or sequences.

To address this problem, we make use of Kernel Principal Component Analysis (KPCA) [22] to project the data in a new space where the dot product approximates the kernel in the original space. We can therefore cast the problem of searching nearest neighbors for an arbitrary PSD kernel into the problem of searching nearest neighbors for the regular inner product. This is similar in spirit to Hamming Embedding techniques [14, 13, 8], which try to learn some embedding functions such that the comparison in the Hamming space reflects the original metric. However, the choice of the Hamming space severely impacts the quality of the embedding. In contrast, by using an explicit embedding derived from KPCA, we will show that the neighborhoods are very well preserved.

The motivation of this approach is illustrated by Figure 1 for the χ^2 distance. The correlation

between these neighborhoods is obvious: the neighbors for the χ^2 distance are ranked in the top ranks for the Euclidean distance measured in the embedded space. This observation suggests a search method that would 1) Select a subset of potential neighbors based on the simpler metric measured in the embedded space ; 2) Compute the costly target kernel only for these few hypotheses to obtain the final ranking. By deriving a simple bound on the kernel approximation and exploiting it in the search process, in some cases it is possible to obtain an *exact* search system that guarantees to find the *exact* nearest neighbor, and this at the cost of performing the exact search for the Euclidean distance.

More importantly, the use of explicit embedding allows to use a state-of-the-art Euclidean indexing algorithm as those recently proposed in the literature [18, 11]. We show that combining the KPCA embedding approach with a recent Euclidean indexing method based on product quantization leads to a state-of-the-art search system for arbitrary kernels, offering, to the best of our knowledge, the best compromise with respect to speed, memory and search quality. In particular, this approach is experimentally shown to significantly outperform the concurrent approaches based on binary codes [14, 13].

The paper is organized as follows. Section 2 introduces the methodological pre-requisites on explicit embeddings and the concurrent techniques. The datasets used to support the analysis and the comparison are presented in Section 3. In section 4 we derive an error bound on the kernel approximation and use it in a first algorithm which aims at returning the exact nearest neighbors. Finally, Section 5 introduces the proposed approximate search technique for arbitrary kernels with short codes, with experimental evaluation and comparison to state-of-the-art techniques.

2 Related work on embedding techniques

This section first describes the methodological background and methods used to approximate a PSD kernel K defined on $\Omega \times \Omega$, also known as *explicit embedding* (or mapping) techniques. We then briefly describe the recent concurrent techniques against which our algorithm will be evaluated in Section 5.

2.1 Theoretical background

Given such a kernel, there exists an application $\Psi : \Omega \to \mathcal{H}$, where $\mathcal{H}$ is a Hilbert space, such that $\forall \mathbf{x}, \mathbf{y} \in \Omega, K(\mathbf{x}, \mathbf{y}) = \langle \Psi(\mathbf{x}), \Psi(\mathbf{y}) \rangle_{\mathcal{H}}$. In the usual case where K is a Mercer kernel, we have $\mathcal{H} = \ell^2(\mathbb{R})$ and $\Psi(\mathbf{x}) = \left[\sqrt{\lambda_i}\Phi_i(\mathbf{x})\right]_{i=1}^{\infty}$, where:

$$\forall \mathbf{x} \in \Omega, \forall i \in \mathbb{N}^*, \lambda_i \Phi_i(\mathbf{x}) = \int_{y \in \Omega} K(\mathbf{x}, \mathbf{y}) \Phi_i(\mathbf{y}) p(\mathbf{y}) d\mathbf{y} \tag{1}$$

$$\forall i, j \in \mathbb{N}^*, \int_{x \in \Omega} \Phi_i(\mathbf{x}) \Phi_j(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} = \delta_i^j, \tag{2}$$

where p is the probability density of the data. K is thus an Euclidean scalar product in $\mathcal{H}$, which is usually called the implicit space. One can therefore define a distance d on Ω by writing:

$$d(\mathbf{x}, \mathbf{y})^2 = \langle \Psi(\mathbf{x}) - \Psi(\mathbf{y}), \Psi(\mathbf{x}) - \Psi(\mathbf{y}) \rangle_{\mathcal{H}} = K(\mathbf{x}, \mathbf{x}) + K(\mathbf{y}, \mathbf{y}) - 2K(\mathbf{x}, \mathbf{y}) \tag{3}$$

for $\mathbf{x}, \mathbf{y} \in \Omega$. Note that in a search scenario, it is natural to assume the data to be normalized with respect to K, *i.e.* $K(\mathbf{x}, \mathbf{x}) = 1, \forall \mathbf{x} \in \Omega$. This ensures that the $\mathbf{x}$ is among its closest neighbors

(and is its closest neighbor if the kernel is definite). In such a case, the distance between $\mathbf{x}$ and $\mathbf{y}$ reduces to

$$d(\mathbf{x},\mathbf{y})^2 = 2(1 - K(\mathbf{x},\mathbf{y})). \tag{4}$$

In the following, the data is assumed to be normalized with respect to K.

2.2 State-of-the-art search techniques for arbitrary kernels

The two recent techniques presented hereafter are binarization techniques, which both aim at mapping vectors of $\mathbb{R}^D$ into a binary space of dimension B while trying to ensure that the locality is preserved, *i.e.* that the nearest neighbors of a vector for K are among the nearest neighbors of this vector in the binary space with respect to the Hamming distance.

