Learning Image Descriptors with Boosting

Tomasz Trzcinski, Mario Christoudias, and Vincent Lepetit

Abstract—We propose a novel and general framework to learn compact but highly discriminative floating-point and binary local feature descriptors. By leveraging the *boosting-trick* we first show how to efficiently train a compact floating-point descriptor that is very robust to illumination and viewpoint changes. We then present the main contribution of this paper — a binary extension of the framework that demonstrates the real advantage of our approach and allows us to compress the descriptor even further. Each bit of the resulting binary descriptor, which we call BinBoost, is computed with a boosted binary hash function, and we show how to efficiently optimize the hash functions so that they are complementary, which is key to compactness and robustness. As we do not put any constraints on the weak learner configuration underlying each hash function, our general framework allows us to optimize the sampling patterns of recently proposed hand-crafted descriptors and significantly improve their performance. Moreover, our boosting scheme can easily adapt to new applications and generalize to other types of image data, such as faces, while providing state-of-the-art results at a fraction of the matching time and memory footprint.

Index Terms—Learning feature descriptors, binary embedding, boosting.

1 Introduction

Representing salient image patches in a way that is invariant to illumination and viewpoint changes remains a significant challenge in Computer Vision, as it lies at the core of many popular applications including visual search, 3D reconstruction and panorama stitching. To model the non-linear nature of these unwanted transformations, well-known local feature descriptors, such as SIFT [1] or SURF [2], typically apply a set of hand-crafted filters and aggregate or *pool* their responses within pre-defined regions of the image patch. The extent, location and shape of these regions defines the *pooling configuration* of the descriptor and recent work shows that optimizing this configuration can result in fairly large performance improvements [3], [4], [5], [6]. Although significant progress has been made, these approaches, however, are either built on top of hand-crafted representations [3], [5] or still require significant parameter tuning, as in [4] that relies on a non-analytical objective that is difficult to optimize.

Learning an invariant feature representation can be seen as finding an appropriate similarity measure which remains invariant to unwanted image transformations. Although several learning methods have been proposed in the literature [3], [7], [8], they have largely focused on finding a linear feature mapping in either the original input or a kernelized feature space. As a result, modeling non-linearities requires choosing an appropriate kernel function that maps the input features to a high-dimensional feature space where the transformations are assumed to be linear. However, selecting the right kernel, which is a crucial element of the algorithm, is often non-intuitive and generally constitutes a complex and challenging problem.

In this paper, we propose a novel supervised learning framework that finds low-dimensional but highly discriminative descriptors. With our approach, image patch appearance is modeled using local non-linear filters that are selected with boosting. We build upon [3] that also relies on boosting to compute a descriptor, and show how we can use it as a way to efficiently select features, from which we compute a compact representation. Analogous to the kernel-trick, our approach can be seen as applying a *boosting-trick* [9] to obtain a non-linear mapping of the input to a high-dimensional feature space. Unlike kernel methods, boosting allows for the definition of intuitive non-linear feature mappings that can share a close connection with existing, prevalent keypoint descriptors. Our learning approach is not limited to any pre-defined sampling pattern and provides a more general framework than previous training-based methods [4], [6], [10]. It also scales linearly with the number of training examples, making it more amenable to large scale problems, and results in highly accurate descriptor matching.

Nevertheless, 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 [11], [5], [12], [13], [10], [14], or even quantized descriptors [4]. In addition, they can be used directly in hash table techniques for efficient Nearest Neighbor search [15], [16], and their similarity can be computed very quickly on modern CPUs based on the Hamming distance.

However, as our experiments show, state-of-the-art binary descriptors often perform worse than their floating-point competitors: some are built on top of existing representa-

- *Tomasz Trzcinski and Mario Christoudias are with the Computer Vision Laboratory, I&C Faculty, Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne CH-1015, Switzerland. E-mail: firstname.lastname@epfl.ch*
- *Vincent Lepetit is with the Institute for Computer Graphics and Vision, Graz University of Technology, Graz, Austria. E-mail: lepetit@icg.tugraz.at*

tions such as SIFT or GIST by relying on training data [11], [5], and are therefore 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 [12], [10], [13], [14], which limit their accuracy.

To address these shortcomings, we extend our learning framework to the binary case and we train a highly discriminative yet compact binary descriptor. This extension demonstrates the real advantage of our approach as it enables us to not only compress the descriptor, but also significantly decrease the processing cost and memory footprint. For each dimension of the resulting binary representation, 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. The resulting binary descriptor, which we refer to as BinBoost, significantly outperforms its binary competitors and exhibits a similar accuracy to state-of-the-art floating-point or quantized descriptors at a fraction of the storage and matching cost. Furthermore it is more complex to optimize, and we show how to efficiently optimize our hash functions using boosting.

This paper extends our previous work [17], [18] in the following way. First, we show in Section 3 that our method provides a general descriptor learning framework that encompasses previously published approaches and the state-of-the-art intensity-based [12], [10], [13], [14] and gradient-based descriptors [1], [2], [5]. Our results show that the ability to effectively optimize over the descriptor filter configuration leads to a significant performance boost at no additional computational cost compared with the original hand-designed representation. We experiment with additional weak learner families from the ones previously used and we show that our learning method performs well independently of the underlying weak learner type. Finally, we provide an exhaustive experimental evaluation of our methods on several challenging datasets, including the Mikolajczyk dataset [19] and UKBench [20]. We also illustrate how our method is not restricted to local feature descriptors and can be successfully extended to new application domains and other types of image data, and we demonstrate this on a face recognition problem.

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 to efficiently construct our set of weak learners, from which we compute a compact floating-point representation. We then explain how to extend this approach to build a binary local feature descriptor. In Section 4 we discuss different weak learner types and in Section 5 we describe the experimental setup of our method and its parameters. Section 6 presents the comparison of our descriptors against the state-of-the-art methods. Finally, in Section 7, we show how our framework can be used to learn representations of other types of data, namely face images.

2 Related Work

Many recent techniques form binary descriptors based on simple pixel intensity comparisons [12], [13], [10]. Huffman coding [21] and product quantization [22] have also been explored to compress histogram of oriented gradient descriptors. Similarly, [23] 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.

For this reason, 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 [11], [24], [25], [26], [27]. Semantic hashing [25] trains a multi-layer neural network to learn compact representative binary codes. Spectral hashing [26] minimizes the expected Hamming distance between similar training examples, and was recently extended to optimize over example affinities [27]. Similarly, [24], [28] find codes whose Hamming distances well approximate the original Euclidean ones. In [29], [11], 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 pre-defined distance or similarity measures and in many cases are limited to the accuracy of the original input space.

Supervised learning approaches can learn feature spaces tailored to specific tasks [8], [30], [5], [29]. They exploit labeled example pairs or triplets that encode the desired proximity relationships of the learned metric. In [8], a Mahalanobis distance metric is learned and optimized with respect to labeled distance constraints. Linear Discriminant Analysis is applied in [11], [5], [14] to learn discriminative feature embeddings. Semi-supervised sequential learning algorithms are proposed in [30], [29] 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. While they are well-suited for image categorization and indexing tasks for which task-specific kernels have been proposed, such as in [31], 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 [4], [6]. In [4], 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 [4], a convex optimization strategy was developed in [6]. 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 for small patch sizes. Fortunately, this is a problem for which AdaBoost [32] and other boosting methods [33], [34] are particularly well-suited. Although greedy, boosting is an 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 [9], [35]. As a result and unlike kernel methods, boosting appears to be an efficient way to find a non-linear transformation of the input that is naturally parameterized over both the descriptor shape and weighting.

In this paper, we introduce a family of boosted descriptors that are trained with boosting for discriminative power and compactness. Our work is inspired by Boosted Similarity Sensitive Coding (SSC) [3] which is the first application of boosting to learn an image similarity measure and was later extended in [36] to be used with a Hamming distance. Boosted SSC, however, only considers linear projections of the input and generally results in fairly high dimensional descriptions. Our methods, on the other hand, rely on more complex weak learners and produce descriptors, both binary and floating-point, of a much lower dimensionality.

We also propose a sequential learning method similar to [29], [30] 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 [25], since each coordinate of the descriptor is computed from a linear combination of pooled image features. As shown in our experiments, this results in highly accurate and compact descriptors whose final performance rivals that of the leading binary and floating point descriptors.

3 METHOD

In this section we describe methods for learning local feature descriptors with boosting. We first formulate our problem by defining the exponential loss objective function we use to learn a similarity embedding between image patches. We then present different similarity measures which, when plugged into our boosting framework, can be used to train floating-point and binary descriptors.

