# BRIEF: Computing a local binary descriptor very fast

Michael Calonder, Vincent Lepetit, Mustafa Özuysal,
Tomasz Trzcinski, Christoph Strecha, and Pascal Fua
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Computer Vision Laboratory, I&C Faculty
CH-1015 Lausanne, Switzerland
{michael.calonder,vincent.lepetit,tomasz.trzcinski,pascal.fua}@epfl.ch
mustafa.oezuysal@a3.epfl.ch
http://cvlab.epfl.ch

**Abstract**

Binary descriptors are becoming increasingly popular as a means to compare feature points very fast and while requiring comparatively small amounts of memory. The typical approach to creating them is to first compute floating-point ones, using an algorithm such as SIFT, and then to binarize them.

In this paper, we show that we can directly compute a binary descriptor we call BRIEF on the basis of simple intensity difference tests. As a result, BRIEF is very fast *both* to build and to match. We compare it against SURF and SIFT on standard benchmarks and show that it yields comparable recognition accuracy, while running in an almost vanishing fraction of the time required by either.

**Index Terms**

Image processing and computer vision, feature matching, augmented reality, real-time matching

## I. Introduction

Feature point descriptors are 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 capabilities, there is a growing need for local descriptors that are fast to compute, fast to match, memory efficient and yet exhibiting good accuracy.

One way to speed up matching and reduce memory consumption is to work with short descriptors. They can be achieved 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 number without performance loss. An even more drastic dimensionality reduction can be achieved by using hash functions that reduce SIFT descriptors to binary strings, as done in [8], [9]. 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 avoided 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 [10] but without requiring a training phase. We refer to this idea as BRIEF and will describe four different variants {U,O,S,D}-BRIEF. They differ by whether or not they are designed to be scale or orientation invariant. For simplicity we will also use BRIEF to refer to any of the four members of this family of algorithms when we can do so unambigously.

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 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 [11] and NEON [12] instruction sets.

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 on well known benchmark datasets.

## II. Related Work

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

This has been addressed by developing descriptors such as SURF [4] that are faster to compute and match while preserving the discriminative power of SIFT. Like SIFT, SURF relies on local gradient histograms but uses integral images to speed up the computation. Different parameter settings are possible but, since a vector of 64 dimensions already yields good recognition performance, that version has become a *de facto* standard. In the Results section, we will therefore compare BRIEF to both SIFT and SURF.

SURF is faster than SIFT but, since the descriptor is a 64-vector of floating points values, its memory footprint is still 256 bytes. This becomes significant when millions of descriptors need to be stored. There are three main classes of approaches towards reducing this number.

The first one involves dimensionality reduction techniques such as Principal Component Analysis (PCA) or Linear Discriminant Embedding (LDE). PCA is very easy to use and can reduce the 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.

Another 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 could be quantized using only 4 bits per coordinate. The same method could most probably be applied to SURF as well, and makes these approaches competitive with BRIEF in terms of memory need. Quantization is a simple operation that not only yields a memory gain but also faster matching because the distance between short vectors can then be computed very efficiently on modern CPUs. In [13], it is shown that for appropriate parameter settings of the DAISY descriptor [14], PCA and quantization can be combined to reduce its size to 60 bits. Quantization can also be achieved by matching the descriptors to a small number of pre-computed centroids as done in source coding to obtain very good results [15]. However, the Hamming distance cannot be used for matching after quantization because the bits cannot be processed independently, which is something that does not happen with BRIEF.

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) [16] to turn floating-point vectors into binary strings. This is done by thresholding the vectors after multiplication by 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 via 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 [17]. A similar binarization method was also used in [18] with a random rotation matrix on quantized SIFT vectors to speed up matching within Voronoi cells. We also developed a method to estimate a matrix and a set of thresholds that optimize the matching performances when applied to SIFT

vectors [9]. Another way to binarize the GIST descriptor is to use non-linear Neighborhood Component Analysis [17], [19], 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 [10] for classification purposes and were shown to be very powerful in spite of their extreme simplicity. Nevertheless, the present approach is very different from [10] and [20] because it does *not* involve any form of online or offline training.

Finally, BRIEF echoes some of the ideas that were proposed in two older methods. The first one is the *Census transform* [21] that was designed to produce a descriptor robust to illumination changes. This descriptor is non-parametric and local to some neighborhood around a given pixel and essentially consists of a pre-defined set of pixel intensity comparisons in a local neighborhood to a central pixel, which produces a sequence of binary values. The bit string is then directly used for matching. Based on this idea, a number of variants have been introduced such as [22], [23], for example.

The second method, developed by Ojala *et al.* [24] and called Locally Binary Patterns (LBP), builds a Census-like bit string where the neighborhoods are taken to be circles of fixed radius. Unlike Census, however, LBPs usually translate the bit strings into its decimal representation and build an histogram of these decimal values. The concatenation of these histogram values have been found to result in stable descriptors. Extensions are numerous and include [25], [26], [27], [28], [29], [30].

## III. Method

Our approach is inspired by earlier work [10], [31] 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 [31] or a Naive Bayesian classifier [10] to recognize patches seen from different viewpoints. Here, we do away with both the classifier and the trees, and simply create a bit string out of the test responses, which we compute after having smoothed the image patch.

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

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

where $I(\mathbf{p},\mathbf{x})$ 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

| | Rotational invariance | Scale invariance |
|---|---|---|
| SURF | good (sliding orientation window) | good (scale space search) |
| U-SURF | limited | good (scale space search) |
| SIFT | good (orientational HoG) | good (scale space search) |
| U-SIFT | limited | good (scale space search) |
| U-BRIEF | limited | fairly good |
| O-BRIEF | good (using external information) | fairly good |
| S-BRIEF | limited | good (using external information) |
| D-BRIEF | very good (template database) | good (template database) |

TABLE I: Invariance properties of SURF, SIFT, and the four different BRIEF variants.

$n_d$-dimensional bit string that corresponds to the decimal counterpart of

$$\sum_{1 \leq i \leq n_d} 2^{i-1} \, \tau(\mathbf{p}; \mathbf{x}_i, \mathbf{y}_i) \quad . \tag{2}$$

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, space, and accuracy.

The above procedure corresponds to the first BRIEF variant, which we call *upright* BRIEF (U-BRIEF). Naturally and by design, U-BRIEF has limited invariance to in-plane rotation which we will quantify later. However, if an orientation estimate for the feature point at hand is available, the tests can be pre-rotated accordingly and hence made rotation invariant. We refer to this as *oriented* BRIEF (O-BRIEF). In practice, the orientation can be estimated by a feature point detector or, when using a mobile device to capture the image, by the device's gravity sensor. Likewise, we can use the scale information provided by the feature detector to grow or shrink the tests accordingly, yielding the *scaled* BRIEF (S-BRIEF) variant of BRIEF. In the absence of externally supplied scale and orientation data, O-BRIEF and S-BRIEF are mostly of academic interest since they would have to rely on a potentially slow feature detector to provide the necessary information. Such is not the case of the fourth and final BRIEF variant, which we refer to as database BRIEF (D-BRIEF). The idea behind D-BRIEF is to achieve full invariance to rotation and scaling by building a database of templates, which are the U-BRIEF descriptors for rotated versions of patches to be recognized. This needs to be done only once and can be done in real-time. Matching a new patch against a database then means matching against all rotated and scaled versions. However, because computing the distance between binary strings is extremely fast, this remains much faster than using either SIFT or SURF and we demonstrate the viability of this approach in an application in section V-E, running at 25 Hz or more. We summarize the different BRIEF variants and its competitors in table I.

