# Efficient Discriminative Projections for Compact Binary Descriptors

Tomasz Trzcinski and Vincent Lepetit

CVLab, EPFL, Lausanne, Switzerland
firstname.lastname@epfl.ch

**Abstract.** Binary descriptors of image patches are increasingly popular given that they require less storage and enable faster processing. This, however, comes at a price of lower recognition performances. To boost these performances, we project the image patches to a more discriminative subspace, and threshold their coordinates to build our binary descriptor. However, applying complex projections to the patches is slow, which negates some of the advantages of binary descriptors. Hence, our key idea is to learn the discriminative projections so that they can be decomposed into a small number of simple filters for which the responses can be computed fast. We show that with as few as 32 bits per descriptor we outperform the state-of-the-art binary descriptors in terms of both accuracy and efficiency.

## 1 Introduction

Local image feature descriptors have been vastly used in many computer vision applications such as image retrieval, pose estimation and 3D reconstruction. Their main goal is to represent a salient image region while remaining invariant to various illumination and viewpoint changes. In practice, however, obtaining such a robust representation becomes a challenging task, especially for mobile devices with limited resources.

Numerous methods have been proposed in the literature to tackle this problem [1, 2], but very recently, several *binary* descriptors computed directly on image patches—BRIEF [3], ORB [4], and BRISK [5]—have appeared. They are both fast to compute and to match, with a very small memory footprint, and their quick success clearly shows a need for such descriptors in real applications, especially for low-end handheld devices.

However, these new descriptors tend to be less robust than slower approaches. In this work, we aim to bridge this performance gap without increasing the computational cost. In order to boost the recognition performances, we adopt a discriminative approach as in [6–8]: We use training data to learn linear projections that map image patches to a more discriminative subspace, and to obtain a binary descriptor, we threshold the projected patches. This way we avoid the intermediate step of computing complex floating-point descriptors, which is common for many binary descriptors [7, 9] but exceeds the capabilities of many mobile platforms.

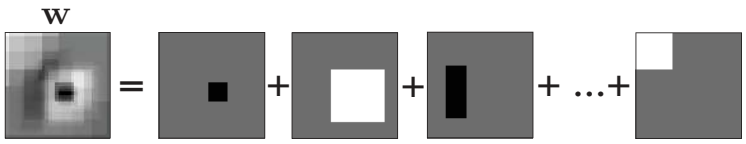



**Fig. 1.** To compute our binary descriptor, we learn from a training set of corresponding image patches several discriminative linear projections that can be computed from a linear combination of a few simple filters. For the example of this figure, we used rectangular filters that can be computed efficiently with integral images, but we also consider box and Gaussian filters which are also efficient to compute. This approach enables us to build our binary descriptor fast while leveraging on training data.

Nevertheless, projecting image patches is computationally expensive and negates the efficiency of binary descriptors, especially when they have to be computed in real time. Thus, in our approach, and this is our main contribution, we train the projections not only to be discriminative but also to be computed as a linear combination of a small number of simple filters from a given dictionary, as shown in Fig. 1. We design the dictionaries in such a way that the filter responses can be computed fast, with box or Gaussian filtering or using integral images. Our key idea is that this can be done by imposing sparsity constraints and using efficient optimization techniques.

To summarize, we build our binary descriptor, which we refer to as D-Brief for Discriminative BRIEF[1], by first projecting image patches to a more discriminant subspace and then concatenating the results of a thresholding operation applied on the projected coordinates. When we use box or Gaussian filters to construct the projections, we can speed up the computation of projected patches by first convolving the entire image with a given filter and then combining only a few values read from the convolved image. With rectangular filters, we use integral images to compute the responses fast. As a result, D-Brief provides better recognition performances than its direct competitors [3–5], while being significantly shorter—only 32 bits to be compared with several hundreds—and less time consuming. D-Brief can also be seen as a much more efficient binary alternative to the short floating-point descriptors of [6], which require more time to be computed, and hence target different applications than binary descriptors.

The rest of this paper is organized as follows. In Section 2, we discuss the related work. In Section 3, we introduce our method to construct a binary descriptor from a set of discriminant projections learnt from a training dataset. We explain in details how we optimize on the projections so that they can be computed using small number of simple filter banks which enables efficient computation of projected patches. Finally, we compare the performances of our D-Brief binary descriptor with the state of the art and we show an example of a real-time application based on D-Brief.

[1] The reference implementation of D-Brief will be made publicly available.

## 2 Related Work

Due to the increasing interest of the research community in real-time applications and the implementation of Computer Vision algorithms on mobile devices, many efficient feature descriptors have been proposed in the literature.

The widely used SURF descriptor [2] is created by summing responses of Haar wavelets, which can be done very fast using integral images. We also rely on integral images for one of our dictionaries of filters, but to speed up the computation of learnt discriminative projections, not to compute handcrafted filters. Our descriptor is also significantly shorter and faster to compute.

Few recent methods propose to construct efficient low bit-rate descriptors using compression or quantization techniques [10, 11]. These approaches, however, were mainly designed for visual search, which is an application of much lower speed requirements than our descriptor aims at. Thus, even though the resulting descriptors are compact, they require complex image gradient computations which often remain prohibitive for mobile devices.

