ELSEVIER

BEBLID: Boosted Efficient Binary Local Image Descriptor

Iago Suárez[a,b], Ghesn Sfeir[a], José M. Buenaposada[c,**], Luis Baumela[a]

[a]*Departamento de Inteligencia Artificial. Universidad Politécnica de Madrid. Campus Montegancedo s/n. 28660 Boadilla del Monte, Spain*
[b]*The Graffter. Centro de Empresas UPM. Campus Montegancedo s/n. 28223 Pozuelo de Alarcón, Spain*
[c]*ETSII. Universidad Rey Juan Carlos. C/ Tulipán, s/n. 28933 Móstoles, Spain*

ABSTRACT

Efficient matching of local image features is a fundamental task in many computer vision applications. However, the real-time performance of top matching algorithms is compromised in computationally limited devices, such as mobile phones or drones, due to the simplicity of their hardware and their finite energy supply. In this paper we introduce BEBLID, an efficient learned binary image descriptor. It improves our previous real-valued descriptor, BELID, making it both more efficient for matching and more accurate. To this end we use AdaBoost with an improved weak-learner training scheme that produces better local descriptions. Further, we binarize our descriptor by forcing all weak-learners to have the same weight in the strong learner combination and train it in an unbalanced data set to address the asymmetries arising in matching and retrieval tasks. In our experiments BEBLID achieves an accuracy close to SIFT and better computational efficiency than ORB, the fastest algorithm in the literature.

1. Introduction

Local image representations are designed to match images in the presence of strong appearance variations, such as illumination changes or geometric transformations. They are a fundamental component of a wide range of Computer Vision tasks, including 3D reconstruction (Agarwal et al., 2009; Schonberger and Frahm, 2016), SLAM (Mur-Artal et al., 2015), image retrieval (Nister and Stewenius, 2006), tracking (Pernici and Del Bimbo, 2014), recognition (Lowe, 1999) or pose estimation (Wohlhart and Lepetit, 2015). They are the most popular image representation approach, because local features are distinctive, view point invariant, robust to partial occlusions and very efficient, since they discard low informative image areas.

To produce a local image representation we must detect a set of salient image structures and provide a description for each of them. There is a plethora of very efficient detectors for various low level structures such as corners (Rosten and Drummond, 2006), segments (Von Gioi et al., 2010), lines (Suarez et al., 2018) and regions (Matas et al., 2002), that may be described

by real valued (Lowe, 2004; Bay et al., 2006) or binary (Calonder et al., 2010; Rublee et al., 2011; Alahi et al., 2012; Balntas et al., 2015; Levi and Hassner, 2016; Leutenegger et al., 2011) descriptors, being the binary ones the fastest to extract and match. In this paper we address the problem of efficient binary feature description.

Although the SIFT descriptor was introduced twenty years ago (Lowe, 1999, 2004), it is still considered the "gold standard" technique. The recent HPatches benchmark has shown, however, that there is still a lot of room for improvement (Balntas et al., 2017). Modern descriptors based on deep models have boosted the mean Average Precision (mAP) in different tasks (Balntas et al., 2017) at the price of a sharp increase in computational requirements. This prevents their use in hardware and battery limited devices such as smartphones, drones or robots. This problem has been studied extensively and many local features detectors (Rosten and Drummond, 2006; Von Gioi et al., 2010; Rublee et al., 2011) and descriptors (Leutenegger et al., 2011; Calonder et al., 2010) have emerged, that enable real-time performance on resource limited devices, at the price of an accuracy significantly lower than SIFT.

We have recently introduced BELID (Suárez et al., 2019), an efficient real-valued descriptor. Our features use the integral image to efficiently compute the difference between the

---

**Corresponding author: Tel.: +0-000-000-0000; fax: +0-000-000-0000;
e-mail: josemiguel.buenaposada@urjc.es (José M. Buenaposada)

mean gray values in a pair of square image regions. In BELID we use the BoostedSSC algorithm (Shakhnarovich, 2005) to discriminatively select a set of features and combine them to produce a strong description. BELID achieves execution times close the fastest technique in the literature, ORB (Rublee et al., 2011), with an accuracy similar to that of SIFT. Specifically, it provides an accuracy better than SIFT in the patch verification and worse in the image matching and patch retrieval tasks of the HPatches benchmark (Balntas et al., 2017). Here we use AdaBoost to improve BELID's feature selection procedure and binarize its description.

In this paper we introduce BEBLID (Boosted Efficient Binary Local Image Descriptor), a very efficient binary local image descriptor. We use AdaBoost to train our new descriptor with an unbalanced data set to address the heavily asymmetric image matching problem. To binarize our descriptor we minimize a new similarity loss in which all weak learners share a common weight. In our experiments BEBLID beats both in terms of accuracy and speed ORB (Rublee et al., 2011), the fastest binary descriptor, BinBoost (Trzcinski et al., 2015) and LATCH (Levi and Hassner, 2016), the top performing binary descriptors among the non-Deep Learning literature.

2. Related work

SIFT is the most well-known feature detection and description algorithm (Lowe, 1999, 2004). It is widely used because it has a good performance in many Computer Vision tasks. However, it is computationally quite demanding requiring the use of a GPU to achieve real-time performance in certain contexts (Björkman et al., 2014).

