BELID: Boosted Efficient Local Image Descriptor

Iago Suárez[1,2][0000−0003−4006−4378], Ghesn Sfeir[1][0000−0002−6600−9409], José M. Buenaposada[3][0000−0002−4308−9653], and Luis Baumela[1][0000−0001−6910−4359]

[1] Departamento de Inteligencia Artificial. Universidad Politécnica de Madrid. Campus Montegancedo s/n. 28660 Boadilla del Monte, Spain
lbaumela@fi.upm.es, g.sfeir@alumnos.upm.es
[2] The Graffter. Campus Montegancedo s/n. Centro de Empresas UPM. 28223 Pozuelo de Alarcón, Spain
iago.suarez@thegraffter.com
[3] ETSII. Universidad Rey Juan Carlos. C/ Tulipán, s/n. 28933 Móstoles, Spain
josemiguel.buenaposada@urjc.es

Abstract. Efficient matching of local image features is a fundamental task in many computer vision applications. Real-time performance of top matching algorithms is compromised in computationally limited devices, due to the simplicity of hardware and the finite energy supply. In this paper we present BELID, an efficient learned image descriptor. The key for its efficiency is the discriminative selection of a set of image features with very low computational requirements. In our experiments, performed both in a personal computer and a smartphone, BELID has an accuracy similar to SIFT with execution times comparable to ORB, the fastest algorithm in the literature.

Keywords: Computer Vision for smartphones · Feature descriptors extraction · Learned descriptors · Boosting

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 such as 3D reconstruction [1,20], SLAM [14], image retrieval [16], tracking [17], place recognition [15] or pose estimation[31]. 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 [18], segments [28], lines [23] and regions [11], that may be described by real valued [10,3] or binary [5,19,8] descriptors, being the binary ones the fastest. In this paper we address the problem of efficient feature description.

Although the SIFT descriptor was introduced twenty years ago [9,10], it is still considered the "golden standard" technique. The recent HPatches benchmark has shown, however, that there is still a lot of room for improvement [2]. Modern descriptors based on deep models have boosted the mean Average Precision (mAP) in different tasks [2] 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 [18,28,19] and descriptors [8,5,8] have emerged. They enable real-time performance on resource limited devices, at the price of an accuracy significantly lower than SIFT [32].

In this paper we present an efficient descriptor. Our features use the integral image to efficiently compute the difference between the mean gray values in a pair of image square regions. We use a boosting algorithm [26] to discriminatively select a set of features and combine them to produce a strong description. In our experiments we show that this approach speeds up the computation and achieves execution times close to the fastest technique in the literature, ORB [19], 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 [2].

2 Related work

SIFT is the most well-known descriptor algorithm [10]. It is widely used because it has a good performance in many Computer Vision tasks. However, it is computationally quite demanding and the only way to use it in a real-time system is using a GPU [4].

A number of different descriptors, such as SURF [3], BRIEF [5], BRISK [8] and ORB [19], have emerged to speed up SIFT. BRIEF, BRISK and ORB use features based on the comparison of pairs of image pixels. The key for their speed is the use of a limited number of binary comparisons. BRIEF uses a fixed size (9×9) smoothing convolution kernel before comparing up to 512 randomly located pixel value pairs (see Fig. 1). BRISK uses a circular pattern (see Fig. 1), smoothing the image with gaussian filters with increasing standard deviation the further away from the center. 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 (see Fig. 1). The main drawback of these methods is that they trade accuracy for speed, performing significantly worse than SIFT.

Descriptors based on learning algorithms may further improve the performance. To this end they learn the descriptor hyper parameters, DAISY [25], and select the most discriminative features using Boosting, BinBoost [26], or Convex Optimization [22]. More recently, the use of Deep Learning has enabled end-to-end learning of descriptors. All CNN-based methods train using pairs or triples of cropped patches. Some train Siamese nets [6], use L2 and hard negative

