Boosting Binary Keypoint Descriptors*

Tomasz Trzcinski, Mario Christoudias, Pascal Fua and Vincent Lepetit
CVLab, EPFL, Lausanne, Switzerland
firstname.lastname@epfl.ch

Abstract

Binary keypoint descriptors provide an efficient alternative to their floating-point competitors as they enable faster processing while requiring less memory. In this paper, we propose a novel framework to learn an extremely compact binary descriptor we call BinBoost that is very robust to illumination and viewpoint changes. Each bit of our descriptor is computed with a boosted binary hash function, and we show how to efficiently optimize the different hash functions so that they complement each other, which is key to compactness and robustness. The hash functions rely on weak learners that are applied directly to the image patches, which frees us from any intermediate representation and lets us automatically learn the image gradient pooling configuration of the final descriptor. Our resulting descriptor significantly outperforms the state-of-the-art binary descriptors and performs similarly to the best floating-point descriptors at a fraction of the matching time and memory footprint.

1. Introduction

Local feature descriptors are ubiquitous in numerous computer vision applications, such as visual search, 3D reconstruction and panorama stitching. They seek a transformation of the input intensity patch that is invariant to unwanted artifacts such as illumination and viewpoint changes and typically involve a high-dimensional floating-point vector that encodes a robust representation of the patch [17, 2]. For increased invariance to local geometric transformations, most methods aggregate or *pool* the local evidence about pre-selected regions within the patch. The extent, location and shape of these regions defines the *pooling configuration* of the descriptor.

As image databases grow in size, modern solutions to local feature-based image indexing and matching must not only be accurate but also highly efficient to remain viable. Binary descriptors are of particular interest as they require far less storage capacity and offer much faster matching times than conventional floating point descriptors [9, 27, 4, 15, 22, 30], or even quantized descriptors [3]. In addition, they can be used directly in hash table techniques for efficient Nearest Neighbor search [20, 18], and their similarity can be computed very quickly on modern CPUs based on the Hamming distance.

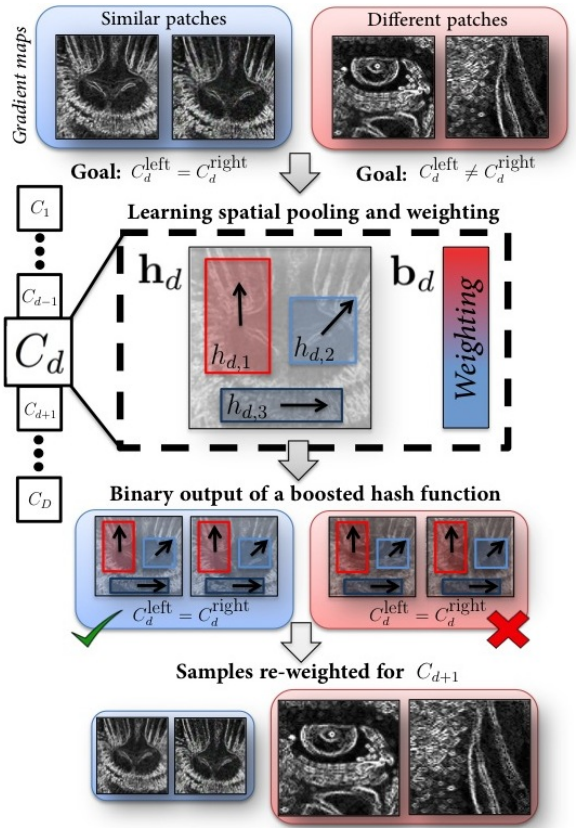


Figure 1. BinBoost learns a boosted hash function C_d for each descriptor bit, jointly optimized over *both* the feature weighting ($\mathbf{b}_d$) and pooling strategy ($\mathbf{h}_d$). The C_d's are iteratively optimized over labeled similar and dis-similar sample pairs of patches. At each iteration incorrectly hashed samples, like the pair of different patches on the right mistakenly assigned to the same C_d, are assigned a larger weight, while the weight of correctly hashed samples is reduced, like the pair on the left. Hence, the next bit tends to correct for the errors of the preceding ones.

However, as our experiments show, state-of-the-art bi-

*This work was supported in part by the Swiss National Science Foundation and the EU project MyCopter.

nary descriptors often perform worse than their floating-point competitors: some are built on top of existing representations such as SIFT or GIST by relying on training data [9, 27], and are limited by the performance of the intermediate representation. Others start from raw image intensity patches, but focus on computation speed and rely on fast-to-compute image features [4, 22, 15, 30], which limit their accuracy.

To address these shortcomings, we propose a novel supervised learning framework that finds a low-dimensional but highly discriminative binary descriptor. As shown in Fig. 1, for each dimension we learn a hash function of the same form as an AdaBoost strong classifier, that is the sign of a linear combination of non-linear weak learners. It is more general and powerful than those used in standard binary descriptors, which often rely on simple thresholded linear projections [30]. It also involves the design of a much more sophisticated objective function, which makes the optimization far more challenging. The resulting binary descriptor which we refer to as BinBoost[1] significantly outperforms its binary competitors. Furthermore, with as few as 64 bits it exhibits a comparable accuracy to state-of-the-art floating point or quantized descriptors at a fraction of the storage and matching cost. Nevertheless, it is more complex to optimize, and we show how to efficiently optimize our hash functions using boosting. As weak learners, we use gradient-based image features that are directly applied to the raw intensity image patches, which frees us from any intermediate representation and lets us automatically learn the image gradient pooling configuration of the final descriptor.