Kernelized Locality Sensitive Hashing. KLSH [14] intends to mimic the LSH method in the implicit space by drawing a random hyperplane in $\mathcal{H}$ from a Gaussian distribution on its normal direction. By considering the data vectors as i.i.d. draws from p and by exploiting the Central Limit theorem that ensures the convergence of the empirical mean of such draws to a Gaussian distribution, it is possible to build explicitly quasi-Gaussian hash functions.

The procedure consists in considering a learning set of vectors $\{\mathbf{x}_1,\ldots,\mathbf{x}_M\}$ and constructing the Gram matrix $\mathbf{K} = (K(\mathbf{x}_i,\mathbf{x}_j))_{i,j=1,\ldots M}$ of this set of vectors. The defined hash functions are of the form $h(\mathbf{x}) = \text{sign}\left(\sum_{i\in I}\mathbf{w}(i)K(\mathbf{x},\mathbf{x}_i)\right)$, where I is a subset of t indices chosen randomly among $\{1,\ldots,M\}$ and $\mathbf{w} = \mathbf{K}^{-\frac{1}{2}}\mathbf{e}_I$, $\mathbf{e}_I$ being the null vector filled with ones at the entries corresponding to the elements of I.

Random Maximum Margin Hashing. RMMH [13] expresses the problem of finding a hyperplane in $\mathcal{H}$ as a SVM problem. Each hash function is chosen as follows: an even number t of vectors $\{\mathbf{x}_1,\ldots,\mathbf{x}_t\}$ are randomly drawn from the training set and half of these vectors are labeled with label $+1$, the other half with label -1. The corresponding hash function is given by

$$h(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{t}\alpha_i y_i K(\mathbf{x},\mathbf{x}_i)\right), \tag{5}$$

where y_i is the label of $\mathbf{x}_i$ and $(\alpha_i)_{i=1}^t$ and b maximize the usal C-SVM cost:

$$\frac{1}{2}\sum_{i,j=1}^{t} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i,\mathbf{x}_j) + C\sum_{i=1}^{t}\xi_i, \tag{6}$$

subject to

$$\forall i,\ \xi_i \geq 0 \text{ and } 1 - \xi_i \leq y_i\left(\sum_{j=1}^{t} y_j\alpha_j K(\mathbf{x}_i,\mathbf{x}_j) + b\right). \tag{7}$$

Intuitively, this formulation finds hash functions that prevent close neighbors from having a different image by the hash function, since the hyperplane is chosen by margin maximization. To the best of our knowledge, this technique is the state-of-the-art for approximate search with arbitrary metrics and short codes.

2.3 Explicit embedding with KPCA

KPCA was introduced in [22] as a way to perform PCA in the implicit space $\mathcal{H}$. it is also a generic way to compute an explicit embedding of the data. Performing PCA on the set of vectors $(\Psi(\mathbf{x}_i))_{i=1}^M$ in implicit space $\mathcal{H}$ consists in projecting them onto the subspace spanned by the first eigenvectors of the compact covariance operator $C(.) = \frac{1}{M}\sum_{i=1}^M \langle \Psi(\mathbf{x}_i), . \rangle_{\mathcal{H}} \Psi(\mathbf{x}_i)$. Let us denote by $(\sigma_i^2)_{i=1}^E$ and $(\mathbf{v}_i)_{i=1}^E$ respectively the first E eigenvalues and eigenvectors of this operator $(E \leq M)$, let $V = \text{span}\langle \mathbf{v}_1, \ldots, \mathbf{v}_E \rangle$ and P_V be the orthogonal projection onto V. It is possible to compute explicitly this projection for any vector of the form $\Psi(\mathbf{x})$ by exploiting the Gram matrix $\mathbf{K} = (K(\mathbf{x}_i, \mathbf{x}_j))_{i,j=1,\ldots M}$. This matrix has the same eigenvalues as the operator C. Let us denote its first E eigenvectors by $(\mathbf{u}_i)_{i=1}^E$ and let $K(\mathbf{x}, .)$ be the vector $[K(\mathbf{x}, \mathbf{x}_1), \ldots, K(\mathbf{x}, \mathbf{x}_M)]^T$. Then the representation of a vector $P_V(\Psi(\mathbf{x}))$ in the basis $(\mathbf{v}_i)_{i=1}^E$ is given by :

$$\forall i \in \{1, \ldots, E\},\ \langle P_V(\Psi(\mathbf{x})), \mathbf{v}_i \rangle_{\mathcal{H}} = \frac{1}{\sigma_i} \langle K(\mathbf{x}, .), \mathbf{u}_i \rangle. \tag{8}$$

KPCA is thus a projection of the vectors onto a E-dimensional subspace which minimizes the following mean squared error:

$$\sum_{i,j=1}^{M} \left(K(\mathbf{x}_i, \mathbf{x}_j) - \langle P_V \Psi(\mathbf{x}_i), P_V \Psi(\mathbf{x}_j) \rangle \right)^2 \tag{9}$$

over all E-dimensional subspaces V. We denote by $\tilde{\Phi}(\mathbf{x})$ the image of $\mathbf{x}$ by the embedding with KPCA and $\tilde{K}$ the approximated kernel, defined by $\tilde{K}(\mathbf{x}, \mathbf{y}) = \langle \tilde{\Phi}(\mathbf{x}), \tilde{\Phi}(\mathbf{y}) \rangle$.