In the remainder of the paper, we will add a postfix to BRIEF, 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 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 V. 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* that will be precisely defined at the beginning of section V. In short, for both images of an image pair that is to be matched and for a given number of corresponding keypoints between them, it quantifies how often the correct match can be established. In the case of BRIEF, the nearest neighbor is identified using the Hamming distance measure. The recognition rate can be computed reliably because the scene is planar and the homography between images is known and can therefore be used to check whether points truly correspond to each other or not.

### A. 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 effect on the recognition rate of increasing amounts of Gaussian smoothing. The variance of the Gaussian kernel has been varied in $[0, 2.75]$. The more difficult the matching, the more important smoothing becomes to achieving good performance. Furthermore, the recognition rates remain relatively constant 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 $7 \times 7$ pixels be necessary and sufficient.

While the Gaussian kernel serves this purpose, its non-uniformity makes it fairly expensive to evaluate, compared to the much cheaper binary tests. We therefore experimented by a box filter. As can be seen in Fig. 1, no accuracy is lost when replacing the Gaussian filter by a box filter, which confirms a similar finding of [26]. The latter, however, is much faster to evaluate since integral images offer an effective way to computing box filter responses independently of the filter size using only three additions.

### B. Spatial Arrangement of the Binary Tests

Generating a length-$n_d$ bit vector leaves many options for selecting the 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

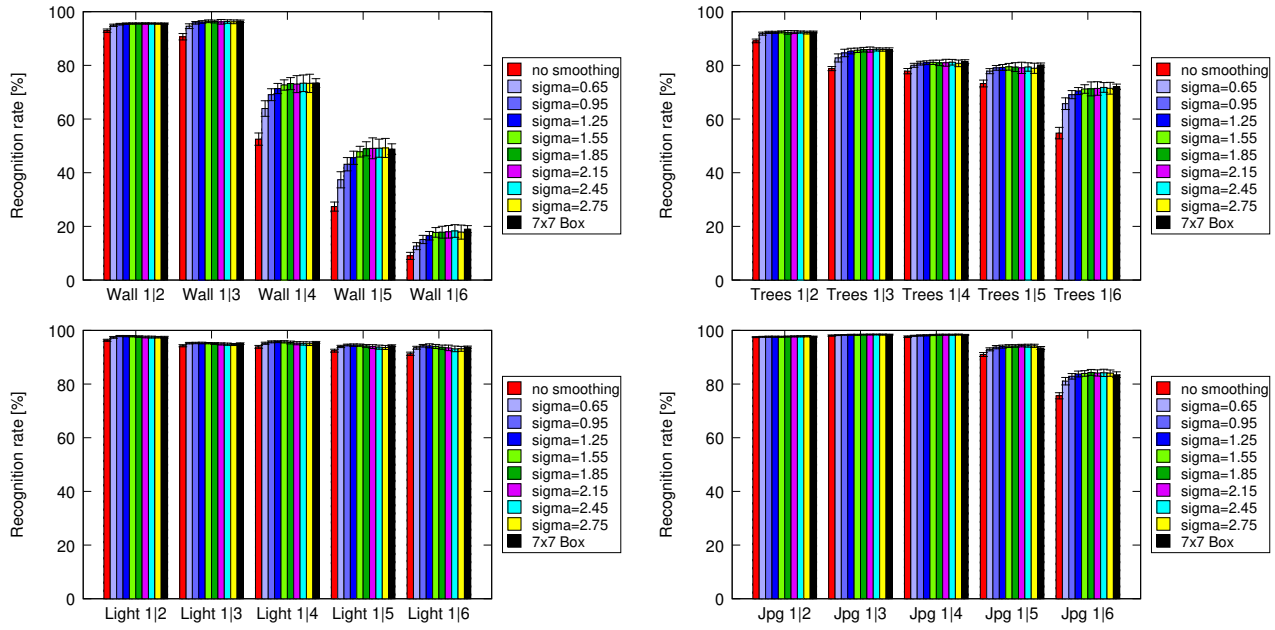



Fig. 1: Recognition rates and their variances for different amounts of smoothing and different benchmark image sets. Each group of 10 bars represents the recognition rates in one specific image pair for increasing levels of smoothing. Especially for the hard-to-match pairs, which are those on the right side of the plot, smoothing is essential in slowing down the rate at which the recognition rate decays. Note that the first 9 bars corresponds to the Gaussian kernel where the rightmost bar of each group corresponds to using a box filter. It appears that using a box filter does not harm accuracy.

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.

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. This is motivated by the fact that pixels at the patch center tend to be more stable under perspective distortion than those near the edges. 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, like the previous, 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

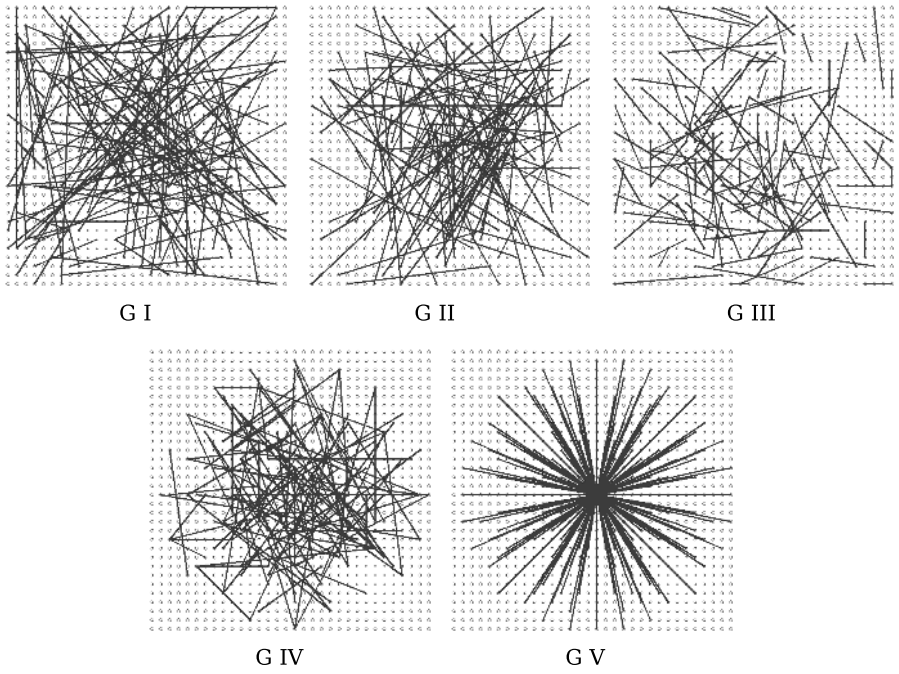



Fig. 2: Different approaches to choosing the test locations. All except the righmost one are selected by random sampling. Showing 128 tests in every image.

spatial quantization. This geometry is of interest because it reflects the behavior of BRIEF when the underlying tests are sampled from a more coarse grid than that of the pixels.

