BRIEF: Binary Robust Independent Elementary Features

Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua

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

Abstract. We propose to use binary strings as an efficient feature point descriptor, which we call BRIEF. We show that it is highly discriminative even when using relatively few bits and can be computed using simple intensity difference tests. Furthermore, the descriptor similarity can be evaluated using the Hamming distance, which is very efficient to compute, instead of the L_2 norm as is usually done.

As a result, BRIEF is very fast both to build and to match. We compare it against SURF and U-SURF on standard benchmarks and show that it yields a similar or better recognition performance, while running in a fraction of the time required by either.

1 Introduction

Feature point descriptors are now at the core of many Computer Vision technologies, such as object recognition, 3D reconstruction, image retrieval, and camera localization. Since applications of these technologies have to handle ever more data or to run on mobile devices with limited computational resources, there is a growing need for local descriptors that are fast to compute, fast to match, and memory efficient.

One way to speed up matching and reduce memory consumption is to work with short descriptors. They can be obtained by applying dimensionality reduction, such as PCA [1] or LDA [2], to an original descriptor such as SIFT [3] or SURF [4]. For example, it was shown in [5,6,7] that floating point values of the descriptor vector could be quantized using very few bits per value without loss of recognition performance. An even more drastic dimensionality reduction can be achieved by using hash functions that reduce SIFT descriptors to binary strings, as done in [8]. These strings represent binary descriptors whose similarity can be measured by the Hamming distance.

While effective, these approaches to dimensionality reduction require first computing the full descriptor before further processing can take place. In this paper, we show that this whole computation can be shortcut by *directly* computing binary strings from image patches. The individual bits are obtained by comparing the intensities of pairs of points along the same lines as in [9] but without requiring a training phase. We refer to the resulting descriptor as BRIEF.

Our experiments show that only 256 bits, or even 128 bits, often suffice to obtain very good matching results. BRIEF is therefore very efficient both to

K. Daniilidis, P. Maragos, N. Paragios (Eds.): ECCV 2010, Part IV, LNCS 6314, pp. 778–792, 2010.

compute and to store in memory. Furthermore, comparing strings can be done by computing the Hamming distance, which can be done extremely fast on modern CPUs that often provide a specific instruction to perform a XOR or bit count operation, as is the case in the latest SSE [10] instruction set.

This means that BRIEF easily outperforms other fast descriptors such as SURF and U-SURF in terms of speed, as will be shown in the Results section. Furthermore, it also outperforms them in terms of recognition rate in many cases, as we will demonstrate using benchmark datasets.

2 Related Work

The SIFT descriptor [3] is highly discriminant but, being a 128-vector, is relatively slow to compute and match. This can be a drawback for real-time applications such as SLAM that keep track of many points as well as for algorithms that require storing very large numbers of descriptors, for example for large-scale 3D reconstruction.

There are many approaches to solving this problem by developing faster to compute and match descriptors, while preserving the discriminative power of SIFT. The SURF descriptor [4] represents one of the best known ones. Like SIFT, it relies on local gradient histograms but uses integral images to speed up the computation. Different parameter settings are possible but, since using only 64 dimensions already yields good recognition performances, that version has become very popular and a *de facto* standard. This is why we compare ourselves to it in the Results section.

SURF addresses the issue of speed but, since the descriptor is a 64-vector of floating points values, representing it still requires 256 bytes. This becomes significant when millions of descriptors must be stored. There are three main classes of approaches to reducing this number.

The first involves dimensionality reduction techniques such as Principal Component Analysis (PCA) or Linear Discriminant Embedding (LDE). PCA is very easy to perform and can reduce descriptor size at no loss in recognition performance [1]. By contrast, LDE requires labeled training data, in the form of descriptors that should be matched together, which is more difficult to obtain. It can improve performance [2] but can also overfit and degrade performance.

A second way to shorten a descriptor is to quantize its floating-point coordinates into integers coded on fewer bits. In [5], it is shown that the SIFT descriptor can be quantized using only 4 bits per coordinate. Quantization is used for the same purpose in [6,7]. It is a simple operation that results not only in memory gain but also in faster matching as computing the distance between short vectors can then be done very efficiently on modern CPUs. In [6], it is shown that for some parameter settings of the DAISY descriptor, PCA and quantization can be combined to reduce its size to 60 bits. However, in this approach the Hamming distance cannot be used for matching because the bits are, in contrast to BRIEF, arranged in blocks of four and hence cannot be processed independently.

A third and more radical way to shorten a descriptor is to binarize it. For example, [8] drew its inspiration from Locality Sensitive Hashing (LSH) [11] to

turn floating-point vectors into binary strings. This is done by thresholding the vectors after multiplication with an appropriate matrix. Similarity between descriptors is then measured by the Hamming distance between the corresponding binary strings. This is very fast because the Hamming distance can be computed very efficiently with a bitwise XOR operation followed by a bit count. The same algorithm was applied to the GIST descriptor to obtain a binary description of an entire image [12]. Another way to binarize the GIST descriptor is to use non-linear Neighborhood Component Analysis [12,13], which seems more powerful but probably slower at run-time.