A number of different descriptors, such as SURF (Bay et al., 2006), BRIEF (Calonder et al., 2010), BRISK (Leutenegger et al., 2011), ORB (Rublee et al., 2011), FREAK (Alahi et al., 2012), BOLD (Balntas et al., 2015) have emerged to speed up SIFT. Binary approaches produce a binary valued descriptor that is very efficient in terms of memory usage and matching speed. The fastest binary approaches, BRIEF, BRISK, FREAK, ORB, and BOLD, use features based on the comparison of pairs of image pixels. The key for their speed is the use of a limited number of comparisons selected to be uncorrelated with an unsupervised approach. BRIEF uses a fixed size (9×9) smoothing convolution kernel before comparing up to 512 randomly located pixel value pairs. BRISK uses a circular pattern, smoothing the pixel with a Gaussian of increasing variance the further the pixel is from the center of the pattern. FREAK chooses uncorrelated pixel pairs from a circular pattern, similar to BRISK, with overlapping Gaussians. The ORB descriptor is an extension of BRIEF that takes into account different orientations of the detected local feature. In this case the smoothing is done with an integral image with a fixed sub-window size. It uses a greedy algorithm to uncorrelate the chosen pixel pairs. BOLD uses pairwise comparisons estimated like ORB, from which it selects a set of patch adapted comparisons that decrease intra-patch distances. The main drawback of these approaches is that they trade accuracy for speed, performing significantly worse than SIFT.
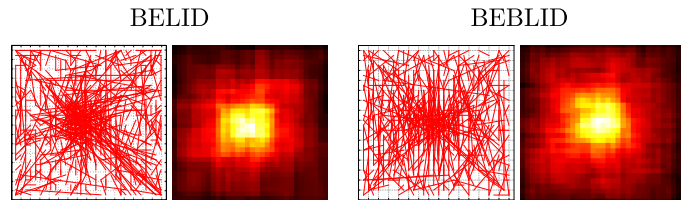
BELID

BEBLID

Fig. 1. Visualization of BELID and BEBLID pixel location sampling pairs (left) and spatial weight heat maps (right) trained on the Liberty patches data set. Both learn a well distributed set of point pairs giving more importance to the center area.

Descriptors based on supervised learning algorithms may further improve the performance. DAISY (Tola et al., 2008) learns pooling regions and how to perform dimensionality reduction. Simonyan et al. (2014) estimate these hyper-parameters with Convex Optimization, whereas Bin-Boost (Trzcinski et al., 2015) and BELID (Suárez et al., 2019) use Boosting. The LATCH descriptor (Levi and Hassner, 2016) compares the gray values in three regions selected to be uncorrelated and discriminative in the patch verification problem.

Deep Learning enables end-to-end supervised learning of descriptors. CNN-based methods are trained using pairs or triples of cropped patches. Some use Siamese nets (Han et al., 2015), L2 based loss and hard negative mining (Tian et al., 2017) or a modified triplet-based loss (Mishchuk et al., 2017). Other methods optimize a loss related to the Average Precision (He et al., 2018), an improved triplet loss to help focus on hard examples in training (Wei et al., 2018) or weigh triplets by their difficulty (Zhang and Rusinkiewicz, 2019). L2Net (Tian et al., 2017) is the most popular CNN descriptor architecture, which is also used in Hardnet (Mishchuk et al., 2017) and DOAP (He et al., 2018). Few Deep Learning methods address the problem of efficiency in the description. TFeat (Vassileios Balntas and Mikolajczyk, 2016) uses triplets in a very efficient way for training and a very shallow CNN for speed. All these methods have improved by a large margin the performance of SIFT in the HPatches benchmark. However, they are computationally more expensive. TFeat, one of the fastest Deep Learning-based descriptors, running in a GPU is 4× slower than ORB in a CPU (Balntas et al., 2017). A larger model, such as L2Net, running in a GPU is 15× times slower than ORB in a CPU (Tian et al., 2017).

In this paper we present BEBLID, a binary descriptor that uses a Boosting scheme to select the most discriminative intensity pairwise tests in a local image region (see Fig. 1). Like the fastest binary approaches, our features are based on differences of gray values. However, as in BELID, we compute the difference of the mean gray values in a box. The box size represents a scale parameter that improves the discrimination (Suárez et al., 2019). In BEBLID, similarly to BinBoost (Trzcinski et al., 2015), we search for the best features using a Boosting scheme. However, each bit in the description produced by BinBoost is a combination of gradient-based features, that are computationally more expensive than simple pairwise tests. In our experiments we prove that our simple and very efficient scaled intensity pairwise tests beat BinBoost's quantized gradient features both in terms of accuracy and speed.
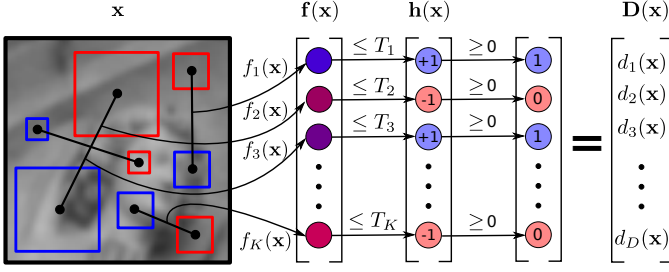
Fig. 2. BEBLID descriptor extraction workflow. To describe an image patch, BEBLID efficiently calculates the mean gray value of the pixels in the red and blue boxes. For each pair of red-blue boxes it subtracts their average values obtaining $\mathbf{f(x)}$, the WL. It then thresholds $\mathbf{f(x)}$ to obtain $\mathbf{h(x)}$ and the binary descriptor $\mathbf{D(x) = h(x) \geq 0}$.

3. Boosted Efficient Binary Local Image Descriptor

In this section we present our binary image descriptor, BE-BLID. To this end, we first introduce a real-valued descriptor based on AdaBoost (see Section 3.1). The use of AdaBoost in our weak learner (WL) selection strategy enables us to train with unbalanced data sets. This is further simplified into a binary descriptor when all WL share the same weights (see Section 3.3). The key for the efficiency of both descriptors lies in the use of a very efficient WL, based on thresholded pairwise tests computed on square patch regions of arbitrary size (see Section 3.2).