The rest of this paper is organized as follows. In Section 2 we discuss related work. In Section 3 we describe our method: we first show how we construct our set of weak learners and how we find the Hamming embedding minimizing the exponential loss function. We then explain how we use this approach to build our binary local feature descriptor and in Section 4 we compare it against the state of the art methods.

2. Related Work

Many recent techniques form binary descriptors based on simple pixel intensity comparisons [4, 15, 22]. Huffman coding [5] and product quantization [13] have also been explored to compress histogram of oriented gradient descriptors. Similarly, [37] develops a binary edge descriptor based on a histogram of normalized gradients. Although more efficient, these hand-designed descriptors are generally not compact and not as accurate as their floating point equivalents.

Machine learning has been applied to improve both the efficiency and accuracy of image descriptor matching. Unsupervised hashing methods learn compact binary descriptors whose Hamming distance is correlated with the similarity in the original input space [9, 14, 23, 36, 35]. Semantic hashing [23] trains a multi-layer neural network to learn representative, compact binary codes. Spectral hashing [36] minimizes the expected Hamming distance between similar training examples, and was recently extended to optimize over example affinities [35]. Similarly, [14, 19] find codes whose Hamming distances well approximate the original Euclidean ones. In [34, 9], iterative and sequential optimization strategies that find projections with minimal quantization error are explored. While these approaches have proven highly effective for finding compact binary codes, they rely on a pre-defined distance or similarity measure and in many cases are limited to the accuracy of the original input space.

Supervised learning approaches can learn feature spaces tailored to a specific task [12, 16, 27, 34]. They exploit labeled example pairs or triplets that encode the desired proximity relationships of the learned metric. A Mahalanobis distance metric is learned in [12] and optimized with respect to labeled distance constraints. Linear Discriminant Analysis is applied in [9, 27, 30] to learn discriminative feature embeddings. Semi-supervised sequential learning algorithms are proposed in [16, 34] for finding discriminative projections. Similar to these approaches, most methods define a linear transformation of the data in either the original or a kernelized feature space, and rely on a pre-specified kernel function to capture non-linearities. They are well-suited for image categorization and indexing tasks for which task-specific kernels have been proposed, *e.g.* [10], however, they are less applicable to local descriptor matching where the appropriate choice of kernel function is less well understood.

Recent descriptor learning methods have emphasized the importance of learning not only the optimal weighting, but also the optimal *shape* or pooling configuration of the underlying representation [3, 26, 29]. In [3], they optimize over different feature selection and pooling strategies of gradient-based features, however, the criterion considered—the area below the ROC—is not analytical making it difficult to optimize. Following [3], a convex optimization strategy was developed in [26]. To make learning tractable, however, a limited set of pooling configurations was considered and restricted to circular, symmetrically arranged pooling regions centered about the patch. As shown in our experiments, our binary descriptor achieves a similar accuracy to these methods at a fraction of the matching cost.

Jointly optimizing over descriptor weighting and shape poses a difficult problem due to the potentially large number of pooling configurations one might encounter. This is especially true for learning generic shapes where the number of pooling regions can easily be in the millions, even

[1] The reference implementation of BinBoost will be made available.

for small patch sizes. Fortunately, this is a problem for which AdaBoost [8] and other boosting methods [7, 32] are particularly well-suited. Although greedy, boosting is a provably effective method for constructing a highly accurate predictor from a large (potentially infinite) collection of constituent parts. The resulting *boosting-trick* like the kernel-trick, maps the input to a high-dimensional feature space, however, the mapping it defines is explicit, with the learned embedding assumed to be sparse [6, 21]. As a result and unlike kernel methods, boosting is an efficient way to find a non-linear transformation of the input that is naturally parameterized over both the descriptor shape and weighting.

The first application of boosting to learn an image similarity measure was Boosted Similarity Sensitive Coding (SSC) [25], which was later extended in [28] to be used with a Hamming distance. Boosted SSC only considers linear projections of the input, however, and generally results in fairly high dimensional descriptions. In [29], we proposed a descriptor we call Low-dimensional Boosted Gradient Map (L-BGM), whose similarity measure models the correlation between weak learners resulting in a compact description. We optimized over gradient-based features resulting in a learned representation that closely resembles the well-known SIFT. Although highly accurate, L-BGM computes a floating point descriptor and therefore its matching time is costly.

In this paper, we introduce a boosted binary descriptor that relies on the same image gradient-based features as [29]. Because it is binary, it is more difficult to optimize, but it is also much more efficient while being as accurate. We define a sequential learning method similar to [16, 34] except, unlike these methods, our boosting approach learns both the optimal shape and weighting of the features associated with each bit. Our descriptor can also be seen as a two layer neural network [23], since each coordinate of the descriptor is computed from a linear combination of pooled image features. As shown in our experiments, this results in a highly accurate and compact binary descriptor. Unlike hand-designed representations, we get similar performance to SIFT with as few as 8 bits, and do significantly better with increasing bit length, our final performance rivaling that of the leading binary and floating point descriptors.

3. The BinBoost Descriptor

In this section, we first describe our BinBoost descriptor and show how to train it efficiently. We then introduce the gradient-based features we use to define our weak learners.

3.1. Problem formulation

Given an image intensity patch $\mathbf{x}$, we look for a binary descriptor $C(\mathbf{x}) = [C_1(\mathbf{x}), \dots, C_D(\mathbf{x})]$ which maps the patch to a D-dimensional binary string. For convenience, we will consider that each bit $C_d(\mathbf{x})$ takes its value in $\{-1, +1\}$ instead of $\{0, 1\}$. We seek to compute each one as:

$$C_d(\mathbf{x}) = \text{sgn}\left(\mathbf{b}_d^T \mathbf{h}_d(\mathbf{x})\right) , \tag{1}$$

which is similar to what an Adaboost [8] classifier does, and where the $\mathbf{h}_d(\mathbf{x}) = [h_{d,1}(\mathbf{x}) \dots h_{d,K}(\mathbf{x})]^T$ are K weak learners weighted by the vector $\mathbf{b}_d = [b_{d,1} \dots b_{d,K}]^T$. Section 3.3 describes the weak learners we use in practice.