While all three classes of shortening techniques provide satisfactory results, relying on them remains inefficient in the sense that first computing a long descriptor then shortening it involves a substantial amount of time-consuming computation. By contrast, the approach we advocate in this paper directly builds short descriptors by comparing the intensities of pairs of points without ever creating a long one. Such intensity comparisons were used in [9] for classification purposes and were shown to be very powerful in spite of their extreme simplicity. Nevertheless, the present approach is very different from [9] and [14] because it does *not* involve any form of online or offline training.

3 Method

Our approach is inspired by earlier work [9,15] that showed that image patches could be effectively classified on the basis of a relatively small number of pairwise intensity comparisons. The results of these tests were used to train either randomized classification trees [15] or a Naive Bayesian classifier [9] to recognize patches seen from different viewpoints. Here, we do away with both the classifier and the trees, and simply create a bit vector out of the test responses, which we compute after having smoothed the image patch.

More specifically, we define test τ on patch $\mathbf{p}$ of size $S \times S$ as

$$\tau(\mathbf{p}; \mathbf{x}, \mathbf{y}) := \begin{cases} 1 & \text{if } \mathbf{p}(\mathbf{x}) < \mathbf{p}(\mathbf{y}) \\ 0 & \text{otherwise} \end{cases} , \tag{1}$$

where $\mathbf{p}(\mathbf{x})$ is the pixel intensity in a smoothed version of $\mathbf{p}$ at $\mathbf{x} = (u, v)^\top$. Choosing a set of n_d $(\mathbf{x}, \mathbf{y})$-location pairs uniquely defines a set of binary tests. We take our BRIEF descriptor to be the n_d-dimensional bitstring

$$f_{n_d}(\mathbf{p}) := \sum_{1 \le i \le n_d} 2^{i-1} \, \tau(\mathbf{p}; \mathbf{x}_i, \mathbf{y}_i) \ . \tag{2}$$

In this paper we consider $n_d = 128$, 256, and 512 and will show in the Results section that these yield good compromises between speed, storage efficiency, and recognition rate. In the remainder of the paper, we will refer to BRIEF descriptors as BRIEF-k, where $k = n_d/8$ represents the number of *bytes* required to store the descriptor.

When creating such descriptors, the only choices that have to be made are those of the kernels used to smooth the patches before intensity differencing and

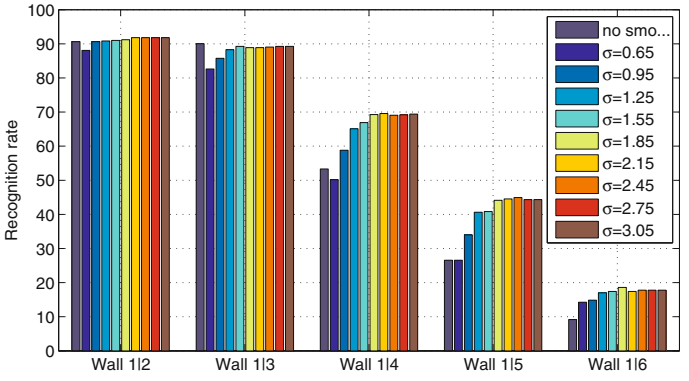

Fig. 1. Each group of 10 bars represents the recognition rates in one specific stereo pair for increasing levels of Gaussian smoothing. Especially for the hard-to-match pairs, which are those on the right side of the plot, smoothing is essential in slowing down the rate at which the recognition rate decreases.

the spatial arrangement of the $(\mathbf{x}, \mathbf{y})$-pairs. We discuss these in the remainder of this section.

To this end, we use the Wall dataset that we will describe in more detail in section 4. It contains five image pairs, with the first image being the same in all pairs and the second image shot from a monotonically growing baseline, which makes matching increasingly more difficult. To compare the pertinence of the various potential choices, we use as a quality measure the *recognition rate* in image pairs that will be precisely defined at the beginning of section 4. In short, for both images of a pair and for a given number of corresponding keypoints between them, it quantifies how often the correct match can be established using BRIEF for description and the Hamming distance as the metric for matching. This rate can be computed reliably because the scene is planar and the homography between images is known. It can therefore be used to check whether points truly correspond to each other or not.

3.1 Smoothing Kernels

By construction, the tests of Eq. 1 take only the information at single pixels into account and are therefore very noise-sensitive. By pre-smoothing the patch, this sensitivity can be reduced, thus increasing the stability and repeatability of the descriptors. It is for the same reason that images need to be smoothed before they can be meaningfully differentiated when looking for edges. This analogy applies because our intensity difference tests can be thought of as evaluating the sign of the derivatives within a patch.

Fig. 1 illustrates the effects of increasing amounts of Gaussian smoothing on the recognition rates for variances of Gaussian kernel ranging from 0 to 3. The more difficult the matching, the more important smoothing becomes to achieving good performance. Furthermore, the recognition rates remain relatively constant

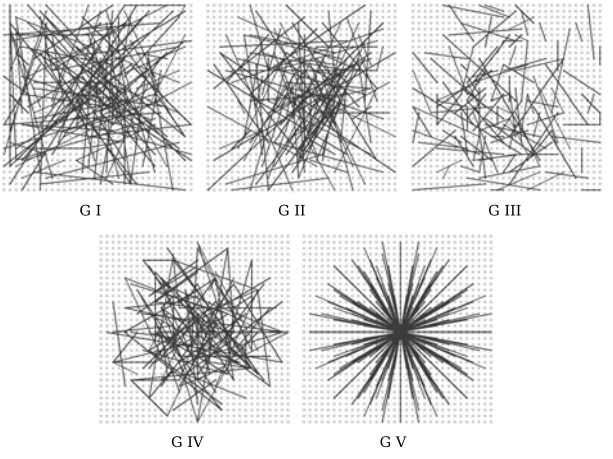

Fig. 2. Different approaches to choosing the test locations. All except the righmost one are selected by random sampling. Showing 128 tests in every image.