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. Note that this geometry corresponds to that of the Census transform [21] discussed in section II.

For each of these test geometries we compute the recognition rate on several benchmark image sets and show the results 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.

### *C. 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

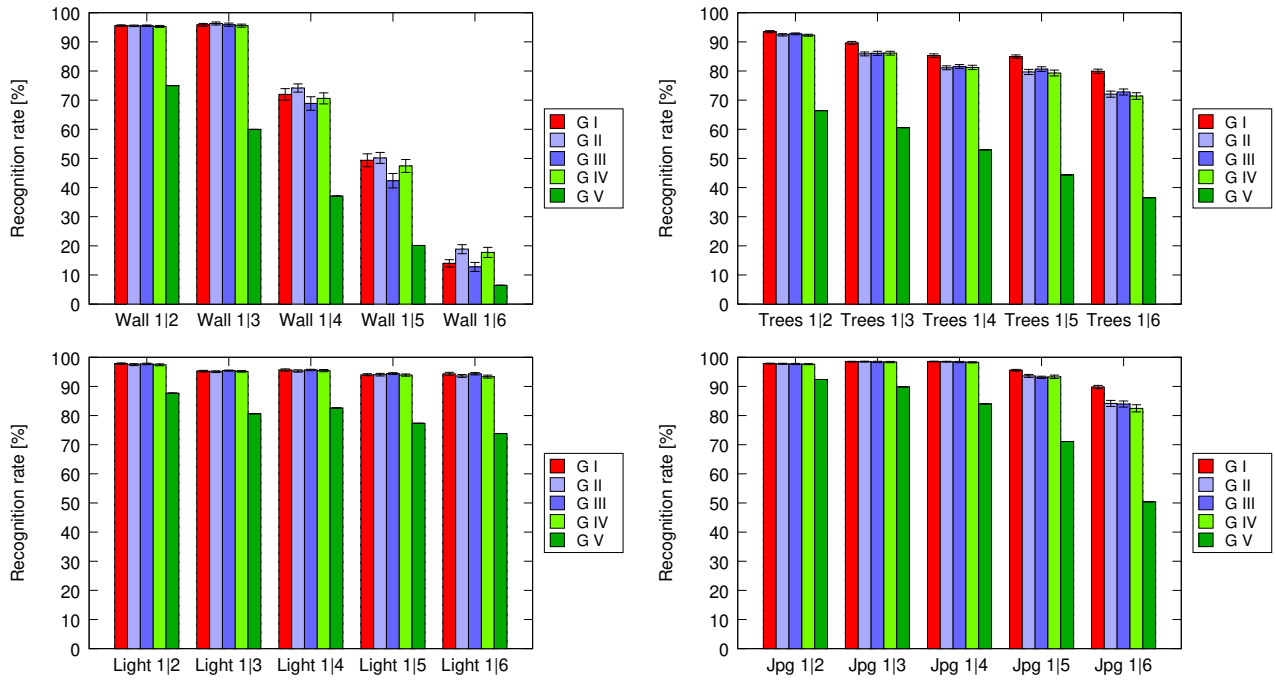



Fig. 3: Recognition rates and their variances for the five different test geometries introduced in Section III-B and different benchmark image sets.

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 V, this is of course what happens with recognition rates being higher in the first pairs of the Wall sequence than in the subsequent ones.

## IV. Analysis

In this section, we use again the Wall dataset of Fig. 10a to analyze the influence of the various design decisions we make when implementing the four variants of the BRIEF descriptor.

### *A. Descriptor Length*

Since many practical problems involve matching a few hundred feature points, we first use $N = 512$ of them to compare the recognition rate of U-BRIEF using either 128, 256, or 512 tests, which we denote as U-BRIEF-16, U-BRIEF-32, and U-BRIEF-64. The main purpose here is to show the dependence of the recognition rate on the descriptor length which is why we only include the recognition rate of U-SURF-64 in the plots. Recall that U-SURF returns 64 floating point numbers requiring 256 bytes of storage–this is 4 to 16 times more than BRIEF.

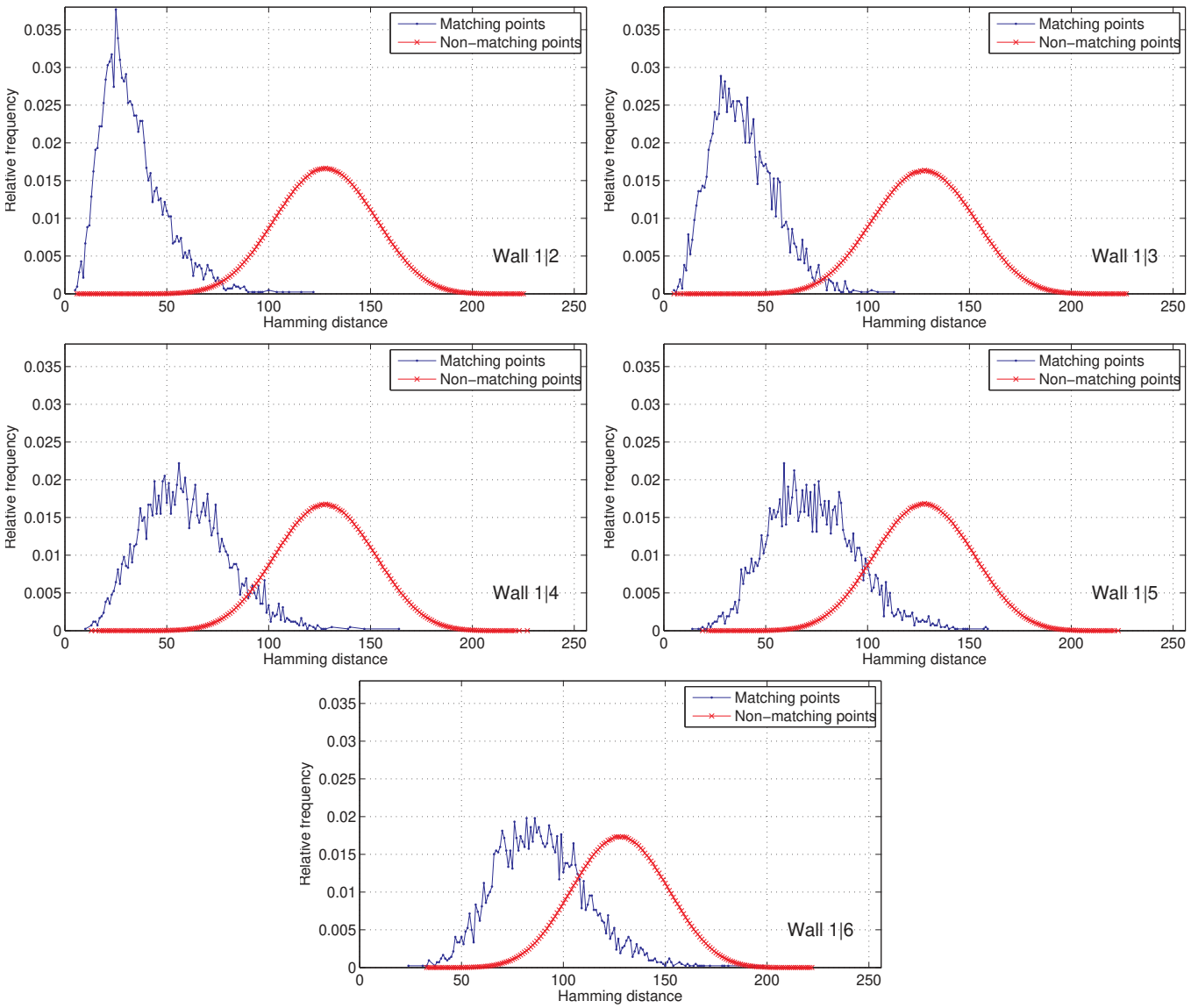



Fig. 4: Distributions of Hamming distances for matching pairs of points (thin blue lines) and for non-matching pairs (thick red lines) in each of the five image pairs of the Wall dataset. They are most separated for the first image pairs, whose baseline is smaller, ultimately resulting in higher recognition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