3.1 Problem formulation

Given an image intensity patch $\mathbf{x}$, we look for a descriptor $C(\mathbf{x}) = [C_1(\mathbf{x}), \ldots, C_D(\mathbf{x})]$ which maps the patch to a D-dimensional vector. This descriptor can be learned by minimizing the exponential loss with respect to a desired similarity function $f\left(C(\mathbf{x}), C(\mathbf{y})\right) = f_C(\mathbf{x}, \mathbf{y})$ defined over image patch pairs:

$$\mathcal{L} = \sum_{i=1}^{N} \exp(-l_i f_C(\mathbf{x}_i, \mathbf{y}_i)) \tag{1}$$

where $\mathbf{x}_i, \mathbf{y}_i \in \mathcal{R}^p$ are training intensity patches and $l_i \in \{-1, 1\}$ is a label indicating whether it is a similar $(+1)$ or dissimilar (-1) pair. Minimizing Equation (1) finds an embedding which maximizes the similarity between pairs of similar patches, while minimizing it for pairs of different patches.

This formulation allows for numerous similarity functions f_C. We consider similarity functions of the form

$$f_C(\mathbf{x}, \mathbf{y}) = C(\mathbf{x})^T \, \mathbf{A} \, C(\mathbf{y}) \tag{2}$$

where $\mathbf{A} \in \mathcal{R}^{D \times D}$ is a symmetric matrix. This defines a general class of symmetric similarity measures that can be factorized to compute a feature descriptor independently over each input and used to define a wide variety of image descriptors. In what follows, we consider different choices of $\mathbf{A}$ and $C(\cdot)$ ordering them in increasing complexity.

3.2 Boosted Similarity Sensitive Coding (SSC)

The Boosted SSC method proposed in [3] considers a similarity function defined by a simply weighted sum of thresholded response functions $\{h_d(\cdot)\}_{d=1}^{D}$:

$$f_{SSC}(\mathbf{x}, \mathbf{y}) = \sum_{d=1}^{D} \alpha_d h_d(\mathbf{x}) h_d(\mathbf{y}) \, . \tag{3}$$

This function is the weighted Hamming distance between $\mathbf{x}$ and $\mathbf{y}$ and corresponds to Equation (2) where $\mathbf{A}$ is restricted to be a diagonal matrix. The importance of each dimension d given by the α_d's and the resulting D-dimensional descriptor is a floating-point vector $C(\mathbf{x}) = [\sqrt{\alpha_d} h_d(\mathbf{x})]_{d=1}^{D}$, where α is constrained to be positive.

Substituting f_{SSC} for f_C in Equation (1) gives

$$\mathcal{L}_{SSC} = \sum_{i=1}^{N} \exp\left(-l_i \sum_{d=1}^{D} \alpha_d h_d(\mathbf{x}_i) h_d(\mathbf{y}_i) \right) \, . \tag{4}$$

In practice the space of h's is prohibitively large, possibly infinite, making the explicit optimization of $\mathcal{L}_{SSC}$ difficult, however, this constitutes a problem for which boosting is particularly well suited [32]. Although boosting is a greedy optimization scheme, it is an effective method for constructing a highly accurate predictor from a collection of weak predictors h. Due to its greedy nature, however, the weak learners found using Boosted SSC often remain highly redundant and hence inefficient. In what follows, we modify the similarity function $f_C(\mathbf{x}, \mathbf{y})$ so that it becomes better suited for learning low-dimensional, discriminative embeddings with boosting.

3.3 FPBoost

To mitigate the potentially redundant embeddings found by boosting we propose an alternative similarity measure f_{FP} that models the correlation between weak response functions:

$$f_{FP}(\mathbf{x},\mathbf{y}) = \sum_{k,k'} \alpha_{k,k'} h_k(\mathbf{x}) h_{k'}(\mathbf{y}) = \mathbf{h}(\mathbf{x})^T \mathbf{A} \mathbf{h}(\mathbf{y}), \quad (5)$$

where $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), \cdots, h_K(\mathbf{x})]$ and $\mathbf{A}$ is a $K \times K$ matrix of coefficients $\alpha_{k,k'}$. This similarity measure is a generalization of Equation (3). In particular, f_{FP} is equivalent to the Boosted SSC similarity measure in the restricted case of a diagonal $\mathbf{A}$.

Substituting the above expression into Equation (1) gives

$$\mathcal{L}_{FP} = \sum_{i=1}^{N} \exp\left(-l_i \sum_{k,k'} \alpha_{k,k'} h_k(\mathbf{x}_i) h_{k'}(\mathbf{y}_i) \right). \quad (6)$$

We optimize $\mathcal{L}_{FP}$ using a two step learning strategy. We first apply AdaBoost to find good weak learners $\{h_k\}_{k=1}^K$ by minimizing Equation (4) on the training samples as in [3]. Then we apply stochastic gradient descent to find an optimal weighting over the selected features that minimizes Equation (6). To guarantee that the similarity function f_{FP} remains symmetric, we restrict the coefficients $\alpha_{k,k'}$ of $\mathbf{A}$ to be symmetric. This optimization strategy is sub-optimal but we found it to work well in practice.

The similarity function of Equation (5) defines an implicit feature mapping over example pairs. In order to compute the feature descriptors independently over each input, we need to factorize $\mathbf{A}$. As we constrain $\mathbf{A}$ to be symmetric, we can factorize it into the following form:

$$\mathbf{A} = \mathbf{B}\mathbf{W}\mathbf{B}^T = \sum_{k=1}^{K} w_k \mathbf{b}_k \mathbf{b}_k^T \quad (7)$$

where $\mathbf{W} = \text{diag}([w_1, \cdots, w_K])$, $w_k \in \{-1, 1\}$, $\mathbf{B} = [\mathbf{b}_1, \cdots, \mathbf{b}_k]$, $\mathbf{b} \in \mathcal{R}^K$.

Equation (5) can then be re-expressed as

$$f_{FP}(\mathbf{x},\mathbf{y}) = \sum_{d=1}^{D} w_d \left(\sum_{k=1}^{K} b_{d,k} h_k(\mathbf{x}) \right) \left(\sum_{k=1}^{K} b_{d,k} h_k(\mathbf{y}) \right). \quad (8)$$

For $D < K$ (i.e., the effective rank of $\mathbf{A}$ is $D < K$) the factorization represents a smoothed version of $\mathbf{A}$ discarding the low-energy dimensions that typically correlate with noise, and in practice leading to further performance improvements. The factorization of Equation (8) defines a signed inner product between the embedded feature vectors and provides increased efficiency with respect to the original similarity measure[1]. Moreover, it is easy to show that the signed inner product is equivalent to the Euclidean distance under the assumption that the descriptors have comparable magnitudes. As shown in Fig. 1, this is the case in practice and, hence, we can leverage the existing methods for fast approximate nearest neighbor search which rely on Euclidean distances.

[1]Matching two sets of descriptors each of size N is $\mathcal{O}(N^2K^2)$ under the original measure and $\mathcal{O}(NKD + N^2D)$ provided the factorization, resulting in significant savings for reasonably sized N and K, and $D \ll K$.

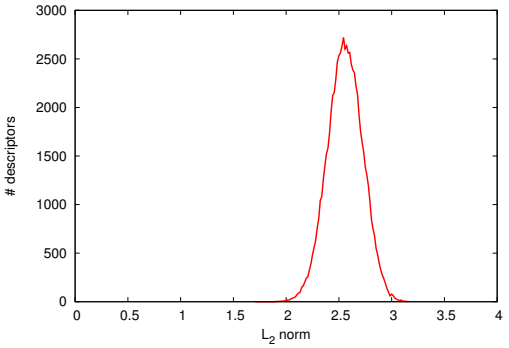


Fig. 1. Histogram of the L_2 norms of 75k FPBoost descriptors extracted from the images of Mikolajczyk dataset [19]. The L_2 norms are upper-bounded by $\sqrt{\sum_{d=1}^{D} |\mathbf{b}_d|_1^2}$ which equals 4 in this case. A significant portion of the descriptors have a comparable magnitude and, hence, we can use Euclidean distance in place of the equivalent inner product to measure descriptor similarity.

The final embedding $C(\mathbf{x}) = \mathbf{B}^T\mathbf{h}(\mathbf{x})$ results in a D-dimensional floating-point descriptor based on K weak learners that we call FPBoost$_K$-D. The projection matrix $\mathbf{B}$ defines a discriminative dimensionality reduction optimized with respect to the exponential loss objective of Equation (6). As seen in our experiments, in the case of redundant weak learners this results in a considerable feature compression, and therefore offering a more compact description than the original input patch. Although compact and highly discriminant, the descriptors learned using FPboost are real-valued and, hence, can be slow to match and costly to store. Next, we consider a modification to FPboost that as we show can be used to learn a highly accurate and efficient binary descriptor.