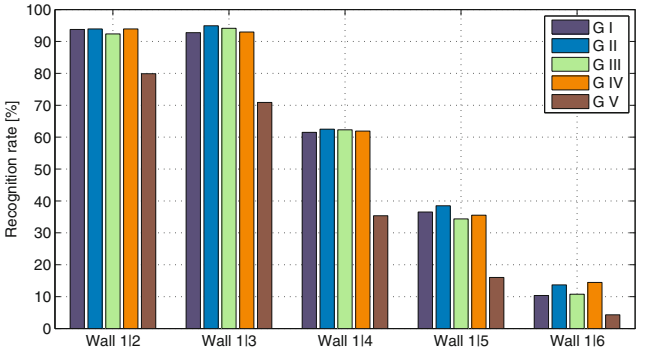

Fig. 3. Recognition rate for the five different test geometries introduced in section 3.2.

in the 1 to 3 range and, in practice, we use a value of 2. For the corresponding discrete kernel window we found a size of 9×9 pixels be necessary and sufficient.

3.2 Spatial Arrangement of the Binary Tests

Generating a length n_d bit vector leaves many options for selecting the n_d test locations $(\mathbf{x}_i, \mathbf{y}_i)$ of Eq. 1 in a patch of size $S \times S$. We experimented with the five sampling geometries depicted by Fig. 2. Assuming the origin of the patch coordinate system to be located at the patch center, they can be described as follows.

I) $(\mathbf{X}, \mathbf{Y}) \sim$ i.i.d. Uniform$(-\frac{S}{2}, \frac{S}{2})$: The $(\mathbf{x}_i, \mathbf{y}_i)$ locations are evenly distributed over the patch and tests can lie close to the patch border.

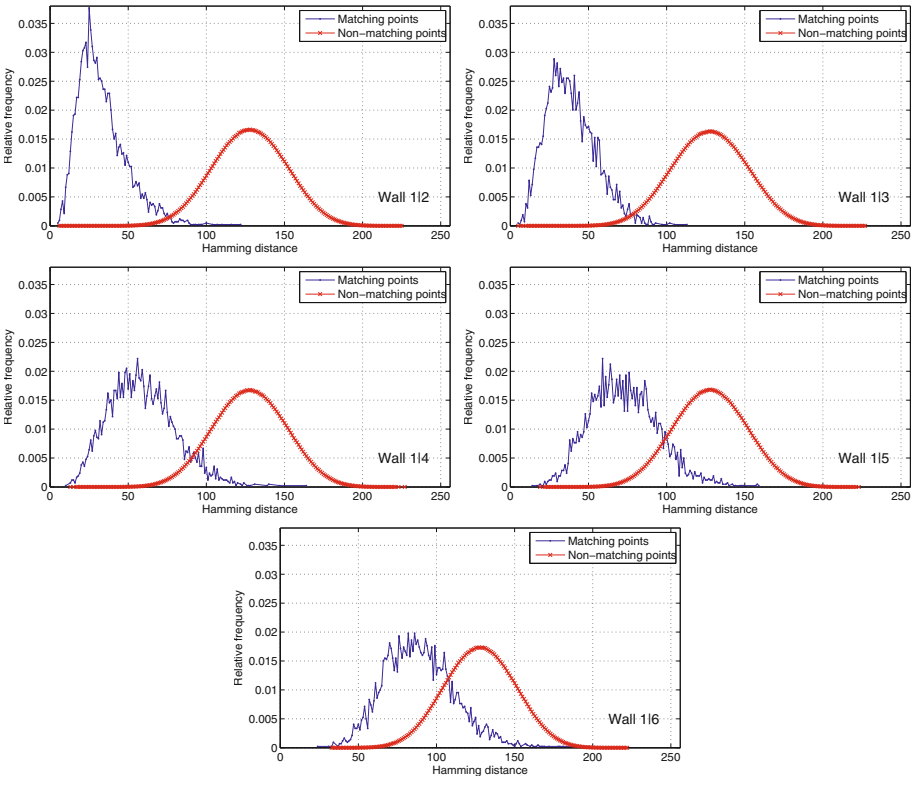

Fig. 4. Distributions of Hamming distances for matching pairs of points (thin blue lines) and for non-matching pairs (thick red lines) in each of the five image pairs of the Wall dataset. They are most separated for the first image pairs, whose baseline is smaller, ultimately resulting in higher recognition rates.

II) $(\mathbf{X}, \mathbf{Y}) \sim$ i.i.d. Gaussian$(0, \frac{1}{25}S^2)$: The tests are sampled from an isotropic Gaussian distribution. Experimentally we found $\frac{s}{2} = \frac{5}{2}\sigma \Leftrightarrow \sigma^2 = \frac{1}{25}S^2$ to give best results in terms of recognition rate.