In Fig. 5 we use Wall to plot the recognition rate as a function of the number of tests. We clearly see a saturation effect beyond 200 tests for the more easy cases and an improvement up to 512 for the others. The motivates the choice for the default descriptor BRIEF-32.

We would like to convince the reader that this behavior is *not* an artifact arising from the fairly low number of feature points used for testing and that BRIEF scales reasonably with the number of features to match. To this end we repeat the testing for values of $N$ ranging from 512 to 4096, adding new feature points by decreasing the detection threshold. We found similar recognition rates, as shown in in Fig. 6. Note how the rates drop with increasing $N$ for *all* the descriptors, as expected; however, the rankings remain unchanged. This behavior changes when $N$ becomes even larger, as discussed in Section V.C.

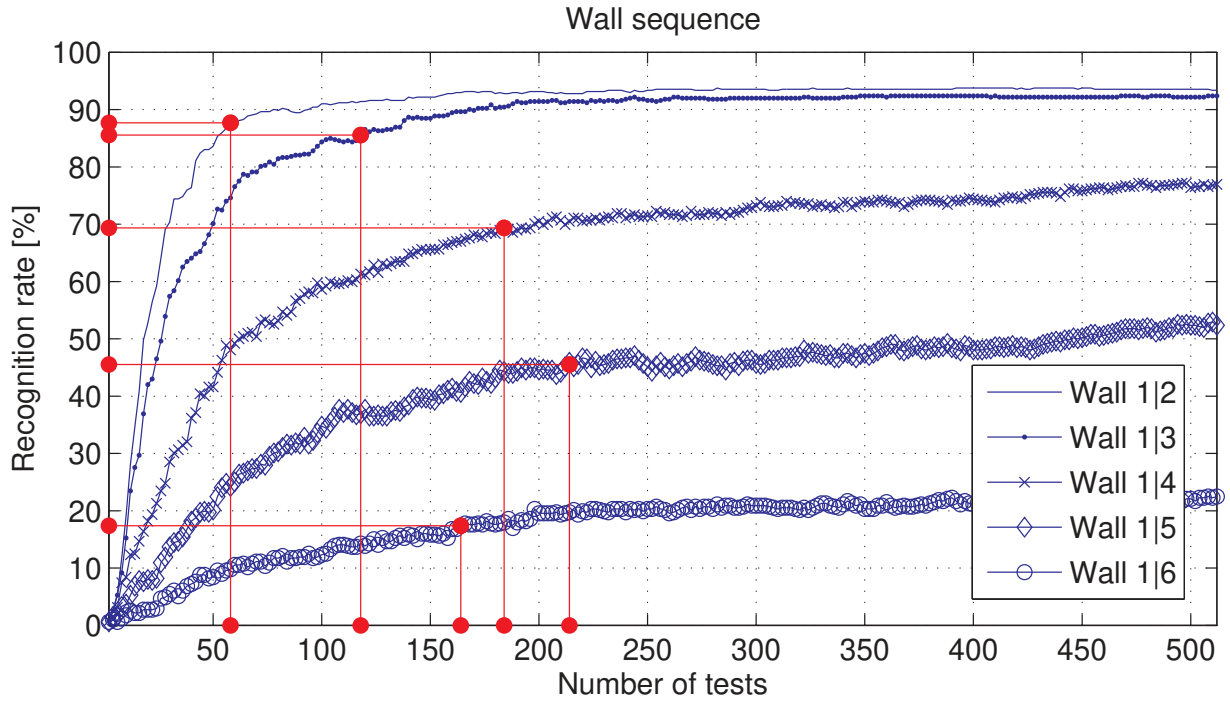



Fig. 5: Varying the number of tests $n_d$ of U-BRIEF. The plot shows the recognition rate as a function of $n_d$ on Wall. The vertical and horizontal lines indicate the number of tests required to achieve the same recognition rate as U-SURF on respective image pairs. In other words, U-BRIEF requires 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).

### *B. Orientation Sensitivity*

U-BRIEF is not designed to be rotationally invariant. Nevertheless, as shown by our results on the six 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 is varied between 0 and 180 degrees.

Figure 7 compares the recognition rate of SURF (●), three versions of BRIEF-32 (×, ◇, △), and U-SURF (○). Since the latter does not correct for orientation either, its behavior is very similar or even slightly worse than that of U-BRIEF: Up to 15 degrees there is little degradation, however, this is followed by a precipitous drop. SURF, which attempts to compensate for orientation changes, does better for large rotations but worse for small ones, highlighting once again that orientation invariance comes at a cost. The typical shape of the SURF-64 curve is a known artifact arising from approximating the Gaussian for scale-space analysis, degrading the performance under in-plane rotation for odd multiples of $\pi/4$ [32], [33].

To complete the experiment, we plot two more curves. The first corresponds to O-BRIEF which is