We note that other algorithms have been proposed to perform explicit embedding for specific kernels in a classification context. Maji and Berg describe in [16] an explicit embedding for the intersection kernel. Vedaldi and Zisserman [26, 27] proposed an embedding algorithm for homogeneous kernels based on Fourier sampling. In [19], Perronnin *et al.* suggested applying KPCA independently in each dimension in the case of additive kernels. Rahimi and Recht [21] also proposed an embedding technique based on random projections for shift-invariant kernels.

In all our experiments, we used KPCA as the embedding method for the following reasons. Firstly, we are aiming at a generic kernel-independent solution, and without any particular assumption on the admissible output dimensionality[1]. Secondly, in practice KPCA yields good approximations on real datasets.

3 Datasets and evaluation protocol

For the sake of reproducibility, we have used publicly available datasets of vectors that are large enough to evaluate the interest of nearest neighbor search techniques. Table 1 gives a brief description of these benchmarks, which all consist of image descriptors extracted from real images. These benchmarks consist of three subsets, which will be used in our case as query set, training set and validation set.

- **SIFT1M**: we consider the benchmark introduced in [11] to evaluate approximate search techniques. It consists of one million local SIFT descriptors [15] computed on patches extracted with the Hessian-Affine detector [17].
- **BIGANN**: [10] provides a ground-truth for subsets of increasing size, from one million to one billion vectors. However, they were only interested in Euclidean search. We have,

[1] For instance, [16, 26, 19] require the output dimensionality to be greater or equal to that of the input space.

Dataset	size	dim	descriptor	online
SIFT1M	1M	128	HesAff [17]+SIFT [15]	http://corpus-texmex.irisa.fr
BIGANN	1M–200M	128	DOG+SIFT [15]	http://corpus-texmex.irisa.fr
Imagenet	1.261M	1,000	Bag-of-words	http://www.image-net.org

Table 1: Datasets used for the evaluation of nearest neighbor search

therefore, computed the ground-truth with respect to the χ^2 kernel, for subsets of 1 to 200 millions vectors. Note that these descriptors were extracted using the DoG detector and are therefore not equivalent to those of the SIFT1M benchmark.

- **Imagenet** [5]: we use the pre-computed 1,000-dimensional bag-of-words (BOW) vectors provided with the images of the 2010 Large-Scale Visual Recognition 2010 (ILSVRC'2010). For our search problem, the larger "train" set is used as the database to be searched, the "val" set to learn the parameters, and we randomly selected a subset of 1,000 queries from the "test" set.

Evaluation protocol. For these three datasets, without loss of generality, we are looking for the most similar vectors with respect to the χ^2 kernel, as done in related works [14, 13, 27]. Note that considering the Gaussian kernel does not bring any benefit in the case of a k-nearest neighbors search, since the ranking of the vectors provided by this similarity measure is the same as the one obtained by the Euclidean distance.

The search performance is measured by the Recall@R measure for different values of R, as proposed in other works on approximate search techniques (*e.g.* [11]). It measures the rate of queries for which the nearest neighbor is returned in the first R positions.

4 Exact search with error bounds

As mentioned in section 2.1, any PSD kernel can be viewed as a dot product in a potentially infinite-dimensional space. It was shown that a finite-dimensional mapping of the data could already yield excellent results in the classification context. Such a finite-dimensional mapping offers two advantages in our large-scale search scenario. The first one, which we exploit in this section for exact search, is that it greatly reduces the cost of computing distances since the dot product or the Euclidean distance[2] are generally significantly faster to compute than non-linear PSD kernels. The second one, which we will exploit in the next section, is that it allows using state-of-the art indexing techniques designed for simple metrics, reducing further the nearest neighbor search time.

4.1 Error bound for the KPCA embedding

Precision bounds on the KPCA provided in the literature are usually probabilistic bounds on the error between the empirical eigenvalues/eigenfunctions computed from the matrix $\mathbf{K}$ (denoted σ_i^2 and $\mathbf{u}_i$ in section 2) and their continuous counterparts λ_i and Φ_i. Such bounds are difficult to obtain and they are either not tight [23] or require oracle knowledge about the implicit embedding [2].

[2] Note that $||\tilde{\Phi}(x)||^2 = \tilde{K}(x,x) \approx K(x,x) = 1$ since we assume the data to be normalized with respect to K. Therefore, we can measure the similarity in the embedded space using the dot product, the Euclidean distance or the cosine similarity.