3.4 BinBoost

To learn a binary descriptor we propose a modified similarity measure that extends f_{FP} to operate within a D-dimensional Hamming space:

$$\begin{aligned} f_B(\mathbf{x},\mathbf{y}) &= \sum_{d=1}^{D} \text{sgn}\left(\mathbf{b}_d^T \mathbf{h}_d(\mathbf{x})\right) \text{sgn}\left(\mathbf{b}_d^T \mathbf{h}_d(\mathbf{y})\right) \\ &= \sum_{d=1}^{D} C_d(\mathbf{x}) C_d(\mathbf{y}) \end{aligned} \quad (9)$$

where $C_d(\mathbf{x}) = \text{sgn}\left(\mathbf{b}_d^T \mathbf{h}_d(\mathbf{x})\right)$ and $\mathbf{h}_d(\mathbf{x}) = [h_{d,1}(\mathbf{x}) \dots h_{d,K}(\mathbf{x})]^T$ are K weak learners weighted by the vector $\mathbf{b}_d = [b_{d,1} \dots b_{d,K}]^T$. The resulting binary descriptor, which we call BinBoost$_K$-D, is a D-dimensional binary vector built using K weak learners by applying $C(\mathbf{x}) = \{C_d(\mathbf{x})\}_{d=1}^D$.

Substituting f_B for f_C in Equation (1) gives

$$\mathcal{L}_B = \sum_{n=1}^{N} \exp\left(-\gamma\, l_n \sum_{d=1}^{D} C_d(\mathbf{x}) C_d(\mathbf{y})\right) . \qquad (10)$$

This optimization problem is closely related to Equation (4), but instead of weighting the dimensions with different α_d values we use a constant weighting factor γ. This enables us to compute the similarity more efficiently as it is now equivalent to the Hamming distance. More importantly, the $\{C_d(\cdot)\}$ functions are much more complex than the weak learners h_d as they are thresholded linear combinations of weak learner responses. The resulting optimization is discontinuous and 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 to solve this difficult problem and jointly optimize 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 $\{C_d(\cdot)\}$ functions iteratively, and at iteration d, the $C_d(\cdot)$ function that minimizes Equation (10) is also the one that maximizes the weighted correlation of its output and the data labels [37]. Using this fact, at iteration d, the optimal $\mathbf{b}_d$ and $\mathbf{h}_d$ can be taken as

$$\arg\max_{\mathbf{b}_d, \mathbf{h}_d} \sum_{n=1}^{N} l_n\, W_d(n) C_d(\mathbf{x}) C_d(\mathbf{y}) , \qquad (11)$$

where

$$W_d(n) = \exp\left(-\gamma l_n \sum_{d'=1}^{d-1} C_{d'}(\mathbf{x}) C_{d'}(\mathbf{y})\right) \qquad (12)$$

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 $C_d(\cdot)$ is non-differentiable, and Equation (11) is thus still hard to solve. We therefore apply the spectral relaxation trick [30], [29] and approximate the sign function using its signed magnitude, $\mathrm{sgn}(x) \approx x$. This yields:

$$\begin{aligned}
& \arg\max_{\mathbf{b}_d, \mathbf{h}_d} \sum_{n=1}^{N} l_n\, W_d(n) C_d(\mathbf{x}) C_d(\mathbf{y}) \\
\approx\ & \arg\max_{\mathbf{b}_d, \mathbf{h}_d} \sum_{n=1}^{N} l_n\, W_d(n) \left(\mathbf{b}_d^T \mathbf{h}_d(\mathbf{x}_n)\right) \left(\mathbf{b}_d^T \mathbf{h}_d(\mathbf{y}_n)\right) \\
=\ & \arg\max_{\mathbf{b}_d, \mathbf{h}_d} \sum_{n=1}^{N} l_n\, W_d(n) \mathbf{h}_d(\mathbf{x}_n)^T \mathbf{b}_d \mathbf{b}_d^T \mathbf{h}_d(\mathbf{y}_n) \\
=\ & \arg\max_{\mathbf{b}_d, \mathbf{h}_d} \mathbf{b}_d^T \left(\sum_{n=1}^{N} l_n\, W_d(n) \mathbf{h}_d(\mathbf{x}_n) \mathbf{h}_d(\mathbf{y}_n)^T\right) \mathbf{b}_d .
\end{aligned} \qquad (13)$$

As for Equation (6), we first select a vector $\mathbf{h}_d(\mathbf{x})$ of suitable weak classifiers by minimizing Equation (4) using the algorithm of [3] on the training samples, this time initially weighted by the $W_d(n)$ weights. The vector $\mathbf{b}_d$ is defined only up to a scale factor, and we then solve for it by looking for

$$\arg\max_{\mathbf{b}_d} \mathbf{b}_d^T \mathbf{M} \mathbf{b}_d, \ \text{s.t.}\ \|\mathbf{b}_d\|_2 = 1 \qquad (14)$$

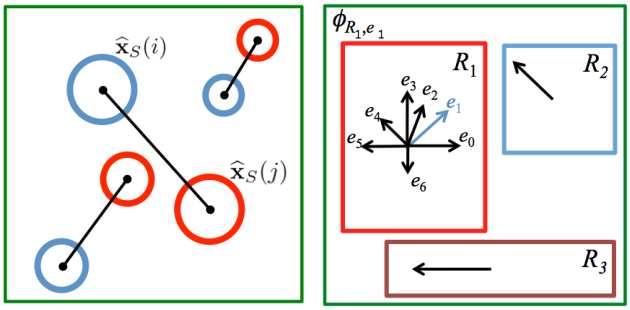
where

$$\mathbf{M} = \sum_{n=1}^{N} l_n\, W_d(n) \mathbf{h}_d(\mathbf{x}_n) \mathbf{h}_d(\mathbf{y}_n)^T . \qquad (15)$$

Eq. (14) 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. (11). Moreover, thanks to our boosting scheme even a sub-optimal selection of $C_d(\cdot)$ 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 $r_1 = \sum_{n=1}^{N} W_1(n)\, l_n\, C_1(\mathbf{x}_n) C_1(\mathbf{y}_n)$ and ν is a shrinkage parameter used to regularize our optimization as described in [38]. In practice, we use $\nu = 0.4$.



(a) Intensity-based (b) Gradient-based

Fig. 2. Overview of the intensity and gradient-based weak learners. To compute the responses of intensity-based weak learners, we compare the image intensity values after Gaussian smoothing at two locations i and j. Using boosting, we optimize both the locations and Gaussian kernel sizes, S. The gradient-based learners consider the orientations of gradients normalized within a given region. Boosting allows us to find the pooling configuration of the gradient regions and optimize the values of the corresponding thresholds.

4 Weak Learners

The employed weak learner family encodes specific design choices and desired descriptor properties. In this section we present two weak learner types inspired from existing, prevalent keypoint descriptors. The simpler, yet less discriminative weak learners are based on pixel intensities. The more complex and computationally expensive weak learners rely on gradient images. Below we provide a detailed description of each along with their parameters.

4.1 Intensity-based learners

The intensity-based weak learners rely on BRIEF-like comparisons of pre-smoothed image intensities. More precisely, we define the output of our weak learner as:

$$h(\widehat{\mathbf{x}}_S; i, j, S) = \begin{cases} 1 & \text{if } \widehat{\mathbf{x}}_S(i) \leq \widehat{\mathbf{x}}_S(j) \\ -1 & \text{otherwise} \end{cases} \quad (16)$$

where $\widehat{\mathbf{x}}_S(i)$ is the pixel intensity of $\mathbf{x}$ pre-smoothed with a Gaussian kernel of size $S \in \{3, 5, 7, \ldots, 15\}$ at position i.

The above formulation allows us to optimize the selection of the sampling points as it was done, *e.g.* in [10], except we minimize a loss function with boosting rather than the responses' corellation with a stochastic algorithm.

Inspired by the sampling scheme of BRISK [13] and FREAK [39], we also optimize the value of the Gaussian kernel size S which defines the amount of smoothing applied to the image before comparing the intensity values, in addition to the positions i and j. This adds an additional degree of freedom to our optimization framework and, therefore, encompasses the formulation of many recently proposed binary feature descriptors, such as BRISK, FREAK and ORB.

4.2 Gradient-based learners

The gradient-based weak learners consider the orientations of intensity gradients over image regions [40]. 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

$$h(\mathbf{x}; R, e, T) = \begin{cases} 1 & \text{if } \phi_{R,e}(\mathbf{x}) \leq T \\ -1 & \text{otherwise} \end{cases}, \quad (17)$$

with

$$\phi_{R,e}(\mathbf{x}) = \sum_{m \in R} \xi_e(\mathbf{x}, m) \; / \sum_{e' \in \Phi, m \in R} \xi_{e'}(\mathbf{x}, m) \,, \quad (18)$$

and

$$\xi_e(\mathbf{x}, m) = \max(0, \cos(e - o(\mathbf{x}, m)) \,, \quad (19)$$

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 is the number of quantization bins. As noted in [40] this representation can be computed efficiently using integral images.

5 Experimental setup

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 descriptors. Our approach improves over the state-of-the-art mostly with the binary version of our boosted descriptors, and we focus here on this version. Nevertheless the optimized parameters remain valid also for the floating-point descriptor.

5.1 Evaluation framework

We evaluate the performance of our methods using three publicly available datasets: Liberty, Notre Dame, and Yosemite [4]. 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 pairs of patches, where 50% correspond to match pairs, and 50% to non-match pairs. In our experiments, we use sub-sampled 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 which is the percent of incorrect matches obtained when 95% of the true matches are found, as in [4].