Much research has been recently focused on designing binary descriptors, since they require much less storage than the real-valued ones. They also enable faster matching as there are efficient indexing schemes for binary vectors [12, 13] and Hamming distances can be computed fast on many architectures, including mobile devices. Thanks to all these properties, binary descriptors are perfect candidates for real-time applications. Both unsupervised and supervised approaches have been proposed. Unsupervised approaches seek for a transformation that preserves the similarity as evaluated in the original space, typically using the Euclidean distance [13–15]. Our method belongs to the category of supervised approaches: These approaches aim to outperform unsupervised ones by exploiting labeled data to produce similar binary representations for data with the same class labels [7, 9, 16, 17].

Our work is closely related to LDAHash [7], which also computes a binary descriptor using discriminative projections but applied to SIFT descriptors instead of image patches. Linear Discriminative Analysis was also applied to image patches and other descriptors in [6]. However, the descriptors in [6] are not binary and much more time-consuming to compute and match than ours. Our main contribution, on the other hand, is to show that the linear projections can be optimized for very fast computation using recent optimization theory, and is therefore not limited to LDA.

Most of the binarization methods mentioned above are applied to a sophisticated descriptor, such as SIFT or Gist, and thus exceed the capabilities of modern mobile devices. To address this problem, we consider here simple intensity image patches. This is motivated by very recent binary feature descriptors [3, 5, 4], which are computed by applying simple tests directly to the image. These tests compare the image intensities at two pixel locations. For noise removal, the image is pre-smoothed with a box filter or a Gaussian filter. BRIEF [3] uses random pre-determined locations, whereas BRISK [5] uses an exhaustive set of comparisons of close locations. ORB [4] relies on optimization like we do and aims at improving the recognition rates by choosing the locations that decorre-

late the tests. However, the approach in ORB is limited to a very specific type of tests and relies on a greedy optimization, while we can deal with tests of general form and incorporate the computational cost as a penalty in our cost function. Finally, our formulation encompasses the computation of these descriptors, as they are built by computing a linear combination of two box or Gaussian filter responses and thresholding the resulting values at zero.

## 3 Learning Framework

Our binary descriptor is computed by applying a set of projections to a real-valued vector made of the intensities of an image patch and then thresholding the results:

$$\forall_{i \in 1,\ldots,N} \quad b_i = \text{sign}(\mathbf{w}_i^\top \mathbf{x} + \tau_i) \,, \tag{1}$$

where the $b_i$ are the $N$ bits of our descriptor, the $\mathbf{w}_i$ the projections, the $\tau_i$ the thresholds, and $\mathbf{x}$ the image patch in vector form. We show in Section 3.1 how to optimize over the $\{\mathbf{w}_i, \tau_i\}$ to obtain our efficient and discriminative descriptor, and in Section 3.2 we explain how it can be computed fast. Finally, we evaluate how our descriptor is influenced by its parameters in Section 3.3.

### 3.1 Approach

In principal, our approach seeks to minimize the expected Hamming distance between binary descriptors that describe similar keypoints while maximizing it for the descriptors that describe different keypoints. To that end, we learn a set of discriminative orthogonal linear projections and the corresponding thresholds from a set $\mathcal{P}$ of pairs of corresponding image patches and another set $\mathcal{N}$ of pairs of different patches. Nevertheless, applying general projections directly on the image patches is computationally expensive. Hence, our key idea is to train the projections $\mathbf{w}_i$ to be a linear combination of a few elements from a predefined set or *dictionary* $D$, which is designed to contain elements for which the responses can be computed fast, for example using box filters.

More formally, we express the projections as $\mathbf{w}_i = D\mathbf{s}_i$, where the dictionary $D$ is defined as a matrix with its columns being the elements of the dictionary. We want most of the coefficients of the $\mathbf{s}_i$ vectors to be equal to zero, that is, the $\mathbf{s}_i$ should be sparse. Our goal can then be formalized as solving the following minimization problem:

$$\begin{aligned} \min_{\{(\mathbf{s}_i,\tau_i)\}} \quad & \sum_{i \in 1,\ldots,N} \sum_{(\mathbf{x},\mathbf{x}') \in \mathcal{N}} \text{sign}((D\mathbf{s}_i)^\top \mathbf{x} + \tau_i)\, \text{sign}((D\mathbf{s}_i)^\top \mathbf{x}' + \tau_i) - \\ & \sum_{(\mathbf{x},\mathbf{x}') \in \mathcal{P}} \text{sign}((D\mathbf{s}_i)^\top \mathbf{x} + \tau_i)\, \text{sign}((D\mathbf{s}_i)^\top \mathbf{x}' + \tau_i) + \lambda |\mathbf{s}_i|_1 \\ & \text{subject to} \quad (D\mathbf{s}_i)^\top (D\mathbf{s}_j) = \delta_{ij} \,, \end{aligned} \tag{2}$$

where the last term encourages sparsity of the $\mathbf{s}_i$ with $\lambda$ determining its sparsity level, and $|.|_1$ denotes the $\ell_1$ norm. The constraint, with $\delta_{ij} = 1$ if $i = j$ and 0 otherwise, makes sure that the projections are orthogonal.

Unfortunately, direct minimization of this objective function is difficult as it involves the non-differentiable sign function. In our case, typical solutions of this

problem, such as smooth approximation with the hinge function [18], would lead to a quadratic non-convex problem which is challenging to solve, as it involves thousands of unknowns.

