Kernel diff-hash

Michael M. Bronstein
Institute of Computational Science
Faculty of Informatics,
Università della Svizzera Italiana
Via G. Buffi 13, Lugano 6900, Switzerland
michael.bronstein@usi.ch

November 3, 2011

Abstract

This paper presents a kernel formulation of the recently introduced diff-hash algorithm for the construction of similarity-sensitive hash functions. Our kernel diff-hash algorithm that shows superior performance on the problem of image feature descriptor matching.

1 Introduction

Efficient representation of data in compact and convenient way to similarity-sensitive hashing methods, first considered in [11] and later in [3, 24, 17, 32, 22]. Similarity-sensitive hashing methods can be regarded as a particular instance of *supervised metric learning* [2, 31], where one tries to construct a hashing function on the data space that preserves known similarity on the training set. Typically, the similarity is binary and can be related to hash collision probability (similar points should collide, and dissimilar points should not collide). Such methods have been enjoying increasing popularity in the computer vision and pattern recognition community in image analysis and retrieval [13, 27, 14, 15, 30, 16], video copy detection [5], and shape retrieval [7].

Shakhnarovich [24] considered parametric hashing functions with affine transformation of the data vectors (projection matrix and threshold vector) followed by the sign function. He posed the problem of similarity-sensitive hash construction as boosted classification, where each dimension of the hash acts as a weak binary classifier. The parameters of the hashing function were learned using AdaBoost. In [25], we used the same setting of the problem and proposed a much simpler algorithm, wherein projections were selected as eigenvectors of the ratio or difference of covariance matrices of similar and dissimilar pairs of data points; the former method was dubbed as LDA-hash and the latter as diff-hash. Applying these methods to SIFT local features in images [18], very compact and accurate binary descriptors were produced.

The inspiration to this paper is the diff-hash method [25]. While being remarkably simple and efficient, this method suffers from two major limitations. First, the length of the hash is limited by the descriptor dimensionality. In some situations, this is a clear disadvantage, as longer hashes allow to produce more accurate matching. Secondly, the affine hashing functions are in many cases too simple and fail to represent correctly the structure of the data. In this paper, we propose a kernel formulation of the diff-hash algorithm which efficiently resolved both problems. We show the performance of the algorithm on the problem of image descriptor matching using the patches dataset from [33] and show that it outperforms the original diff-hash.

2 Background

Let $X \subseteq \mathbb{R}^n$ denote the data space. We denote by $\mathcal{P}$ the set of pairs of similar data points (*positives*) and by $\mathcal{N}$ the set of pairs of dissimilar data points (*negatives*). The problem of similarity-sensitive hashing is to represent the data in a common space $\mathbb{H}^m = \{-1, +1\}^m$ of m-dimensional binary vectors with the Hamming metric $d_{\mathbb{H}^m}(a, b) = \frac{m}{2} - \frac{1}{2}\sum_{i=1}^{m} a_i b_i$ by means of a map $\xi : X \to \mathbb{H}^m$ such that $d_{\mathbb{H}^m} \circ (\xi \times \xi)|_{\mathcal{P}} \approx 0$ on and $d_{\mathbb{H}^m} \circ (\xi \times \xi)|_{\mathcal{N}} \approx m$. Alternatively, this can be expressed as having $\mathbb{E}\{d_{\mathbb{H}^m} \circ (\xi \times \eta)|\mathcal{P}\} \approx 0$ (i.e., the hash has high collision probability on the set of positives) and $\mathbb{E}\{d_{\mathbb{H}^m} \circ (\xi \times \eta)|\mathcal{N}\} \gg 0$. The former can be interpreted as the *false negative rate* (FNR) and the latter as the *false positive rate* (FPR).

2.1 Similarity-sensitive hashing (SSH)

To further simplify the problem, Shakhnarovich [24] considered parametric hashing function of the form $\xi(\mathbf{x}) = \text{sign}(\mathbf{P}\mathbf{x} + \mathbf{a})$, where $\mathbf{P}$ is $m \times n$ *pro-*

jection matrix and $\mathbf{a}$ is an $m \times 1$ *threshold* vector. The similarity-sensitive hashing (SSH) algorithm considers the hash construction as boosted binary classification, where each hash dimension acts as a weak binary classifier. For each dimension, AdaBoost is used to maximize the following loss function

$$\min_{\mathbf{p}_i, a_i} \sum_{(\mathbf{x},\mathbf{x}') \in \mathcal{P} \cup \mathcal{N}} w_i(\mathbf{x},\mathbf{x}') s(\mathbf{x},\mathbf{x}') \xi_i(\mathbf{x}) \xi_i(\mathbf{x}'), \tag{1}$$

where $\xi_i(\mathbf{x}) = \mathrm{sign}(\mathbf{p}_i^{\mathrm{T}}\mathbf{x} + a_i)$, $s(\mathbf{x},\mathbf{x}') = 1$ for $(\mathbf{x},\mathbf{x}') \in \mathcal{N}$ and 0 for $(\mathbf{x},\mathbf{x}') \in \mathcal{P}$ and $w_i(\mathbf{x},\mathbf{x}')$ is the AdaBoost weigh for pair $(\mathbf{x},\mathbf{x}')$ at ith iteration. Shakhnarovich [24] selected $\mathbf{p}_i$ as the axis projection onto which minimizes the objective. In [5, 8], minimization problem (1) was relaxed in the following way : First, removing the non-linearity and setting $a_i = 0$, find the projection vector $\mathbf{p}_i$. Then, fixing the projection $\mathbf{p}_i$, find the threshold a_i. The disadvantages of the boosting-based SSH is first high computational complexity, and second, the tendency to find unnecessary long hashes.[1]