5.2 Weak learner types

To analyze the impact of the weak learner type on descriptor performances, we train a BinBoost$_1$-256 descriptor where each bit corresponds to one weak learner. For our gradient-based descriptor we use $q = 8$ orientation bins, as this is equal to the number of bins proposed for SIFT.

First, we compared the sampling patterns employed in the state-of-the-art binary intensity-based descriptors, such as BRIEF, BRISK and ORB, with the pooling learned with our framework when using intensity-based weak learners. Fig. 3 shows the visualization of intensity tests and heat maps of the spatial weighting employed by each descriptor. For BRIEF, intensity tests are from an isotropic Gaussian distribution with the origin of the coordinate system located at the patch center [12]. By contrast, the sampling pattern of BRISK is deterministic. The intensity tests of ORB are selected to increase the variance of the responses, while reducing their correlation. This results in a pronounced vertical trend which can also be seen in the case of BinBoost. Nevertheless, the heat maps show that the tests for BinBoost-Intensity are more dense around the center of the patch, similar to BRIEF, while the ones used in ORB present a more uniform distribution.

To evaluate the influence of the weak learner type on performance, in Fig. 4 we compared the results obtained with BinBoost-Intensity and BinBoost-Gradient with those of Boosted SSC, BRIEF, ORB, BRISK, D-Brief [14] and SIFT. The performance of Boosted SSC remains inferior to the other descriptors as the weak learners proposed in [3] rely on thresholding single pixel intensity values and do not provide enough discriminative power. Our experiments show that even though BinBoost-Intensity with variable Gaussian kernel size performs the best out of all the intensity-based descriptors, it is only slightly better than BinBoost-Intensity with filter size equal to 3. As shown in Fig. 5, our learning framework does not find a clear correlation between the size of the smoothing kernel and

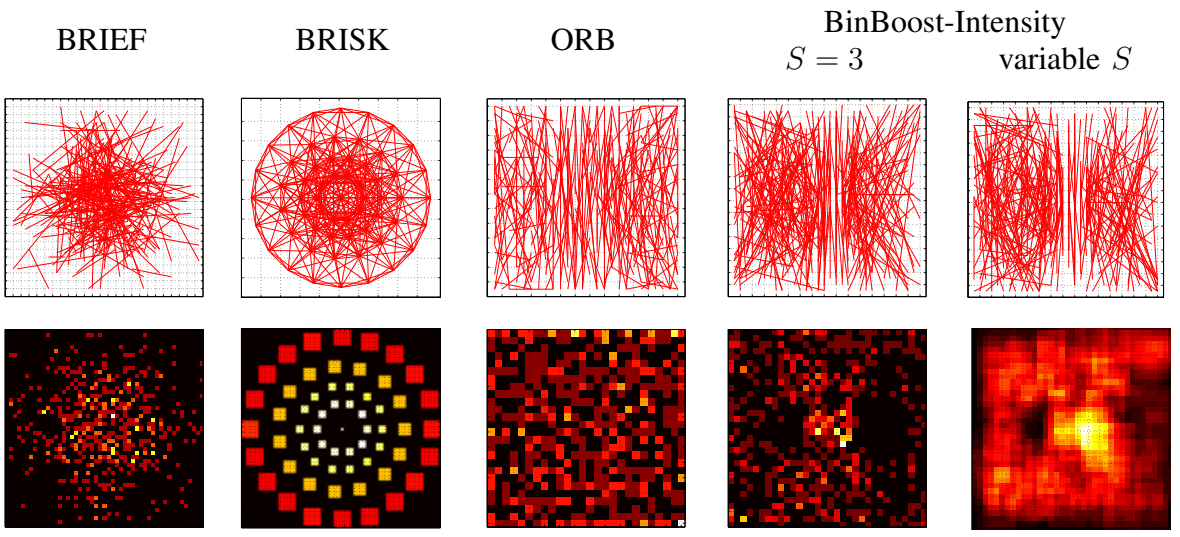


Fig. 3. Visualization of the intensity tests (first row) and spatial weight heat maps (second row) employed by BRIEF, ORB, BRISK and our BinBoost$_1$-256 descriptor trained with intensity-based weak learners on rectified patches from the Liberty dataset. BRIEF picks its intensity tests from an isotropic Gaussian distribution around the center of the patch, while the sampling pattern of BRISK is deterministic. The intensity tests of ORB are selected to increase the variance of the responses, while reducing their correlation. This results in a pronounced vertical trend which can also be seen in the case of BinBoost. Nevertheless, the heat maps show that the tests for BinBoost-Intensity are — similarly to BRIEF — more dense around the center of the patch while the ones used in ORB present a more uniform distribution.

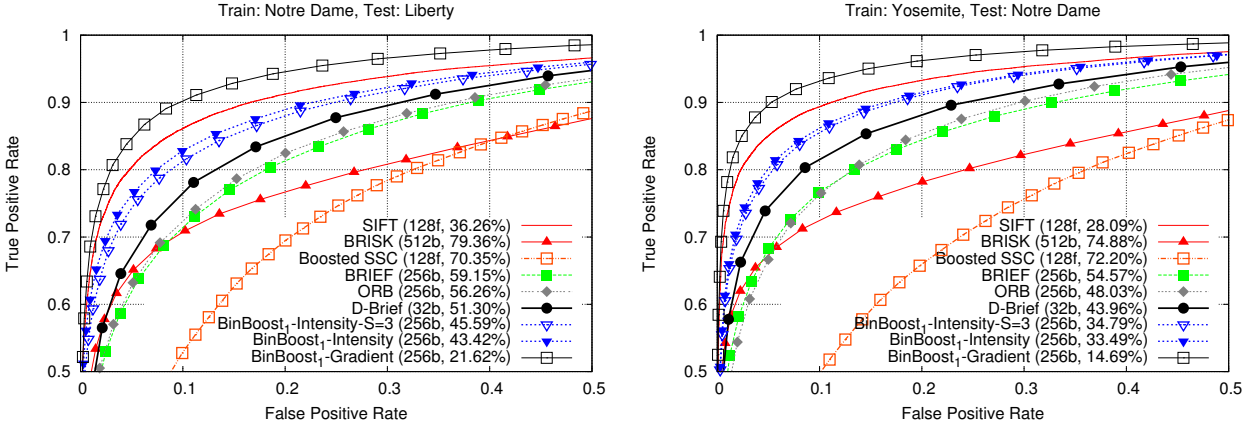


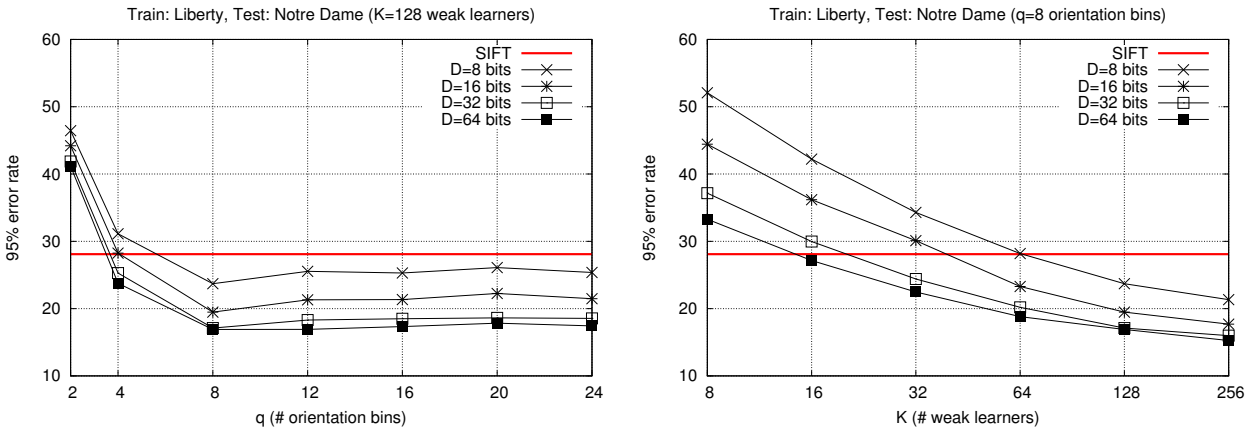
Fig. 4. Performance of BinBoost$_1$-256 with different weak learner types compared with the state-of-the-art binary descriptors and SIFT as a baseline. Out of all descriptors based on intensity tests, BinBoost-Intensity performs the best. This shows that our framework is able to effectively optimize over the other state-of-the-art binary descriptors and boost their performances at no additional computational cost. Nevertheless, the performance of BinBoost-Intensity cannot match that of floating-point SIFT which is outperformed when using the more discriminative gradient-based weak learners (BinBoost-Gradient).

the distance to the patch center, contrary to the sampling pattern of BRISK. Interestingly, even though the optimized sampling scheme of ORB resembles this of BinBoost-Intensity, our framework improves the results over BRIEF much more than ORB. This may be explained when looking at the spatial weighting employed by BinBoost and ORB, where we can see that certain parts of the patch are much more densely sampled in the case of BinBoost, whereas the sampling scheme of ORB is rather uniform.