Thus, we drop the sign function and minimize the related objective function as it is done in [7]:

$$\min_{\{\mathbf{s}_i\}} \quad \sum_i \frac{\sum_{(\mathbf{x},\mathbf{x}')\in\mathcal{P}} ((D\mathbf{s}_i)^\top(\mathbf{x}-\mathbf{x}'))^2}{\sum_{(\mathbf{x},\mathbf{x}')\in\mathcal{N}} ((D\mathbf{s}_i)^\top(\mathbf{x}-\mathbf{x}'))^2} + \lambda|\mathbf{s}_i|_1 \qquad (3)$$
$$\text{subject to} \quad (D\mathbf{s}_i)^\top(D\mathbf{s}_j) = \delta_{ij}$$

The above objective is independent of the thresholds $\tau_i$. Hence, after finding the projections $\mathbf{w}_i = D\mathbf{s}_i$, the optimal thresholds are obtained by minimizing the original objective of Eq. (2) using the training sets $\mathcal{P}$ and $\mathcal{N}$. With the projections $\mathbf{w}_i$ being fixed, this requires simple one-dimensional search, as explained in [7].

A possible method to solve the minimization problem of Eq. (3) is by using Stochastic Gradient Descent, with soft-thresholding as the proximal operator of the $\ell_1$ norm [19]. However, even after dropping the sign function, Eq. (3) remains non-convex and the optimization is likely to get stuck in a local minimum. Therefore, it becomes essential to initialize the optimization properly, because random initialization may not give satisfactory results, as we show below.

We propose to set the initialization point of the optimization using the following approach: We start by minimizing the first term of Eq. (3) and we obtain an initial set of discriminant projections $\{\mathbf{w}_i^0\}$ using Linear Discriminant Embedding (LDE) [6]:

$$\{\mathbf{w}_i^0\} = \arg\min_{\{\mathbf{w}_i\}} \quad \sum_i \frac{\sum_{(\mathbf{x},\mathbf{x}')\in\mathcal{P}} (\mathbf{w}_i{}^\top(\mathbf{x}-\mathbf{x}'))^2}{\sum_{(\mathbf{x},\mathbf{x}')\in\mathcal{N}} (\mathbf{w}_i^\top(\mathbf{x}-\mathbf{x}'))^2} \,. \qquad (4)$$

As one can see, the LDE includes the orthogonality constraint from Eq. (3), as the resulting projections are based on the orthogonal eigenvectors. We then address the sparsity constraint of Eq. (3) and approximate each $\mathbf{w}_i^0$ projection with a sparse linear combination of elements from dictionary $D$ by minimizing the following objective:

$$\{\mathbf{s}_i^0\} = \arg\min_{\mathbf{s}_i} \quad \|\mathbf{w}_i^0 - D\mathbf{s}_i\|_2^2 + \lambda|\mathbf{s}_i|_1 \,, \qquad (5)$$

where the first term corresponds to the quality of approximation, and the second one to the sparsity of the filter representation. To simplify our optimization, we do not constrain our approximated projections to be orthogonal, although this could certainly lead to an interesting extension of our approach.

The advantage of the stepwise approach discussed above is that Eq. (4) can be solved in closed-form as shown in [6], while Eq. (5) is convex and can be solved with efficient recent techniques [19]. In practice, we use the MATLAB `lasso` function which implements [20] and lets the user define $|\mathbf{s}|_0^{\max}$, the maximal number of non-zero coefficients in the representation, which is a more convenient way of controlling the sparsity of the approximation compared to tuning $\lambda$.

To evaluate the quality of our approach, we performed the following experiments. We took two sets of projections: Random ones $\{\mathbf{w}_i^{\mathbf{R}}\}$ and those obtained using our stepwise approach $\{\mathbf{w}_i^{\mathbf{S}} = D\mathbf{s}_i^0\}$. We then minimized Eq. (3) with Stochastic Gradient Descent using first $\{\mathbf{w}_i^{\mathbf{R}}\}$ and then $\{\mathbf{w}_i^{\mathbf{S}}\}$ as initializers. As a result we obtained two optimized sets of projections, $\{\mathbf{w}_i^{\mathbf{R-Opt}}\}$ and $\{\mathbf{w}_i^{\mathbf{S-Opt}}\}$ respectively. To make the comparison fair, we set the number of projections to 32 and tuned the parameters so that each projection was a linear combination of 64 columns of $D$. We then found the corresponding optimal thresholds and evaluated the resulting descriptors on all the test sets, using the setup of Section 3.3. We present here two representative ROC curves.

As Fig. 2 shows, optimizing over the random projections significantly improves the results. Nevertheless, the projections obtained using the stepwise initialization scheme we propose perform better even without optimization. Moreover, our extensive evaluation shows that applying a global optimization on the $\{\mathbf{w}_i^{\mathbf{S}}\}$ does not lead to any significant improvement and, hence, in the remainder of this paper we use our stepwise approach without further optimizing it.