Our problem formulation is similar to [25] in the sense that [25] also learned a descriptor $C^{\text{SSC}}(\mathbf{x})$ by minimizing its exponential loss with Adaboost. Expression (1), however, is more complex than the one used in [25], which considered functions of the simpler form $C_d^{\text{SSC}}(\mathbf{x}) = b_d h_d(\mathbf{x})$, with b_d a scalar and h_d a single weak learner. It is also more general than the one used in most of the previous work on binary descriptors. It is therefore reasonable to expect that this expression will make our descriptors more compact, as is confirmed by our experiments. However, it also results in a more challenging optimization problem.

Let $\{(\mathbf{x}_n, \mathbf{y}_n, l_n)\}_{n=1}^N$ be a set of N labeled training pairs such that $l_n = +1$ if image patches $\mathbf{x}_n$ and $\mathbf{y}_n$ correspond to the same physical point, and $l_n = -1$ otherwise. We solve for the $\{\mathbf{b}_d, \mathbf{h}_d\}_{d=1}^D$ in the expression of $C(\cdot)$ by minimizing the exponential loss on the training data:

$$\mathcal{L} = \min_{\{\mathbf{b}_d, \mathbf{h}_d\}_{d=1}^D} \sum_{n=1}^N \exp\left(-\gamma\, l_n \sum_{d=1}^D c_d(\mathbf{x}_n, \mathbf{y}_n; \mathbf{b}_d, \mathbf{h}_d)\right) , \tag{2}$$

where

$$\begin{aligned} c_d(\mathbf{x}, \mathbf{y}; \mathbf{b}_d, \mathbf{h}_d) &= C_d(\mathbf{x}) C_d(\mathbf{y}) \\ &= \text{sgn}\left(\mathbf{b}_d^T \mathbf{h}_d(\mathbf{x})\right) \text{sgn}\left(\mathbf{b}_d^T \mathbf{h}_d(\mathbf{y})\right) , \end{aligned} \tag{3}$$

and γ is a parameter of our method—we explain below how we pick it in practice. Minimizing Eq. (2) aims at reducing the Hamming distances between descriptors of patches from positive pairs ($l_n = +1$) while increasing the Hamming distances between descriptors of patches from negative pairs ($l_n = -1$).

The optimization problem of Eq. (2) is closely related to the standard AdaBoost formulation [8], with two differences. First the c_d functions are not weighted, because for efficiency reasons we want to use the regular Hamming distances between descriptors instead of the weighted one. Second, and more importantly, the c_d functions are much more complex than the ones that are usually used, since they are a product of two strong classifiers. The resulting optimization is discontinuous and highly non-convex and in practice the space of all possible weak learners h is discrete and prohibitively large. In what follows we develop a greedy optimization algorithm for solving this difficult problem.

3.2. Greedy optimization

In this section we present a greedy algorithm for jointly optimizing over the weak classifiers of each bit, $\mathbf{h}_d$ and their associated weights $\mathbf{b}_d$. We first proceed as in regular AdaBoost. We optimize the $\{C_d\}$ functions iteratively, and at iteration d, the C_d function that minimizes Eq. (2) is also the one that maximizes the weighted correlation of its output and the data labels [24]. Using this fact, at iteration d, the optimal $\mathbf{b}_d$ and $\mathbf{h}_d$ can be taken as

$$\max_{\mathbf{b}_d,\mathbf{h}_d} \sum_{n=1}^{N} l_n \, W_d(n) c_d(\mathbf{x}_n, \mathbf{y}_n; \mathbf{b}_d, \mathbf{h}_d) \, , \tag{4}$$

where

$$W_d(n) = \exp\left(-\gamma l_n \sum_{d'=1}^{d-1} c_{d'}(\mathbf{x}_n, \mathbf{y}_n; \mathbf{b}_d, \mathbf{h}_d)\right) \tag{5}$$

is a weighting that is very similar to the one used in regular Adaboost. This means that pairs that are incorrectly classified by the previous iterations are assigned a higher weight, whereas the weight of those correctly classified is decreased.

The sign function in c_d is non-differentiable, and Eq. (4) is thus still hard to solve. We therefore apply the spectral relaxation trick [16, 34] and approximate the sign function using its signed magnitude, $\text{sgn}(x) \approx x$. This yields:

$$\begin{aligned}
&\max_{\mathbf{b}_d,\mathbf{h}_d} \sum_{n=1}^{N} l_n \, W_d(n) c_d(\mathbf{x}_n, \mathbf{y}_n; \mathbf{b}_d, \mathbf{h}_d) \\
\approx\ &\max_{\mathbf{b}_d,\mathbf{h}_d} \sum_{n=1}^{N} l_n \, W_d(n) \left(\mathbf{b}_d^T \mathbf{h}_d(\mathbf{x}_n)\right) \left(\mathbf{b}_d^T \mathbf{h}_d(\mathbf{y}_n)\right) \\
=\ &\max_{\mathbf{b}_d,\mathbf{h}_d} \sum_{n=1}^{N} l_n \, W_d(n) \left(\mathbf{b}_d^T \mathbf{h}_d(\mathbf{x}_n)\right) \left(\mathbf{h}_d(\mathbf{y}_n)^T \mathbf{b}_d\right) \\
=\ &\max_{\mathbf{b}_d,\mathbf{h}_d} \mathbf{b}_d^T \left(\sum_{n=1}^{N} l_n \, W_d(n) \mathbf{h}_d(\mathbf{x}_n) \mathbf{h}_d(\mathbf{y}_n)^T\right) \mathbf{b}_d \, .
\end{aligned} \tag{6}$$