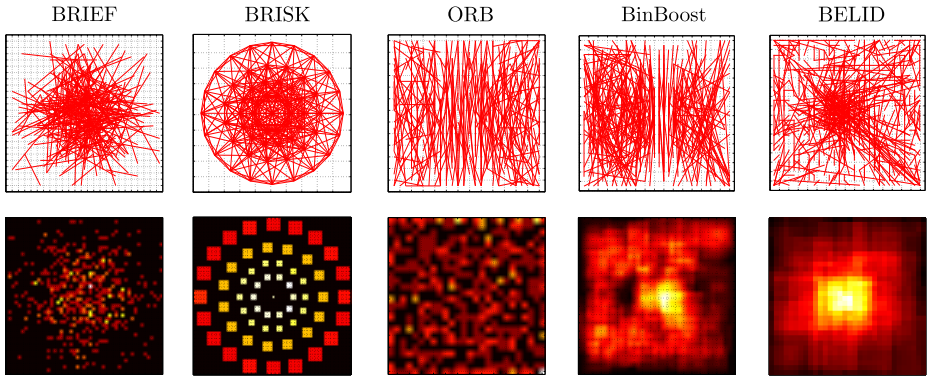

Fig. 1. Visualization of the sampling pairs of pixel locations (first row) and spatial weight heat maps (second row) employed by BRIEF, BRISK, ORB, BinBoost and our BELID trained on the Notre Dame patches dataset. BELID learns a well distributed set of point pairs giving most importance to the center area, making it close to a gaussian weight distribution. BRIEF, BRISK, ORB and BinBoost images taken from [26].

mining [24] or modified triplet-based loss [13]. Other methods optimize a loss related to the Average Precision [7] or an improved triplet loss to help focus on hard examples in training [29]. These methods have improved by a large margin the performance of SIFT in the HPatches benchmark. However, all of them incur in a much larger computational cost.

In this paper we present BELID, a descriptor trained with Boosting that is able to select the best features for the task at hand. Like BRIEF, BRISK and ORB, our features are based on differences of gray values. However, in our descriptor, we compute the difference of the mean gray values within a box. The box size represents a scale parameter that improves the discrimination. Another important difference is that in BELID the search for the best features is guided by a discriminative objective.

3 Boosted Efficient Local Image Descriptor (BELID)

In this section we present an efficient algorithm for describing image local regions that is as fast as ORB and as accurate as SIFT. The key for its speed is the use of few, fast and discriminatively selected features. Our descriptor uses a set of K features selected using the BoostedSCC algorithm [21]. This algorithm is a modification of AdaBoost to select the weak learner (WL) that maximizes the difference between the True Positive Rate (TR) and the False Positive Rate (FP).

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 $l_i = -1$ if different. The

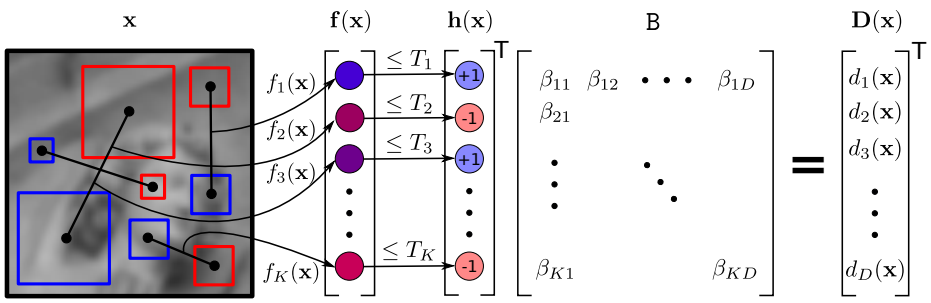

Fig. 2. Descriptor extraction workflow: To describe an image patch BELID efficiently calculates the mean gray value of the pixels in the red and blue boxes in the left. Next, for each pair of red-blue boxes (weak learner) we subtract the red box average gray value from the blue one average, obtaining $\mathbf{f}(\mathbf{x})$. We apply a set of thresholds to these values obtaining $\mathbf{h}(\mathbf{x})$ and finally we multiply by matrix $\mathtt{B}$, to produce the descriptor $\mathbf{D}(\mathbf{x})$.

training process minimizes the loss

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

where $h_k(\mathbf{z}) \equiv h_k(\mathbf{z}; f, T)$ corresponds to the k-th WL that depends on a feature extraction function $f : \mathcal{X} \to \mathbb{R}$ and a threshold T. Given f and T we define our weak learners by thresholding $f(\mathbf{x})$ with T,

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

3.1 Thresholded Average Box weak learner

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

$$f(\mathbf{x}; \mathbf{p}_1, \mathbf{p}_2, s) = \frac{1}{s^2}\left(\sum_{\mathbf{q} \in R(\mathbf{p}_1, s)} I(\mathbf{q}) - \sum_{\mathbf{r} \in R(\mathbf{p}_2, s)} I(\mathbf{r})\right), \tag{3}$$

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 $R(\mathbf{p}_1, s)$. To speed up the computation of f, we use the integral image S 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 4 arithmetic operations.