Nevertheless, BinBoost-Intensity cannot match the performance of SIFT as the discriminative power of the underlying weak learners is not sufficient. When using gradient-based weak learners, we are able to outperform 128-dimensional floating-point SIFT with only 256 bits. Since the performance of gradient-based weak learners remains superior to the intensity-based learners, we use only the former to compute our BinBoost descriptor.

5.3 Numerical parameters

Our boosting framework defines a generic optimization strategy that unlike many previous approaches, such as [4], does not require fine tuning of multiple parameters. BinBoost has only three main parameters that provide a clear trade-off between the performance and complexity of



(a) (b)

Fig. 6. 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 $K = 128$.

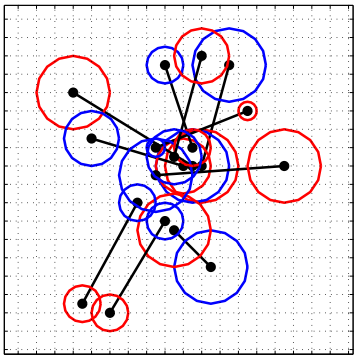


Fig. 5. Visualization of the first ten intensity-based weak learners with variable kernel size S trained on the Liberty dataset. When optimizing on both the pixel positions and the sizes of the Gaussian kernels, our boosting framework does not yield a clear pattern, in particular there is no clear correlation between the size of the smoothing kernel and the distance to the patch center, contrary to the sampling pattern proposed for BRISK. It nevertheless outperforms BRISK in our experiments.

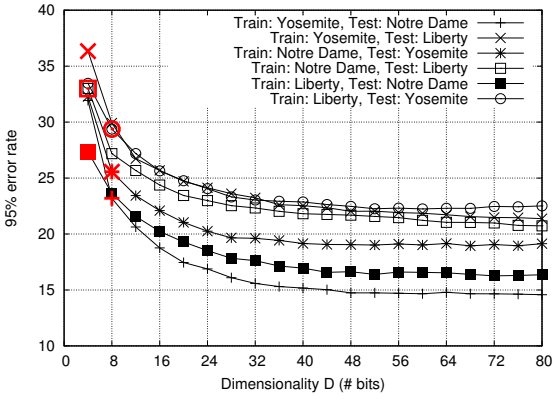


Fig. 7. Performance for different dimensionalities D. With $D = 64$ bits, BinBoost reaches its optimal performance as increasing the dimensionality further does not seem to improve the results. In bold red we mark the dimensionality for which BinBoost starts outperforming SIFT. Best viewed in color.

the final descriptor: the number of orientation bins used by the weak learners, 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. 6(a) shows the results obtained for different values of q and D. For most values of D, the performance is optimal for $q = 8$ as finer orientation quantization does not improve the performance 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. 6(b) we show the 95% error rates for different values of K. Increasing the value of K results in increased computational cost and since performance seems 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. 7 shows that with $D = 64$ bits, our descriptor reaches its optimal performance as increasing the dimensionality further does not seem to improve the results.

Using these parameters we trained our compact BinBoost descriptor on the Notre Dame dataset. A visualization of the learned weighting and pooling configuration is shown in Fig. 8 for the first 8 bits of the descriptor. 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.

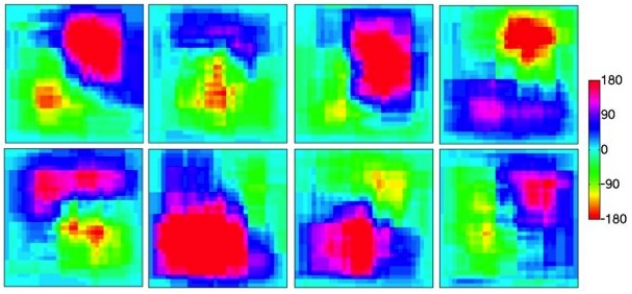


Fig. 8. Visualization of the selected weak learners for the first 8 bits learned on 200k pairs of 32×32 patches from the Notre Dame dataset (best viewed on screen). For each pixel of the figure we show the average orientation weighted by the weights of the weak learners $\mathbf{b}_d$. For different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

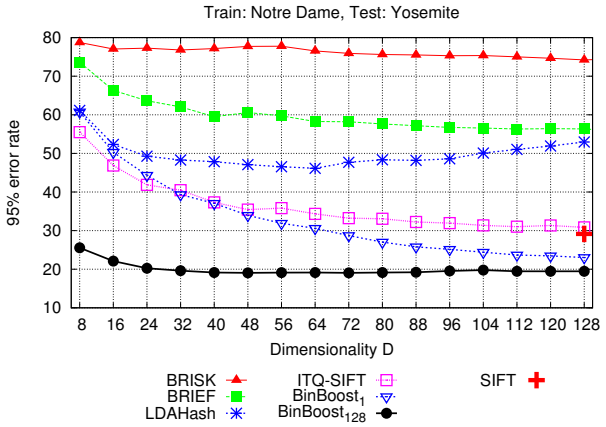


Fig. 9. 95% error rates for binary descriptors of different dimensionality. For reference, we plot the results obtained with SIFT. BinBoost outperforms the state-of-the-art binary descriptors and the improvement is especially visible for lower dimensionality.

6 Results

In this section we provide an extensive comparison of our method against the state-of-the-art descriptors on the Brown [4] and Mikolajczyk [19] datasets. We also show the performance our descriptor for performing visual search on the UKBench dataset [20].

We compare our approach against SIFT [1], SURF [2], the binary LDAHash descriptor [5], Boosted SSC [3], the binary ITQ descriptor applied to SIFT [11], and the fast binary BRIEF [12], ORB [10] and BRISK [13] descriptors.

6.1 Implementation

For SIFT, we use the publicly available implementation of A. Vedaldi [41]. For SURF, LDAHash, BRIEF, BRISK, ORB and ITQ 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. When matching the descriptors we use a fast `POPCOUNT`-based implementation for computing Hamming distances between binary descriptors and matched floating-point descriptors using their Euclidean distance.

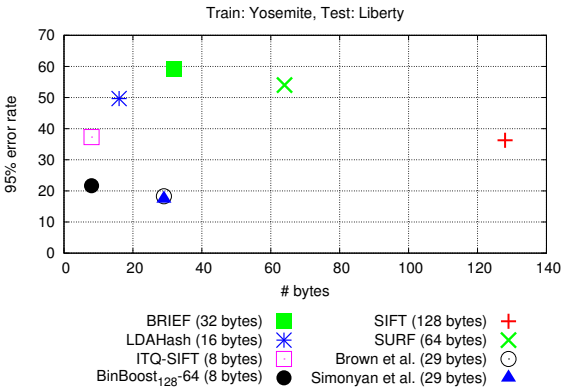


Fig. 10. Descriptor performances as a function of their memory footprint. For floating-point descriptors we assume 1 byte per dimension as this quantization was reported as sufficient for SIFT [41]. Our BinBoost descriptor offers a significantly lower memory footprint than the floating-point descriptors while providing competitive performances.

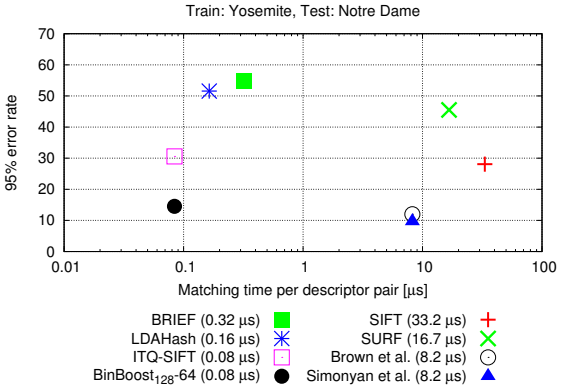


Fig. 11. Descriptor performances as a function of their matching times. The reported times were computed from 100k test pairs (*i.e.* 100k distance computations were performed) on a Macbook Pro with an Intel i7 2.66 GHz CPU using the `POPCOUNT` instruction and averaged over 100 runs. To make the comparison 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.