We first select a vector $\mathbf{h}_d(\mathbf{x})$ of suitable weak classifiers using the algorithm of [25] on the training samples initially weighted by the $W_d(n)$ weights. The sign function in the expression of C_d makes $\mathbf{b}_d$ defined only up to a scale factor, and given an estimate for $\mathbf{h}_d(\mathbf{x})$, we solve for $\mathbf{b}_d$ by looking for

$$\max_{\mathbf{b}_d} \mathbf{b}_d^T \mathbf{M} \mathbf{b}_d, \text{ s.t. } \|\mathbf{b}_d\|_2 = 1 \tag{7}$$

where

$$\mathbf{M} = \sum_{n=1}^{N} l_n \, W_d(n) \mathbf{h}_d(\mathbf{x}_n) \mathbf{h}_d(\mathbf{y}_n)^T \, . \tag{8}$$

Eq. (7) defines a standard eigenvalue problem and the optimal weights $\mathbf{b}_d$ can therefore be found in closed-form as the eigenvector of $\mathbf{M}$ associated with its largest eigenvalue.

Although not globally optimal, this solution returns a useful approximation to the solution to Eq. (4). Moreover, thanks to our boosting scheme even a sub-optimal selection of C_d allows for an effective minimization.

We still have to explain how we choose the γ parameter. Note that its value is needed for the first time at the end of the first iteration, and we set this parameter after finding C_1 using the formula from regular Adaboost. We use the rule $\gamma = \nu \cdot \frac{1}{2} \log \frac{1+r_1}{1-r_1}$ where $r_1 = \sum_{n=1}^{N} W_1(n) \, l_n \, c_1(\mathbf{x}_n, \mathbf{y}_n)$ and ν is a shrinkage parameter used to regularize our optimization as described in [11]. In practice, we use $\nu = 0.4$.

3.3. Weak learners

In our implementation, we rely on weak learners that consider the orientations of intensity gradients over image regions [1, 29]. They are parameterized by a rectangular region R over the image patch $\mathbf{x}$, an orientation e, and a threshold T, and are defined as

$$h(\mathbf{x}; R, e, T) = \begin{cases} 1 & \text{if } \phi_{R,e}(\mathbf{x}) \leq T \\ -1 & \text{otherwise} \end{cases} \, , \tag{9}$$

with

$$\phi_{R,e}(\mathbf{x}) = \sum_{m \in R} \xi_e(\mathbf{x}, m) \, / \sum_{e' \in \Phi, m \in R} \xi_{e'}(\mathbf{x}, m) \, , \tag{10}$$

and

$$\xi_e(\mathbf{x}, m) = \max(0, \cos(e - o(\mathbf{x}, m)) \, , \tag{11}$$

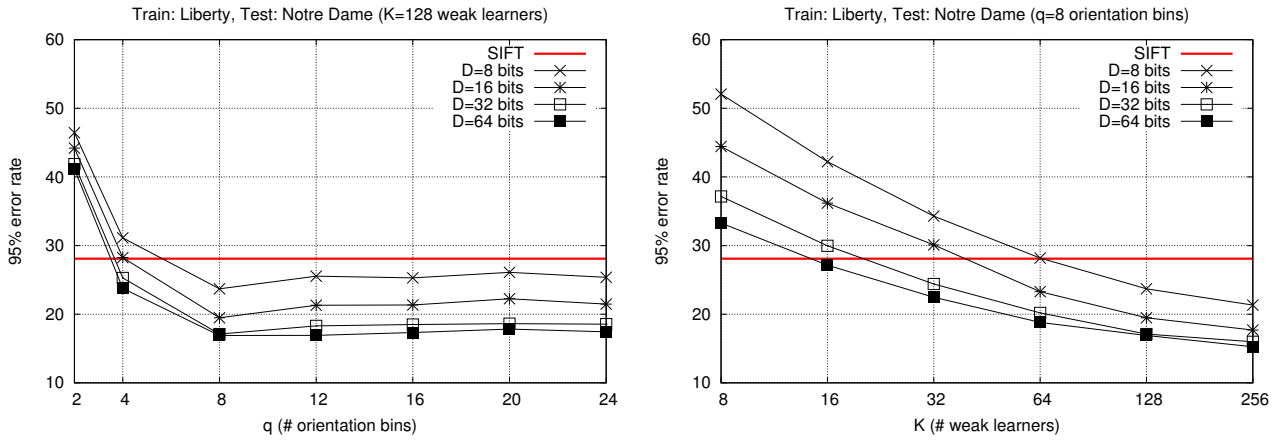
where $o(\mathbf{x}, m)$ is the orientation of the image gradient in $\mathbf{x}$ at location m. The orientation e is quantized to take values in $\Phi = \{0, \frac{2\pi}{q}, \frac{4\pi}{q}, \cdots, (q-1)\frac{2\pi}{q}\}$ with q the number of quantization bins. As noted in [1] this representation can be computed efficiently using integral images.

4. Results

In this section, we first describe our evaluation framework. We then present a set of initial experiments which validate our approach and allow us to select the correct parameters for our BinBoost descriptor. Finally, we compare BinBoost with the state-of-the-art binary and floating point descriptors.