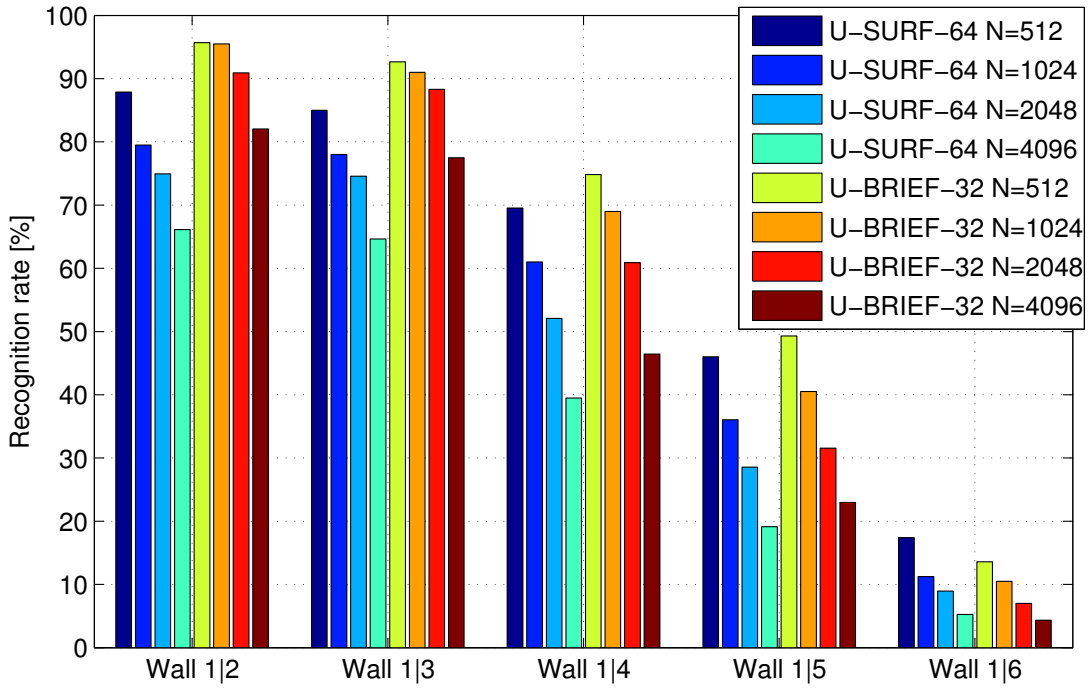



Fig. 6: Scalability. For each of the five image pairs of Wall, we plot on the left side four sets of rates for $N$ being 512, 1024, 2048, and 4096 when using U-SURF-64, and four equivalent sets when using U-BRIEF-32. Note that the recognition rate consistently decreases with increasing $N$. However, for the same $N$, BRIEF outperforms SURF, except in the last image pair where the recognition rate is equally low for both.

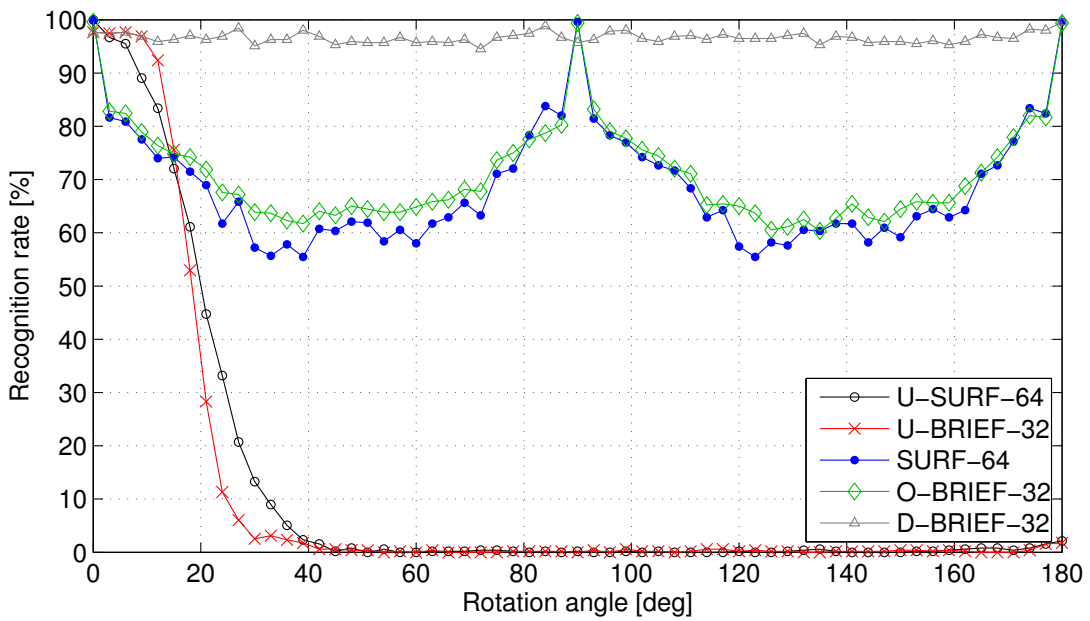



Fig. 7: Recognition rate when matching the first image of the Wall dataset against a rotated version of itself, as a function of the rotation angle.

identical to U-BRIEF except that the tests are rotated according to the orientation estimation of SURF. O-BRIEF-32 is not meant to represent a practical approach but to demonstrate 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 the O-BRIEF-32 and SURF curves are almost perfectly superposed.

The second curve is labeled D-BRIEF-32. It is applicable to problems where full rotational invariance is required while no other source for orientation correction is available–a case seen less and less often. As discussed at the beginning of section III, D-BRIEF exploits BRIEF's characteristic speed by pre-computing a database of U-BRIEF descriptors for several orientations, matching incoming frames against the entire database and picking the set of feature points with the probably highest number of correct matches. Doing so is practically viable and lets applications process incoming frames at 30 Hz. As a welcome side-effect, D-BRIEF also avoids the characteristic accuracy loss observed in scale space-based methods based on approximations of Gaussians as can be seen in Fig. 7.

### *C. Sensitivity to Scaling*

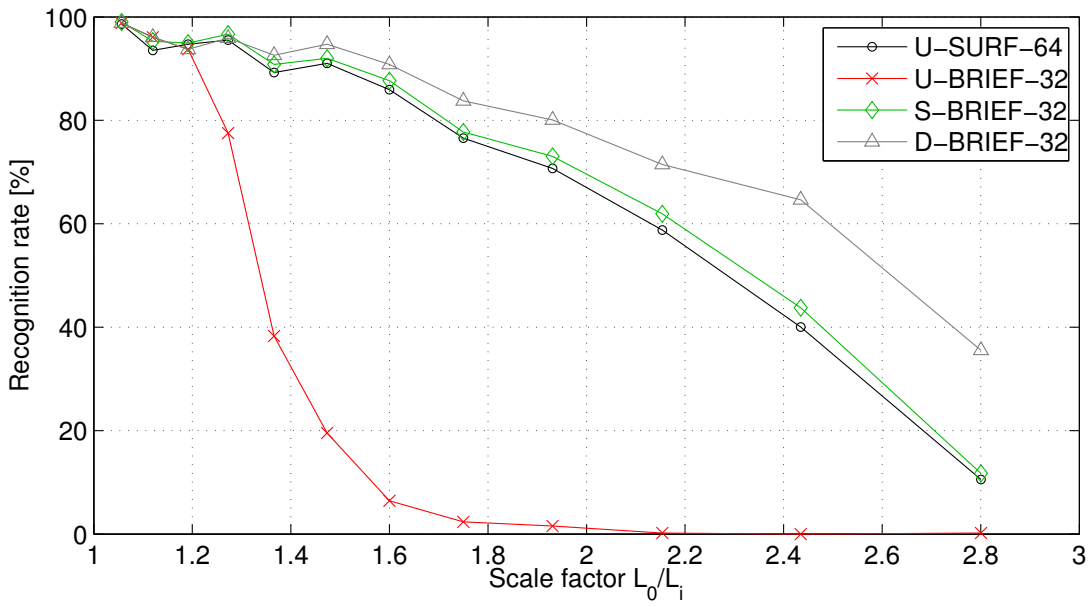



Fig. 8: Recognition rate under (down)scaling using the first image from Wall. $L_0$: original image width, $L_i$: width of matched image $i$.

By default, U- and O-BRIEF are not taking scale information into account, although the smoothing provides a fair amount of robustness to scale changes as confirmed in the experiments below. We assess the extent of this tolerance in Figure 8 and plot the recognition rate as a function of the image scaling factor. The first two curves in the plot, U-SURF-64 and U-BRIEF-32, show that U-BRIEF is inherently

robust to small and intermediate scale changes, at least to some limited extent. Larger changes result in a notable loss in accuracy.