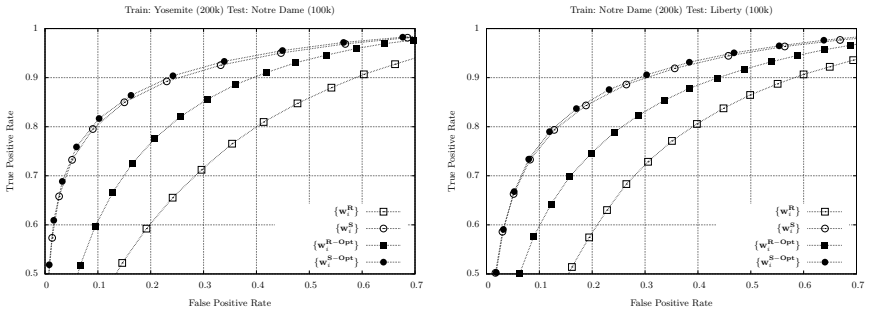



**Fig. 2.** Results obtained using Stochastic Gradient Descent applied to Eq. (3) and initialized with random projections or projections found with our stepwise approach. We used 32 projections and tuned the parameters so that each projection is a linear combination of 64 columns of BOX dictionary. We then found the corresponding optimal thresholds. Initializing gradient descent with the stepwise approach instead of random projections boosts the results significantly. Interestingly, optimization improves the quality of the projections only slightly while requiring additional processing time. Thus, to build D-Brief we use projections computed with the stepwise approach without further optimization.

### 3.2 Dictionaries

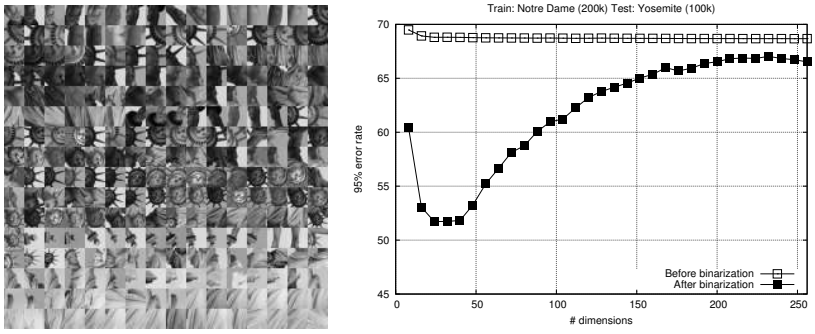
After finding a set of projections $\{\mathbf{w}_i = D\mathbf{s}_i\}$, they can be applied on an image patch $\mathbf{x}$ as:

$$\mathbf{w}_i^\top \mathbf{x} = (D\mathbf{s}_i)^\top \mathbf{x} = \sum_{j \text{ such that } \mathbf{s}_{ij} \neq 0} \mathbf{s}_{ij} D_j^\top \mathbf{x}\,, \tag{6}$$

where the $D_j$ are the columns of matrix $D$ and contain the dictionary elements. The dictionary in $D$ is designed so that the dot product $D_j^\top \mathbf{x}$ can be computed efficiently. In our experiments we use three different dictionaries that contain:

a) **Box filters (BOX)**: To create this dictionary we generate a set of box filters of size $5 \times 5$ that are centered at each coordinate of the image patch. Since our subsampled patches are of size $32 \times 32$, there are 1,024 elements in this dictionary.
b) **Gaussian filters (GAUSS)**: Similarly, we generate a set of Gaussian filters with $\sigma = 3$ centered at each coordinate of the image patch. The size of this dictionary is also 1,024.
c) **Rectangular filters (RECT)**: We create this dictionary by generating a set of rectangular filters of different sizes centered at each coordinate. We subsample the space of all possible rectangular filters by considering those whose horizontal or vertical edge is equal to $1, 4, 7, 10, \ldots$. The resulting dictionary size is 34,596.

At run-time, to compute the $D_j^\top \mathbf{x}$ values, we first convolve the image patch with a box filter or a Gaussian filter, or compute the integral image for the patch. All these operations can be done very efficiently. Then, the $D_j^\top \mathbf{x}$ can be obtained by reading a single value in the result of the convolution, or four values in the integral image in the case of the RECT dictionary.




(a) (b)

**Fig. 3. (a)** Some images patches from the *Liberty* dataset [6] used to train the discriminative projections. **(b)** 95% error rates for different numbers of LDE projections obtained from Eq. (4) and used for dimensionality reduction before and after applying the thresholds (the 95% error rate is the percent of incorrect matches obtained when 95% of the true matches are found). Without binarization, the error rates do not change significantly for different dimensionalities. After binarization, however, the dimensionality has much greater influence, likely because the projections are ranked by decreasing discriminative power, while the binarization gives them equal importance. The minimum of the plotted curve corresponding to the best performance is obtained for 32 dimensions.

### 3.3 Experiments

To train our projections, we use three publicly available dataset: *Liberty*, *Notre Dame* and *Yosemite* [6]. Each of them contains over 400k scale- and rotation-normalized $64 \times 64$ 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 datasets created this way exhibit substantial perspective distortion and various lighting conditions. Sample patches from the *Liberty* dataset are shown in Fig. 3(a). The ground truth available for each of those datasets describes 100k, 200k and 500k pairs of patches, where 50% correspond to match pairs, and 50% to non-match pairs. To avoid overfitting when training the LDE projections we apply a regularization method proposed in [6] with clipping parameter $\alpha = 0.01$. We tried different combinations of training and testing datasets, but as in [6], we found that choosing a specific combination does not have a strong influence on the final results.