However, it is possible to get tighter precision bounds by exploiting the data we consider. As shown earlier, KPCA is an orthogonal projection onto a subspace V in the implicit space. The approximation error $\epsilon(\mathbf{x},\mathbf{y}) = K(\mathbf{x},\mathbf{y}) - \tilde{K}(\mathbf{x},\mathbf{y})$ can thus be written as a scalar product in the implicit space: $\epsilon(\mathbf{x},\mathbf{y}) = \langle P_{V^\perp}\Psi(\mathbf{x}), P_{V^\perp}\Psi(\mathbf{y})\rangle_{\mathcal{H}}$, where $P_{V^\perp}$ is the orthogonal projection on the subspace orthogonal to V. Let us denote $R(\mathbf{x}) = ||P_{V^\perp}\Psi(\mathbf{x})||_{\mathcal{H}}$. Since the data is supposed normalized with respect to K, this quantity is equal to $\left(1 - ||\tilde{\Phi}(\mathbf{x})||^2\right)^{\frac{1}{2}}$ and can be easily computed in practice. The Cauchy-Schwarz inequality provides a bound on $\epsilon(\mathbf{x},\mathbf{y})$:

$$|\epsilon(\mathbf{x},\mathbf{y})| \leq R(\mathbf{x})R(\mathbf{y}). \tag{10}$$

This bound can be used for nearest neighbor search. Despite its simplicity, we believe it has never been used in such a context. Let us assume that we look for the nearest neighbor of a query $\mathbf{q}$ in the vector dataset $\mathcal{X}$ and let us denote by $\mathbf{x}_1$ and $\tilde{\mathbf{x}}_1$ respectively its nearest neighbor and approximate nearest neighbor. We have the following series of inequalities:

$$K(\mathbf{q},\mathbf{x}_1) \geq K(\mathbf{q},\tilde{\mathbf{x}}_1) \geq \tilde{K}(\mathbf{q},\tilde{\mathbf{x}}_1) - R(\mathbf{q})R(\tilde{\mathbf{x}}_1). \tag{11}$$

For any other basis vector $\mathbf{x}$,

$$K(\mathbf{q},\mathbf{x}) \leq \tilde{K}(\mathbf{q},\mathbf{x}) + R(\mathbf{q})R(\mathbf{x}). \tag{12}$$

Combining (11) and (12), we obtain that, if for some $\mathbf{x}$ we have:

$$\tilde{K}(\mathbf{q},\mathbf{x}) + R(\mathbf{q})R(\mathbf{x}) < \tilde{K}(\mathbf{q},\tilde{\mathbf{x}}_1) - R(\mathbf{q})R(\tilde{\mathbf{x}}_1), \tag{13}$$

then $K(\mathbf{q},\mathbf{x}) < K(\mathbf{q},\mathbf{x}_1)$ and $\mathbf{x}$ cannot be the nearest neighbor of $\mathbf{q}$. The following section describes a procedure that uses this result to avoid kernel computations while still being sure to find the nearest neighbor of a query.

4.2 Exact search procedure

Assuming that the quantities $R(\mathbf{x})$ are computed and stored while performing the embedding of all vectors in dataset $\mathcal{X}$, the nearest neighbor of the query $\mathbf{q}$ is retrieved with probability 1 by using the following procedure, which is illustrated in Figure 2 (left):

1. Compute $\tilde{\Phi}(\mathbf{q})$ and $R(\mathbf{q})$.
2. Compute $\tilde{K}(\mathbf{q},\mathbf{x}) = \langle\tilde{\Phi}(\mathbf{q}),\tilde{\Phi}(\mathbf{x})\rangle$ for all $\mathbf{x}\in\mathcal{X}$.
3. Find $\tilde{\mathbf{x}}_1 = \underset{\mathbf{x}\in\mathcal{X}}{\operatorname{argmax}}\ \tilde{K}(\mathbf{q},\mathbf{x})$
4. Find all elements $\mathbf{y}\in\mathcal{X}$ such that:
   $\tilde{K}(\mathbf{q},\tilde{\mathbf{x}}_1) - R(\mathbf{q})R(\tilde{\mathbf{x}}_1) \leq \tilde{K}(\mathbf{q},\mathbf{y}) + R(\mathbf{q})R(\mathbf{y})$.
   Let us denote by $\mathcal{Y}$ the set of these vectors.
5. The true nearest neighbor of the query is $\mathbf{x}_1 = \underset{\mathbf{y}\in\mathcal{Y}}{\operatorname{argmax}}\ K(\mathbf{q},\mathbf{y})$.

4.3 Complexity Analysis and Experiments

Let N be the size of the base. If the embedding is computed using M vectors and the set $\mathcal{Y}$ is of cardinality k, this method is guaranteed to find the nearest neighbor of $\mathbf{q}$ by computing $M+k$ kernel values and N Euclidean scalar products, vs. N kernel values for the exhaustive search. If