2.2 Diff-hash

In [25], we proposed a simpler approach, computing the similarity-sensitive hashing by minimizing

$$\begin{aligned} L(\xi) &= \alpha \mathbb{E}\{d_{\mathbb{H}^m} \circ (\xi \times \xi) | \mathcal{P}\} - \mathbb{E}\{d_{\mathbb{H}^m} \circ (\xi \times \xi) | \mathcal{N}\} \\ &= \tfrac{m(\alpha-1)}{2} + \tfrac{1}{2}\mathbb{E}\{\xi^{\mathrm{T}}\xi | \mathcal{N}\} - \tfrac{\alpha}{2}\mathbb{E}\{\xi^{\mathrm{T}}\xi | \mathcal{P}\} \end{aligned} \tag{2}$$

w.r.t. the map ξ. Problem (2) is equivalent, up to constants, to minimizing the correlations

$$\begin{aligned} L(\mathbf{P}, \mathbf{a}) &= \mathbb{E}\{\mathrm{sign}(\mathbf{P}\mathbf{x} + \mathbf{a})^{\mathrm{T}} \mathrm{sign}(\mathbf{P}\mathbf{x} + \mathbf{a}) | \mathcal{N}\} \\ &- \alpha \mathbb{E}\{\mathrm{sign}(\mathbf{P}\mathbf{x} + \mathbf{a})^{\mathrm{T}} \mathrm{sign}(\mathbf{P}\mathbf{x} + \mathbf{a}) | \mathcal{P}\} \end{aligned} \tag{3}$$

w.r.t. the projection matrix $\mathbf{P}$ and threshold vector $\mathbf{a}$. The first and second terms in (3) can be thought of as FPR and FNR, respectively. The parameter α controls the tradeoff between FPR and FNR. The limit case $\alpha \gg 1$ effectively considers only the positive pairs ignoring the negative set.

Problem (3) is a highly non-convex non-linear optimization problem difficult to solve straightforwardly. Following [5, 8], we simplify the problem in

[1] The second problem can be partially resolved by using sequential probability testing [6] which creates hashes of minimum expected length.

the following way. First, ignore the threshold and solve a simplified problem without the sign non-linearity for projection matrix $\mathbf{P}$,

$$\begin{aligned}
\min_{\mathbf{P}^{\mathrm{T}}\mathbf{P}=\mathbf{I}} \quad & \mathbb{E}\{(\mathbf{Px})^{\mathrm{T}}(\mathbf{Px})|\mathcal{N}\} - \alpha\mathbb{E}\{(\mathbf{Px})^{\mathrm{T}}(\mathbf{Px})|\mathcal{P}\} = \\
\min_{\mathbf{P}^{\mathrm{T}}\mathbf{P}=\mathbf{I}} \quad & \mathrm{tr}\,(\mathbf{P}^{\mathrm{T}}\mathbb{E}\{\mathbf{xx}^{\mathrm{T}}|\mathcal{N}\}\mathbf{P}) - \alpha\mathrm{tr}\,(\mathbf{P}^{\mathrm{T}}\mathbb{E}\{\mathbf{xx}^{\mathrm{T}}|\mathcal{P}\}\mathbf{P}) = \\
\min_{\mathbf{P}^{\mathrm{T}}\mathbf{P}=\mathbf{I}} \quad & \mathrm{tr}\,(\mathbf{P}^{\mathrm{T}}(\mathbf{\Sigma}_{\mathcal{N}} - \alpha\mathbf{\Sigma}_{\mathcal{P}})\mathbf{P}),
\end{aligned} \tag{4}$$

where $\mathbf{\Sigma}_{\mathcal{P}}, \mathbf{\Sigma}_{\mathcal{N}}$ denote the $n \times n$ covariance matrices of the positive and negative data. The solution of (4) is given explicitly as $\mathbf{P} = [\lambda_{n-m+1}^{1/2}\mathbf{v}_{n-m+1}, \ldots, \lambda_n^{1/2}\mathbf{v}_n]^{\mathrm{T}}$, the m smallest eigenvectors of the matrix $\mathbf{\Sigma}_{\mathcal{N}} - \alpha\mathbf{\Sigma}_{\mathcal{P}} = \mathbf{V\Lambda V}^{\mathrm{T}}$ of weighted covariance differences.[2]

Second, fixing the projections find optimal threshold vector $\mathbf{a}$,

$$\begin{aligned}
\min_{\mathbf{a}} \quad & \mathbb{E}\{\mathrm{sign}(\mathbf{Px}+\mathbf{a})^{\mathrm{T}}\mathrm{sign}(\mathbf{Px}'+\mathbf{a})|\mathcal{N}\} \\
& -\alpha\mathbb{E}\{\mathrm{sign}(\mathbf{Px}+\mathbf{a})^{\mathrm{T}}\mathrm{sign}(\mathbf{Px}'+\mathbf{a})|\mathcal{P}\} = \\
\min_{\{a_i\}} \quad & \textstyle\sum_{i=1}^{m} \mathbb{E}\{\mathrm{sign}(\mathbf{p}_i^{\mathrm{T}}\mathbf{x}+a_i)\mathrm{sign}(\mathbf{p}_i^{\mathrm{T}}\mathbf{x}+a_i)|\mathcal{N}\} \\
& -\alpha\textstyle\sum_{i=1}^{m} \mathbb{E}\{\mathrm{sign}(\mathbf{p}_i^{\mathrm{T}}\mathbf{x}+a_i)\mathrm{sign}(\mathbf{p}_i^{\mathrm{T}}\mathbf{x}+a_i)|\mathcal{P}\}.
\end{aligned}$$