As shown in Fig. 3(b) the best performances are obtained when using the first 32 projections. When a larger number of projections is used, the performances start deteriorating. This performance peak can be explained by the fact that the binarization step gives equal importance to all the projections, while the projections computed with LDE are ranked by decreasing discriminative power. Moreover, using the least ranked projections, which correspond to the dimensions of lower energy, introduces classification noise and deteriorates performance. We therefore keep the top 32 projections in the rest of the paper, and our descriptor is made of 32 bits.

We performed a set of experiments to qualitatively and quantitatively assess the influence of different parameters on Eq. (5). Fig. 4 shows how the dictionary type and the number of elements used influences the results on two sample LDE projections.

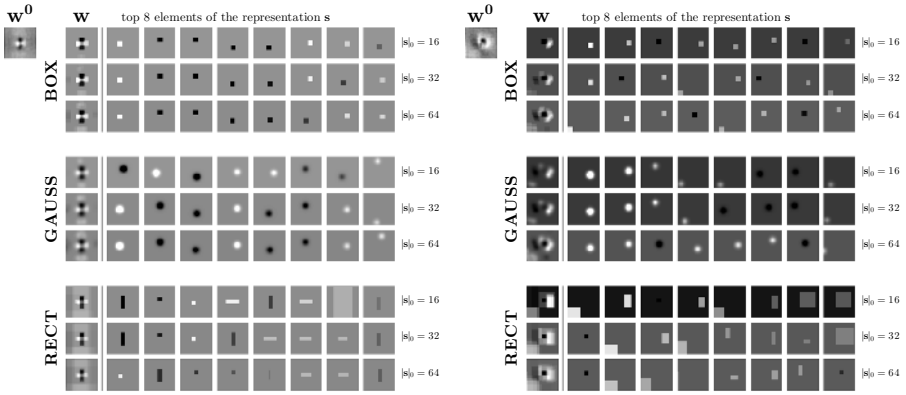
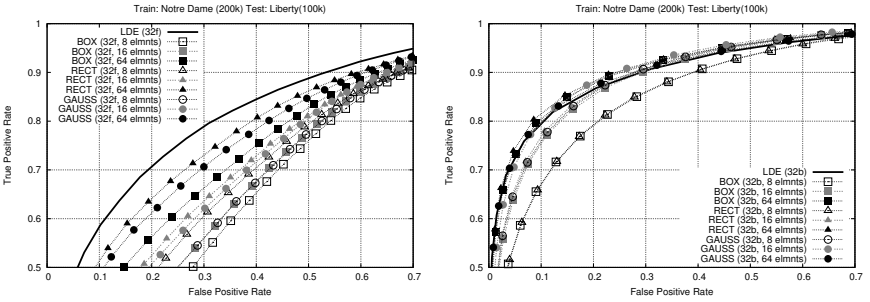



**Fig. 4.** Projections obtained by optimizing Eq. (5), from two LDE projections obtained with Eq. (4). We varied the dictionaries and the maximal numbers of dictionary elements per projection. The columns on the right show the first 8 elements of the sparse representation **s** which are best viewed on a monitor.

Fig. 5 shows the recognition rates for projections obtained by optimizing Eq. (5), from the top 32 LDE projections obtained with Eq. (4) with different dictionaries and various numbers of dictionary elements used. Before binarization, the RECT dictionary provides the best results, followed by the GAUSS and BOX dictionaries. While optimizing the projections with Eq. (5) hurts the

recognition performances before binarization, it is not true anymore after binarization. There is actually a minor improvement, which may come from the fact that the LDE projections are typically noisy and the optimization introduces some regularization which makes the thresholds generalize better. Surprisingly, the GAUSS dictionary performs better than the others after binarization for small numbers of elements, but then gets outperformed again by the RECT dictionary for large numbers of elements.




(a) Before binarization (b) After binarization

**Fig. 5.** Performances of the projections obtained by optimizing Eq. (5), from the top 32 LDE projections obtained with Eq. (4). We varied the dictionaries and the maximal numbers of dictionary elements per projection. We also give the results for the projections directly obtained with Eq. (4), referred to as LDE. In parentheses: Number of floating point (f) or binary (b) coordinates, and number of elements. Before binarization, the RECT dictionary provides the best approximation, followed by the GAUSS and BOX dictionary. While the approximation hurts the recognition performances before binarization, it is not true anymore after binarization. The minor improvement can be explained by the smoothing effect of the approximation applied on typically noisy LDE projections which makes the thresholds generalize better.

As explained in Section 3.1, to build the descriptors efficiently at run-time we first preprocess the image patch either by convolving it with a box or Gaussian filter or by computing the integral image. Then, the $D_j^\top \mathbf{x}$ values of Eq. (6) are obtained by reading either one or four values from the output of this preprocessing stage. Convolution with a box filter is faster than convolution with a Gaussian filter, and when using the RECT dictionary, we need to read four values for each $D_j^\top \mathbf{x}$ instead of only one as with the BOX and GAUSS dictionaries. The computation times are therefore different for each dictionary and Table 2 presents the times required for computing our descriptors using different dictionaries. With our approach, we can speed up the description time with respect to regular projections by over an order of magnitude.

Moreover, there is a trade-off between the accuracy and efficiency of the dictionaries: The slower one yields the best recognition performance, and *vice versa*. In practice, this means that we can adapt our method to the need of the final application.