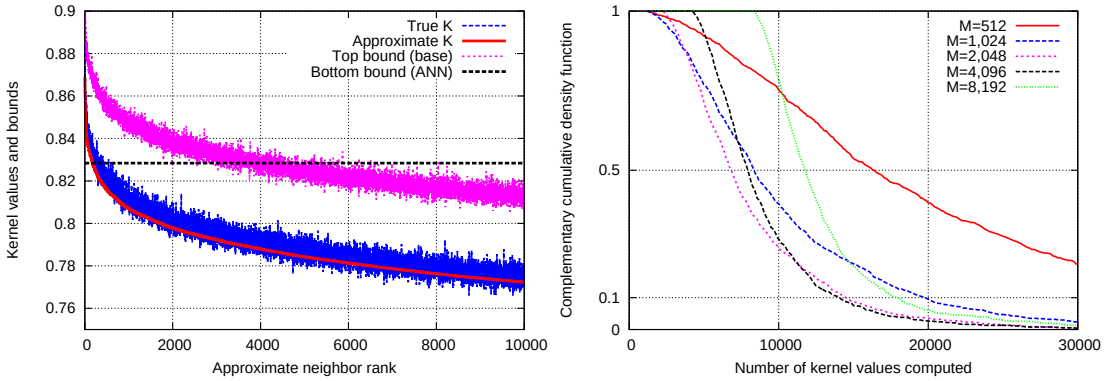


Figure 2: Exact nearest neighbor search in a database of 10M 128-dimensional SIFT vectors, embedded in dimension 128 with KPCA on χ^2 kernel. *Left:* Procedure for a given query. The reranked vectors are the neighbors which have a top bound value higher than the bottom bound of the approximate nearest neighbor. *Right:* one minus the cumulative density function of the number of kernel values computed to find the exact nearest neighbor of a query (for different values of M). The M kernel calculations required to compute the embedding of the query are taken into account.

the cost of the Euclidean scalar product is lower than the cost of computing the kernel (which is very likely to be the case, especially for complicated kernels), and if $M + k \ll N$, we can be sure to retrieve the nearest neighbor while performing much less calculations.

Figure 2 (right) illustrates this method on 10M SIFT and shows that it reduces the number of kernel values computed. Indeed, for $M = 2,048$, 90% of the tested queries require less than 15,000 kernel computations, which represents a comparison with less than 0.15% of the base with respect to the distance we are interested in. A tradeoff has to be found on the number of vectors M used for the embedding: with more vectors the Cauchy-Schwarz bound will be tighter because the residues norms $R(\mathbf{x})$ will decrease, but using too many vectors will result in more kernel computations to perform the embedding on the query. In this example, $M = 2,048$ performs on average better than $M = 8,192$ (8,000 average kernel computations vs. more than 11,000). See also Figures 3 and 4 for an analysis of the impact of parameters M and E on the search quality (independently of the number of kernel computations).

On the other hand, this approach does not perform well on the Imagenet BOW database. In this case, the Cauchy-Schwarz inequality is too pessimistic to provide a good selection of the vectors that are likely to be in the close neighborhood of the query. Indeed, experiments showed that the number of vectors that needed to be reranked was about 400,000 on average, which is a significant part of the database (more than 30%). In this case, it is still possible to select fewer vectors by relaxing the Cauchy-Schwarz inequality and considering that $\alpha R(\mathbf{x})R(\mathbf{y}) \leq \epsilon(\mathbf{x},\mathbf{y}) \leq \beta R(\mathbf{x})R(\mathbf{y})$, where α and β are simply lower and upper bounds for the cosine between vectors $\mathbf{x}$ and $\mathbf{y}$. However, even if we can empirically estimate values of α and β that work for many vectors, we have no guarantee to retrieve the nearest neighbor(s) anymore. Without such guarantees, it is far more efficient to use a state-of-the-art approximate search algorithm designed for the Euclidean distance. This is the topic of the next section.

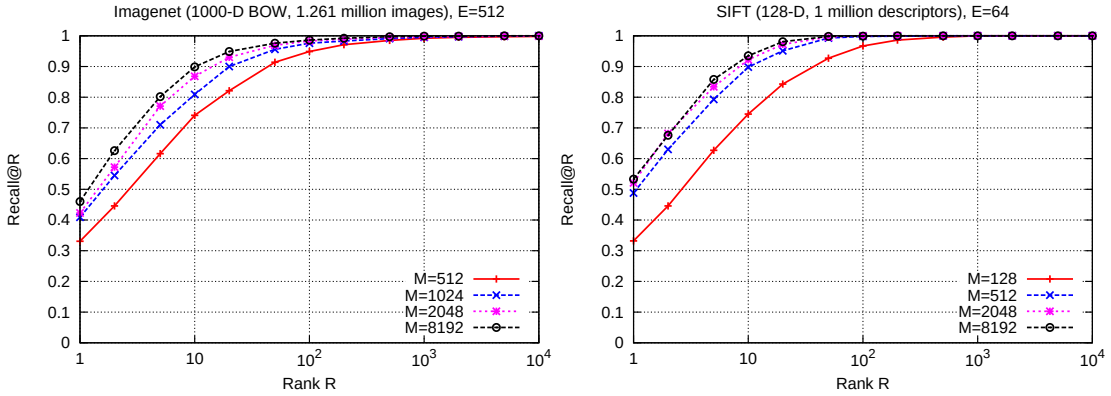


Figure 3: Impact of the size M of the learning set on the ranking performance, measured using recall@R curves. For the two datasets considered (Imagenet and SIFT1M), the vector output by the explicit embedding is about one half of the original descriptor dimensionality (E=512 and E=64, respectively).