Scale information, if available from a detector, can be readily integrated into BRIEF, which we demonstrate with the third curve in the plot, labeled S-BRIEF-32. The scale of a keypoint is used to adjust both the area which the tests cover and the smoothing kernel size before the tests are applied. Apparently, this results in a descriptor that behaves similarly to SURF under scale changes where potentially using a larger-than-standard kernel size for smoothing comes at no cost, thanks to integral images used for computing the box filter response.

The D-BRIEF curve in Fig. 8 describes the case where no scale information is available. As for rotational invariance, we can resort to pre-computing a database of scaled versions of the original image. The scales are chosen such that they cover the expected range of observable ones, which is in the present case $[1.0, 0.60, 0.43, 0.33]$. In a more general setup this interval may be centered around 1.0 to account for larger and smaller scales. However, since a scaling factor of 0.33 is smaller than what practical applications typically demand, that scale can be removed and another, larger one added. Therefore, 4 or even 3 scales might suffice for most applications.

Alternatively, a point detector that also returns a scale estimate can be used to achieve such scale-invariant behavior and in practice, a faster detector such as CenSurE [34] would be given preference over the still fairly slow Fast Hessian detector underlying SURF, let alone SIFT's DoG detector. In any case, changing the underlying point detector does not influence BRIEF's behavior, as we are going to show in the next section.

### *D. Robustness to Underlying Feature Detector*

To perform the experiments described above, we used SURF keypoints so that we could run both 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 CenSurE [34]. To test the validity of this approach, we therefore re-computed our recognition rates on the Wall sequence using CenSurE features[1] instead of SURF keypoints. As can be seen in Figure 9, U-BRIEF works even slightly better for CenSurE points than for SURF points.

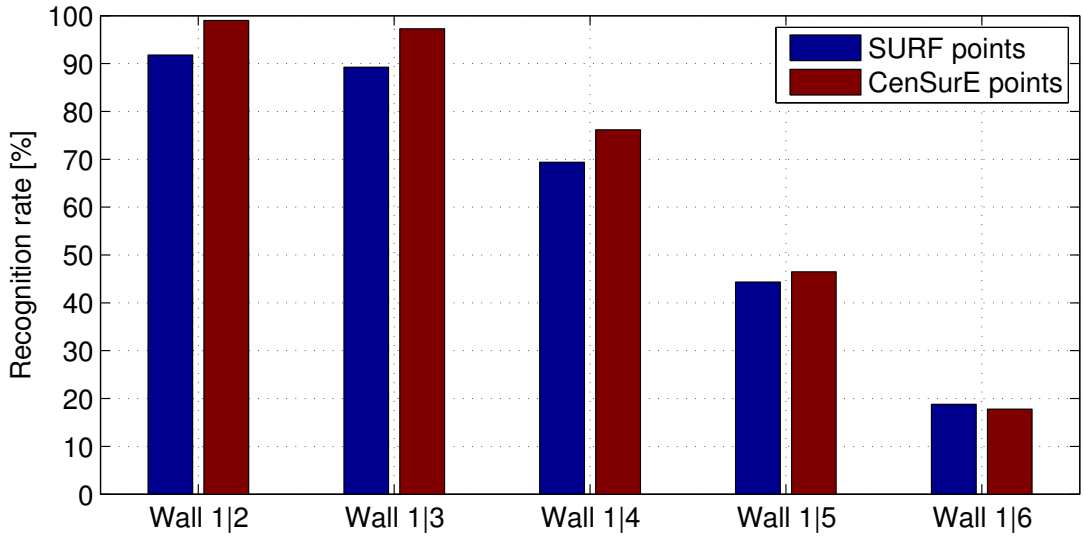



Fig. 9: Using CenSurE feature points instead of SURF ones. U-BRIEF works slightly better with CenSurE feature points rather than with those from SURF.

| | | {U,O,S}-BRIEF-X (CPU) | | | U-SURF-64 | |
|---|---|---|---|---|---|---|
| | | X=16 | X=32 | X=64 | CPU | GPU |
| Detection | | 1.61 | 1.61 | 1.61 | 160 | 6.93 |
| Description | Gaussian | 6.59 | 6.81 | 7.40 | 101 | 3.90 |
| | Uniform | 6.05 | 6.07 | 6.30 | | |
| | Uniform-Integral† | 1.74 | 2.69 | 4.65 | | |
| | Uniform-Integral‡ | 1.01 | 1.98 | 3.91 | | |
| Matching | $n^2$-Matching w/o `POPCNT` | 2.62 | 5.03 | 9.56 | 28.3 | 6.80 |
| | $n^2$-Matching w/ `POPCNT` | 0.433 | 0.833 | 1.58 | | |

TABLE II: CPU wall clock time in [ms] for {U,O,S}-BRIEF of length 16, 32, or 64 and U-SURF-64 on 512 points, median over 10 trials. Values for matching using D-BRIEF could be obtained by multiplying the given value by the number of orientations and/or scales used. The four row of numbers in the *Description* part of the table correspond to four different methods to smooth the patches before performing the binary tests. The two rows of numbers in the *Matching* part of the table correspond to whether or not the POPCNT instruction is part of the instruction set of the CPU being used. Neither the kernel choice or the presence of the POPCNT instruction affect SURF. †Integral image pre-computed. ‡Integral image re-used from a detector such as CenSurE.

### *E. Speed Estimation*

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

[1]Center Surrounded Extrema [34], or CenSurE for short, has been implemented in OpenCV 2.0 with some improvements and hence received a new name: *Star detector*, hinting at the shape of the bi-level filters.

possible. Table II gives timing results in milliseconds measured on a 2.66 GHz/Linux x86-64 laptop for 512 keypoints. We give individual times for the three main computational steps depending on various implementation choices. We also provide SURF timings on both CPU and GPU for comparison purposes.

1) **Detecting feature points.** For scale-invariant methods such as SURF or SIFT, the first step can involve a costly scale-space search for local extrema. In the case of BRIEF, any fast detector such as CenSurE [34] or FAST [35], [36] can be used. BRIEF is therefore at an advantage there.
2) **Computing descriptors.** When computing the BRIEF descriptor, the most time-consuming part of the computation is smoothing the image-patches. In Table II, we provide times when using Gaussian smoothing, simple box filtering, and box filtering using integral images. The latter is of course much faster. Furthermore, as discussed in Section III-A and shown in Fig. 1, it does not result in any matching performance loss.

   In the specific case of the BRIEF-32 descriptor, we observe a 38-fold speed-up over U-SURF, without having to resort to a GPU which may not exist on a low-power device.
3) **Matching descriptors.** This involves performing nearest neighbor search in descriptor space. For benchmarking purposes, we used the simplest possible matching strategy by computing distances of every descriptor against every other. Even though the computation time is quadratic in terms of the number of points being matched, BRIEF is fast enough to so that it remains practical for real-time purposes over a broad range. Furthermore, for larger point sets, it would be easy to incorporate a more elaborate matching strategy [37].