In the remainder of this paper, we use projections learnt on the *Notre Dame* dataset, unless stated otherwise, and the RECT dictionary with 64 elements to

approximate the projections, since its performance is superior while the processing time still enables real-time applications, as shown in the next section.

As reported in [6], pre-normalizing the patches by subtracting the intensity mean value and dividing by the standard deviation and post-normalizing the descriptor to unit length improves the performance, in case of a real-valued descriptor. Inspired by these findings, we tried normalizing our patches and descriptors as in [6]. Fig. 6 confirms the observed performance improvement for real-valued coordinates. However, these steps do not have much influence on the results after binarization. This is because we train the thresholds to be discriminative after applying the projections and, hence, they adapt well to the light variations. Hence, we skip the normalization, as it entails additional computational overhead.

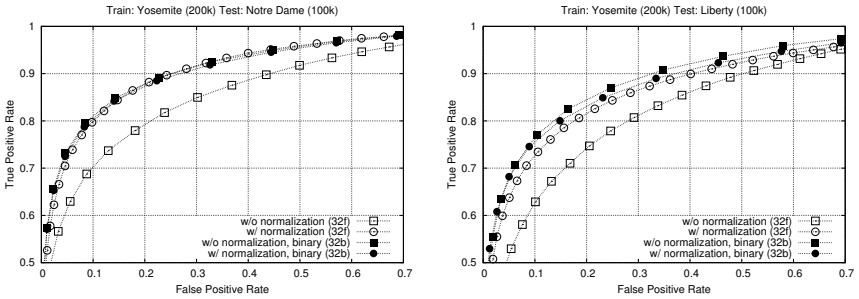



**Fig. 6.** Influence of normalization applied on the D-Brief descriptors before (left) and after (right) binarization. In parentheses: Number of floating point (f) or binary (b) coordinates. Normalization improves the performance only for the real-valued coordinates. The thresholds applied in the binarization step adapt to the discriminative subspace and, we can hence skip the normalization steps with no loss of accuracy.

## 4 Results

In this section, we first compare the performance of our approach to other descriptors including recent binary descriptors. We then confirm the generalization of our approach by providing an example of a real-time object detection application based on the compact binary descriptor we learnt.

### 4.1 Descriptor Comparison

We compare here the performance of our D-Brief descriptor against the state-of-the-art descriptors on the *Yosemite*, *Notre Dame*, and *Liberty* datasets from [6] and report the results in terms of ROC curves and 95% error rate as in [6]. We focus on fast binary descriptors, and consider the very recent BRIEF, BRISK, and ORB as they are the direct competitors to our approach. For reference, we also provide results obtained with SIFT, SURF, and a real-valued descriptor computed by applying LDE projections on bias-gain normalized patches. For this descriptor, which we refer to as LDE, we use the optimal number of projections found in [6]. We compute BRIEF and ORB using OpenCV implementations. For SIFT, we use the publicly available implementation of A. Vedaldi[2]. For BRISK

[2] `www.vlfeat.org/~vedaldi/code/siftpp.html`

and SURF, we use the implementations available on the websites of the authors, and for LDE, our own implementation. All the experiments were performed on a Macbook Pro with an Intel i7 2.66 GHz CPU.

Table 1 clearly shows that D-Brief provides up to 32% improvement over BRISK and up to 11% improvement over BRIEF in terms of 95% error rate, while requiring only 4 bytes instead of 64 for BRISK and 32 for BRIEF. It also shows that D-Brief remains competitive to the much longer and much more computationally expensive floating-point SURF.

**Table 1.** 95% error rates for different training and testing configurations and the corresponding results for BRIEF, BRISK and SURF (the results for ORB were identical to those of BRIEF, despite the fact that ORB optimizes its tests). As a baseline, we give the results for SIFT which is over 3 orders of magnitude slower to compute than our descriptor. For each configuration, we learnt the first 32 projections, approximated them using a dictionary of rectangular filters and learnt the thresholds. Below the descriptor names we write the number of bytes used to encode them. D-Brief outperforms its binary competitors, while requiring significantly less memory.

| Train | Test | D-Brief *4 bytes* | BRIEF\|ORB *32 bytes* | BRISK *64 bytes* | SURF *64 bytes* | SIFT *128 bytes* |
|---|---|---|---|---|---|---|
| Yosemite | Notre Dame | 43.96 | 54.57 | 74.88 | 45.51 | 28.09 |
| Yosemite | Liberty | 53.39 | 59.15 | 79.36 | 54.01 | 36.26 |
| Notre Dame | Yosemite | 46.22 | 54.96 | 73.21 | 43.57 | 29.14 |
| Notre Dame | Liberty | 51.30 | 59.15 | 79.36 | 54.01 | 36.26 |
| Liberty | Notre Dame | 43.10 | 54.57 | 74.88 | 45.51 | 28.09 |
| Liberty | Yosemite | 47.29 | 54.96 | 73.21 | 43.57 | 29.14 |

Table 2 shows that the performance improvement of D-Brief is not reached at the price of a loss of its efficiency. We extracted 3000 feature points using the SURF detector from the 1000×700 `wall1.ppm` image from Mikolajczyk dataset, and averaged the timings to compute the SURF, BRISK, BRIEF, LDE and D-Brief descriptors over 300 runs. Since both BRIEF and ORB are built using the same number of intensity comparisons, their timings are equal by definition. Our descriptor is 2-3 times faster to compute than BRISK and slightly faster than BRIEF when the BOX or GAUSS dictionaries are used. Also, its compact representation makes it much faster to match.