The problem is separable and can be solved independently in each dimension i. The above terms are the false positive and negative rates as function of the threshold a_i,

$$\begin{aligned}
\mathrm{FNR}(a_i) \quad = & \quad \Pr(\mathbf{p}_i^{\mathrm{T}}\mathbf{x}+a_i<0 \;\text{ and }\; \mathbf{p}_i^{\mathrm{T}}\mathbf{x}'+a_i>0|\mathcal{P}) \\
+ & \quad \Pr(\mathbf{p}_i^{\mathrm{T}}\mathbf{x}+a_i>0 \;\text{ and }\; \mathbf{p}_i^{\mathrm{T}}\mathbf{x}'+a_i<0|\mathcal{P})
\end{aligned}$$

and

$$\begin{aligned}
\mathrm{FPR}(a_i) \quad = & \quad \Pr(\mathbf{p}_i^{\mathrm{T}}\mathbf{x}+a_i<0 \;\text{ and }\; \mathbf{p}_i^{\mathrm{T}}\mathbf{x}'+a_i<0|\mathcal{N}) \\
+ & \quad \Pr(\mathbf{p}_i^{\mathrm{T}}\mathbf{x}+a_i>0 \;\text{ and }\; \mathbf{p}_i^{\mathrm{T}}\mathbf{x}'+a_i>0|\mathcal{N}).
\end{aligned}$$

The above probabilities can be estimated from histograms (cumulative distributions) of $\mathbf{p}_i^{\mathrm{T}}\mathbf{x}$ and $\mathbf{q}_i^{\mathrm{T}}\mathbf{y}$ on the positive and negative sets. The optimal threshold

$$a_i^* \quad = \quad \operatorname*{argmin}_{a} \;\alpha\mathrm{FNR}(a) + \mathrm{FPR}(a) \tag{5}$$

is obtained by means of one-dimensional exhaustive search.

[2] The name of the algorithm *diff-hash* refers in fact to this covariance difference matrix.

3 Kernel diff-hash

An obvious disadvantage of diff-hash (and spectral methods in general) compared to AdaBoost-based methods is that it must be *dimensionality-reducing*: since we compute projection $\mathbf{P}$ as the eigenvectors of a covariance matrix of size $n \times n$, the dimensionality of the embedding space must be $m \leq n$. This restriction is limiting in many cases, as first it depends on the data dimensionality, and second, such a dimensionality may be too low and a longer hash would achieve better performance. Furthermore, the affine parametric form of the embedding ξ is in many cases an oversimplification, and some more generic map is required.

In this paper, we cope with both problems using a kernel formulation, which transforms the data into some feature space that is never dealt with explicitly (only inner products in this space, referred to as *kernel* [23], are required). In order to simplify the following discussion, since the problem is separable (as we have seen, projection in each dimension corresponds to a eigenvector of the covariance matrix difference), we consider one-dimensional projections. The whole method is summarized in Algorithm 1.

3.1 Projection computation

Let $k_X : X \times X \to \mathbb{R}$ be a positive semi-definite kernel, and let $\phi : \mathbf{x} \mapsto k_X(\cdot, \mathbf{x})$. Thus, ϕ maps the data into some feature space, which we represent here as a Hilbert space $\mathcal{V}$ (possibly of infinite dimension) with an inner product $\langle \cdot, \cdot \rangle_{\mathcal{V}}$, and satisfies $k_X(\mathbf{x}, \mathbf{x}') = \langle k_X(\cdot, \mathbf{x}), k_X(\cdot, \mathbf{x}') \rangle_{\mathcal{V}} = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{V}}$.

The idea of kernelization is to replace the original data X with the corresponding feature vectors $\phi(X)$, replacing the linear projection $\mathbf{p}^{\mathrm{T}}\mathbf{x}$ with $p(\mathbf{x}) = \sum_{i=1}^{l} \beta_i \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle_{\mathcal{V}} = \boldsymbol{\beta}^{\mathrm{T}}[k_X(\mathbf{x}_1, \mathbf{x}) \ldots k_X(\mathbf{x}_l, \mathbf{x})]$. Here, $\boldsymbol{\beta}$ is a vector of unknown linear combination coefficients, and $\mathbf{x}_1, \ldots, \mathbf{x}_l$ denote some representative points in the data space.

In this formulation, at the projection computation stage we minimize, for each dimension