3.1. Real valued Boosted Efficient Local Image Descriptor

Let $\{(\mathbf{x}_i, \mathbf{y}_i, l_i)\}_{i=1}^{N}$ be a training set composed of pairs of image patches, $\mathbf{x}_i, \mathbf{y}_i \in \mathcal{X}$, and labels $l_i \in \{-1, 1\}$. Where $l_i = 1$ means that both patches correspond to the same salient image structure and $l_i = -1$ that they are different. We use AdaBoost to minimize the loss

$$\mathcal{L}_{BELID} = \sum_{i=1}^{N} \exp\left(-\gamma l_i \underbrace{\sum_{k=1}^{K} \alpha_k h_k(\mathbf{x}_i) h_k(\mathbf{y}_i)}_{g_s(\mathbf{x}_i, \mathbf{y}_i)}\right), \qquad (1)$$

where γ is the shrinkage or learning rate parameter and $h_k(\mathbf{z}) \equiv h_k(\mathbf{z}; f, T)$ corresponds to the k-th WL combined with weight α_k in the ensemble g_s. The WL depends on a feature extraction function $f : \mathcal{X} \to \mathbb{R}$ and a threshold T. Given f and T we define our WL by thresholding $f(\mathbf{x})$ with T,

$$h(\mathbf{x}; f, T) = \begin{cases} +1 & \text{if } f(\mathbf{x}) \leq T \\ -1 & \text{if } f(\mathbf{x}) > T \end{cases}. \qquad (2)$$

The loss in Eq. 1 can be seen as a similarity learning function given by g_s and $\mathbf{h(x)}$ is the vector of K WL responses for image patch $\mathbf{x}$. The descriptor of this patch is given by

$$\mathbf{D(x)} = \mathbf{A}^{\frac{1}{2}} \mathbf{h(x)} = [\sqrt{\alpha_1} \cdot h_1(\mathbf{x}), \dots, \sqrt{\alpha_K} \cdot h_K(\mathbf{x})]^{\top} \qquad (3)$$

where $\mathbf{A} = \text{diag}(\alpha_1, \dots, \alpha_k)$ and α_i is the AdaBoost weight for the i-th WL, $h_i(\mathbf{x})$. We denote this descriptor as BELID-U-ADA (Boosted Efficient Local Image Descriptor, Un-optimized, trained with AdaBoost) in contrast to BELID (Suárez et al., 2019) that learns a complete matrix $\mathbf{A}$ modeling the correlations among WLs (see Section 3.4).

3.2. Thresholded Average Box weak learner

The key for BEBLID's efficiency is selecting an $f(\mathbf{x})$ that is both discriminative and fast to compute. We define our feature extraction function, $f(\mathbf{x})$,

$$f(\mathbf{x}; \mathbf{p}_1, \mathbf{p}_2, s) = \frac{1}{s^2}\left(\sum_{\mathbf{q} \in R(\mathbf{p}_1, s)} I(\mathbf{q}) - \sum_{\mathbf{r} \in R(\mathbf{p}_2, s)} I(\mathbf{r})\right), \qquad (4)$$

where $I(\mathbf{t})$ is the gray value at pixel $\mathbf{t}$ and $R(\mathbf{p}, s)$ is the square box centered at pixel $\mathbf{p}$ with size s. Thus, f computes the difference between the mean gray values of the pixels in $R(\mathbf{p}_1, s)$ and $R(\mathbf{p}_2, s)$. The red and blue squares in Fig. 2 represent, respectively, $R(\mathbf{p}_2, s)$ and $R(\mathbf{p}_1, s)$.

On each AdaBoost iteration, we find the best WL by evaluating: 1) a fixed number, N_p, of pixel pairs (p_1, p_2); 2) all square regions of size s; and 3) all thresholds T. Inspired by BoostedSSC (Shakhnarovich, 2005) we have developed an efficient algorithm (see Alg. 1) to select the best discrete threshold for a given WL candidate without an exhaustive evaluation. The algorithm takes as input the responses of $f(\mathbf{x}; \mathbf{p}_1, \mathbf{p}_2, s)$ at each pair of patches and finds the threshold that minimizes the weighted classification error. The algorithm has $O(P \log P)$ $(P = 2N)$ complexity that derives from the sorting step in line 9, this allows us for a fast search over all possible thresholds.

Algorithm 1 ThresholdRate(P, f, W): Evaluation of projection thresholds given similarity-labeled examples.

Input: Set of labeled pairs $P = \{(\mathbf{x}_i, \mathbf{y}_i, l_i)\}_{i=1}^{N} \subset \mathcal{X} \times \{-1, 1\}$
Input: A feature extraction function $f : \mathcal{X} \to \mathbb{R}$
Input: Data weights $W = [w_1, \dots, w_N]$
Output: $\{(T_t, \epsilon_t)\}_{t=1}^{n}$, where ϵ_t is accuracy with threshold T_t.

1: Let $v_{i,1} := f(\mathbf{x}_i)$; $v_{i,2} := f(\mathbf{y}_i)$, $i = 1, \dots, N$
2: Let $u_1 < \dots < u_{n-1}$ be the $n-1$ unique values of $\lfloor v_{i,p} \rceil$
3: Let $\Delta_j := (u_{j+1} - u_j)/2$, $j = 1, \dots, n-2$
4: Let $T_1 := u_1 - \Delta_1$, and $T_{j+1} := u_j + \Delta_j$, $j = 1, \dots, n-1$
5: **for all** $i = 1, ..., N$ **do**
6:      Let $d_{i,1} := \begin{cases} -l_i w_i & \text{if } v_{i,1} \leq v_{i,2} \\ +l_i w_i & \text{if } v_{i,1} > v_{i,2} \end{cases}$
7:      Let $d_{i,2} := \begin{cases} -l_i w_i & \text{if } v_{i,1} > v_{i,2} \\ +l_i w_i & \text{if } v_{i,1} \leq v_{i,2} \end{cases}$
8: **end for**
9: { Sort in ascending order by $v_{i,p}$ value: }
10: $\{(v^{(k)}, d^{(k)})\}_{k=1}^{2N} \leftarrow sort\left(\{(v_{i,p}, d_{i,p})\}_{i=1,\dots,N, p=1,2}\right)$
11: $\epsilon_0 := \sum_{i=1}^{N} \mathbb{1}\{l_i = +1\} \cdot w_i$ { $\mathbb{1}$ is the indicator function }
12: t:=1
13: **for all** $j = 1, ..., t$ **do**
14:      $\epsilon_j := \epsilon_{j-1}$
15:      **while** $v^{(t)} \leq T_j$ **do**
16:          $\epsilon_j := \epsilon_j + d^{(t)}$
17:          t:=t+1
18:      **end while**
19: **end for**

To speed up the computation of f, we use S, the integral of the input image. Once S is available, the sum of gray levels in

a square box can be computed with 4 memory accesses and 3 arithmetic operations. To make our descriptor invariant to euclidean transformations, we orient and scale our measurements with the underlying local structure.