Detectors usually compute the orientation and scale of the local structure. To make our descriptor invariant to euclidean transformations, we orient and scale our measurements with the underlying local structure.

3.2 Optimizing weak learner weights

The BoostedSCC algorithm selects K weak learners with their corresponding weights. The loss function optimized by BoostedSCC in eq. 1 can be seen as a metric learning approach in which the metric matrix $\mathtt{A}$ is diagonal

$$\mathcal{L}_{BSCC} = \sum_{i=1}^{N} \exp\left(-l_i \mathbf{h}(\mathbf{x}_i)^\top \underbrace{\begin{bmatrix} \alpha_1^2 & & \\ & \ddots & \\ & & \alpha_K^2 \end{bmatrix}}_{\mathtt{A}} \mathbf{h}(\mathbf{y}_i) \right), \tag{4}$$

where $\mathbf{h}(\mathbf{w})$ is the vector with the responses of the K weak learners for the image patch $\mathbf{w}$. In this case we are not considering the dependencies between different weak learners responses. At this point the BELID-U (un-optimized) descriptor of a given image patch $\mathbf{w}$ is calculated as $\mathbf{D}(\mathbf{w}) = \mathtt{A}^{\frac{1}{2}}\mathbf{h}(\mathbf{x})$, where $\mathtt{A}^{\frac{1}{2}}$ is such that $\mathtt{A} = \mathtt{A}^{\frac{1}{2}}\mathtt{A}^{\frac{1}{2}}$.

Further, estimating the whole matrix $\mathtt{A}$ improves the similarity function by modeling the correlation between features, $s(\mathbf{x}, \mathbf{y}) = \mathbf{h}(\mathbf{x})^\top \mathtt{A}\mathbf{h}(\mathbf{y})$. FP-Boost [26] estimates $\mathtt{A}$ minimizing

$$\mathcal{L}_{FP} = \sum_{i=1}^{N} \exp\left(-l_i \sum_{k,r} \alpha_{k,r} h_k\left(\mathbf{x}_i\right) h_r\left(\mathbf{y}_i\right) \right) = \sum_{i=1}^{N} \exp\left(-l_i \mathbf{h}(\mathbf{x})^\top \mathtt{A}\mathbf{h}(\mathbf{y}) \right). \tag{5}$$

It uses Stochastic Gradient Descent to estimate a symmetric $\mathtt{A}$. Jointly optimizing $\mathtt{A}$ and $h_i(\mathbf{x})$ from scratch is difficult. Thus the algorithm starts from the K weak learners and α's found by BoostedSCC. This second learning step is quite fast because all weak learners responses can be pre-computed.

As in the case of the un-optimized descriptor we have to factorize the similarity function $s(\mathbf{x}, \mathbf{y})$ to compute the independent descriptors for $\mathbf{x}$ and $\mathbf{y}$. Given that $\mathtt{A}$ is a symmetric matrix we can use its eigen-decomposition selecting the D eigenvectors with largest eigenvalues

$$\mathtt{A} = \mathtt{B}\mathtt{W}\mathtt{B}^\top = \sum_{d=1}^{D} w_d \mathbf{b}_d \mathbf{b}_d^\top, \tag{6}$$

where $\mathtt{W}$ =diag$([w_1, \cdots, w_D])$, $w_d \in \{-1, 1\}$, $\mathtt{B} = [\mathbf{b}_1, \cdots, \mathbf{b}_D], \mathbf{b} \in \mathbb{R}^K$, and $D \leq K$. The final descriptor of a given image patch $\mathbf{w}$ is given by $\mathbf{D}(\mathbf{w}) = \mathtt{B}^\top \mathbf{h}(\mathbf{w})$ (see Fig. 2). It will be denoted using the final dimension D, as BELID-D (e.g. BELID-128 when $D = 128$).

4 Experiments

In our experiments we use the popular dataset of patches [4] from Winder *et al.* [30] for training. It consists of 64×64 cropped image patches from three

[4] `http://matthewalunbrown.com/patchdata/patchdata.html`

different scenes: Notre Dame cathedral, Yosemite National Part and Liberty statue in New York. The patches are cropped around local structures detected by SIFT.