$$
\begin{aligned}
&\min_{\boldsymbol{\beta}} \quad \frac{1}{|\mathcal{N}|}\sum_{(\mathbf{x},\mathbf{y})\in\mathcal{N}} p(\mathbf{x})q(\mathbf{y}) - \frac{\alpha}{|\mathcal{P}|}\sum_{(\mathbf{x},\mathbf{y})\in\mathcal{P}} p(\mathbf{x})q(\mathbf{y}) = \\
&\min_{\boldsymbol{\beta}} \quad \frac{1}{|\mathcal{N}|}\sum_{(\mathbf{x},\mathbf{x}')\in\mathcal{N}}\sum_{i,j=1}^{l} \beta_i\langle\phi(\mathbf{x}_i),\phi(\mathbf{x})\rangle_{\mathcal{V}}\beta_j\langle\phi(\mathbf{x}_j),\phi(\mathbf{x}')\rangle_{\mathcal{V}} = \\
&\qquad -\frac{\alpha}{|\mathcal{P}|}\sum_{(\mathbf{x},\mathbf{x}')\in\mathcal{P}}\sum_{i,j=1}^{l} \beta_i\langle\phi(\mathbf{x}_i),\phi(\mathbf{x})\rangle_{\mathcal{V}}\beta_j\langle\phi(\mathbf{x}_j),\phi(\mathbf{x}')\rangle_{\mathcal{V}} = \\
&\min_{\boldsymbol{\beta}} \quad \frac{1}{|\mathcal{N}|}\boldsymbol{\beta}^{\mathrm{T}}\mathbf{K}_{\mathcal{N}}\mathbf{K}_{\mathcal{N}}^{\mathrm{T}}\boldsymbol{\beta} - \frac{\alpha}{|\mathcal{P}|}\boldsymbol{\beta}^{\mathrm{T}}\mathbf{K}_{\mathcal{P}}\mathbf{K}_{\mathcal{P}}^{\mathrm{T}}\boldsymbol{\beta} = \min_{\boldsymbol{\beta}}\boldsymbol{\beta}^{\mathrm{T}}\mathbf{K}\boldsymbol{\beta},
\end{aligned}
$$

where $\mathbf{K}_{\mathcal{N}}$ and $\mathbf{K}_{\mathcal{P}}$ denote $l\times|\mathcal{N}|$ and $l\times|\mathcal{P}|$ matrices with elements $k_X(\mathbf{x}_i,\mathbf{x})$. The optimal projection coefficients $\boldsymbol{\alpha}$ minimizing are given as the smallest eigenvectors of the $l \times l$ matrix $\mathbf{K} = \frac{1}{|\mathcal{N}|}\mathbf{K}_{\mathcal{N}}\mathbf{K}_{\mathcal{N}}^{\mathrm{T}} - \frac{\alpha}{|\mathcal{P}|}\mathbf{K}_{\mathcal{P}}\mathbf{K}_{\mathcal{P}}^{\mathrm{T}}$.

The kernel k_X can be selected to account correctly for the structure of the data space X. In our formulation, the dimensionality of the hash is bounded by the number of the basis vectors, $m \leq l$, which is limited only by the training set size and computational complexity.

3.2 Threshold selection

As previously, the threshold should be selected to minimize the false positive and false negative rates, that can be expressed, as previously, as

$$
\begin{aligned}
\mathrm{FNR}(a) &= \Pr(p(\mathbf{x})+a<0 \text{ and } p(\mathbf{x}')+a>0|\mathcal{P}) \\
&+ \Pr(p(\mathbf{x})+a>0 \text{ and } p(\mathbf{x}')+a<0|\mathcal{P}), \\
\mathrm{FPR}(a) &= \Pr(p(\mathbf{x})+a<0 \text{ and } p(\mathbf{x}')+a<0|\mathcal{N}) \\
&+ \Pr(p(\mathbf{x})+a>0 \text{ and } p(\mathbf{x}')+a>0|\mathcal{N}),
\end{aligned}
$$

The optimal threshold is obtained as

$$
a^* = \operatorname*{argmin}_{a} \ \alpha\mathrm{FNR}(a) + \mathrm{FPR}(a). \tag{6}
$$

3.3 Hash function application

Once the coefficients $\mathbf{B}$ and threshold $\mathbf{a}$ are computed, given a new data point $\mathbf{x}$, the corresponding m-dimensional binary hash vector is constructed

```
Algorithm 1: Kernel diff-hash algorithm.
Input: Positives set $\mathcal{P} \subset X \times X$, Negatives set $\mathcal{N} \subset X \times X$;
       Dimensionality of the hash $m$; Kernel $k_X$; Set of vectors
       $\mathbf{x}_1, \dots, \mathbf{x}_l$.
Output: Optimal combination coefficient matrix $\mathbf{B}$ of size $m \times l$;
        optimal offset vector $\mathbf{a}$ of size $m \times 1$.
Compute the kernel matrices $\mathbf{K}_{\mathcal{P}}, \mathbf{K}_{\mathcal{N}}$ of size $l \times |\mathcal{P}|$ and $l \times |\mathcal{N}|$,
respectively.
Compute the matrix $\mathbf{K} = \frac{1}{|\mathcal{N}|}\mathbf{K}_{\mathcal{N}}\mathbf{K}_{\mathcal{N}}^{\mathrm{T}} - \frac{\alpha}{|\mathcal{P}|}\mathbf{K}_{\mathcal{P}}\mathbf{K}_{\mathcal{P}}^{\mathrm{T}}$.
Perform eigendecomposition $\mathbf{K} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^{\mathrm{T}}$.
for $i = 1, \dots, m$ do
    Set the $i$th row of the coefficient matrices to be the $i$th smallest
    eigenvectors, $\boldsymbol{\beta}_i^{\mathrm{T}} = \lambda_{n-1+1}\mathbf{v}_{n-i+1}^{\mathrm{T}}$.
    Compute the projection $p_i(\mathbf{x}) = \boldsymbol{\beta}_i^{\mathrm{T}}\mathbf{K}_X$.
    Compute the rates $\mathrm{FNR}(a_i)$ and $\mathrm{FPR}(a_i)$ for $p_i(\mathbf{x}) + a_i$, as
    function of threshold $a_i$.
    Compute the optimal thresholds
        $a_i^* = \operatorname*{argmin}_a \alpha\mathrm{FNR}(a) + \mathrm{FPR}(a)$.
```

as $\xi(\mathbf{x}) = \mathrm{sign}(\mathbf{B}(k_X(\mathbf{x}_1, \mathbf{x}), \dots, k_X(\mathbf{x}_l, \mathbf{x}))^{\mathrm{T}} + \mathbf{a})$. Note that this embedding is kernel-dependent and has a more generic form than the affine transformation used in [24, 25].