III) $\mathbf{X} \sim$ i.i.d. Gaussian$(0, \frac{1}{25}S^2)$, $\mathbf{Y} \sim$ i.i.d. Gaussian$(\mathbf{x}_i, \frac{1}{100}S^2)$: The sampling involves two steps. The first location $\mathbf{x}_i$ is sampled from a Gaussian centered around the origin while the second location is sampled from another Gaussian centered on $\mathbf{x}_i$. This forces the tests to be more local. Test locations outside the patch are clamped to the edge of the patch. Again, experimentally we found $\frac{S}{4} = \frac{5}{2}\sigma \Leftrightarrow \sigma^2 = \frac{1}{100}S^2$ for the second Gaussian performing best.

IV) The $(\mathbf{x}_i, \mathbf{y}_i)$ are randomly sampled from discrete locations of a coarse polar grid introducing a spatial quantization.

V) $\forall i : \mathbf{x}_i = (0, 0)^\top$ and $\mathbf{y}_i$ is takes all possible values on a coarse polar grid containing n_d points.

For each of these test geometries we compute the recognition rate and show the result in Fig. 3. Clearly, the symmetrical and regular G V strategy loses out against all random designs G I to G IV, with G II enjoying a small advantage over the other three in most cases. For this reason, in all further experiments presented in this paper, it is the one we will use.

3.3 Distance Distributions

In this section, we take a closer look at the distribution of Hamming distances between our descriptors. To this end we extract about 4000 matching points from the five image pairs of the Wall sequence. For each image pair, Fig. 4 shows the normalized histograms, or distributions, of Hamming distances between corresponding points (in blue) and non-corresponding points (in red). The maximum possible Hamming distance being $32 \cdot 8 = 256$ bits, unsurprisingly, the distribution of distances for non-matching points is roughly Gaussian and centered around 128. As could also be expected, the blue curves are centered around a smaller value that increases with the baseline of the image pairs and, therefore, with the difficulty of the matching task.

Since establishing a match can be understood as classifying pairs of points as being a match or not, a classifier that relies on these Hamming distances will work best when their distributions are most separated. As we will see in section 4, this is of course what happens with recognition rates being higher in the first pairs of the Wall sequence than in the subsequent ones.

4 Results

In this section, we compare our method against several competing approaches. Chief among them is the latest OpenCV implementation of the SURF descriptor [4], which has become a *de facto* standard for fast-to-compute descriptors. We use the standard SURF64 version, which returns a 64-dimensional floating point vector and requires 256 bytes of storage. Because BRIEF, unlike SURF, does not correct for orientation, we also compare against U-SURF [4], where the U stands for upright and means that orientation also is ignored [4].

To this end, we use the six publicly available test image sequences depicted by Fig. 5. They are designed to test robustness to

- viewpoint changes (Wall[1], Graffiti[1], Fountain[2]),
- compression artifacts (Jpeg[1]),
- illumination changes (Light[1]),
- and image blur (Trees[1]).

For each one, we consider 5 image pairs by matching the first of 6 images to the other five they contain. Note that, the Wall and Graffiti scenes being planar,

[1] http://www.robots.ox.ac.uk/~vgg/research/affine
[2] http://cvlab.epfl.ch/~strecha/multiview