We compare the performance using three measures:

- **FPR-95**. This is the False Positive Rate at 95% of recall in a patch verification problem (*i.e.* given two patches deciding if they are similar - positive class - or not). When we develop a descriptor, we want to be able to match most of the local structures, lets say a 95% of recall, but with the lowest possible number of false positives. Thus, a descriptor is better the lower FPR-95 it achieves in the patch verification problem.
- **AUC**. Area Under the ROC Curve in a patch verification problem. It provides a good overall measurement, since it considers all the operation points of the curve, instead of just one as in the FPR-95 case.
- **mAP**. Mean Average Precision, as defined in the HPatches benchmark [2] for each of the three tasks: patch verification, image matching and patch retrieval.

We have implemented in Python BoostedSCC, FP-Boost and the learning and testing part of the Thresholded Average Box weak learner of Sec. 3.1. For optimizing the `A` matrix we use the Stochastic Gradient Descent algorithm with a fixed learning rate of 10^{-8} and a batch size of 2000 samples. We have also implemented in C++, using OpenCV, the descriptor extraction algorithm to process the input images (i.e. not cropped patches). We use this implementation to measure the execution time of BELID in different platforms.

4.1 Patch verification experiments

Here we first explore the effect of the number of dimensions, K, in BELID-U and D in BELID (optimized) descriptors. In Fig. 3 we show the AUC and FPR-95 values as a function of the number of dimensions ("N Dimensions"). In the case of BELID, we use $K = 512$ weak learners and compute `B` to reduce from 512 dimensions to the one given in the plots.

We train using a balanced set of 100K positive and 100K negative patch pairs from the Yosemite sequence. The testing set comprises 50K positive and 50K negative pairs from the Liberty statue. We first run BoostedSSC selecting 512 weak learners. We change the number of dimensions of the BELID-U curve in Fig. 3 by removing the last weak learners from this initial set. For BELID, we discard the last columns of `B`, that correspond to the scaled eigen-vectors associated with the smallest eigenvalues.

We can see in Fig. 3 that the boosting process selects features that, up to one point, contribute to the final discrimination. After 128 weak learners the improvement provided by each new feature is very small. After 256 we do not get any improvement at all, which means that the last ones are redundant. The performances of the optimized BELID are always better than those of BELID-U. This proves the interest of the optimization process. BELID gets the lowest

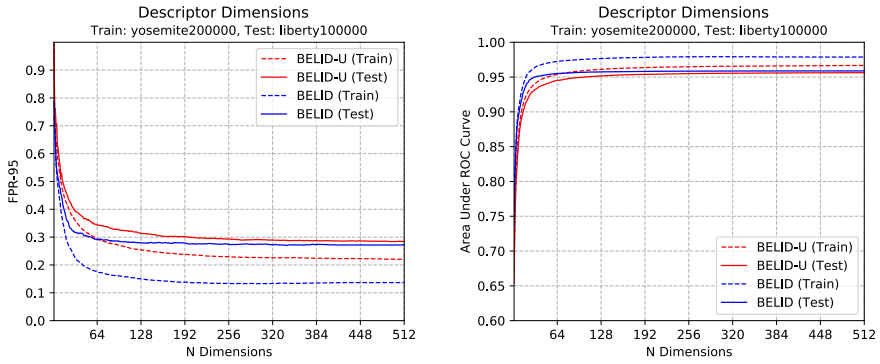

Fig. 3. BELID-U and BELID, training and testing, AUC ("Area Under the ROC Curve") and FPR-95 ("Error") as a function of the number of dimensions in a patch verification problem.

FPR-95 at 128 dimensions that, interestingly, is the same number of dimensions used by SIFT. In consequence, BELID-128 is our best descriptor.

In the next experiment we compare our descriptor with SIFT, the "golden standard", and ORB, a representative of the descriptors developed for computational efficiency. We also evaluate LBGM [27], a descriptor using more informative, but computationally expensive, features based on the gradients orientation and the optimization in Sec. 3.2 to estimate $\mathbf{A}$. For these features we use the implementations in OpenCV. We have trained in the 200K patch balanced set from Notre Dame and tested in the 100K patch balanced set from the Liberty statue datasets (see Fig. 4 left). We have also trained in the 200K patch balanced set from Yosemite sequence and tested with the 100K patch balanced set from Notre Dame (see Fig. 4 right). Fig. 4 shows the ROC curves for the testing sets. In terms of accuracy, ORB is the worst descriptor. BELID-128 is better than SIFT and marginally worse than LBGM and BinBoost, both using the same boosting scheme for selecting gradient-based features. Comparing different versions of our algorithm, we can see that BELID-U gets slightly higher FPR-95 values than BELID (as we have seen in the previous experiments) when training and testing sets are from the same domain (Notre Dame/Liberty) and a comparable FPR-95 when they are from different ones (Yosemite/Notre Dame).