4.1. Evaluation framework

We evaluate the performance of our methods using three publicly available datasets: Liberty, Notre Dame, and Yosemite [3]. Each of them contains over 400k scale- and rotation-normalized 64×64 patches. These patches are sampled around interest points detected using Difference of Gaussians and the correspondences between patches are found using a multi-view stereo algorithm. The resulting datasets exhibit substantial perspective distortion and changing lighting conditions. The ground truth available for each of these datasets describes 100k, 200k and 500k



(a) (b)

Figure 2. Influence of **(a)** the number of orientation bins q and **(b)** the number of weak learners K on the descriptor performance for dimensionalities $D = 8, 16, 32, 64$ bits. The performances are optimal with $q = 8$ orientation bins, which is also the number used in SIFT. Increasing the number of weak learners K from $K = 128$ to $K = 256$ provides only a minor improvement—at greatly increased computational cost—and, hence, we choose for our final descriptor $K = 128$.

pairs of patches, where 50% correspond to match pairs, and 50% to non-match pairs. In our experiments, we use subsampled patches of size 32×32 and the descriptors are trained on each of the 200k datasets and we use the held-out 100k dataset for testing. We report the results of the evaluation in terms of ROC curves and 95% error rate as in [3].

4.2. Initial experiments

Our boosting framework defines a generic optimization strategy that unlike many previous approaches, such as [3], does not require the fine tuning of multiple parameters. BinBoost has only three main parameters that provide a clear trade-off between the performance and complexity of the final descriptor: the number of orientation bins used by the weak learner, the number of weak learners, and the final dimensionality of the descriptor. We study below the influence of each of them on the performance of our descriptor.

Number of orientation bins q defines the granularity of the gradient-based weak learners. Fig. 2(a) shows the results obtained for different values of q and D. For most of the values for D, the performances are optimal for $q = 8$ as finer orientation quantization does not lead to any performance improvement and we keep $q = 8$ in the remaining experiments. Interestingly, this is also the number of orientation bins used in SIFT.

Number of weak learners K determines how many gradient-based features are evaluated per dimension and in Fig. 2(b) we show the 95% error rates for different values of K. Increasing the value of K results in increased computational cost and since the performances seem to saturate after $K = 128$, we keep this value for our final descriptor.

Dimensionality D is the number of bits of our final descriptor. Fig. 3 shows that with $D = 64$ bits, our descriptor

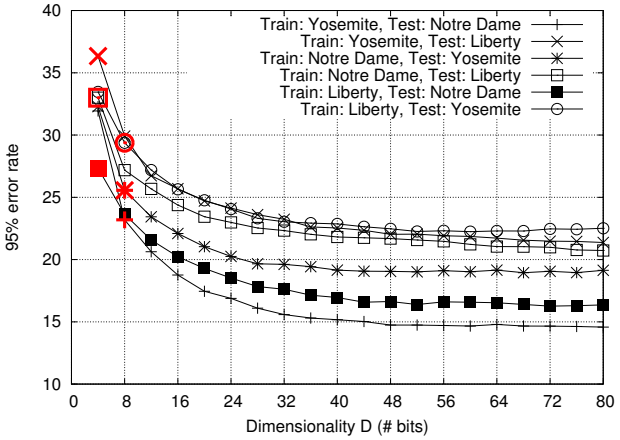


Figure 3. Performances for different dimensionalities D. With $D = 64$ bits, BinBoost reaches its optimal performance as increasing the dimensionality further does not seem to improve the results. In bold red we mark the dimensionality for which BinBoost outperforms SIFT, which is always less or equal to 8.

reaches its optimal performance as increasing the dimensionality further does not seem to improve the results.

Using the above-mentioned parameters for our compact BinBoost descriptor, we trained it using the Notre Dame dataset. To visualize the weighting and pooling scheme found with our approach, we show in Fig. 4 the weak learners and their weighted orientations chosen for computing the first 8 bits. The weak learners of similar orientations tend to cluster about different regions for each bit thus illustrating the complementary nature of the learned hash functions.

4.3. Comparison with the state of the art

In this section we compare our approach against SIFT [17], SURF [2], the binary LDAHash descriptor [27],

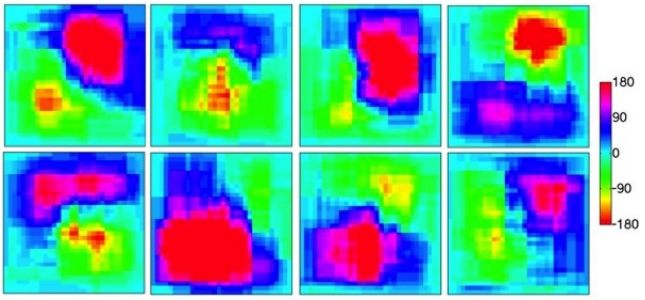


Figure 4. Visualization of the selected weak learners for the first 8 bits learned on 200k pairs of 32×32 patches from the Notre Dame dataset (best viewed on screen). For each pixel of the figure we show the average orientation weighted by the weights of the weak learners $\mathbf{b}_d$. For different bits, the weak learners cluster about different regions and orientations illustrating their complementary nature.

the binary BGM descriptor [29], Boosted SSC [25], L-BGM [29], the binary ITQ descriptor applied on SIFT descriptors [9], and the fast binary BRIEF [4] and BRISK [15] descriptors. Like our approach, Boosted SSC, BGM, and L-BGM are based on boosting. ITQ is based on rotations applied to an intermediate representation such as SIFT. BRIEF and BRISK are computed from simple image intensity comparisons.