3.3. Binary descriptor learning

To obtain a binary description we optimize the loss

$$\mathcal{L}_{BEBLID} = \sum_{i=1}^{N} \exp\left(-\gamma l_i \sum_{k=1}^{K} h_k(\mathbf{x})h_k(\mathbf{y})\right), \quad (5)$$

where γ is the common WLs weight. In practice it acts as a shrinkage parameter that determines the training speed. Since we stop the training process if the algorithm is not able to find a WL better than random guessing, γ also determines the number of selected WLs.

Finally, to have a {0, 1} output, we convert the -1 output to 0 and the +1 output to 1 (see Fig. 2). This new binary descriptor is termed BEBLID, that stands for Boosted Efficient Binary Local Image Descriptor.

This is a Boosting scheme in which all WLs have the same contribution to the final strong decision. The intuition behind this scheme is the following. In an AdaBoost-based minimization, such as that used to obtain the BELID-U-ADA descriptor in Section 3.1, the contribution of each WL is weighted by α_k. This constant depends on the success of the k-th WL in solving a binary patch classification problem. However, we are interested on using our descriptor for solving many other image related problems. So, the α_ks are biased by the patch verification problem used to compute them. A descriptor in which all WLs have the same weight actually performs better in other tasks, such as for example image matching and retrieval. In our experiments we prove that this intuition is correct.

3.4. BELID, BELID-U and BELID-U-ADA

In our previous work (Suárez et al., 2019) we used Boost-edSSC (Shakhnarovich, 2005) to compute the BELID-U descriptor by minimizing Eq. 1. As described in Section 3.1, in this paper we also optimize it with AdaBoost to produce BELID-U-ADA.

Further, estimating the whole matrix $\mathbf{A}$ improves the similarity by modeling the correlation between WLs. FP-Boost (Trzcinski et al., 2015) estimates a symmetric $\mathbf{A}$ minimizing

$$\mathcal{L}_{FP} = \sum_{i=1}^{N} \exp\left(-l_i \mathbf{h}(\mathbf{x})^\top \mathbf{A} \mathbf{h}(\mathbf{y})\right) \quad (6)$$

with Stochastic Gradient Descent.

BELID (Boosted Efficient Local Image Descriptor) (Suárez et al., 2019) describes an image patch $\mathbf{x}$ as $\mathbf{D}(\mathbf{x}) = \mathbf{B}^\top \mathbf{h}(\mathbf{x})$, where $\mathbf{B} = [\mathbf{b}_1, \cdots, \mathbf{b}_D], \mathbf{b} \in \mathbb{R}^K$ are the eigenvectors associated to the D largest eigenvalues of matrix $\mathbf{A}$, estimated with FP-Boost.

4. Experiments

In our experiments we train our models with the popular Brown data set (Winder and Brown, 2007)[1]. It contains 64×64 SIFT detected and cropped gray level image patches from three different scenes: Notre Dame cathedral, Yosemite National Park and Liberty statue in New York.

We evaluate our results with the HPatches benchmark (Balntas et al., 2017). It provides patches extracted from images of various scenes under different capturing conditions and tested in verification, matching and retrieval tasks. The set of images are organized in 6 splits: "a", "b", "c", "illum", "view", and "full". In the experiments of this paper we use the "full" split that contains all the scenes in the dataset, whereas in Suárez et al. (2019) we tested our models in the "a" split.

We evaluate the performance using three measures:

- **FPR-95**. False Positive Rate at 95% recall in a patch verification problem.

- **AUC**. Area Under the ROC Curve in a patch verification problem. It is a good global measure since it considers all the curve operation points.

- **mAP**. Mean Average Precision, as defined in the HPatches benchmark, for each of the three tasks: patch verification, image matching and patch retrieval.

We have implemented AdaBoost and the learning and testing part of the Thresholded Average Box WL in Python. Using OpenCV 4.1.0 we have also implemented a C++ version[2] of our descriptor extraction algorithms. We use this implementation to evaluate their execution time in Section 4.5.

In all our experiments we train our models with the Liberty Statue patches scaled to 32×32 pixels. The size values in the Average Box WL are $\mathcal{S} = \{3, 5, 7, 9, 11, 13, 15\}$, and its location constrained to fall inside the image. We quantize $f(\mathbf{x})$ to an integer, to reduce the set $\mathcal{T}$ of WL thresholds.

4.1. AdaBoost vs. BoostedSSC

In the first experiment we compare AdaBoost with Boost-edSSC in verification and evaluate the relevance of selecting a good WL. First we train our model with BoostedSSC, a BELID-U descriptor as in Suárez et al. (2019). Then we train three versions of the BELID-U-ADA descriptor:

- **BELID-U-ADA-Rand**. In each AdaBoost iteration we use 500 candidate WLs randomly selecting location (p_1, p_2), scale s from $\mathcal{S}$ and threshold T from $\mathcal{T}$.

- **BELID-U-ADA**. In each AdaBoost iteration we randomly select $N_p = 500$ candidate pixel locations (p_1, p_2). Then we exhaustively evaluate all scales $s \in \mathcal{S}$ and all thresholds $t \in \mathcal{T}$ for each (p_1, p_2) pair.