4.2 Experiments on the Hpatches dataset

The recent HPatches benchmark [2] solves some of the shortcomings of previous data sets in terms of data and task diversity, evaluation metrics and experimental reproducibility. The benchmark provides patches taken from images of different scenes under real and varying capturing conditions, that are tested in patch verification, image matching and patch retrieval problems. We have trained with the balanced 200K patches pairs from Notre Dame and evaluated on the testing HPatches dataset using the Python code provided by the authors.

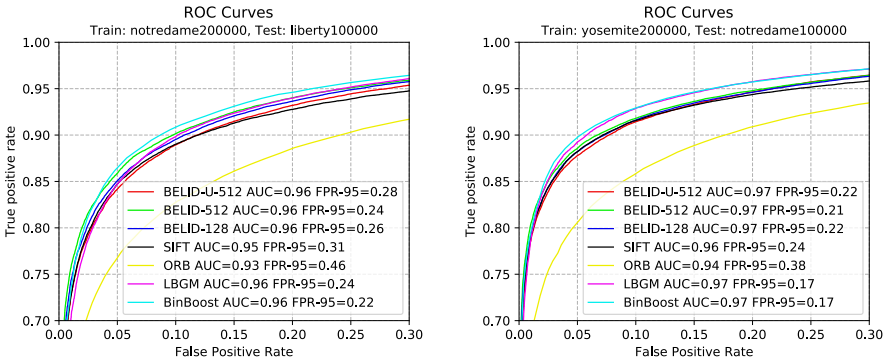

Fig. 4. Test sets ROC curves for the patch verification experiments.

Fig. 5 shows the results of various BELID configurations and those of other competing approaches obtained with the HPathces tool. In the patch verification problem, the one we use to optimize our descriptor, we get the same situation of the previous experiments. All BELID configurations are better than SIFT, 69.57 vs 63.35, and much better than ORB, 58.21. However, in the other two tasks our descriptor is falling behind SIFT. This is an expected result since we are not optimizing our descriptor for these tasks. Altogether, depending on the configuration considered, BELID may provide results close to SIFT and better than ORB in all tasks.

We have added to Fig. 5 Hardnet [13], a representative CNN-based descriptor. Hardnet beats by a large margin all handcrafted (BRIEF, ORB, SIFT) and Boosting-based descriptors (BinBoost [26], BELID), but it has a much higher computational and energy requirements.

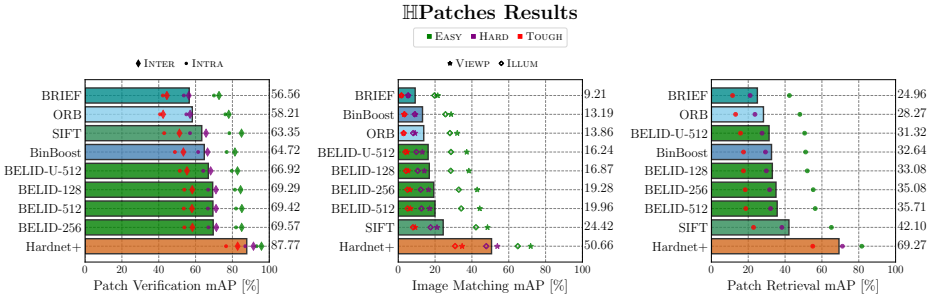

Fig. 5. Comparison in the HPatches dataset for three different tasks.

4.3 Execution time in different platforms

In the last experiment we test our C++ implementation of BELID processing images (i.e. no cropped patches) in a desktop CPU, Intel Core i7-6700HQ at 2.60GHz and 16GB RAM, and in the limited CPU, Exynox Octa 7870 at 1.59 GHz and 2GB RAM of a Samsung Galaxy J5-2017 smartphone. We report the execution time in the Mikolajczyk [12] dataset composed by 48 800×640 images from 8 different scenes. We detect a maximum of 2000 local structures per image with SURF.