For SIFT, we use the publicly available implementation of A. Vedaldi [31]. For SURF, LDAHash, BRIEF, BRISK, ITQ, BGM and L-BGM we use the implementation available from their authors. For the other methods, we use our own implementation or we report the results from the literature. For Boosted SSC, we use 128 dimensions as this obtained the best performance.

Fig. 7 shows the ROC curves for BinBoost and the state-of-the-art methods. Table 1 summarizes the 95% error rates. Both show that BinBoost significantly outperforms the baselines. It performs almost twice as well as SIFT in terms of 95% error rate, while requiring only 64 bits (8 bytes) instead of 128 bytes for SIFT. Moreover, since BinBoost can be efficiently implemented using integral images, the computation time of our descriptor is comparable with that of SIFT using Vedaldi's implementation—approximately 1ms per descriptor on a Macbook Pro with an Intel i7 2.66 GHz CPU. The performance improvement of BinBoost with respect to the recent binary descriptors, such as LDAHash or BRIEF, is even greater, BinBoost achieving a 95% error rate that is almost a factor of 3 lower than that obtained with these methods. More results are presented in the supplementary material.

Since the dimensionality of the other binary descriptors can be varied depending on the required performance quality, Fig. 5 compares the 95% error rates of these descriptors for different numbers of bits used. BinBoost clearly outperforms them across all dimensions at the lower end of the spectrum. However, the biggest improvement can be seen

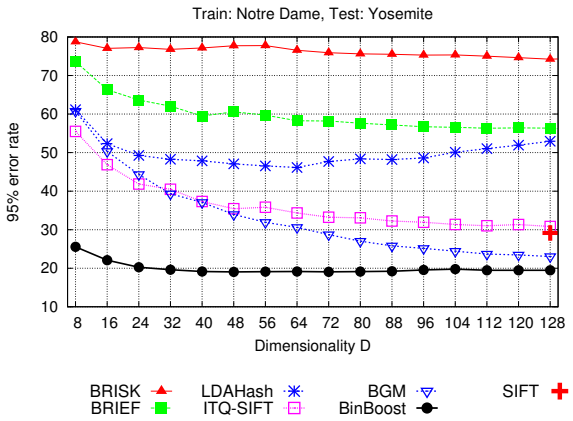


Figure 5. 95% error rates for binary descriptors of different dimensionality. For reference, we plot the results obtained with SIFT. BinBoost outperforms the state-of-the-art binary descriptors and the improvement is especially visible for lower dimensionality.

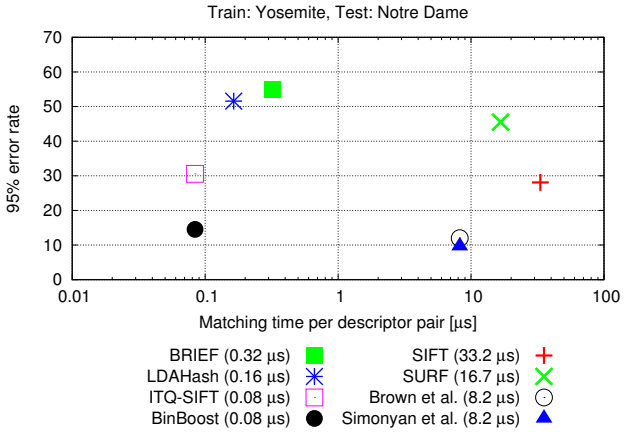


Figure 6. Descriptor performances as a function of their matching times. The reported times were computed for 100k pairs from a test dataset (*i.e.* 100k distance computations were performed) on a Macbook Pro with an Intel i7 2.66 GHz CPU (with the `POPCOUNT` instruction enabled) and averaged over 100 runs. To make the comparison fair, we optimized the matching strategy for floating-point descriptors by representing them with unsigned characters. The advantage of binary descriptors, out of which BinBoost performs the best in terms of 95% error rate, is clear.

for lower dimensionality. In fact, with as few as 16 bits BinBoost performs as well as the next best descriptor, BGM, which is 128 bits long.

Moreover, our BinBoost descriptor remains competitive to the best descriptors of [3] and [26], even though the memory footprint of their descriptors is almost 4 times greater. The real advantage of BinBoost, however, is its binary nature which allows for extremely fast similarity computation using the Hamming distance[2], whereas the descriptors of [3] and [26] are floating-point and cannot benefit from the same optimization, even when quantized very coarsely.

[2]On modern CPUs this can be implemented as a bitwise XOR operation on the descriptors followed by a `POPCOUNT` instruction which counts the number of bits set to 1.