---

[1] http://matthewalunbrown.com/patchdata/patchdata.html
[2] The C++ code with the pre-trained descriptors BEBLID-256-M and BEBLID-512-M (explained in section 4.4) has been made public in https://github.com/iago-suarez/BEBLID
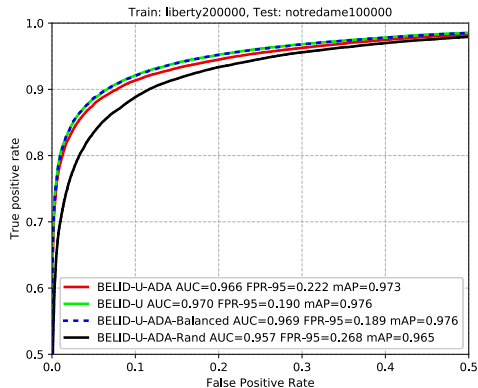
Fig. 3. ROC Curve for the verification task in the Brown data sets. We compare BoostedSSC with AdaBoost selecting in each iteration the best WL of a random selection (BELID-U-ADA-Rand), or by exhaustively searching for some of the WL parameters (BELID-U-ADA), or further normalizing the weights of the positive and negative classes.

- **BELID-U-ADA-Balanced**. Same as in BELID-U-ADA, but in this case we normalize the data weights to sum 0.5 for the negative and positive classes.

In Fig. 3 we show the results for the descriptors trained on a balanced data set of 200K patches pairs from the Liberty scene. We test the methods in a balanced data set of 100K patches pairs from the Notredame scene. We observe a reduction in FPR-95 from 26.8%, with BELID-U-ADA-Rand, to 22.2%, with BELID-U-ADA. Since the only difference between both algorithms is the exhaustive search along the scale, s, and threshold, T, parameters in the BELID-U-ADA approach, then we can infer that to achieve top performance it is important to search for good WLs. This justifies the use of Alg. 1 to speed up the optimal threshold search. Also, the performance of BELID-U-ADA-Balanced is equivalent to that of BELID-U, based on BoostedSSC. Hence we experimentally prove that BoostedSSC is just AdaBoost with the assumption of equal priors for positive and negative classes. Finally, an equal prior algorithm, BELID-U-ADA-Balanced, marginally improves the performance of BELID-U-ADA in this balanced verification problem. This is a first hint of the importance of selecting the appropriate priors when training the descriptor. In the next section we analyze this in more detail.

4.2. Asymmetric training

Here we exploit the fact that our key problems, matching and retrieval, are asymmetric. We evaluate the performance of descriptors trained with AdaBoost using unbalanced data sets from the Liberty scene. In our experiments we fix the number of training data to 1 million. To train with a data set with X% positives, we first randomly select X*10k positive samples. Then randomly generate negative pairs up to 1 million. In Table 1 we show the results for BELID-U-ADA and BEBLID descriptors with 512 components. In Table 2 we provide the learning rates used when training these descriptors.

The test set for the HPatches verification problem consists of 200K positives and 1 million negative examples (see "Verification" results in Table 1). However, we have also added results
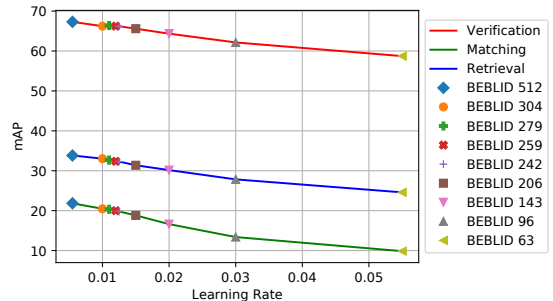

Fig. 4. BEBLID learning rate selection experiment. We show the mAP for verification, matching and retrieval in the "full" split of Hpatches for models trained with different learning rates, γ, in the Liberty data set.

for a fully balanced test set (see "Verification-balanced" in Table 1).

The BELID-U-ADA results in Table 1 for the verification-balanced problem provide the best AUC=85.59% with the balanced training data set. We get the best result for matching, mAP=20.11%, with the most unbalanced set (5% positives). In the retrieval problem we also get the best results with the most unbalanced training set. With BEBLID we have similar results. We get the best descriptor for the verification-balanced with the balanced training set, AUC=85.52%. In matching and retrieval we also get the best result with an unbalanced training. However, in this case, the results with 5% positives are worse than those with 20%. We speculate this is due to the small number of positives, 50K pairs in this case, that is scarce for training a BEBLID descriptor. We have experienced the same problem with 1% positives.

In summary, dealing with the asymmetry in the target problem is fundamental to improve the accuracy of image descriptors. Specifically, matching and retrieval tasks on one side, and verification on the other, need different descriptors. Here we have considered the use of AdaBoost trained with unbalanced data sets to address this issue.

4.3. Tuning BEBLID learning rate

In this experiment we use a training set from Liberty with 20% positives, selected as described in section 4.2. We train BEBLID with different learning rates, γ. In Fig. 4 we show the accuracy results in HPatches. We also display the number of bits of the resultant descriptor. As expected, the larger the learning rate the lower the number of iterations and bits of the descriptor. To get the desired number of bits (=WLs), K, we select a small enough value for γ and keep the first K WLs, that are also the most significant ones. We get the best results with 512 WLs and $\gamma = 0.0055$.

4.4. Comparison with the state-of-the-art

In this section we compare our binary descriptor with the most relevant approaches in the literature. Fig. 5 shows the results of various BEBLID configurations and those of other competitors. Here we compare our binary descriptor trained for the balanced verification problem ("V" suffix) and for the matching problem ("M" suffix) with ORB, BRISK, FREAK, LATCH, BinBoost, BELID and SIFT. We train descriptors with suffixes

Table 1. Results in the "full" split of HPatches when training with different ratios of positive samples (50%, 20% and 5%) from the Liberty data set.

	Verification - balanced (AUC)			Verification (mAP)			Matching (mAP)			Retrieval (mAP)		
	50%	20%	5%	50%	20%	5%	50%	20%	5%	50%	20%	5%
BELID-U-ADA (512f)	85.59	85.41	84.33%	67.41%	67.34%	66.17%	17.89%	18.73%	20.11%	30.00%	30.79%	32.22%
BEBLID (512b)	85.52	84.97	84.44	67.14%	67.31%	66.52%	17.44%	21.84%	21.69%	29.87%	33.82%	33.74%

Table 2. Learning rates for the descriptors in Table 1.