We compare the execution time with other relevant descriptors in the OpenCV library. To this end we use the C++ interface. Specifically we run ORB [19], SIFT [10], LBGM [27] and BinBoost [26]. In Table 1 we show the size of the descriptors in terms of the number of components that can be floating point numbers (f) or bits (b) and the average execution time per image in the experiment.

In terms of speed, BELID-U (without optimization) is comparable to ORB. In fact, BELID-U is as fast as ORB in desktop (0.41 ms vs 0.44 ms) and faster in the limited CPU (2.54 ms vs 6.49 ms). This was expected since both use as features a set of gray value differences. LBGM uses the same feature selection algorithm as BELID, but with slower features. Thus, this descriptor requires the same processing time as SIFT in the desktop setup (19.77 ms vs 22.22 ms) with a slightly better FPR-95 (see section 4.1).

BELID-128 takes only 3.08 ms in the desktop CPU, around $7\times$ the time of BELID-U and ORB. In the Exynos Octa smartphone CPU the time of BELID-128 is also around $7\times$ slower than BELID-U, as expected.

These results support the claim that our descriptor if a faster alternative to SIFT that is able to run in real-time on low performance devices, while preserving the accuracy.

Table 1. Average execution time per image of various descriptors.

	Size	Intel Core i7	Exynox Octa
SIFT	128f	22.22 ms	163.2 ms
ORB	256b	0.44 ms	6.49 ms
LBGM	64f	19.77 ms	64.24 ms
BinBoost	256b	12.57 ms	42.39 ms
BELID-512	512f	10.48 ms	61.47 ms
BELID-128	128f	3.08 ms	17.13 ms
BELID-U	512f	**0.41 ms**	**2.54 ms**

5 Conclusion

In this paper we presented BELID, an efficient learned image descriptor. In our experiments we proved that it has very low computational requirements, similar

to those of ORB, the fastest descriptor in the literature. This is due to the use of very fast image features, based on gray value differences computed with the integral image. In terms of accuracy BELID is better than ORB and close to SIFT, the golden standard reference in the literature. We believe this is due to the discriminative scheme used to select the image features and the possibility of learning the best smoothing filter scale, represented in BELID by the feature box sizes. Our feature selection scheme optimizes a patch verification problem. This is why BELID achieves better accuracy than SIFT in the HPatches patch verification task and worse in the image matching and patch retrieval tasks.

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 it 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, BELID represents also an excellent trade-off.

There are various ways to improve the results in this work. First we may change the feature selection process to optimize the performance not only in a patch verification task but also in image matching and patch retrieval. We may also binarize the output descriptor to decrease the model storage requirements and achieve higher matching speed. Finally, we also plan to improve the implementation to optimize speed in different types of processors.

Acknowledgments

The following funding is gratefully acknowledged. Iago Suárez, grant Doctorado Industrial DI-16-08966; José M. Buenaposada and Luis Baumela, Spanish MINECO project TIN2016-75982-C2-2-R.

References

1. Agarwal, S., Snavely, N., Simon, I., Seitz, S.M., Szeliski, R.: Building rome in a day. In: 2009 IEEE 12th international conference on computer vision. pp. 72–79. IEEE (2009)
2. Balntas, V., Lenc, K., Vedaldi, A., Mikolajczyk, K.: Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5173–5182 (2017)
3. Bay, H., Tuytelaars, T., Van Gool, L.: Surf: Speeded up robust features. In: European conference on computer vision. pp. 404–417. Springer (2006)
4. Bjrkman, M., Bergstrm, N., Kragic, D.: Detecting, segmenting and tracking unknown objects using multi-label mrf inference. Computer Vision and Image Understanding **118**, 111 – 127 (2014). https://doi.org/https://doi.org/10.1016/j.cviu.2013.10.007, `http://www.sciencedirect.com/science/article/pii/S107731421300194X`