4 Results

In order to test our approach, we applied it to the problem of image feature matching. This problem is a core of many modern Internet-scale computer vision applications, including city scale reconstruction [1]. The basic underlying task in these problems, repeated millions and billions of times, is the comparison of local image features (SIFT [18] or similar methods [21, 4, 26]). Typically, these features are represented by means of multidimensional descriptors vectors (e.g. SIFT is 128-dimensional) and compared using the

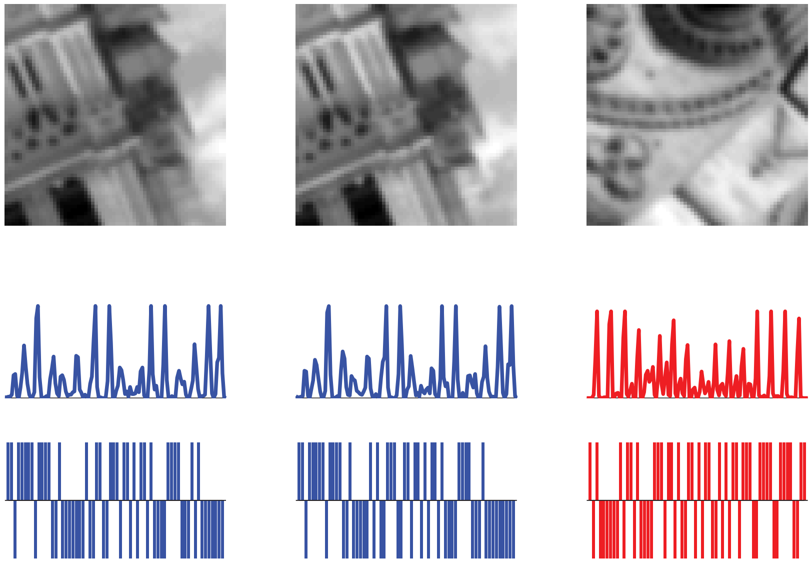


Figure 1: Example of a positive (left, middle) and negative (left, right) pair of image patches and corresponding descriptors. First row: patches, second row: SIFT descriptors, third row: binary descriptors of length 32 produced using kDIF.

Euclidean distance. With very large datasets (containing $10^6 - 10^9$ feature points), severe scalability issues are encountered, including problems of storage and similarity query on feature descriptors. Efficient representation and comparison of feature descriptors have been addressed in many recent works in the computer vision community (see, e.g., [20, 19, 28, 12, 33, 34, 10, 9]). In [25], we proposed using similarity-sensitive hashing methods to produce compact *binary descriptors* [25]. Such descriptors have several appealing properties that make them especially suitable in large-scale applications. First, they are compact (typically, $64-256$ bits, compared to at least 1024 required for the standard SIFT) and easy to store in standard databases. Second, the comparison of binary descriptors is done using the Hamming metric, which amounts to XOR and bit count – an operation that can be carried out extremely efficiently on modern CPU architectures, significantly faster than the computation of Euclidean or other L_p distances. Finally, the construction of the binarization transformations involves metric learning, thus modeling more correctly the distance between the descriptors, which is usually non-Euclidean. In particular, this allows to compensate for imperfect invariance of the descriptor (since viewpoint transformations are only approximately locally affine) and cope with descriptor variability in pairs of images with wide baseline. As a result of this last property, the use of similarity-sensitive hashing reduces the descriptor size while actually *improving* its performance [25], unlike other methods that typically come at the price of decreased performance.

In our experiments, we used data from [33]. The datasets contained rectified and normalized 64×64 patches extracted from multiple images depicting three different scenes (Trevi fountain, Notre Dame cathedral, and Half Dome). The first two scenes were similar representing architectural landmarks; the last scene was different representing a natural mountain environment. In each scene, a total of nearly 100K patches corresponding to around 30K different feature points were available; each feature appeared multiple times. For training, we used 100K pairs of patches corresponding to different views of the same points as positives, and 200K pairs of patches from different points as negatives (Figure 1). For testing, a different subset of the dataset containing 50K positive and 50K negative pairs was used.

In each patch, a 128-dimensional (8-bit per dimension) SIFT descriptor was computed using the toolbox of Vedaldi [29]. We compared the performance of binary descriptor obtained by means of the diff-hash method of Strecha *at al.* [25] (DIF) and our kernel version (kDIF). Diff-hash appeared to be the best performing algorithm in an extensive set of evaluations done in [25]. Since kDIF is an extended version of DIF, we choose to compare to this method. In both methods, we used the value $\alpha = 25$ which was experimentally found to produce the best results. In kDIF, we used a Gaussian kernel with the Mahalanobis distance of the form $k_X(\mathbf{x}, \mathbf{x}') = \exp\{-(\mathbf{x} - \mathbf{x}')^{\mathrm{T}} \mathbf{\Sigma}_X^{-1/2} (\mathbf{x} - \mathbf{x}')\}$. The same training and testing data were used for all methods. For reference, we show the Euclidean distance between the original SIFT descriptors.

Figures 2–3 show the performance of different hashing algorithms as a function of m on different datasets. Several conclusions can be drawn from this figure. First, kDIF appears to consistently outperform DIF on all three scenes for the same hash length m. Second, for sufficiently large m, our method outperforms SIFT while still being more compact. Third, the learned hashing functions generalize gracefully to other scenes, though slight performance degradation is noticeable when training on mountain scene (Half Dome) and using the learned hash in an architectural scene (Note Dame).