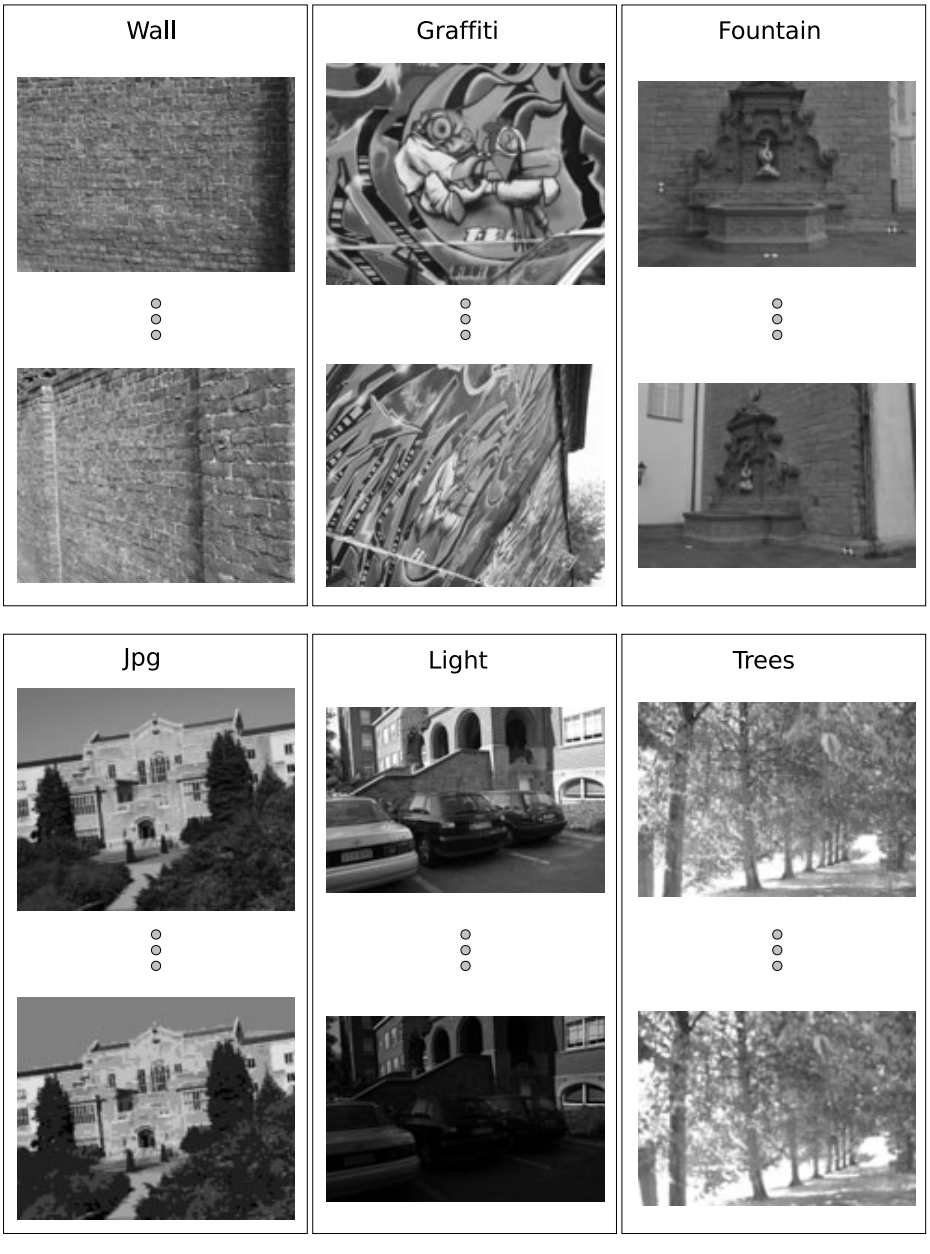

Fig. 5. Data sets used for comparison purposes. Each one contains 6 images and we consider 5 image pairs by matching the first one against all others.

the images are related by homographies that are used to compute the ground truth. The viewpoints for Jpeg, Light, and Trees being almost similar, the images are also taken to be related by a homography, which is very close to being

the identity. By contrast, the Fountain scene is fully three-dimensional and the ground truth is computed from laser scan data. Note also that the 5 pairs in Wall and Fountain are sorted in order of increasing baseline so that pair 1|6 is much harder to match than pair 1|2, which negatively affects the performance of *all* the descriptors considered here.

For evaluation purposes, we rely on two straightforward metrics, elapsed CPU time and recognition rate. The former simply is averaged measured wall clock time over many repeated runs. Given an image pair, the latter is computed as follows:

- Pick N interest points from the first image, infer the N corresponding points in the other from the ground truth data, and compute the $2N$ associated descriptors using the method under consideration.
- For each point in the first set, find the nearest neighbor in the second one and call it a match.
- Count the number of correct matches n_c and take the recognition rate to be $r = n_c/N$.

Although this may artificially increase the recognition rates, only the absolute recognition rate numbers are to be taken with caution. Since we apply the same procedure to all descriptors, and not only ours, the relative rankings we obtain are still valid and speak in BRIEF's favor. To confirm this, we detected SURF-points in both images of each test pair and computed their (SURF- or BRIEF-) descriptors, matched these descriptors to their nearest neighbor, and applied a standard left-right consistency check. Even though this setup now involves outliers, BRIEF continued to outperform SURF in the same proportion as before.

In the remainder of this section we will use these metrics to show that the computational requirements of BRIEF are much lower than those of all other methods while achieving better recognition rates than SURF on all sequences except Graffiti. This is explained both by the fact that this dataset requires strong rotation invariance, which BRIEF does not provide, and by the very specific nature of the Graffiti images. They contain large monochrome areas on which our intensity difference tests are often uninformative. In other words, this data set clearly favors descriptors based on gradient histograms, as has already been noted [7]. When comparing recognition rates against those of U-SURF, BRIEF still does better on Wall, Fountain, Trees, and similarly on Light and Jpg.

In other words, on data sets such as those that involve only modest amounts of in-plane rotation, there is a cost not only in terms of speed but also of recognition rate to achieving orientation invariance, as already pointed out in [4]. This explains in part why both BRIEF and U-SURF outperform SURF. Therefore, when not actually required, orientation correction should be avoided. This is an important observation because there are more and more cases, such as when using a mobile phone equipped with an orientation sensor, when orientation invariance stops being a requirement. This is also true in many urban contexts where photos tend to be taken at more or less canonical orientations and for mobile robotics where the robot's attitude is known.

Recognition Rate as a Function of Descriptor Size. Since many practical problems involve matching a few hundred feature points, we first use $N = 512$ to compare the recognition rate of BRIEF using either 128, 256, or 512 tests, which we denote as BRIEF-16, BRIEF-32, and BRIEF-64. The trailing number stands for the number of bytes required to store the descriptor. Recall that since both SURF and U-SURF return 64 floating point numbers, they require 256 bytes of storage and are therefore at least four times bigger.