5. Calonder, M., Lepetit, V., Strecha, C., Fua, P.: Brief: Binary robust independent elementary features. In: European conference on computer vision. pp. 778–792. Springer (2010)
6. Han, X., Leung, T., Jia, Y., Sukthankar, R., Berg, A.C.: Matchnet: Unifying feature and metric learning for patch-based matching. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2015)
7. He, K., Lu, Y., Sclaroff, S.: Local descriptors optimized for average precision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 596–605 (2018)
8. Leutenegger, S., Chli, M., Siegwart, R.: Brisk: Binary robust invariant scalable keypoints. In: 2011 IEEE international conference on computer vision (ICCV). pp. 2548–2555. Ieee (2011)
9. Lowe, D.G.: Object recognition from local scale-invariant features. In: Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2. pp. 1150–. ICCV '99, IEEE Computer Society, Washington, DC, USA (1999), `http://dl.acm.org/citation.cfm?id=850924.851523`
10. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International journal of computer vision **60**(2), 91–110 (2004)
11. Matas, J., Chum, O., Urban, M., Pajdla, T.: Robust wide baseline stereo from maximally stable extremal regions. In: Proc. BMVC. pp. 36.1–36.10 (2002), doi:10.5244/C.16.36
12. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE transactions on pattern analysis and machine intelligence **27**(10), 1615–1630 (2005)
13. Mishchuk, A., Mishkin, D., Radenovic, F., Matas, J.: Working hard to know your neighbor's margins: Local descriptor learning loss. In: Advances in Neural Information Processing Systems. pp. 4826–4837 (2017)
14. Mur-Artal, R., Montiel, J.M.M., Tards, J.D.: Orb-slam: A versatile and accurate monocular slam system. IEEE Transactions on Robotics **31**(5), 1147–1163 (Oct 2015). https://doi.org/10.1109/TRO.2015.2463671
15. Mur-Artal, R., Tards, J.D.: Fast relocalisation and loop closing in keyframe-based slam. In: 2014 IEEE International Conference on Robotics and Automation (ICRA). pp. 846–853 (May 2014). https://doi.org/10.1109/ICRA.2014.6906953
16. Nister, D., Stewenius, H.: Scalable recognition with a vocabulary tree. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06). vol. 2, pp. 2161–2168 (June 2006)
17. Pernici, F., Del Bimbo, A.: Object tracking by oversampling local features. IEEE transactions on pattern analysis and machine intelligence **36**(12), 2538–2551 (2014)
18. Rosten, E., Drummond, T.: Machine learning for high-speed corner detection. In: European conference on computer vision. pp. 430–443. Springer (2006)
19. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.R.: Orb: An efficient alternative to sift or surf. In: ICCV. vol. 11, p. 2. Citeseer (2011)
20. Schonberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4104–4113 (2016)
21. Shakhnarovich, G.: Learning task-specific similarity. Ph.D. thesis, Massachusetts Institute of Technology (2005)
22. Simonyan, K., Vedaldi, A., Zisserman, A.: Learning local feature descriptors using convex optimisation. IEEE Transactions on Pattern Analysis and Machine Intelligence **36**(8), 1573–1585 (2014)

23. Suarez, I., Muñoz, E., Buenaposada, J.M., Baumela, L.: FSG: A statistical approach to line detection via fast segments grouping. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 97–102 (Oct 2018). https://doi.org/10.1109/IROS.2018.8594434
24. Tian, Y., Fan, B., Wu, F.: L2-net: Deep learning of discriminative patch descriptor in euclidean space. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6128–6136 (July 2017). https://doi.org/10.1109/CVPR.2017.649
25. Tola, E., Lepetit, V., Fua, P.: A fast local descriptor for dense matching. In: 2008 IEEE conference on computer vision and pattern recognition. pp. 1–8. IEEE (2008)
26. Trzcinski, T., Christoudias, M., Lepetit, V.: Learning image descriptors with boosting. IEEE transactions on pattern analysis and machine intelligence **37**(3), 597–610 (2015)
27. Trzcinski, T., Christoudias, M., Lepetit, V., Fua, P.: Learning image descriptors with the boosting-trick. In: Advances in neural information processing systems. pp. 269–277 (2012)
28. Von Gioi, R.G., Jakubowicz, J., Morel, J.M., Randall, G.: Lsd: A fast line segment detector with a false detection control. IEEE transactions on pattern analysis and machine intelligence **32**(4), 722–732 (2010)
29. Wei, X., Zhang, Y., Gong, Y., Zheng, N.: Kernelized subspace pooling for deep local descriptors. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1867–1875 (June 2018). https://doi.org/10.1109/CVPR.2018.00200
30. Winder, S.A., Brown, M.: Learning local image descriptors. In: 2007 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–8. IEEE (2007)
31. Wohlhart, P., Lepetit, V.: Learning descriptors for object recognition and 3d pose estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3109–3118 (2015)
32. Yan, W., Shi, X., Yan, X., Wang, L.: Computing opensurf on opencl and general purpose gpu. International Journal of Advanced Robotic Systems **10**(10), 375 (2013)