   More technically, matching BRIEF's bit vectors mainly involves computing Hamming distances. The distance between two BRIEF vectors $\mathbf{b}_1$ and $\mathbf{b}_2$ is computed as POPCNT$(\mathbf{b}_1$ XOR $\mathbf{b}_2)$ where POPCNT is returning the number of bits set to 1. Older CPUs require POPCNT to be implemented in native C++ and BRIEF descriptors can be matched about 6 times faster than their SURF counterparts, as shown in Tab. II. Furthermore, since POPCNT is part of the SSE4.2 [11] instruction set that newer CPUs, such as the Intel Core i7, support, the corresponding speed-up factor of BRIEF over SURF becomes 34. Note that recent ARM processors, which are typically used in low-power devices, also have such an instruction, called VCNT [12].

In short, for all three steps involved in matching keypoints, BRIEF is significantly faster than the already fast SURF without any performance loss and without requiring a GPU. This is summarized graphically in Fig. 16.

## V. 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 SURF-64 version, which returns a 64-dimensional floating point vector and requires 256 bytes of storage. Because {U,S}-BRIEF, unlike SURF, do not correct for orientation, we also compare against U-SURF-64 [4], where the U also stands for *upright* and means that orientation is ignored [4].

As far as D-BRIEF is concerned, because its database of rotated patches has to be tailored to a particular application, we demonstrate it for real-time Augmented Reality purposes.

### A. Datasets

We compare the methods on real-world data, using the six publicly available test image sequences depicted by Fig. 10a. They are designed to test robustness to typical image disturbances occurring in real-world scenarios. They include

- viewpoint changes: Wall[2], Graffiti[2], Fountain[3], Liberty[4];
- compression artifacts: Jpg[2];
- illumination changes: Light[2];
- image blur Trees[2].

For all sequences except Libertywe consider five image pairs by matching the first image to the remaining five. The five 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. The pairs from Jpg, Light, and Treesare similarly sorted in order of increasing difficulty.

The Wall and Graffiti scenes being planar, the images are related by homographies that we use to compute the ground truth. Both sequences involve substantial out-of-plane rotation and scale changes. Although the Jpg, Light, and Trees scenes are non-planar, the ground truth can also be represented accurately enough by a homography that is close to identity since there is almost no change in viewpoint.

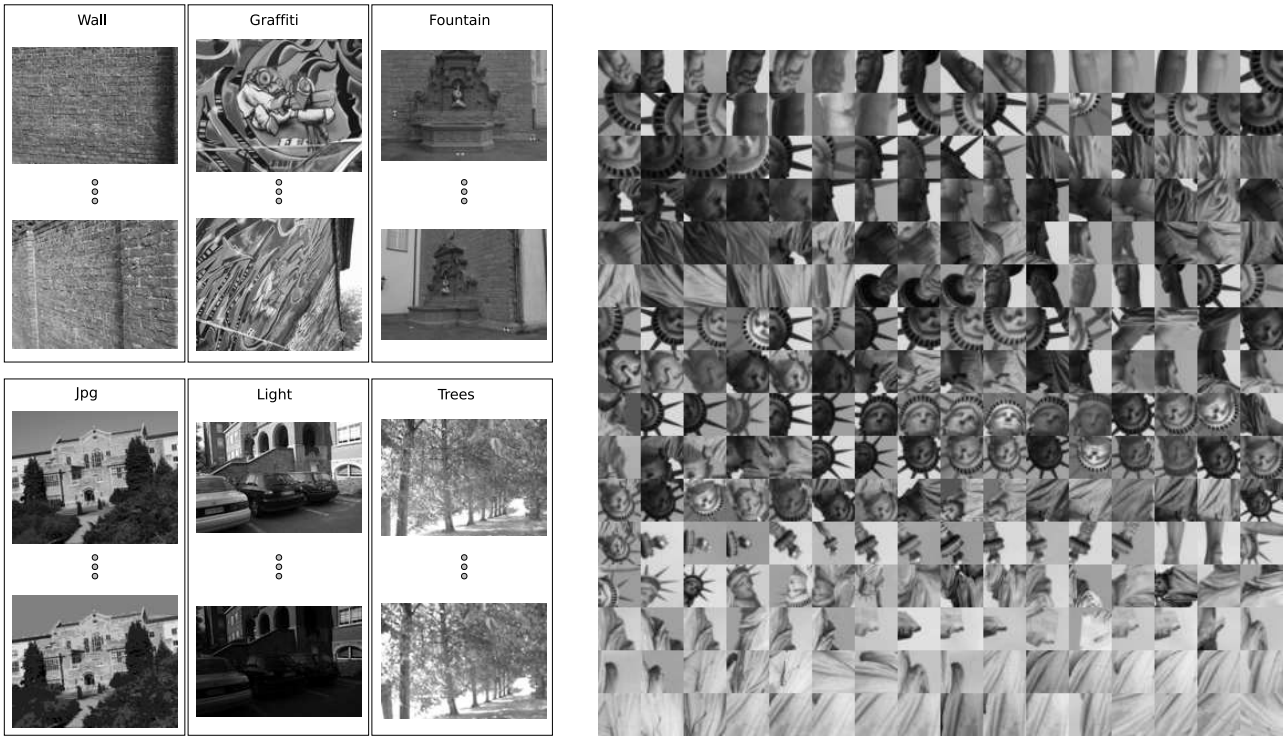
By contrast, the Fountain scene is fully three-dimensional and contains substantial perspective distortion. As this precludes using a homography to encode the ground truth, we use instead accurate laser scan data, which is also available, to compute the flow field for each image pair.

These six datasets are representative of the challenges that an algorithm designed to match image pairs might face. However, they only involve relatively small numbers of keypoints to be matched in each individual image. To test the behavior of our algorithm on the much larger libraries of keypoints that applications such as image retrieval might involve, we used the publicly available Liberty dataset depicted by Fig. 10b. It contains over 400k scale and rotation normalized patches sampled from a 3D reconstruction

[2]http://www.robots.ox.ac.uk/~vgg/research/affine [38]

[3]http://cvlab.epfl.ch/~strecha/multiview [39]

[4]http://cvlab.epfl.ch/~brown/patchdata/patchdata.html [13]




(a) (b)

Fig. 10: (a) Test image sequences. The left column shows the first image, the right column the last image. Each sequence contains 6 images and hence we consider 5 image pairs per sequence by matching the first one against all subsequent images. This way, except for Graffiti, the pairs are sorted in ascending difficulty. More details in the text. (b) Some of the images patches from the Liberty dataset used in our evaluation.

of the New York Statue of Liberty. The 64×64 patches are sampled around interest points detected using Difference of Gaussians and the correspondences between patches are found using multi-view stereo algorithm. The dataset created this way represent substantial perspective distortion of the 3D statue seen under various lighting and viewing conditions.

### *B. Comparing Performance for Image Matching Purposes*

This section finally compares the BRIEF-32 descriptor against SIFT and SURF. For SIFT we employ the `siftpp` implementation by Vedaldi [40] that computes the standard 128-vector of real numbers, requiring 512 bytes of memory. For SURF we rely on the latest OpenCV implementation [41] where we use the standard SURF-64 version. The default parameter settings are used.

Although BRIEF can be computed on arbitrary kinds of feature points, this comparison is based on SIFT or SURF points, depending on which of the two we are actually comparing BRIEF to. While doing so removes the direct comparability of SIFT's and SURF's recognition rates, it is vital for a fair comparison to BRIEF. In practical applications, however, we would rather employ FAST [35] or CenSurE [34] features

for computational efficiency. While FAST is faster to compute than CenSurE, the latter provides BRIEF with integral images that can be exploited for additional speed, see Tab. II.