As shown in Fig. 6, BRIEF-64 outperforms SURF and U-SURF in all sequences except Graffiti while using a descriptor that is four times smaller. Unsurprisingly, BRIEF-32 does not do quite as well but still compares well against SURF and U-SURF. BRIEF-16 is too short and shows the limits of the approach.

To show that this behavior is *not* an artifact for the number N of feature points used for testing purposes, we present similar recognition rates for values of N ranging from 512 to 4096 in Fig. 7. As could be expected, the recognition rates drop as N increases for *all* the descriptors but the rankings remain unchanged.

In Fig. 8, we use the wall data set to plot recognition rates as a function of the number of tests. We clearly see a saturation effect beyond 200 tests for the easy cases and an improvement up to 512 for the others. This tallies with the results of Fig. 6 showing that BRIEF-32 (256 tests) yields near optimal results for the short baseline pairs and that BRIEF-64 (512 tests) is more appropriate for the others.

Influence of Feature Detector. To perform the experiments described above, we used SURF keypoints so that we could run both SURF, U-SURF, and BRIEF on the same points. This choice was motivated by the fact that SURF requires an orientation and a scale and U-SURF a scale, which the SURF detector provides.

However, in practice, using the SURF detector in conjunction with BRIEF would negate part of the considerable speed advantage that BRIEF enjoys over SURF. It would make much more sense to use a fast detector such as [16]. To test the validity of this approach, we therefore recomputed our recognition rates on the Wall sequence using CenSurE keypoints[3] instead of SURF keypoints. As can be seen in Fig. 9-left, BRIEF works even slightly better for CenSurE points than for SURF points.

Orientation Sensitivity. BRIEF is not designed to be rotationally invariant. Nevertheless, as shown by our results on the 5 test data sets, it tolerates small amounts of rotation. To quantify this tolerance, we take the first image of the Wall sequence with $N = 512$ points and match these against points in a rotated version of itself, where the rotation angle ranges from 0 to 180 degrees.

Fig. 9-right depicts the recognition rate of BRIEF-32, SURF, and U-SURF. Since the latter does not correct for orientation either, its behavior is very similar or even a bit worse than that of BRIEF: Up to 10 to 15 degrees, there is little degradation followed by a precipitous drop. SURF, which attempts to

[3] Center Surrounded Extrema, or CenSurE for short, is implemented in OpenCV 2.0 under the alias name *Star detector*, following the shape of the CenSurE detector.

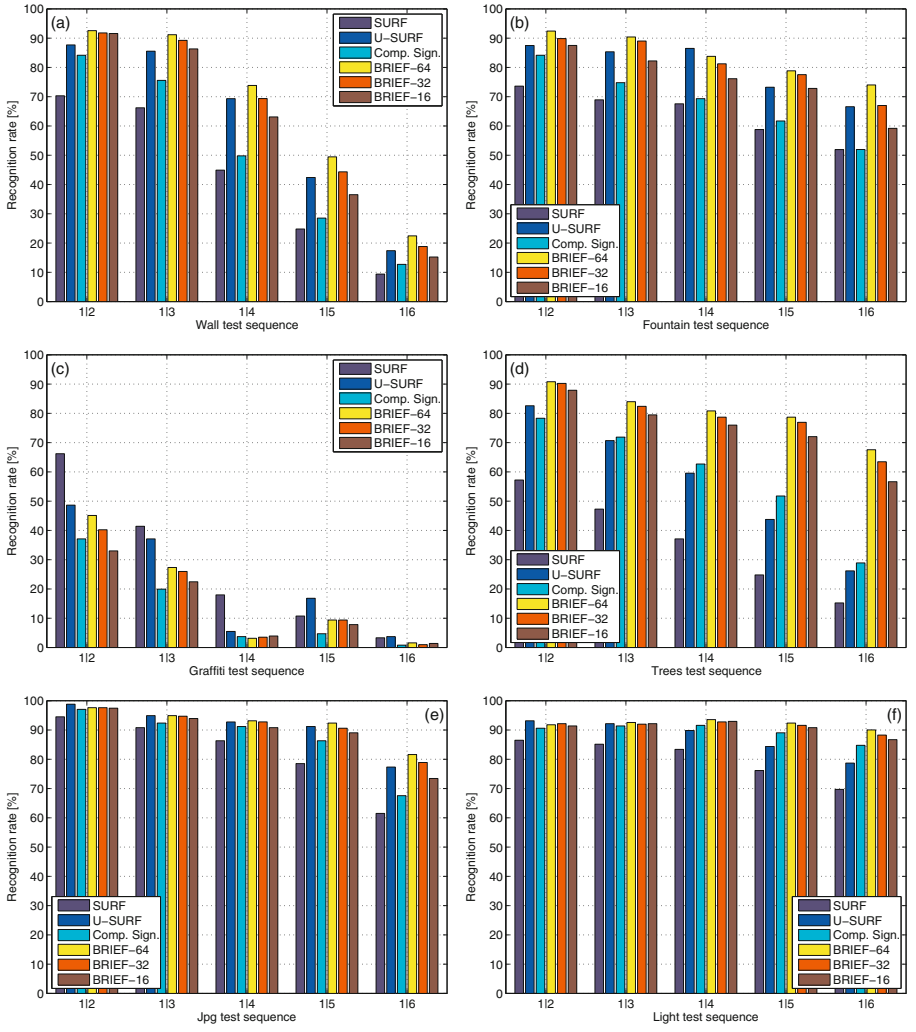

Fig. 6. Recognition rates on (a) Wall (b) Fountain. (c) Graffiti (d) Trees (e) Jpg (f) Light. The trailing 16, 32, or 64 in the descriptor's name is its length in bytes. It is much shorter than those of SURF and U-SURF, which both are 256. For completeness, we also compare to a recent approach called Compact Signatures [7] which has been shown to be very efficient. We obtained the code from OpenCV's SVN repository.

compensate for orientation changes, does better for large rotations but worse for small ones, highlighting once again that orientation-invariance comes at a cost.