Figure 4 compares the performance of different descriptors in terms of FNR at two low FPR points (0.1% and 0.01%). Binary descriptors outperform raw SIFT while being 2-4 more compact (to say nothing about the lower computational complexity of the Hamming distance compared to the Euclidean distance). Second, kDIF consistently outperforms DIF. Third, one can see that using longer hash ($m > 128$) increases the performance.

Figure 5 shows a few examples of first matches between patch descriptors obtained using Euclidean distance and the Hamming distance on the hashed descriptors using our method. Our method provides superior performance.

5 Conclusions

We presented kernel formulation of diff-hash similarity-sensitive hashing algorithm and showed how this method can be used to produce efficient and compact binary feature descriptors. Though we showed results with SIFT, the method is generic and can be applied to any local feature descriptor. Our method showed superior results compared to the original diff-hash proposed in [25], and is more generic as it allows to obtain hashes of any length and also incorporate nonlinearity through the choice of the kernel.

References

[1] S. Agarwal, N. Snavely, I. Simon, S.M. Seitz, and R. Szeliski. Building Rome in one day. In *Proc. ICCV*, 2009.

[2] V. Athitsos, J. Alon, S. Sclaroff, and G. Kollios. Boostmap: a method for efficient approximate similarity ranking. In *Proc. CVPR*, 2004.

[3] M. Bawa, T. Condie, and P. Ganesan. LSH forest: self-tuning indexes for similarity search. In *Proc. Int. Conf. World Wide Web*, pages 651–660. ACM, 2005.

[4] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. *CVIU*, 10(3):346–359, 2008.

[5] A. M. Bronstein, M. M. Bronstein, and R. Kimmel. The video genome. Technical Report arXiv:1003.5320v1, 2010.

[6] A. M. Bronstein, M. M. Bronstein, M. Ovsjanikov, and L. J. Guibas. WaldHash: sequential similarity-preserving hashing",. Technical Report CIS-2010-03, Technion, Israel, 2010.

[7] A.M. Bronstein, M.M. Bronstein, M. Ovsjanikov, and L.J. Guibas. Shape Google: geometric words and expressions for invariant shape retrieval. *ACM TOG*, 2010.

[8] M. M. Bronstein, A. M. Bronstein, F. Michel, and N. Paragios. Data fusion through cross-modality metric learning using similarity-sensitive hashing. In *Proc. CVPR*, 2010.

[9] M. Brown, G. Hua, and S. A. Winder. Discriminative learning of local image descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(PrePrints), 2010.

[10] V. Chandrasekhar, G. Takacs, D. M. Chen, S.S. Tsai, R. Grzeszczuk, and B. Girod. Chog: Compressed histogram of gradients a low bit-rate feature descriptor. In *Proc. CVPR*, pages 2504–2511, 2009.

[11] A. Gionis, P. Indik, and R. Motwani. Similarity Search in High Dimensions via Hashing. In *Int. Conf. Very Large Databases*, 2004.

[12] G. Hua, M. Brown, and S. Winder. Discriminant embedding for local image descriptors. In *Proc. ICCV*, 2007.

[13] P. Jain, B. Kulis, and K. Grauman. Fast image search for learned metrics. In *Proc. CVPR*, 2008.

[14] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *Proc. ECCV*, pages 304–317, 2008.

[15] H. Jégou, M. Douze, and C. Schmid. Packing Bag-of-Features. In *Proc. ICCV*, 2009.

[16] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *Trans. PAMI*, 2010.

[17] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *Proc. NIPS*, pages 1042–1050, 2009.

[18] D.G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 20(2):91–110, 2004.

[19] K. Mikolajczyk and J. Matas. Improving descriptors for fast tree matching by optimal linear projection. In *Proc. ICCV*, 2007.

[20] K. Mikolajczyk and C. Schmid. A Performance Evaluation of Local Descriptors. In *Proc. CVPR*, pages 257–263, June 2003.

[21] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *IJCV*, 65(1/2):43–72, 2005.

[22] M. Raginsky and S. Lazebnik. Locality-Sensitive Binary Codes from Shift-Invariant Kernels. *Proc. NIPS*, 2009.

[23] B. Schölkopf, A. Smola, and K.R. Müller. Kernel principal component analysis. *Proc. ICANN*, pages 583–588, 1997.

[24] G. Shakhnarovich. *Learning Task-Specific Similarity.* PhD thesis, MIT, 2005.

[25] C. Strecha, A. M. Bronstein, M. M. Bronstein, and P. Fua. LDAHash: improved matching with smaller descriptors. *Trans. PAMI*, 2011.

[26] E. Tola, V. Lepetit, and P. Fua. Daisy: an Efficient Dense Descriptor Applied to Wide Baseline Stereo. *Trans. PAMI*, 32(5):815–830, 2010.

[27] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: a large dataset for non-parametric object and scene recognition. *Trans. PAMI*, 30(11):1958–1970, 2008.

[28] T. Tuytelaars and C. Schmid. Vector quantizing feature space with a regular lattice. *Proc. ICCV*, 2007.

[29] A. Vedaldi. An open implementation of the SIFT detector and descriptor. Technical Report 070012, UCLA CSD, 2007.