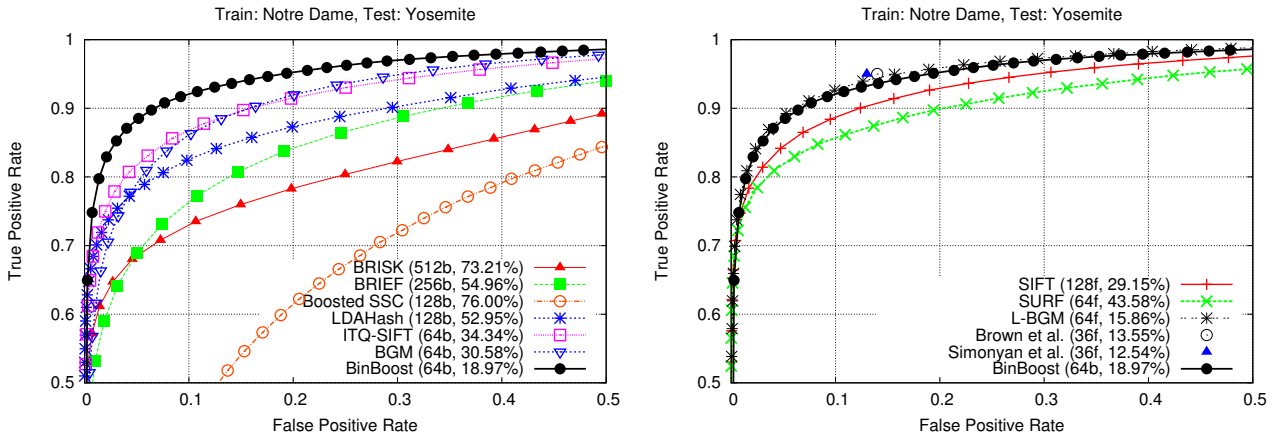


Figure 7. Comparison of our BinBoost descriptor to the state-of-the-art binary (**left**) and floating-point (**right**) descriptors. In parentheses: the number of floating-point (f) or binary (b) dimensions and the 95% error rate. Our BinBoost descriptor significantly outperforms its binary competitors for all false positive rates. It also outperforms SIFT and provides similar performances to the recent floating-point descriptors, even though it is much faster to match and has a lower memory footprint. More results are shown in the supplementary material.

		Binary						**Floating-point**				
Train	**Test**	BinBoost *8 bytes*	BGM [29] *8 bytes*	ITQ-SIFT [9] *8 bytes*	LDAHash [27] *16 bytes*	BRIEF *32 bytes*	BRISK *64 bytes*	SURF *64 bytes*	SIFT *128 bytes*	L-BGM [29] *64 bytes*	Brown [3] *29 bytes*	Simonyan [26] *29 bytes*
Yosemite	Notre Dame	**14.54**	26.80	30.56	51.58	54.57	74.88	45.51	28.09	13.73	11.98	9.67
Liberty		**16.90**	29.60	31.07						14.15	-	-
Yosemite	Liberty	**21.67**	33.54	37.31	49.66	59.15	79.36	54.01	36.27	21.03	18.27	17.44
Notre Dame		**20.49**	31.90	36.95						18.05	16.85	14.51
Notre Dame	Yosemite	**18.97**	30.58	34.34	52.95	54.96	73.21	43.58	29.15	15.86	13.55	12.54
Liberty		**22.88**	38.13	34.43						19.63	-	-

Table 1. 95% error rates for different training and testing configurations and the corresponding results for BinBoost with 64 and 8 bits and its competitors. For the descriptors that do not depend on the training data, we write one result per testing dataset, for others we give the results for two different training datasets. Below the descriptor names we write the number of bytes used to encode them. For the floating point descriptors (SIFT, SURF, L-BGM [29], Brown *et al*. [3], Simonyan *et al*. [26]) we assume 1 byte per dimension, as this quantization was reported as sufficient for SIFT [31]. BinBoost significantly outperforms its binary competitors, while requiring less memory. For reference, we also give the results of the floating-point descriptors: BinBoost performs similarly to the best floating-point descriptors even though it is shorter and binary which enables a significant speedup in processing time (See Fig. 6).

As presented in Fig. 6, this results in a speedup of over 2 orders of magnitude in terms of similarity search.

To verify the performance of our descriptor, we also compare it to several binarization techniques applied on the recently proposed floating-point L-BGM descriptor that outperforms SIFT on the Liberty, Notre Dame and Yosemite datasets. Results are displayed in Fig. 8. Binarizing the L-BGM coordinates by thresholding them at an optimal threshold found as in [27] results in large binarization errors significantly decreasing the accuracy of the resulting binary representation. This error can be reduced using Iterative Quantization [9], however, the orthogonality constraints used in this approach largely limit the extent to which it can be minimized. In contrast, sequential projection learning (S3PLH) [34] can find non-orthogonal projections that more faithfully mitigate binarization error, however, it requires a fairly large number of bits to recover L-BGM's original performance. Unlike these methods, by effectively combining multiple weak learners within each hash function, our algorithm results in a more accurate predictor with far fewer bits.

5. Conclusion

In this paper we presented an efficient framework to train highly discriminative binary local feature descriptors. Leveraging the boosting-trick, we simultaneously optimize both the descriptor weighting and pooling strategy. The proposed sequential learning scheme finds a single boosted hash function per dimension as a linear combination of non-linear gradient-based weak learners. Since we train our descriptor from intensity patches, our final binary descriptor does not rely on any pre-computed representation, and it outperforms the state of the art with only 64 bits per descriptor. The generalization of our approach to different evaluation conditions and application domains, including medical and underwater imaging, are important problems that we plan to address as part of future research.

References

[1] K. Ali, F. Fleuret, D. Hasler, and P. Fua. A Real-Time Deformable Detector. *PAMI*, 34(2):225–239, 2012. 4

[2] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. In *ECCV'06*. 1, 5

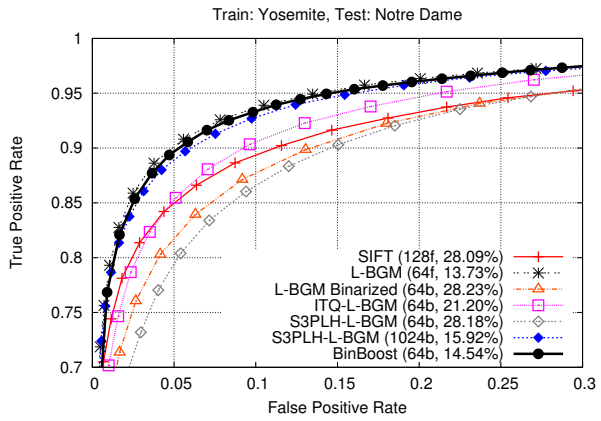


Figure 8. Our BinBoost descriptor's performance compared with different binarization methods applied on L-BGM. Binarizing the discriminative projections found with L-BGM either by simple thresholding or with Iterative Quantization (ITQ) results in large binarization errors significantly reducing its accuracy. On the other hand, the sequential projection learning of S3PLH requires a fairly large number of bits to recover L-BGM's original performance. In contrast, by jointly optimizing over the feature weighting and pooling strategy of each bit, our BinBoost approach results in a highly compact and accurate binary descriptor whose performance is similar with L-BGM but at a fraction of the storage cost.