To complete the experiment, we plot a fourth curve labeled as O-BRIEF-32, where the "O" stands for orientation correction. In other words, we run BRIEF-32 on an image rotated using the orientation estimated by SURF. O-BRIEF-32 is not meant to represent a practical approach but to demonstrate

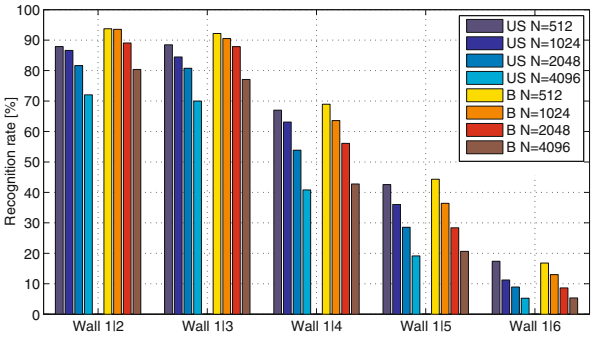

Fig. 7. Influence of the number N of keypoints being considered on the recognition rates. For each of the five image pairs of Wall, we plot on the left side four sets of rates for N being 512, 1024, 2048, and 4096 when using U-SURF, and four equivalent sets when using BRIEF-32. The first are denoted as US and the second as B in the color chart. Note that the recognition rates consistently decrease when N increases but that, for the same N, BRIEF-32 outperforms U-SURF, except in the last image pair where the recognition rate is equally low for both.

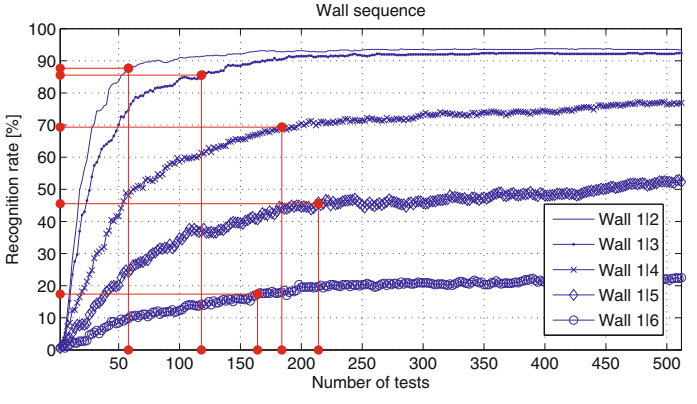

Fig. 8. Recognition rate as a function of the number of tests on Wall. The vertical and horizontal lines denote the number of tests required to achieve the same recognition rate as U-SURF on respective image pairs. In other words, BRIEF requires only 58, 118, 184, 214, and 164 bits for Wall 1|2, ..., 1|6, respectively, which compares favorably to U-SURF's $64 \cdot 4 \cdot 8 = 2048$ bits (assuming 4 bytes/float).

that the response to in-plane rotations is more a function of the quality of the orientation estimator rather than of the descriptor itself, as evidenced by the fact that O-BRIEF-32 and SURF are almost perfectly superposed.

Estimating Speed. In a practical setting where either speed matters or computational resources are limited, not only should a descriptor exhibit the highest possible recognition rates but also be computationally as cheap as

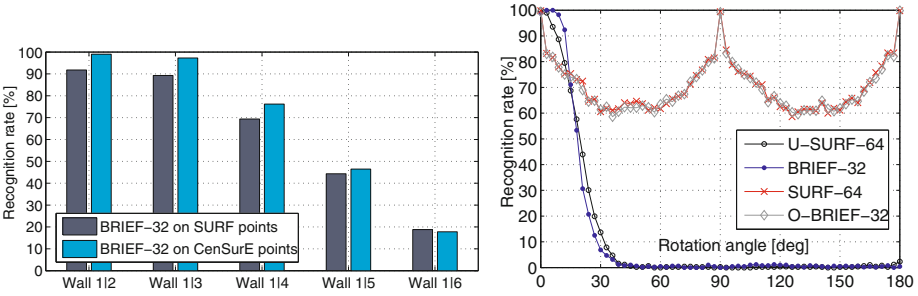

Fig. 9. Left: Using CenSurE keypoints instead of SURF keypoints. BRIEF works slightly better with CenSurE than with SURF keypoints. Right: Recognition rate when matching the first image of the Wall dataset against a rotated version of itself, as a function of the rotation angle.

possible. Matching a number of points between two images typically involves three steps:

1) Detecting the feature points.
2) Computing the description vectors.
3) Matching, which means finding the nearest neighbor in descriptor space.