	50%	20%	5%
BELID-U-ADA (512f)	0.1	0.1	0.4
BEBLID (512b)	0.0055	0.0055	0.0025

"M" and "V" with 20% and 50% positives respectively (see section 4.2). We do not display results for BOLD and BRIEF since they are respectively worse than BinBoost and ORB (Balntas et al., 2017). We use the OpenCV implementation of BRISK, FREAK and LATCH. The results of SIFT, ORB and BinBoost come from the HPatches benchmark database.

In the HPatches verification testing set, with 16.66% positives (200K positives, 1M negatives), all boosting-based descriptors (BELID, BEBLID, BinBoost) are better than SIFT (mAP=65.12%) while LATCH, ORB, FREAK are worse. The real-valued BELID descriptors (Suárez et al., 2019), trained in the balanced Liberty data set, get the best results among non-CNN descriptors. The performance of BEBLID is behind that of BELID because of the binarization and because it does not take into account the correlations between WLs. The balanced version of our new binary descriptor, BEBLID-512-V, is marginally behind BEBLID-512-M because of the unbalanced testing set. Moreover, BEBLID-512-M, with mAP=67.31%, and all other versions of BEBLID are better than BinBoost, the best binary descriptor, and SIFT. This is remarkable since both use gradient based features whereas BEBLID uses simple average gray level differences. This is not surprising, however, since gray level differences for different box sizes is an approximation to the gradient at different scales.

Our best binary descriptor in the matching problem is BEBLID-512-M. Trained with unbalance data it gets mAP=21.84%, which is worse than SIFT, mAP=25.44% but, as we will see in the next section, it is two orders of magnitude faster. BRISK and FREAK are the worse descriptors both in image matching and patch retrieval. In the former problem BinBoost also shows poor accuracy, mAP=14.73%. Here the main difference with the patch verification problem is the asymmetry of the matching problem. Two key differences between BEBLID-512-M and BinBoost are that we use unbalanced training and a simpler ensemble with common weights. LATCH gets better results in matching than FREAK, BRISK, BinBoost and ORB. However, in patch retrieval, it is worse than BinBoost. BEBLID-512-M beats all its binary competitors both in the image matching and patch retrieval problems. This result validates our decisions in Section 3.3.

The number of bits used by a binary descriptor is important. When we halve the number of bits, from 512 to 256, the performance of BEBLID-M drops 1.07 in verification, 1.94 in matching and 1.53 in retrieval. Something similar happens with BEBLID-V. Here BEBLID-256-M and BEBLID-256-V

are comparable with ORB and BinBoost, since all of them use 256 bits. In this case we beat both descriptors in matching. In patch retrieval, BEBLID-256-M is marginally worse than BinBoost. However, we get marginally better results than BinBoost using 512 simple WLs (BEBLID-512-M) while BinBoost uses gradient based WLs. In the next section we will see that this is an important drawback for BinBoost in terms of efficiency.

We have added to Fig. 5 Hardnet (Mishchuk et al., 2017), a representative CNN-based descriptor. Hardnet beats by a large margin all handcrafted and learned descriptors, but it has much higher computational requirements.

In summary, we have shown that with our approach we get the best accuracy among non CNN-based binary descriptors in the verification, matching and retrieval problems. This is due to the two key ideas: WLs based on thresholded and scaled pairwise comparisons, and the adaptation of the training process to the level of asymmetry of the problem.

4.5. Execution time in different platforms

In the last experiment we test the C++ implementation of BEBLID processing full images (not cropped patches) in a desktop CPU, Intel Core i7-8750H, and in two limited CPUs, Exynox Octa 7870 and Snapdragon 855. We report the description execution time in the Mikolajczyk and Schmid (2005) data set, composed by 48 800×640 images from 8 different scenes. In each of them we detect a maximum of 2000 local structures with SURF. In this case we use the implementation of BinBoost in OpenCV, $BinBoost_{32}$-256, with a descriptor of 256 bits and 32 gradient based WLs per bit, with 8192 WLs evaluated per descriptor.

In Table 3 we show the average execution time per image and the size of each descriptor in terms of the number of components, that can be floating point numbers (f) or bits (b). We compare the execution time with C++ implementations of BELID (Suárez et al., 2019) and other relevant descriptors in the OpenCV library: SIFT (Lowe, 2004), ORB (Rublee et al., 2011), BRISK (Leutenegger et al., 2011), FREAK (Alahi et al., 2012), LATCH (Levi and Hassner, 2016), BinBoost (Trzcinski et al., 2015).

On average, it takes 0.21 ms in a desktop CPU and 0.64 in a smartphone for the most accurate BEBLID implementation, with 512 bits, to process a 800×640 image with 2000 keypoints. This is roughly 20× faster than LATCH, the most recent competing binary descriptor. The 256 bits implementation is slightly less accurate, but roughly 2× faster. This means that our implementation of BEBLID with 256 bits is 4× faster than OpenCV's ORB, the fastest binary descriptor in the literature, and 50× faster than BinBoost, the best binary descriptor in terms of accuracy. Compared with other competing floating point approaches, BEBLID 512b is as fast as the 512f version of BELID-U-ADA and more than 20× faster than BELID