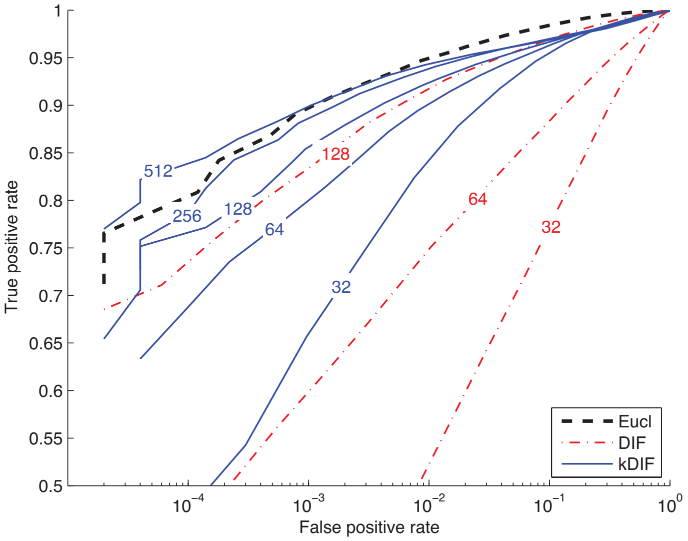
[30] J. Wang, S. Kumar, and S. F. Chang. Semi-supervised hashing for scalable image retrieval. In *CVPR*, 2010.

[31] J. Wang, S. Kumar, and S. F. Chang. Sequential projection learning for hashing with compact codes. In *ICML*, 2010.

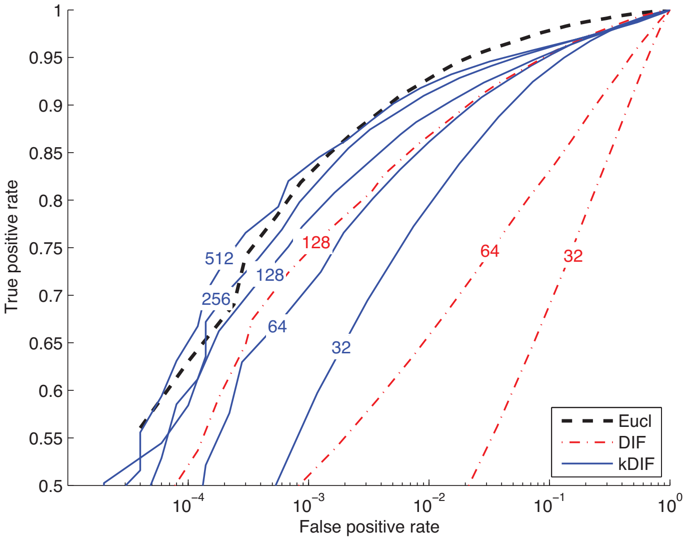
[32] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. *Proc. NIPS*, 21:1753–1760, 2009.

[33] S. A. Winder and M. Brown. Learning local image descriptors. In *Proc. CVPR*, Minneapolis, MI, June 2007.

[34] S. A. Winder, G. Hua, and M. Brown. Picking the best DAISY. In *Proc. CVPR*, June 2009.

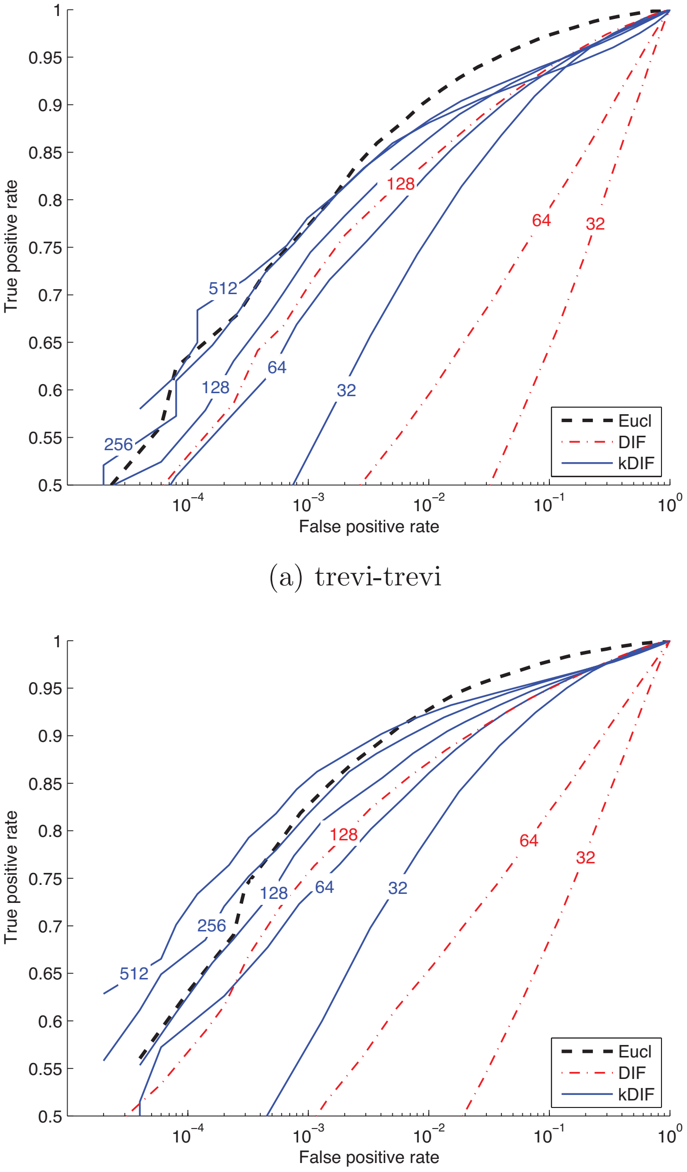


(a) halfdome-halfdome



(b) halfdome-notredame

Figure 2: ROC curves showing the performance of Euclidean distance between SIFT descriptors (dashed black) and Hamming distance between binary vectors of different dimension $m = 32, 64, \ldots, 512$ constructed using DIF (dash-dot red) and kDIF (solid blue) hashing algorithms. Captions follow the convention *training-test*.