Table 2 also recalls the advantages of binary descriptors directly computed from the image. The description times for BRIEF and for D-Brief are over two orders of magnitude shorter than for SURF and LDE. The matching time for D-Brief is also much shorter than for SURF and LDE thanks to its binary nature.

Overall, we can build and match our descriptor over three times faster than BRIEF (when working on a CPU on which the `POPCOUNT` instruction is not available[3]) and almost two times faster when the `POPCOUNT` instruction is enabled.

[3] The `POPCOUNT` instruction computes the number of bits set to 1, and can be used after a bitwise `XOR` operation to compute the Hamming distance very efficiently.

**Table 2.** Description and matching timings per descriptor for SURF, LDE, BRISK, BRIEF, ORB and D-Brief with different dictionaries (BOX, GAUSS and RECT) and 64 elements to approximate the projections. Results computed for 3000 descriptors extracted from the `wall1.ppm` image of Mikolajczyk dataset and averaged over 300 runs. For SIFT, we use its detector, for the other descriptors we use the one of SURF. Exhaustive matching is performed. The subscripts of the descriptor names indicate its number of binary or floating-point coordinates. With D-Brief we can speed up the description and matching time by over an order of magnitude w.r.t. SURF and [6] and over three times w.r.t. BRIEF and ORB (without the `POPCOUNT` instruction).

| | | Description [$\mu$s] | Matching [$\mu$s] | **Total** [$\mu$s] |
|---|---|---|---|---|
| $\text{SIFT}_{128f}$ | | 2115.487 | 392.088 | 2507.575 |
| $\text{SURF}_{64f}$ | | 143.701 | 201.996 | 345.697 |
| LDE $[6]_{14f}$ | | 113.149 | 57.649 | 170.798 |
| without `POPCOUNT`: | | | | |
| $\text{BRISK}_{512b}$ | | 13.245 | 186.671 | 199.916 |
| $\text{BRIEF}\|\text{ORB}_{256b}$ | | 3.736 | 99.125 | 102.861 |
| $\text{D-Brief}_{32b}$ | *No approx.* | 80.694 | 26.510 | 107.204 |
| | BOX | 3.007 | | 29.517 |
| | GAUSS | 3.200 | | 29.710 |
| | RECT | 7.470 | | 33.980 |
| with `POPCOUNT`: | | | | |
| $\text{BRISK}_{512b}$ | | 13.245 | 15.770 | 29.015 |
| $\text{BRIEF}\|\text{ORB}_{256b}$ | | 3.736 | 7.045 | 10.781 |
| $\text{D-Brief}_{32b}$ | *No approx.* | 80.694 | 3.080 | 83.774 |
| | BOX | 3.007 | | 6.087 |
| | GAUSS | 3.200 | | 6.280 |
| | RECT | 7.470 | | 10.55 |

Moreover, while we could not test it on mobile devices, it is reasonable to expect that D-Brief presents a few more advantages on such platforms. The ARM processors, which they typically use, are 32-bit processors, and since our descriptor is made of 32 bits, it fits into a single register. This is very advantageous as many operations could be performed in one cycle.

Finally, Fig. 7 provides the ROC curves for two combinations of training and test sets. D-Brief performs better than its binary competitors at all error rates. D-Brief remains competitive to SURF, especially for the higher false positive rates, even though it has a much shorter representation. The results of D-Brief are comparable to those obtained with LDE. However, our descriptor is binary which enables further speed-up when computing the similarities, and can be computed much faster, as shown above. SIFT performs the best of all tested descriptors, though its complexity is prohibitive for real-time application. BRISK performs better than ORB and BRIEF at low True Positive rates and joins our descriptor at such rates, but it is much longer.

### 4.2 Real-time Application

To demonstrate the real-time performance of D-Brief we implemented a simple real-time application for planar object detection. The user can select the object

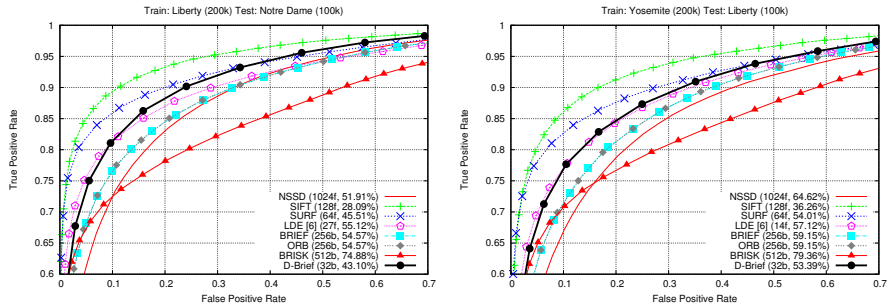



**Fig. 7.** Recognition rates for the floating point and binary descriptors for different training and testing datasets. In parentheses: The number of floating point (f) or binary (b) coordinates, and the 95% error rate. Our D-Brief descriptor outperforms all the other binary descriptors based on the intensity tests, namely BRIEF, ORB, and BRISK, while remaining competitive to SURF for higher false positive rates. It is also much shorter than all the other descriptors. The results obtained with LDE [6] applied to normalized patches are similar to those of D-Brief, but it is much more computationally expensive to compute and match (see Table 2).