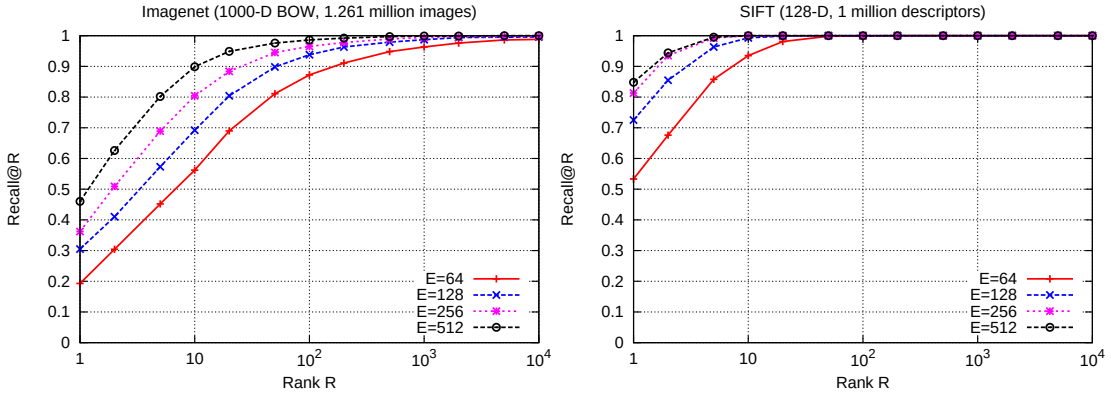


Figure 4: Ranking quality (Recall@R curves) as a function of the embedded vector size E. The number of learning vectors is here fixed to $M = 8,192$. As expected, the longer BOW vectors from Imagenet require a higher embedding dimensionality E.

5 Approximate search with short codes

We now describe and evaluate the proposed technique for approximate search. It relies on two steps:

1. The data is first embedded in an Euclidean space using KPCA.

2. Encoding is performed in the embedded space.

Note that at step 2, any compression technique which aims at approximating the dot-product, the Euclidean distance or the cosine similarity could be employed. This includes among others LSH [3, 4] which approximates the cosine or SH [28] and LSBC [20] which approximate the Euclidean distance. In this section, we focus on a recently proposed technique known as Product Quantization (PQ) which has been shown to be state-of-the-art for the Euclidean distance.

Dataset	SIFT1M			Imagenet		
Pre-processing	None	RR	RP	None	RR	RP
Recall@1	0.17	0.15	**0.19**	0.04	0.04	**0.05**
Recall@10	0.42	0.39	**0.51**	0.12	0.12	**0.14**
Recall@100	0.75	0.74	**0.85**	0.30	0.33	**0.37**
Recall@1000	0.96	0.97	**0.99**	0.60	0.67	**0.69**

Table 2: Choice of the pre-processing: performance of KPCA+PQ (Recall@R) with no pre-processing [11], Random rotation (RR) [12] and Random permutation (RP). Parameters: $M = 1,024$, $E = 64$ and $s = 8$ bytes for the two datasets considered. Averages on 10 experiments.

5.1 Product quantization and variations

Most of the techniques based on short codes consider binary signatures [14, 13, 8]. Recently, Jégou *et al.* proposed [11] instead an approximate search algorithm which produces non-binary codes which allow the estimation of Euclidean distances in the compressed domain. For this purpose, a product quantizer is used to decompose the feature space into a Cartesian product of s subspaces. A given database vector $\mathbf{x}$ is decomposed into s subvectors of equal lengths, each of which is quantized separately using a k-means quantizer with a limited number of centroids. It is typically set to 256 such that each subvector is quantized as a 1-byte code (8 bits) and therefore the vector is represented by a s-bytes code. The estimation of $d(\mathbf{q}, \mathbf{x})$ is done in an asymmetrical manner, *i.e.* the query is not quantized to prevent it from being approximated. Computing a distance estimate requires s table look-ups and additions, which remains competitive compared with operations on binary vectors.

Pre-processing of vectors. The PQ technique was used jointly with PCA for image retrieval [12]. In this context, the authors argue that the PQ technique is more effective if a rotation matrix is used on the output of the PCA and prior to PQ encoding. The rotation matrix is either learned or randomly generated, leading to comparable results. The motivation for a random rotation is to balance the variance between the components, so that all the quantizers have a vector of comparable energy to encode. The drawback is that the individual product quantizers partially encode the same information, since the decorrelation performed by PCA is lost.

In this paper, we also consider the use of a random permutation of the components as an alternative to the random rotation. The random permutation is cheaper to compute. Another benefit is that it ensures that the components remain uncorrelated. Although the random permutation leads to less balanced variance, we will see in our experiments that it leads to better results overall.

5.2 Experiments on approximate search

Impact of the pre-processing. We first study the impact of the three pre-processing techniques we just discussed: (i) no pre-processing is done, as in [11], (ii) a random rotation is performed as done in [12] or (iii) a random permutation of the components is performed as proposed in this paper. Table 2 shows that the choice of the pre-processing stage has an impact on the search quality. On SIFT1M, the random permutation is clearly the best choice. On Imagenet, the difference is less clear but the random permutation still yields better results. Moreover, this pre-processing is cheaper to compute than the random rotation. We will therefore adopt this choice in all subsequent experiments.