In this case we use the implementation of BinBoost in OpenCV, BinBoost$_{32}$-256, with a descriptor of 256 bits and 32 gradient based WLs per bit, with 8192 WLs evaluated per descriptor.
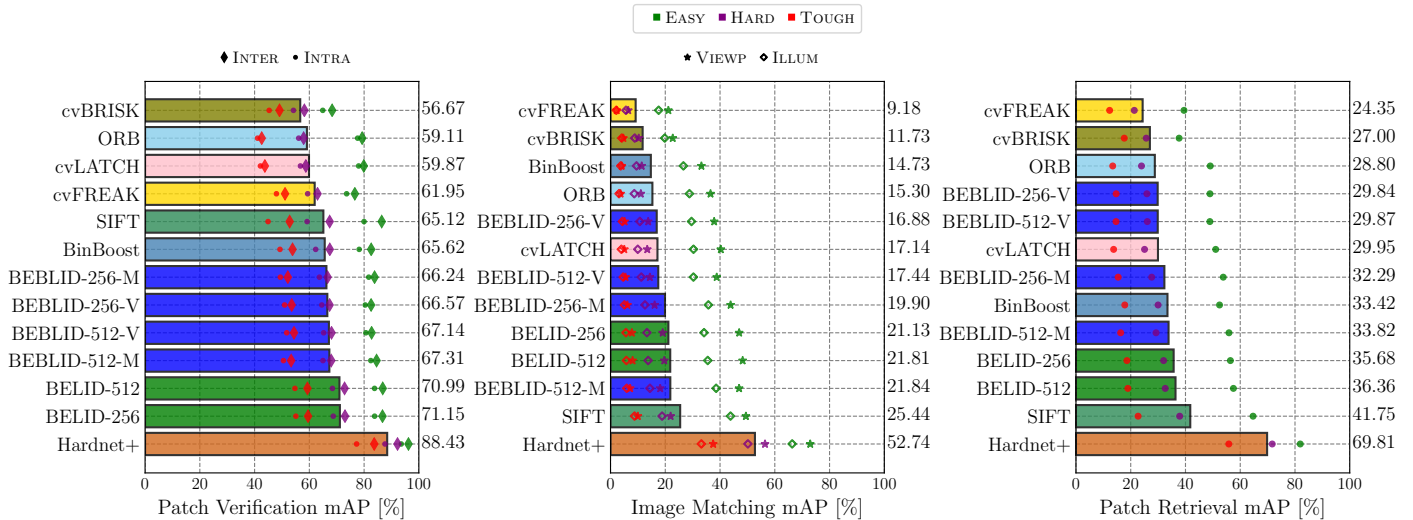
$\mathbb{H}$Patches Results

Fig. 5. Comparison of the state-of-the-art descriptors in the "full" split of the HPatches data set. The marker color indicates the noise level: EASY, HARD, and TOUGH. INTRA makers in Patch Verification task corresponds to Patch pairs obtained from the same sequence whereas INTER markers are from different ones. In the Image Matching task, the VIEWP markers refer to scenes with viewpoint distortions and the ILLUM to scenes with illumination changes. The bar length represents the mean of the six variants of each task.

512f, the comparable floating point descriptors. BEBLID 256b is roughly two orders of magnitude faster than SIFT.

The key for the computational efficiency of BELID and BEBLID lies in the use of very efficient WLs based on pairwise comparisons computed on the integral image. For this reason BEBLID computational requirements should be similar to those of ORB. The differences are caused by the fact that we extract our features in parallel, whereas the present ORB implementation in OpenCV does not. BELID is less efficient than BEBLID because it requires an extra multiplication of the WLs measurements, $\mathbf{h}(\mathbf{w})$, with matrix $\mathbf{B}$. However, BEBLID is as efficient as BELID-U-ADA, since in it $\mathbf{B}$ is the identity.

From the results in this section we can conclude that BEBLID is the most efficient binary descriptor in the literature. Our new binary descriptor is the best compromise between mAP and speed. These results support the claim that our descriptor is a faster alternative to SIFT that is able to run in real-time on low performance devices.

5. Conclusion

In this paper we introduce BEBLID, the best non CNN-based binary descriptor in the state of the art in terms of accuracy and the most efficient in terms of computational requirements. In our experiments we proved that it is faster than the popular OpenCV implementation of ORB, the fastest descriptor in the literature. This is due to the use of very efficient image features, based on gray value differences computed with the integral image. In terms of accuracy BEBLID is better than BinBoost, the best binary descriptor in the literature, and close to SIFT, the "gold standard" reference. This is due to the discriminative scheme used to select the image features and the possibility of learning the feature scale, represented by the feature box

Table 3. Average description time per image, in milliseconds, of various descriptors in three platforms, two of them power-limited (Exynox Octa S and Snapdragon 855). The column "Size" reports the descriptor size in floating-point (f) or binary (b) values.

	Size	Intel Core i7 8750H	Exynox Octa S	Snapdragon 855
SIFT	128f	14.29	152.30	53.34
ORB	256b	0.45	5.49	1.22
BRISK	512b	0.92	8.27	1.92
FREAK	512b	0.47	4.70	1.25
LATCH	512b	5.21	62.78	8.33
BinBoost$_{32}$-256	256b	6.55	52.63	12.82
BELID	512f	5.46	40.70	13.95
BELID	256f	2.83	21.46	7.26
BELID-U-ADA	512f	0.25	2.27	0.69
BEBLID	512b	0.21	2.09	0.64
BEBLID	256b	**0.11**	**1.32**	**0.42**

size. Furthermore, we provide different BEBLID descriptors trained with unbalanced data sets, to model the asymmetry in the matching and retrieval problems, which significantly improves the evaluation results.

As discussed in the introduction, feature matching is required in many other higher level computer vision tasks. In most of them it is a mid-level process often followed by model fitting, e.g. RANSAC. This robust fitting step fixes the errors occurred in the matching procedure. This is possibly one of the reasons why SIFT is still the most widely used descriptor. Although SIFT is not the best performing approach in terms of accuracy, it provides a reasonable trade-off between accuracy and computational requirements. In the context of real-time performance on computationally limited devices, BEBLID represents the best trade-off as it is faster than ORB with an accuracy close to that of SIFT.

Acknowledgments

References

Agarwal, S., Snavely, N., Simon, I., Seitz, S.M., Szeliski, R., 2009. Building Rome in a day, in: Proc. of International Conference on Computer Vision, IEEE. pp. 72–79.

Alahi, A., Ortiz, R., Vandergheynst, P., 2012. FREAK: Fast retina keypoint, in: Proc. Conference on Computer Vision and Pattern Recognition, IEEE. pp. 510–517.

Balntas, V., Lenc, K., Vedaldi, A., Mikolajczyk, K., 2017. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors, in: Proc. Conference on Computer Vision and Pattern Recognition, pp. 5173–5182.