[3] M. Brown, G. Hua, and S. Winder. Discriminative Learning of Local Image Descriptors. *PAMI*, 2011. 1, 2, 4, 5, 6, 7

[4] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua. BRIEF: Computing a Local Binary Descriptor Very Fast. *PAMI*, 34(7):1281–1298, 2012. 1, 2, 6

[5] V. Chandrasekhar, G. Takacs, D. Chen, S. Tsai, R. Grzeszczuk, and B. Girod. CHoG: Compressed Histogram of Gradients a Low Bit-Rate Feature Descriptor. In *CVPR'09*. 2

[6] O. Chapelle, P. Shivaswamy, S. Vadrevu, K. Weinberger, Y. Zhang, and B. Tseng. Boosted Multi-Task Learning. *Machine Learning*, 2010. 3

[7] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral Channel Features. In *BMVC'09*. 3

[8] Y. Freund and R. Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. In *European Conference on Computational Learning Theory*, 1995. 3

[9] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative Quantization: A Procrustean Approach to Learning Binary Codes for Large-Scale Image Retrieval. *PAMI*, 2012. 1, 2, 6, 7

[10] K. Grauman and T. Darrell. The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features. In *ICCV'05*. 2

[11] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. 2001. 4

[12] P. Jain, B. Kulis, J. Davis, and I. Dhillon. Metric and Kernel Learning Using a Linear Transformation. *JMLR*, 2012. 2

[13] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating Local Descriptors into a Compact Image Representation. In *CVPR'10*. 2

[14] B. Kulis and T. Darrell. Learning to Hash with Binary Reconstructive Embeddings. In *NIPS'09*. 2

[15] S. Leutenegger, M. Chli, and R. Siegwart. BRISK: Binary Robust Invariant Scalable Keypoints. In *ICCV'11*. 1, 2, 6

[16] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised Hashing with Kernels. In *CVPR'12*. 2, 3, 4

[17] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 20(2):91–110, 2004. 1, 5

[18] M. Malekesmaeili, R. Ward, and M. Fatourechi. A Fast Approximate Nearest Neighbor Search Algorithm in the Hamming Space. *PAMI*, 2012. 1

[19] M. Norouzi and D. Fleet. Minimal Loss Hashing for Compact Binary Codes. In *ICML'11*. 2

[20] M. Norouzi, A. Punjani, and D. Fleet. Fast Search in Hamming Space with Multi-Index Hashing. In *CVPR'12*. 1

[21] S. Rosset, J. Zhu, and T. Hastie. Boosting as a Regularized Path to a Maximum Margin Classifier. *JMLR*, 2004. 3

[22] E. Rublee, V. Rabaud, K. Konolidge, and G. Bradski. ORB: An Efficient Alternative to SIFT or SURF. In *ICCV'11*. 1, 2

[23] R. Salakhutdinov and G. Hinton. Semantic Hashing. *International Journal of Approximate Reasoning*, 2009. 2, 3

[24] R. E. Schapire and Y. Singer. Improved Boosting Algorithms Using Confidence Rated Predictions. *Machine Learning*, 1999. 4

[25] G. Shakhnarovich. *Learning Task-Specific Similarity*. PhD thesis, 2006. 3, 4, 6

[26] K. Simonyan, A. Vedaldi, and A. Zisserman. Descriptor Learning Using Convex Optimisation. In *ECCV'12*. 2, 6, 7

[27] C. Strecha, A. Bronstein, M. Bronstein, and P. Fua. LDAHash: Improved Matching with Smaller Descriptors. *PAMI*, 34(1), 2012. 1, 2, 5, 7

[28] A. Torralba, R. Fergus, and Y. Weiss. Small Codes and Large Databases for Recognition. In *CVPR'08*. 3

[29] T. Trzcinski, M. Christoudias, V. Lepetit, and P. Fua. Learning Image Descriptors with the Boosting-Trick. In *NIPS'12*. 2, 3, 4, 6, 7

[30] T. Trzcinski and V. Lepetit. Efficient Discriminative Projections for Compact Binary Descriptors. In *ECCV'12*. 1, 2

[31] A. Vedaldi. `http://www.vlfeat.org/~vedaldi/code/siftpp.html`. 6, 7

[32] P. Viola and M. Jones. Rapid Object Detection Using a Boosted Cascade of Simple Features. In *CVPR'01*. 3

[33] G. Wang and Q. Wu. Quasi-Perspective Projection Model: Theory and Application to Structure and Motion Factorization from Uncalibrated Image Sequences. *IJCV*, 87(3):213–234, 2010.

[34] J. Wang, S. Kumar, and S.-F. Chang. Sequential Projection Learning for Hashing with Compact Codes. In *ICML'10*. 2, 3, 4, 7

[35] Y. Weiss, R. Fergus, and A. Torralba. Multidimensional Spectral Hashing. In *ECCV'12*. 2

[36] Y. Weiss, A. Torralba, and R. Fergus. Spectral Hashing. *NIPS*, 21:1753–1760, 2009. 2

[37] C. Zitnick. Binary Coherent Edge Descriptors. In *ECCV'10*. 2