6.2 Brown datasets

We first compare our method using the Liberty, Notre Dame and Yosemite datasets [4] according to the evaluation protocol described in Sec. 5.1. Fig. 12 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

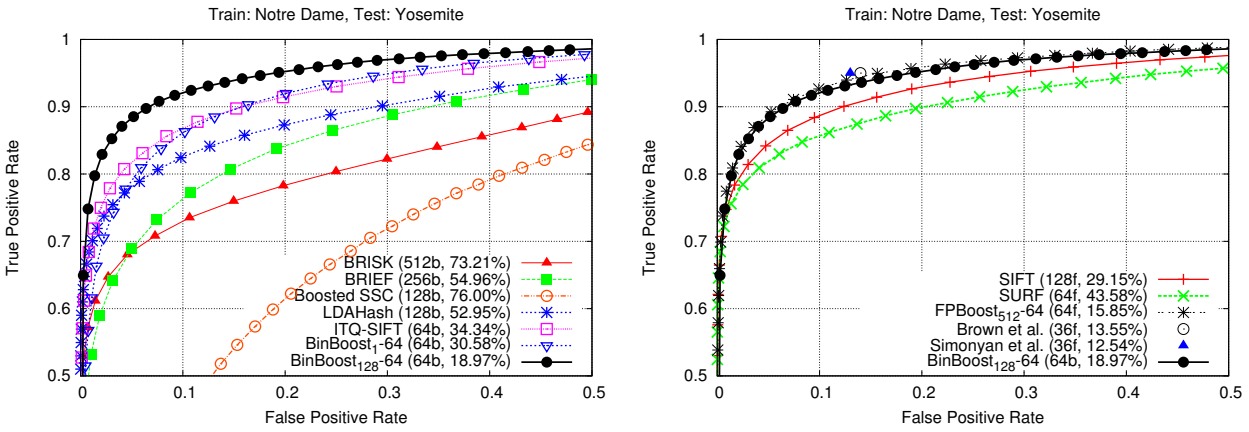


Fig. 12. Comparison of our BinBoost descriptor to the state-of-the-art binary (**left**) and floating-point (**right**) descriptors. In parentheses: the number of floating-point (f) or binary (b) dimensions and the 95% error rate. Our BinBoost descriptor significantly outperforms its binary competitors across 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.

		Binary						**Floating-point**				
Train	**Test**	BinBoost$_{128}$-64 *8 bytes*	BinBoost$_1$-64 *8 bytes*	ITQ-SIFT [11] *8 bytes*	LDAHash [5] *16 bytes*	BRIEF *32 bytes*	BRISK *64 bytes*	SURF *64 bytes*	SIFT *128 bytes*	FPBoost$_{512}$-64 *64 bytes*	Brown [4] *29 bytes*	Simonyan [6] *29 bytes*
Yosemite	Notre Dame	**14.54**	26.80	30.56	51.58	54.57	74.88	45.51	28.09	14.80	11.98	9.67
Liberty		**16.90**	29.60	31.07						14.68	-	-
Yosemite	Liberty	**21.67**	33.54	37.31	49.66	59.15	79.36	54.01	36.27	22.39	18.27	17.44
Notre Dame		**20.49**	31.90	36.95						17.90	16.85	14.51
Notre Dame	Yosemite	**18.97**	30.58	34.34	52.95	54.96	73.21	43.58	29.15	15.85	13.55	12.54
Liberty		**22.88**	38.13	34.43						20.85	-	-

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, FPBoost, Brown *et al.* [4], Simonyan *et al.* [6]) we assume 1 byte per dimension, as this quantization was reported as sufficient for SIFT [41]. 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 matching time as shown in Fig. 11.

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.

Since the dimensionality of the other binary descriptors can be varied depending on the required performance quality, Fig. 9 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 for lower dimensionality.

Moreover, our BinBoost descriptor remains competitive to the best descriptors of [4] and [6], even though the memory footprint of their descriptors is almost 4 times greater as shown in Fig. 10. The real advantage of BinBoost, however, is that it allows for extremely fast similarity computation using the Hamming distance[2], whereas the descriptors of [4] and [6] are floating-point and cannot benefit from the same optimization, even when quantized very coarsely. As presented in Fig. 11, 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 to FPBoost. Results are displayed in Fig. 13. Binarizing the FPBoost coordinates by thresholding them at an optimal threshold found as in [5] results in large binarization errors significantly decreasing the accuracy of the resulting binary

[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.

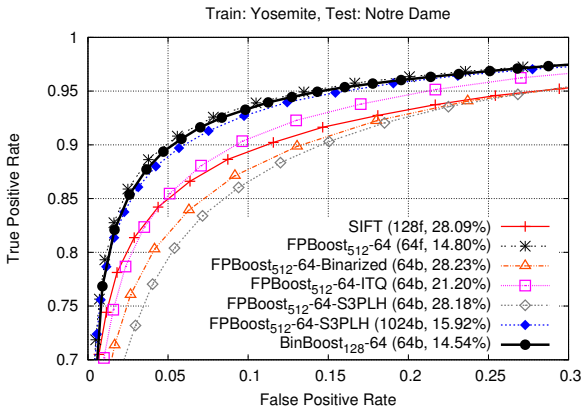


Fig. 13. Performance of our BinBoost descriptor compared with different binarization methods applied on FPBoost. Binarizing the discriminative projections found with FPBoost either by simple thresholding or with Iterative Quantization (ITQ) results in large binarization errors significantly reducing its accuracy. On the other hand, the sequential projection learning of S3PLH requires a fairly large number of bits to recover the original performance of FPBoost. 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 FPBoost but at a fraction of the storage cost.

representation. This error can be reduced using Iterative Quantization [11], however, the orthogonality constraints used in this approach largely limit the extent to which it can be minimized. In contrast, sequential projection learning (S3PLH) [29] can find non-orthogonal projections that more faithfully mitigate binarization error, however, it requires a fairly large number of bits to recover FPBoost'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.

6.3 Mikolajczyk dataset

We tested the generalization performance of our descriptor when trained on the Brown datasets and evaluated on the significantly different Mikolajczyk dataset [19]. We report results using the Notre Dame dataset, however, similar results were found for all the Brown datasets. We followed the evaluation protocol of [19] that compares descriptors using a single keypoint detector, and used the OpenCV SURF Hessian-based detector. For each image pair we detect 1000 keypoints per image and match them using exhaustive search. We then filter outliers using a distance ratio threshold of 0.8 as in [1]. We evaluate each descriptor in terms of the recognition rate which is the number of correctly matched keypoints.

Fig. 14 shows the results obtained for the `bark`, `boat`, `graf`, `trees`, `ubc` and `wall` sequences. In all the sequences $\text{BinBoost}_1\text{-}256$ and $\text{FPBoost}_{512}\text{-}64$ outperform the other descriptors. The performance of $\text{BinBoost}_{128}\text{-}64$, however, does not perform as well as when evaluated on the Brown datasets, which indicates that there is an inherent efficiency tradeoff when training on a different condition. Nonetheless, the extended $\text{BinBoost}_{128}\text{-}128$ and $\text{BinBoost}_{128}\text{-}256$ descriptors outperform the other methods while being shorter or of the same length.

6.4 Visual Search on UKBench

We further evaluate our approach for performing visual search using the University of Kentucky Benchmark (UKBench) dataset [20] that contains over 10k images of 2600 objects, each object being depicted in 4 images taken from different viewpoints. As in other approaches, we first build a database of the almost one million descriptors extracted from all the dataset images. For each query image, we then search for the nearest neighbors in the database using their associated keypoint descriptors to vote for the most similar images in the database. Finally, we sort the database images according to the number of votes they receive and retrieve those associated with the highest number of votes. As with our previous experiments, we consider descriptors trained using the Notre Dame dataset with similar results seen for the other Brown datasets. In our evaluation we randomly selected 500 query images from the dataset and use the remaining 10k images to create a database. We ran the experiment three times and report the average results along with the standard deviation values.

Table 2 summarizes the results we obtained for different descriptors. To evaluate the performance we report mean average precision (mAP) and percentage of correct number of images retrieved at the top of the list (Correct@1). Out of all the evaluated descriptors $\text{BinBoost}_{128}\text{-}256$ performs the best followed by $\text{BinBoost}_1\text{-}256$. FPBoost performs slightly worse than BinBoost, while still outperforming SIFT and other intensity-based binary descriptors. Overall, the boosted keypoints descriptors provide the best performance of all the tested descriptors, even though they were trained on a significantly different dataset.

7 Face Descriptors

In this section we evaluate our method for matching face images. While this constitues a rather different problem than modeling the appearance of local keypoints, as we show our method is generic and can be easily adapted to new application domains. For our evaluation we used a dataset of face images [42] that consists of faces imaged under different viewpoints. From this dataset we created two sets of 100k and 200k pairs of images. Similarly to the Liberty, Notre Dame and Yosemite datasets, each set is balanced and contains an equal number of image pairs belonging to the same person as those of different people. We used the 200k dataset to train our descriptors and the 100k set to test them.