Balntas, V., Tang, L., Mikolajczyk, K., 2015. BOLD - Binary online learned descriptor for efficient image matching, in: Proc. Conference on Computer Vision and Pattern Recognition, pp. 2367–2375.

Bay, H., Tuytelaars, T., Van Gool, L., 2006. SURF: Speeded up robust features, in: Proc. European Conference on Computer Vision, Springer. pp. 404–417.

Björkman, M., Bergström, N., Kragic, D., 2014. Detecting, segmenting and tracking unknown objects using multi-label MRF inference. Computer Vision and Image Understanding 118, 111 – 127.

Calonder, M., Lepetit, V., Strecha, C., Fua, P., 2010. BRIEF: Binary robust independent elementary features, in: Proc. European Conference on Computer Vision, Springer. pp. 778–792.

Han, X., Leung, T., Jia, Y., Sukthankar, R., Berg, A.C., 2015. MatchNet: Unifying feature and metric learning for patch-based matching, in: Proc. Conference on Computer Vision and Pattern Recognition, pp. 3279–3286.

He, K., Lu, Y., Sclaroff, S., 2018. Local descriptors optimized for average precision, in: Proc. Conference on Computer Vision and Pattern Recognition, pp. 596–605.

Leutenegger, S., Chli, M., Siegwart, R., 2011. BRISK: Binary robust invariant scalable keypoints, in: Proc. of International Conference on Computer Vision, IEEE. pp. 2548–2555.

Levi, G., Hassner, T., 2016. LATCH: Learned arrangements of three patch codes, in: IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE. pp. 1–9.

Lowe, D.G., 1999. Object recognition from local scale-invariant features, in: Proc. of International Conference on Computer Vision, IEEE. pp. 1150–1157.

Lowe, D.G., 2004. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60, 91–110.

Matas, J., Chum, O., Urban, M., Pajdla, T., 2002. Robust wide baseline stereo from maximally stable extremal regions, in: Proc. British Machine Vision Conference, pp. 36.1–36.10.

Mikolajczyk, K., Schmid, C., 2005. A performance evaluation of local descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence 27, 1615–1630.

Mishchuk, A., Mishkin, D., Radenovic, F., Matas, J., 2017. Working hard to know your neighbor's margins: Local descriptor learning loss, in: Advances in Neural Information Processing Systems, pp. 4826–4837.

Mur-Artal, R., Montiel, J.M.M., Tardós, J.D., 2015. ORB-SLAM: A versatile and accurate monocular SLAM system. IEEE Transactions on Robotics 31, 1147–1163.

Nister, D., Stewenius, H., 2006. Scalable recognition with a vocabulary tree, in: Proc. Conference on Computer Vision and Pattern Recognition, pp. 2161–2168.

Pernici, F., Del Bimbo, A., 2014. Object tracking by oversampling local features. IEEE Transactions on Pattern Analysis and Machine Intelligence 36, 2538–2551.

Rosten, E., Drummond, T., 2006. Machine learning for high-speed corner detection, in: Proc. European Conference on Computer Vision, Springer. pp. 430–443.

Rublee, E., Rabaud, V., Konolige, K., Bradski, G., 2011. ORB: An efficient alternative to SIFT or SURF, in: Proc. of International Conference on Computer Vision, pp. 2564–2571.

Schonberger, J.L., Frahm, J.M., 2016. Structure-from-motion revisited, in: Proc. Conference on Computer Vision and Pattern Recognition, pp. 4104–4113.

Shakhnarovich, G., 2005. Learning Task-Specific Similarity. Ph.D. thesis. Massachusetts Institute of Technology.

Simonyan, K., Vedaldi, A., Zisserman, A., 2014. Learning local feature descriptors using convex optimisation. IEEE Transactions on Pattern Analysis and Machine Intelligence 36, 1573–1585.

Suarez, I., Muñoz, E., Buenaposada, J.M., Baumela, L., 2018. FSG: A statistical approach to line detection via fast segments grouping, in: Proc. of Int. Conf. on Intell. Robots Systems, pp. 97–102.

Suárez, I., Sfeir, G., Buenaposada, J.M., Baumela, L., 2019. BELID: Boosted efficient local image descriptor, in: Proc. of Iberian Conference on Pattern Recognition and Image Analysis, Springer International Publishing, Cham. pp. 449–460.

Tian, Y., Fan, B., Wu, F., 2017. L2-Net: Deep learning of discriminative patch descriptor in euclidean space, in: Proc. Conference on Computer Vision and Pattern Recognition, pp. 6128–6136.

Tola, E., Lepetit, V., Fua, P., 2008. A fast local descriptor for dense matching, in: Proc. Conference on Computer Vision and Pattern Recognition, IEEE. pp. 1–8.

Trzcinski, T., Christoudias, M., Lepetit, V., 2015. Learning image descriptors with boosting. IEEE Transactions on Pattern Analysis and Machine Intelligence 37, 597–610.

Vassileios Balntas, Edgar Riba, D.P., Mikolajczyk, K., 2016. Learning local feature descriptors with triplets and shallow convolutional neural networks, in: Proc. British Machine Vision Conference, pp. 119.1–119.11.

Von Gioi, R.G., Jakubowicz, J., Morel, J.M., Randall, G., 2010. LSD: A fast line segment detector with a false detection control. IEEE Transactions on Pattern Analysis and Machine Intelligence 32, 722–732.

Wei, X., Zhang, Y., Gong, Y., Zheng, N., 2018. Kernelized subspace pooling for deep local descriptors, in: Proc. Conference on Computer Vision and Pattern Recognition, pp. 1867–1875.

Winder, S.A., Brown, M., 2007. Learning local image descriptors, in: Proc. Conference on Computer Vision and Pattern Recognition, IEEE. pp. 1–8.

Wohlhart, P., Lepetit, V., 2015. Learning descriptors for object recognition and 3D pose estimation, in: Proc. Conference on Computer Vision and Pattern Recognition, pp. 3109–3118.

Zhang, L., Rusinkiewicz, S., 2019. Learning local descriptors with a CDF-Based dynamic soft margin, in: Proc. of International Conference on Computer Vision.