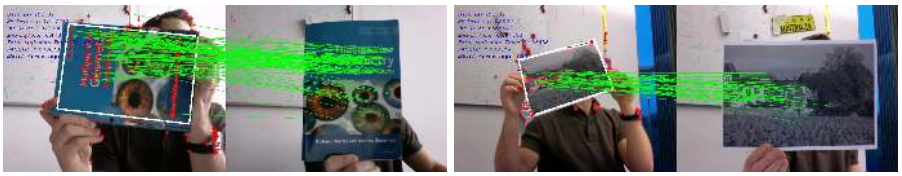


**Fig. 8.** Screenshots of the object detection application. The user first draws a rectangle around the target. The invariance to large scale changes and rotation is obtained by computing the feature point descriptors under different scales and orientations. This is performed on-the-fly and detecting the target runs in real-time with 27 frames per second.

of interest by drawing a rectangle around it in a reference view. The application then extracts feature points using FAST [21] and builds a database of D-Brief descriptors for these feature points in 18 rotated views at 3 scales, totaling up to 54 views. This is a simple way to make our descriptor invariant to scale and rotation changes, and was used for BRIEF in [3]. Alternatively, one could estimate the scale and orientation of the feature points, and compute the descriptors on the rectified patches as was done in ORB for example.

At run-time, for each input image, the application simply extracts feature points, computes their D-Brief descriptors, matches them against all the descriptors of the database, and finally computes the homography between the reference view and the input image using RANSAC. Some screenshots are shown in Fig. 8. One shall note that the projection matrix and thresholds of D-Brief are learnt on images from *Notre Dame* dataset, whose quality differs significantly from the quality of webcam images. Despite of that, the matching results are very consistent and the correct homography is easily found.

## 5 Conclusion

We presented a new method to learn discriminative projections which can be computed efficiently as a linear combination of a few simple filters from a given dictionary. This approach enables us to learn a compact and discriminative binary descriptor we call D-Brief. With only 32 bits per descriptor, D-Brief outperforms its binary state-of-the-art competitors in terms of accuracy and efficiency, while significantly reducing the memory footprint.

## References

1. Lowe, D.: Distinctive Image Features from Scale-Invariant Keypoints. IJCV 20, 91–110 (2004)
2. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: Speeded Up Robust Features. In: ECCV (2006)
3. Calonder, M., Lepetit, V., Ozuysal, M., Trzcinski, T., Strecha, C., Fua, P.: BRIEF: Computing a Local Binary Descriptor Very Fast. PAMI (2011)
4. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: an efficient alternative to SIFT or SURF. In: ICCV (2011)
5. Leutenegger, S., Chli, M., Siegwart, R.: BRISK: Binary Robust Invariant Scalable Keypoints. In: ICCV (2011)
6. Brown, M., Hua, G., Winder, S.: Discriminative Learning of Local Image Descriptors. PAMI 33, 43–57 (2011)
7. Strecha, C., Bronstein, A., Bronstein, M., Fua, P.: LDAHash: Improved Matching with Smaller Descriptors. PAMI 1, 66-78 (2012)
8. Cai, H., Mikolajczyk, K., Matas, J.: Learning Linear Discriminant Projections for Dimensionality Reduction of Image Descriptors. PAMI 33, 338-352 (2011)
9. Torralba, A., Fergus, R., Weiss, Y.: Small Codes and Large Databases for Recognition. In: CVPR (2008)
10. Chandrasekhar, V., Takacs, G., Chen, D., Tsai, S., Grzeszczuk, R., Girod, B.: CHoG: Compressed histogram of gradients A low bit-rate feature descriptor. In: CVPR (2009)
11. Jégou, H., Douze, M., Schmid, C.: Product quantization for nearest neighbor search. PAMI 33, 117–128 (2011)
12. Gionis, A., Indik, P., Motwani, R.: Similarity Search in High Dimensions via Hashing. In: International Conference on Very Large Databases (1999)
13. Weiss, Y., Torralba, A., Fergus, R.: Spectral Hashing. In: NIPS (2009)
14. Andoni, A., Indyk, P.: Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. Communications of the ACM (2008)
15. Gong, Y., Lazebnik, S.: Iterative Quantization: A Procrustean Approach to Learning Binary Codes. In: CVPR (2011)
16. Shakhnarovich, G.: Learning Task-Specific Similarity. PhD thesis (2005)
17. Wang, J., Kumar, S., S.-F.Chang: Semi-Supervised Hashing for Scalable Image Retrieval. In: CVPR (2010)
18. Bergamo, A., Torresani, L., Fitzgibbon, A.: Picodes: Learning a compact code for novel-category recognition. In: NIPS (2011)
19. Bach, F., Jenatton, R., Mairal, J., Obozienski, G.: Optimization with Sparsity-Inducing Penalties. Foundations and Trends in Machine Learning 4, 1-106 (2012)
20. Tibshirani, R.: Regression Shrinkage and Selection via the Lasso. Journal of the Royal Statistical Society (1996)
21. Rosten, E., Porter, R., Drummond, T.: Faster and Better: A Machine Learning Approach to Corner Detection. PAMI 32, 105–119 (2010)