Fig. 15 compares the learned spatial weightings obtained with the Brown and Faces datasets. When we train our

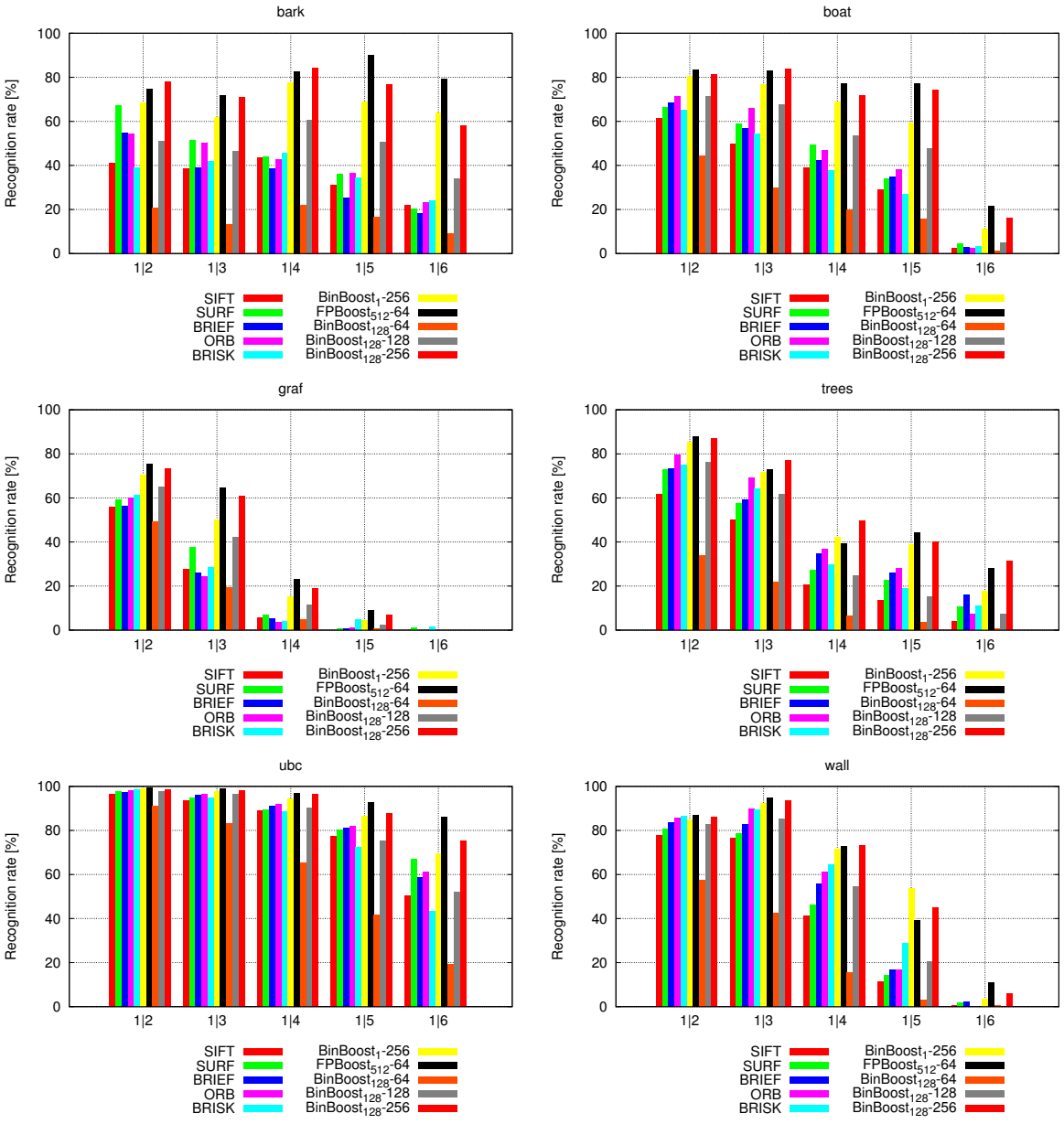


Fig. 14. Recognition rates for the Mikolajczyk dataset. The proposed BinBoost and FPBoost descriptors significantly outperform the competitors, both binary and floating-point, while being shorter or of the same length. This shows that, although generalization of the learned descriptors remains a challenging task, they can still perform well under conditions that differ greatly from the training conditions.

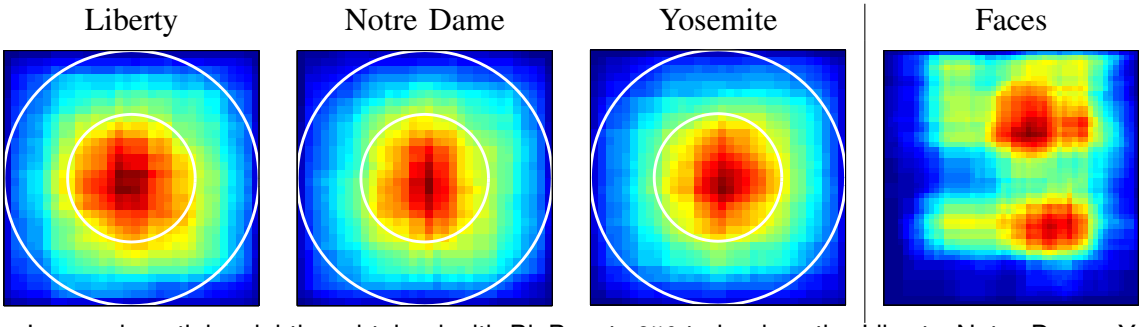


Fig. 15. Learned spatial weighting obtained with BinBoost$_1$-256 trained on the Liberty, Notre Dame, Yosemite and Faces datasets. For the first three datasets, the learned weighting closely resembles the Gaussian weighting employed by SIFT (white circles indicate $\sigma/2$ and σ used by SIFT). However, the learned spatial weighting on the Faces dataset is focused about the eyes and mouth that constitute discriminative facial features.

Descriptor	mAP $\pm \sigma$	Correct@1 $\pm \sigma$
BRISK	0.402 $\pm$ 0.006	61.830 $\pm$ 0.884
ORB	0.418 $\pm$ 0.005	64.902 $\pm$ 1.931
SIFT	0.455 $\pm$ 0.008	68.235 $\pm$ 2.183
BRIEF	0.457 $\pm$ 0.014	68.562 $\pm$ 0.493
FPBoost$_{512}$-64	0.476 $\pm$ 0.006	70.000 $\pm$ 1.709
BinBoost$_{128}$-128	0.493 $\pm$ 0.017	72.222 $\pm$ 2.747
BinBoost$_{1}$-256	0.533 $\pm$ 0.010	76.144 $\pm$ 2.467
BinBoost$_{128}$-256	**0.556** $\pm$ **0.008**	**79.216** $\pm$ **1.870**

TABLE 2

Results of visual search on the UKBench dataset [20]: mean average precision (mAP) and percentage of correctly retrieved images at the first position (Correct@1) are reported. Average results are shown across three random train and test splits along with the standard deviation. BinBoost$_{128}$-256 outperforms the other descriptors, even though it is trained on the Notre Dame dataset. The other learned descriptors, namely BinBoost$_{1}$-256 and FPBoost$_{512}$-64, achieve worse results, though their performance is still better than SIFT and the other intensity-based descriptors.

Fig. 16. Matching results on the Faces dataset using our 256-bit BinBoost$_{1}$-256 at the 95% error rate, *i.e.* when 95% of the positive image pairs are correctly classified. BinBoost remains robust to significant viewpoint changes and motion blur. The mis-classified examples are mostly due to occlusion and extreme variations in viewpoint such as side views.

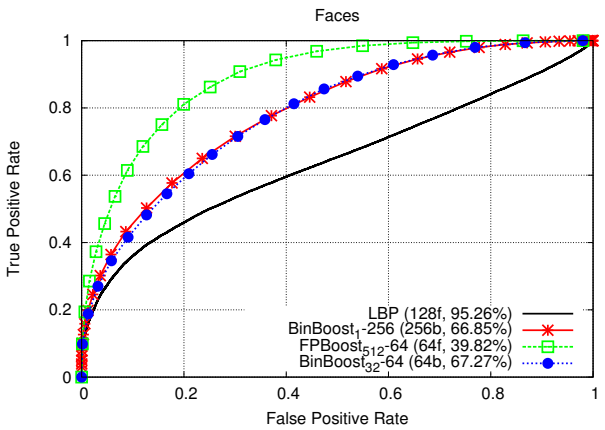


Fig. 17. The performance of our boosted descriptors on the Faces dataset compared with the commonly used LBP face descriptor [43]. BinBoost$_{1}$-256 significantly outperforms LBP. Similarly to the results obtained for local feature descriptors, we can see that BinBoost$_{128}$-64 performs equally to BinBoost$_{1}$-256, but with only 64 bits per descriptor. FPBoost performs even better with the 95% error rate reduced by more than twice compared with the LBP baseline.

BinBoost descriptor on the images extracted around interest points, the weak learners clearly concentrate around the center of the patch. In fact, the obtained weighting closely resembles the Gaussian weighting employed by SIFT. In contrast, for face images the weak learners concentrate about the lower and upper image regions that correspond to the location of the eyes and mouth, and as also observed in [43] constitute discriminative facial features. This further demonstrates the flexibility of our approach and its ability to adapt to new types of image data.

Fig. 16 shows the qualitative results obtained using BinBoost$_{1}$-256. BinBoost remains largely invariant to the significant viewpoint and intensity changes present in this dataset, while still being able to discriminate between different people. Most of the mis-classifications are due to occlusions and extreme viewpoint variation such as side views.