For affine-invariant methods such as SURF, the first step can involve a costly scale-space search for local maxima. In the case of BRIEF, any fast detector such as CenSurE [16] or FAST [17] can be used. BRIEF is therefore at an advantage there.

The following table gives timing results for the second and third steps for 512 keypoints, measured on a 2.66 GHz/Linux x86-64 machine, in milliseconds:

	BRIEF-16	BRIEF-32	BRIEF-64	SURF-64
Descriptor computation	8.18	8.87	9.57	335
Matching (exact NN)	2.19	4.35	8.16	28.3

As far as building the descriptors is concerned, we observe a 35- to 41-fold speed-up over SURF where the time for performing and storing the tests remains virtually constant. U-SURF being about 1/3 faster than SURF [4], the equivalent number should be an 23- to 27-fold speed increase. Because BRIEF spends by far the most CPU time with smoothing, approximate smoothing techniques based on integral images may yield extra speed. For matching, we observe a 4- to 13-fold speed-up over SURF. The matching time scales quadratically with the number of bits used in BRIEF but the absolute values remain extremely low within the useful range. Furthermore, in theory at least, these computation times could be driven almost to zero using the POPCNT instruction from SSE4.2 [10]. Because only the latest Intel Core i7 CPUs support this instruction, we were unable to exploit it and used a straight-forward SSE2/SSE4.1 implementation instead.

5 Conclusion

We have introduced the BRIEF descriptor that relies on a relatively small number of intensity difference tests to represent an image patch as a binary string.[4] Not only is construction and matching for this descriptor much faster than for other state-of-the-art ones, it also tends to yield higher recognition rates, as long as invariance to large in-plane rotations is not a requirement.

It is an important result from a practical point of view because it means that real-time matching performance can be achieved even on devices with very limited computational power. It is also important from a more theoretical viewpoint because it confirms the validity of the recent trend [18,12] that involves moving from the Euclidean to the Hamming distance for matching purposes.

In future work, we will incorporate orientation and scale invariance into BRIEF so that it can compete with SURF and SIFT in a wider set of situations. Using fast orientation estimators, there is no theoretical reason why this could not be done without any significant speed penalty.

References

1. Mikolajczyk, K., Schmid, C.: A Performance Evaluation of Local Descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence 27, 1615–1630 (2004)
2. Hua, G., Brown, M., Winder, S.: Discriminant Embedding for Local Image Descriptors. In: International Conference on Computer Vision (2007)
3. Lowe, D.: Distinctive Image Features from Scale-Invariant Keypoints. Computer Vision and Image Understanding 20, 91–110 (2004)
4. Bay, H., Ess, A., Tuytelaars, T., Gool, L.V.: Surf: Speeded Up Robust Features. Computer Vision and Image Understanding 10, 346–359 (2008)
5. Tuytelaars, T., Schmid, C.: Vector Quantizing Feature Space With a Regular Lattice. In: International Conference on Computer Vision (2007)
6. Winder, S., Hua, G., Brown, M.: Picking the Best Daisy. In: Conference on Computer Vision and Pattern Recognition (2009)
7. Calonder, M., Lepetit, V., Konolige, K., Bowman, J., Mihelich, P., Fua, P.: Compact Signatures for High-Speed Interest Point Description and Matching. In: International Conference on Computer Vision (2009)
8. Shakhnarovich, G.: Learning Task-Specific Similarity. PhD thesis, Massachusetts Institute of Technology (2005)
9. Ozuysal, M., Calonder, M., Lepetit, V., Fua, P.: Fast Keypoint Recognition Using Random Ferns. IEEE Transactions on Pattern Analysis and Machine Intelligence 32, 448–461 (2010)
10. Intel: SSE4 Programming Reference: software.intel.com/file/18187 (2007)
11. Gionis, A., Indik, P., Motwani, R.: Similarity Search in High Dimensions Via Hashing. In: International Conference on Very Large Databases (2004)
12. Torralba, A., Fergus, R., Weiss, Y.: Small Codes and Large Databases for Recognition. In: Conference on Computer Vision and Pattern Recognition (2008)
13. Salakhutdinov, R., Hinton, G.: Learning a Nonlinear Embedding by Preserving Class Neighbourhood Structure. In: International Conference on Artificial Intelligence and Statistics (2007)

[4] The BRIEF code being very simple, we will be happy to make it publicly available.

14. Taylor, S., Rosten, E., Drummond, T.: Robust feature matching in 2.3s. In: IEEE CVPR Workshop on Feature Detectors and Descriptors: The State of the Art and Beyond (2009)
15. Lepetit, V., Fua, P.: Keypoint Recognition Using Randomized Trees. IEEE Transactions on Pattern Analysis and Machine Intelligence 28, 1465–1479 (2006)
16. Agrawal, M., Konolige, K., Blas, M.: Censure: Center Surround Extremas for Realtime Feature Detection and Matching. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part IV. LNCS, vol. 5305, pp. 102–115. Springer, Heidelberg (2008)
17. Rosten, E., Drummond, T.: Machine Learning for High-Speed Corner Detection. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 430–443. Springer, Heidelberg (2006)
18. Shakhnarovich, G., Viola, P., Darrell, T.: Fast Pose Estimation With Parameter-Sensitive Hashing. In: International Conference on Computer Vision (2003)