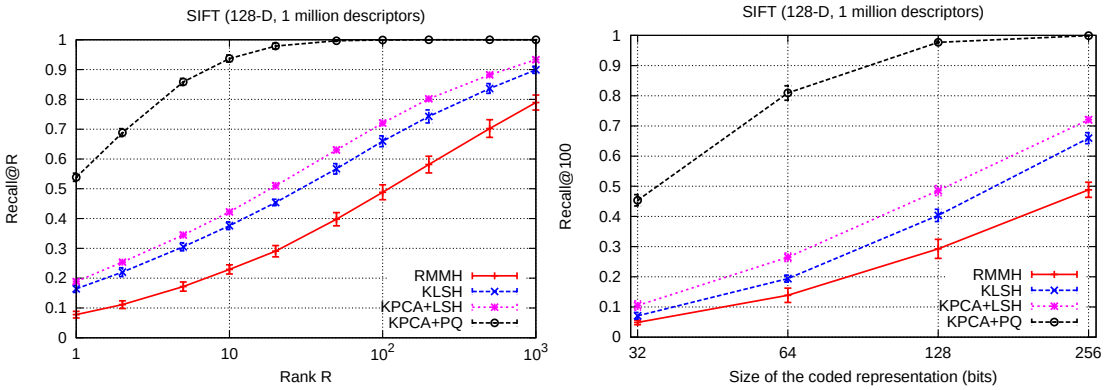


Figure 5: Comparison with the state-of-the-art: search quality for the χ^2 kernel with coded representations for 1M SIFT. *Left*: recall@R curves for a fixed size of B=256 bits. *Right:* recall@100 as a function of the number of bits (32–256). Parameters: KPCA+LSH $\rightarrow E = B$ and M=1,024 ; KPCA+PQ $\rightarrow E$=128 and M=1,024.

Comparison with the state of the art. We now compare the proposed coding scheme based on KPCA+PQ with the state-of-the-art techniques: KLSH [14] and RMMH [13]. We note that the proposed approach differs in two significant ways from KLSH and RMMH: (i) KLSH and RMMH rely on an implicit embedding while we rely on an explicit KPCA embedding and (ii) KLSH and RMMH rely on binary coding while we encode a sample with PQ as a vector of bytes. To better understand the impact of these two differences, we also experimented with a coding scheme based on KPCA+LSH which performs an explicit embedding of the data (as is the case of the proposed scheme) but encodes a data sample as a binary vector (as is the case of KLSH and RMMH).

Given a target number of bits B and a target number of kernel computations M, we still need to choose a number of output dimensions E for KPCA. For KPCA+PQ, we set $E = 128$. For KPCA+LSH, we set $E = B$ which led to optimal or near optimal results in preliminary experiments.

We report results for SIFT1M on Figure 5 and for Imagenet on Figure 6. We repeated all experiments 10 times (with different random draws of the training samples, rotations, permutations, etc.) and we show the average and standard deviation.

We can first observe that, as reported in [13], RMMH outperforms KLSH on the BOW Imagenet features. However, for SIFT features, KLSH performs significantly better than RMMH. For instance, for $B = 128$ bits, the improvement in recall@100 is on the order of 10% absolute. Second, we can see that the simple KPCA+LSH scheme outperforms KLSH and RMMH at all ranks and code sizes. This difference can be significant, *e.g.* on the order of 10% absolute on SIFT1M for 128 bits. This clearly shows the practical superiority of explicit embedding with respect to implicit embedding for approximate search on these datasets. Finally, we can observe that KPCA+PQ performs significantly better than KPCA+LSH, which confirms the superiority of a non-binary encoding scheme as opposed to a binary coding scheme [11].

Overall, the improvement of KPCA+PQ with respect to KLSH and RMMH can be considerable, especially for a small number of bits. For instance, for SIFT1M and for $B = 64$, KPCA+PQ achieves on the order of 80% recall@100 while KLSH achieves only 20%. For Imagenet and for the same number of bits B, the recall@1000 of RMMH is somewhat above 30% while KPCA+PQ achieves on the order of 70%.

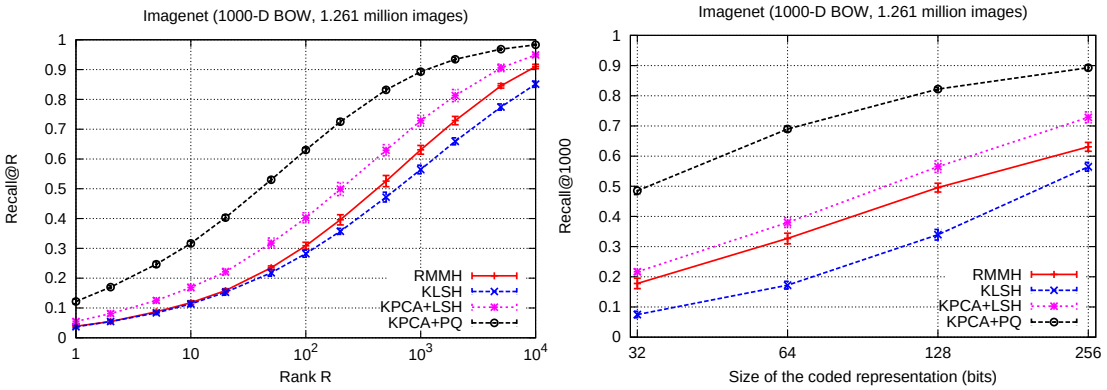


Figure 6: Comparison with the state-of-the-art: search quality for the χ^2 kernel with coded representations for 1.2M BOW. *Left*: recall@R curves for a fixed size of B=256 bits. *Right:* recall@1000 as a function of the number of bits (32–256). Parameters: KPCA+LSH $\rightarrow E = B$ and $M = 1,024$; KPCA+PQ $\rightarrow E = 128$ and $M = 1,024$.