In Fig. 17 we plot the quantitative results of BinBoost$_{1}$-256, FPBoost$_{256}$-64 and BinBoost$_{128}$-64 descriptors compared with LBP, a widely used face descriptor [43]. Our boosted descriptors result in a significant improvement over the baseline. Furthermore, compared with LBP our FPBoost descriptor achieves a reduction in 95% error rate by more than a factor of 2. Similar to [44], [45] this demonstrates the potential advantages of exploiting image data to learn a face descriptor. More importantly, it illustrates the flexibility of our approach beyond local keypoint descriptors.

8 Conclusion

In this paper we presented an efficient framework to train highly discriminative and compact local feature descriptors that leverages the boosting-trick to simultaneously optimize both the weighting and sampling strategy of a set of non-linear feature responses. We first showed how boosting can be used to result in an accurate yet compact floating-point descriptor. We then considered a binary extension of our approach that shares a similar accuracy but operates at a fraction of the matching and storage cost. We explored the use of both intensity- and gradient-based features within our learning framework and performed an evaluation across a variety of descriptor matching tasks. In each task, our

approach achieved a signficant improvement over the state-of-the-art descriptors, such as BRIEF and BRISK, in both accuracy and efficiency by optimizing their sampling patterns. Finally, we showed that our method can be easily generalized to new applicaiton domains, such as faces.

REFERENCES

[1] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *IJCV*, vol. 20, no. 2, pp. 91–110, 2004.
[2] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded Up Robust Features," *ECCV*, 2006.
[3] G. Shakhnarovich, "Learning Task-Specific Similarity," Ph.D. dissertation, MIT, 2006.
[4] M. Brown, G. Hua, and S. Winder, "Discriminative Learning of Local Image Descriptors," *PAMI*, 2011.
[5] C. Strecha, A. Bronstein, M. Bronstein, and P. Fua, "LDAHash: Improved Matching with Smaller Descriptors," *PAMI*, vol. 34, no. 1, 2012.
[6] K. Simonyan, A. Vedaldi, and A. Zisserman, "Descriptor Learning Using Convex Optimisation," *ECCV*, 2012.
[7] S. Shen, W. Shi, and Y. Liu, "Monocular 3D Tracking of Inextensible Deformable Surfaces Under L2-Norm," *ACCV*, 2009.
[8] P. Jain, B. Kulis, J. Davis, and I. Dhillon, "Metric and Kernel Learning Using a Linear Transformation," *Journal of Machine Learning Research*, 2012.
[9] O. Chapelle, P. Shivaswamy, S. Vadrevu, K. Weinberger, Y. Zhang, and B. Tseng, "Boosted Multi-Task Learning," *Machine Learning*, 2010.
[10] E. Rublee, V. Rabaud, K. Konolidge, and G. Bradski, "ORB: An Efficient Alternative to SIFT or SURF," *ICCV*, 2011.
[11] 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.
[12] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua, "BRIEF: Computing a Local Binary Descriptor Very Fast," *PAMI*, vol. 34, no. 7, pp. 1281–1298, 2012.
[13] S. Leutenegger, M. Chli, and R. Siegwart, "BRISK: Binary Robust Invariant Scalable Keypoints," *ICCV*, 2011.
[14] T. Trzcinski and V. Lepetit, "Efficient Discriminative Projections for Compact Binary Descriptors," *ECCV*, 2012.
[15] M. Norouzi, A. Punjani, and D. Fleet, "Fast Search Hamming Space with Multi-Index Hashing," *CVPR*, 2012.
[16] M. Malekesmaeili, R. Ward, and M. Fatourechi, "A Fast Approximate Nearest Neighbor Search Algorithm the Hamming Space," *PAMI*, 2012.
[17] T. Trzcinski, M. Christoudias, V. Lepetit, and P. Fua, "Learning Image Descriptors with the Boosting-Trick," *NIPS*, 2012.
[18] T. Trzcinski, M. Christoudias, P. Fua, and V. Lepetit, "Boosting Binary Keypoint Descriptors," *CVPR*, 2013.
[19] 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*, vol. 65, no. 1/2, pp. 43–72, 2005.
[20] D. Nister and H. Stewenius, "Scalable Recognition with a Vocabulary Tree," *CVPR*, 2006.
[21] V. Chandrasekhar, G. Takacs, D. Chen, S. Tsai, R. Grzeszczuk, and B. Girod, "CHoG: Compressed Histogram of Gradients a Low Bit-Rate Feature Descriptor," *CVPR*, 2009.
[22] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating Local Descriptorsto a Compact Image Representation," *CVPR*, 2010, pp. 3304–3311.
[23] C. Zitnick, "Binary Coherent Edge Descriptors," *ECCV*, 2010.
[24] B. Kulis and T. Darrell, "Learning to Hash with Binary Reconstructive Embeddings," *NIPS*, 2009, pp. 1042–1050.
[25] R. Salakhutdinov and G. Hinton, "Semantic Hashing," *International Journal of Approximate Reasoning*, vol. 50, no. 7, 2009.
[26] Y. Weiss, A. Torralba, and R. Fergus, "Spectral Hashing," *NIPS*, vol. 21, pp. 1753–1760, 2009.
[27] Y. Weiss, R. Fergus, and A. Torralba, "Multidimensional Spectral Hashing," *ECCV*, 2012.
[28] M. Norouzi and D. Fleet, "Minimal Loss Hashing for Compact Binary Codes," *ICML*, 2011.
[29] J. Wang, S. Kumar, and S.-F. Chang, "Sequential Projection Learning for Hashing with Compact Codes," *ICML*, 2010.
[30] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised Hashing with Kernels," *CVPR*, 2012.
[31] K. Grauman and T. Darrell, "The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features," *ICCV*, 2005, pp. 1458–1465.
[32] Y. Freund and R. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *European Conference on Computational Learning Theory*, 1995, pp. 23–37.
[33] P. Dollár, Z. Tu, P. Perona, and S. Belongie, "Integral Channel Features," *BMVC*, 2009.
[34] P. Viola and M. Jones, "Rapid Object Detection Using a Boosted Cascade of Simple Features," *CVPR*, 2001.
[35] S. Rosset, J. Zhu, and T. Hastie, "Boosting as a Regularized Path to a Maximum Margin Classifier," *Journal of Machine Learning Research*, 2004.
[36] A. Torralba, R. Fergus, and Y. Weiss, "Small Codes and Large Databases for Recognition," *CVPR*, 2008.
[37] R. E. Schapire and Y. Singer, "Improved Boosting Algorithms Using Confidence Rated Predictions," *Machine Learning*, vol. 37, no. 3, pp. 297–336, 1999.
[38] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer, 2001.
[39] A. Alahi, L. Jacques, Y. Boursier, and P. Vandergheynst, "Sparsity Driven People Localization with a Heterogeneous Network of Cameras," *Journal of Mathematical Imaging and Vision*, 2011.
[40] K. Ali, F. Fleuret, D. Hasler, and P. Fua, "A Real-Time Deformable Detector," *PAMI*, vol. 34, no. 2, pp. 225–239, 2012.
[41] A. Vedaldi, "http://www.vlfeat.org/ vedaldi/code/siftpp.html," 2005.
[42] M. Zervos, "Multi-Camera Face Detection and Recognition Applied to People Tracking," Master's thesis, EPFL, 2013.
[43] T. Ahonen, A. Hadid, and M. Pietikïinen, "Face Description with Local Binary Patterns: Application to Face Recognition," *PAMI*, vol. 28, no. 12, pp. 2037–2041, 2006.
[44] Z. Cao, Q. Yin, X. Tang, and J. Sun, "Face Recognition with Learning-Based Descriptor," *CVPR*, 2010.
[45] X. Wang, C. Zhang, and Z. Zhang, "Boosted Multi-Task Learning for Face Verification with Applications to Web Image and Video Search," *CVPR*, 2009.

Tomasz Trzcinski received his MSc degree in Research on Information and Communication Technologies from Universitat Politècnica de Catalunya and MSc degree in Electronics Engineering from Politecnico di Torino in 2010. He joined EPFL in 2010 where he is currently pursuing his PhD in computer vision. His research interests include visual search, augmented reality and machine learning. He worked with Telefonica R&D, Qualcomm and Google.

Mario Christoudias received a BS degree in electrical and computer engineering from Rutgers University in 2002 and MS and PhD degrees in computer science from the Massachusetts Institute of Technology in 2004 and 2009. He is currently a postdoctoral researcher in the Computer Vision Laboratory at EPFL. His main research interests are in the areas of computer vision and machine learning.

Vincent Lepetit is a Professor at the Institute for Computer Graphics and Vision, TU Graz and a Visiting Professor at the Computer Vision Laboratory, EPFL. He received the engineering and master degrees in Computer Science from the ESIAL in 1996. He received the PhD degree in Computer Vision in 2001 from the University of Nancy, France, after working in the ISA INRIA team. He then joined the Virtual Reality Lab at EPFL as a post-doctoral fellow and became a founding member of the Computer Vision Laboratory. His research interests include vision-based Augmented Reality, 3D camera tracking, object recognition and 3D reconstruction.