(a) trevi-trevi

(b) trevi-notredame

Figure 3: ROC curves showing the performance of Euclidean distance between SIFT descriptors (dashed black) and Hamming distance between binary vectors of different dimension $m = 32, 64, \ldots, 512$ constructed using DIF (dash-dot red) and kDIF (solid blue) hashing algorithms. Captions follow the convention *training-test*.

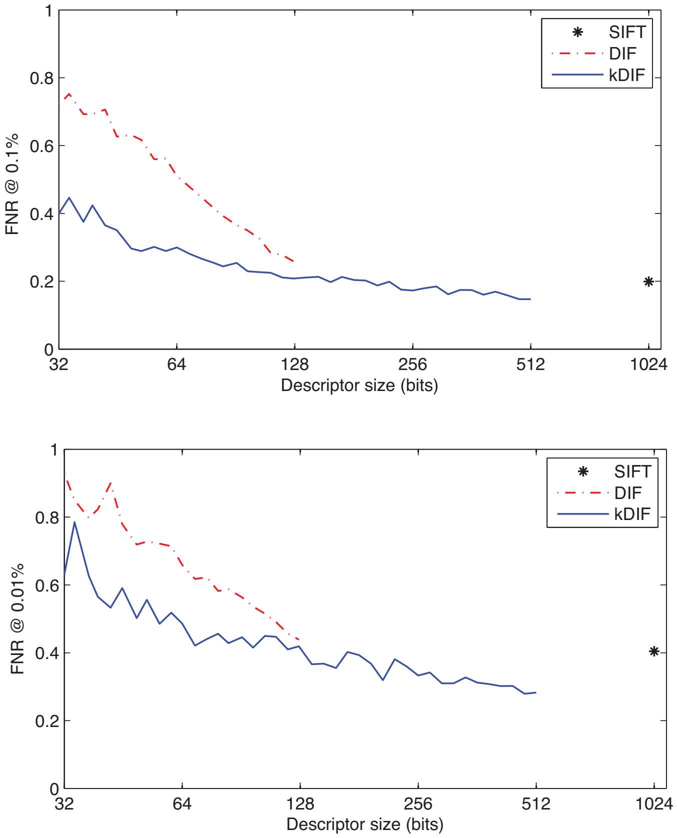


Figure 4: Performance (FNR at 0.1% and 0.01% FPR; the smaller the better) of different methods as function of descriptor size in bits. Training was done on trevi dataset; testing on notredame dataset.

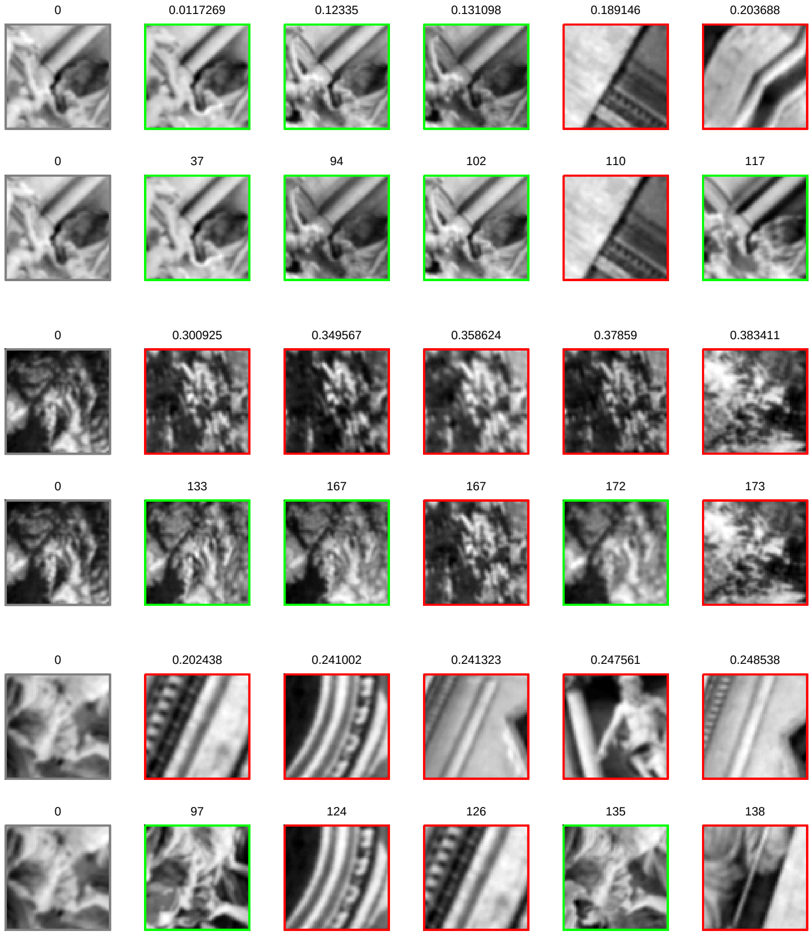


Figure 5: First matches using Euclidean distance between SIFT descriptors (odd rows) and Hamming distance between 512-dimensional binary vectors constructed using our kDIF hashing algorithms (even rows). Query image is shown on the left, first five matches are shown on the right. Numbers indicate the distance from query. Wrong matches are marked in red, correct matches are marked in green.