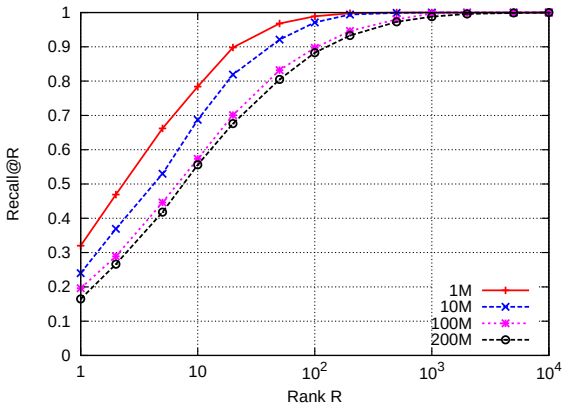


Figure 7: Performance as a function of the database size (1M to 200M vectors). Parameters for KPCA+PQ: $s = 16$, $E = 64$ and $M = 1,024$.

Large scale experiments. Figure 7 reports the Recall@R measured on subsets of 1M–200M SIFT descriptors taken from the BIGANN dataset. These results show that the proposed KPCA+PQ scheme can scale to very large datasets without a drastic loss in performance.

6 Conclusion

We now summarize our main findings. First, the proposed exact search algorithm based on the Cauchy-Schwarz inequality works very well on the small SIFT descriptors, yet is less interesting for higher dimensional representations such as BOW vectors, for which it is necessary to resort to approximate search algorithms. Second, the proposed encoding scheme based on explicit embedding significantly outperforms techniques such as KLSH or RMMH which rely on an implicit embedding. The combination of KPCA+PQ defines the new state-of-the-art for large-scale approximate search with PSD kernels.

References

[1] O. Boiman, E. Shechman, and M. Irani. In defense of nearest neighbor based image classification. In *CVPR*, June 2008.

[2] M. Braun. Accurate error bounds for the eigenvalues of the kernel matrix. *Journal of Machine Learning Research*, 7:2303–2328, 2006.

[3] M. Charikar. Similarity estimation techniques from rounding algorithms. In *ACM STOC*, 2002.

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

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

[6] Wei Dong, Zhe Wang, William Josephson, Moses Charikar, and Kai Li. Modeling LSH for performance tuning. In *Conference on Information and Knowledge Management*, October 2008.

[7] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimension via hashing. In *Proceedings of the International Conference on Very Large DataBases*, pages 518–529, 1999.

[8] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, June 2011.

[9] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *ICCV*, 2005.

[10] H. Jégou, R. Tavenard, M. Douze, and L. Amsaleg. Searching in one billion vectors: re-rank with source coding. In *ICASSP*, Prague Czech Republic, 2011.

[11] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *Trans. PAMI*, 33(1):117–128, January 2011.

[12] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, June 2010.

[13] A. Joly and O. Buisson. Random maximum margin hashing. In *CVPR*, 2011.

[14] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *ICCV*, October 2009.

[15] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[16] Subhransu Maji and Alexander Berg. Max-margin additive models for detection. In *ICCV*, 2009.

[17] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, Jiri Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *IJCV*, 65(1/2):43–72, 2005.

[18] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP*, February 2009.

[19] F. Perronnin, J. Sánchez, and Y. Liu. Large-scale image categorization with explicit data embedding. In *CVPR*, 2010.

[20] M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *NIPS*, 2010.

[21] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *NIPS*, 2007.

[22] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem, 1998.

[23] J. Shawe-Taylor, C. Williams, N. Cristianini, and J. Kandola. On the eigenspectrum of the gram matrix and the generalization error of kernel pca. IEEE *Transactions on Information Theory*, 51(7):2510–2522, 2005.

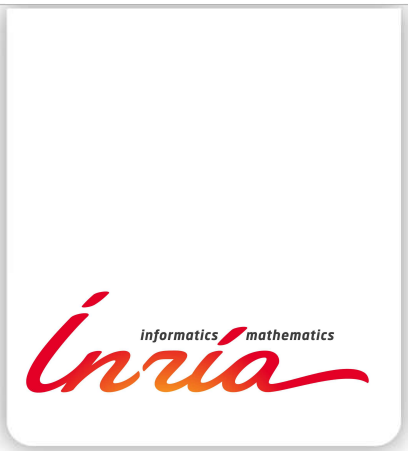
[24] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, October 2003.

[25] A. Torralba, R. Fergus, and Y. Weiss. Small codes and large databases for recognition. In *CVPR*, June 2008.

[26] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. In *CVPR*, 2010.

[27] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. *Trans. PAMI*, to appear.

[28] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, 2008.



RESEARCH CENTRE
RENNES – BRETAGNE ATLANTIQUE

Campus universitaire de Beaulieu
35042 Rennes Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399