We measure the performance of the descriptors using Nearest Neighbor (NN) correctness. We refer to this measure as *recognition rate* $\rho$. It simply computes the fraction of descriptors that are NN in feature space while belonging to the same real-world point, and hence being a true match. Given an image pair, this is done as follows.

- Pick $N$ feature points from the first image, infer the $N$ corresponding points in the second image using the known geometric relation between them, and compute the $2N$ associated descriptors using the method under consideration.
- For each point in the first set, find the NN in the second one and call it a match.
- Count the number of correct matches $n_c$ and compute $\rho := n_c/N$.

Note that not detecting feature points in the second image but using geometry instead prevents repeatability issues in the detector from influencing our results. Since we apply the same procedure to all methods, none is favored over any other.

The recognition rate is of great importance when the feature vectors are matched using exhaustive NN search. This is feasible in real-time when the number of features is not exceeding a few hundred or maybe one thousand as typical for SLAM or object detection applications. Even though approximate NN schemes exist, exact search is done whenever possible because the accuracy of such schemes quickly deteriorates in high-dimensional spaces.

In the following we show a large number of graphs to compare the different variants of BRIEF to SURF and SIFT. More specifically we use 6 image sequences, each containing 5 image pairs, to compare

- U-SURF and U-BRIEF that do not correct for orientation, and
- SIFT, SURF, and O-BRIEF that do using the same orientation estimate.

This means that a thorough comparison encompasses $2 \times 6 \times 2 = 24$ recognition rate plots. The factor 2 from SIFT and SURF cannot be dropped because—although the two are conceptually very similar—they are not working with the same set of feature points and forcing either to use the other's will distort the results. In some cases we use points extracted by SIFT and in others by SURF.

In the title of each plot we give the name of the test sequence and the number of points that were matched for each image pair. When the detected points in a pair give rise to more than 1000 ground truth-compatible matches, 1000 of them are selected at random; otherwise the maximum available number of matches is used. Keeping the number of points constant for all evaluations makes comparisons easier.

Figs. 11 through 14 show the recognition rate of BRIEF together with those of SURF and SIFT for comparison. Each of the four figures contains six graphs, one for each image sequence. Based on these

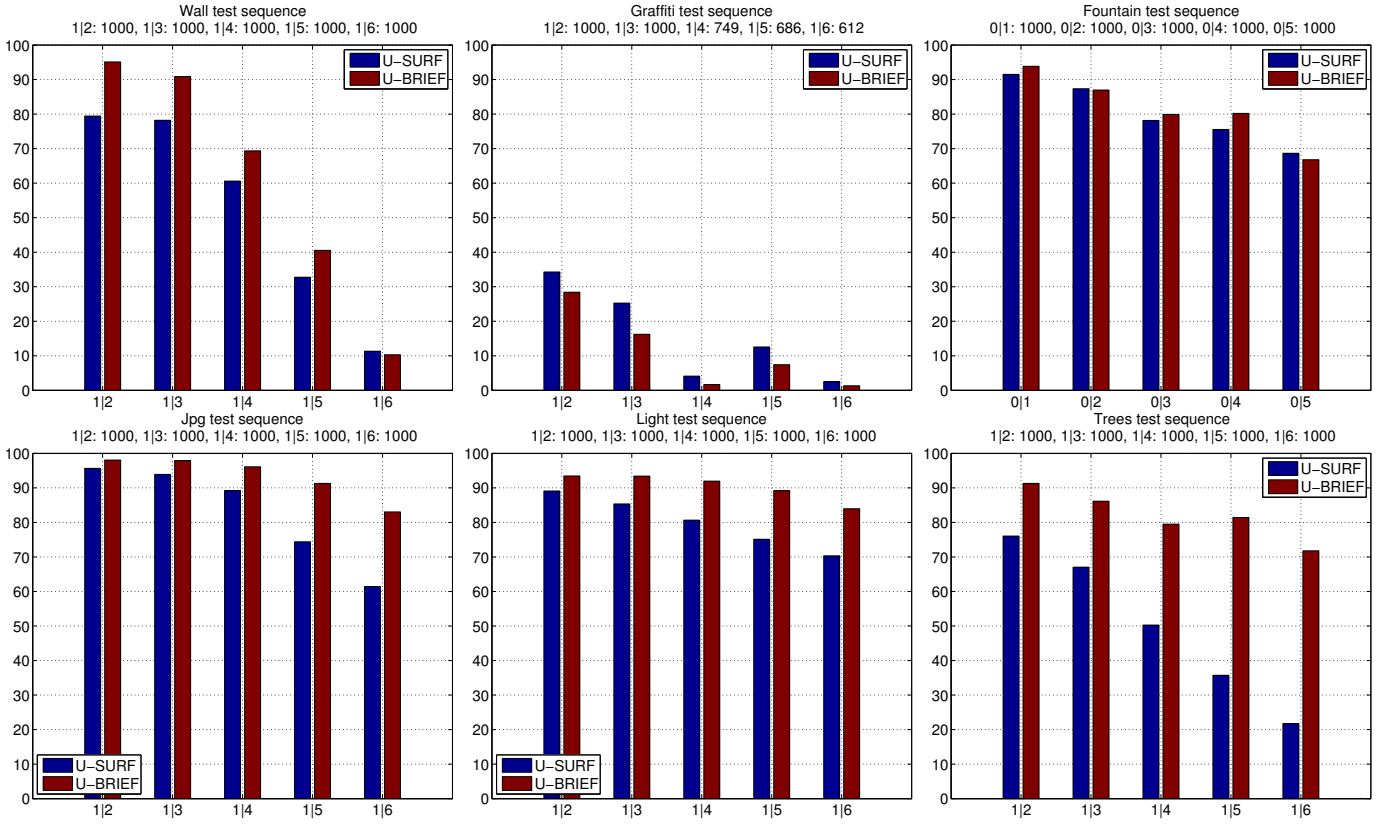



Fig. 11: Recognition rate vs. image pairs of all sequences. Comparing to U-SURF.

plots, we make the following observations:

- On all sequences except on Graffiti, U-BRIEF outperforms U-SURF, as can be seen in Fig. 11.
- Using orientation correction, O-BRIEF also outperforms SURF except on Graffiti. O-BRIEF does slightly better than U-BRIEF on Graffiti but not much (Figure 12).
- SURF and SIFT are both substantially more sensitive to image blur than BRIEF is, and slightly more sensitive to compression and illumination artifacts (Figures 11 to 14).
- O-BRIEF works better with SURF's orientation assignment than with that of SIFT features (Figures 12 and 14).
- SIFT is more robust to viewpoint changes than both BRIEF and SURF.

In summary, the rough ordering 'SIFT $\gtrsim$ BRIEF $\gtrsim$ SURF' applies in terms of robustness to common image disturbances but, as we will see below, BRIEF is much faster than both. Note, however, that these results were obtained for $N = 1000$ keypoints per image. This is representative of what has to be done when matching two images against each other but not of the more difficult problem of retrieving keypoints within a very large database. We now turn to this problem.

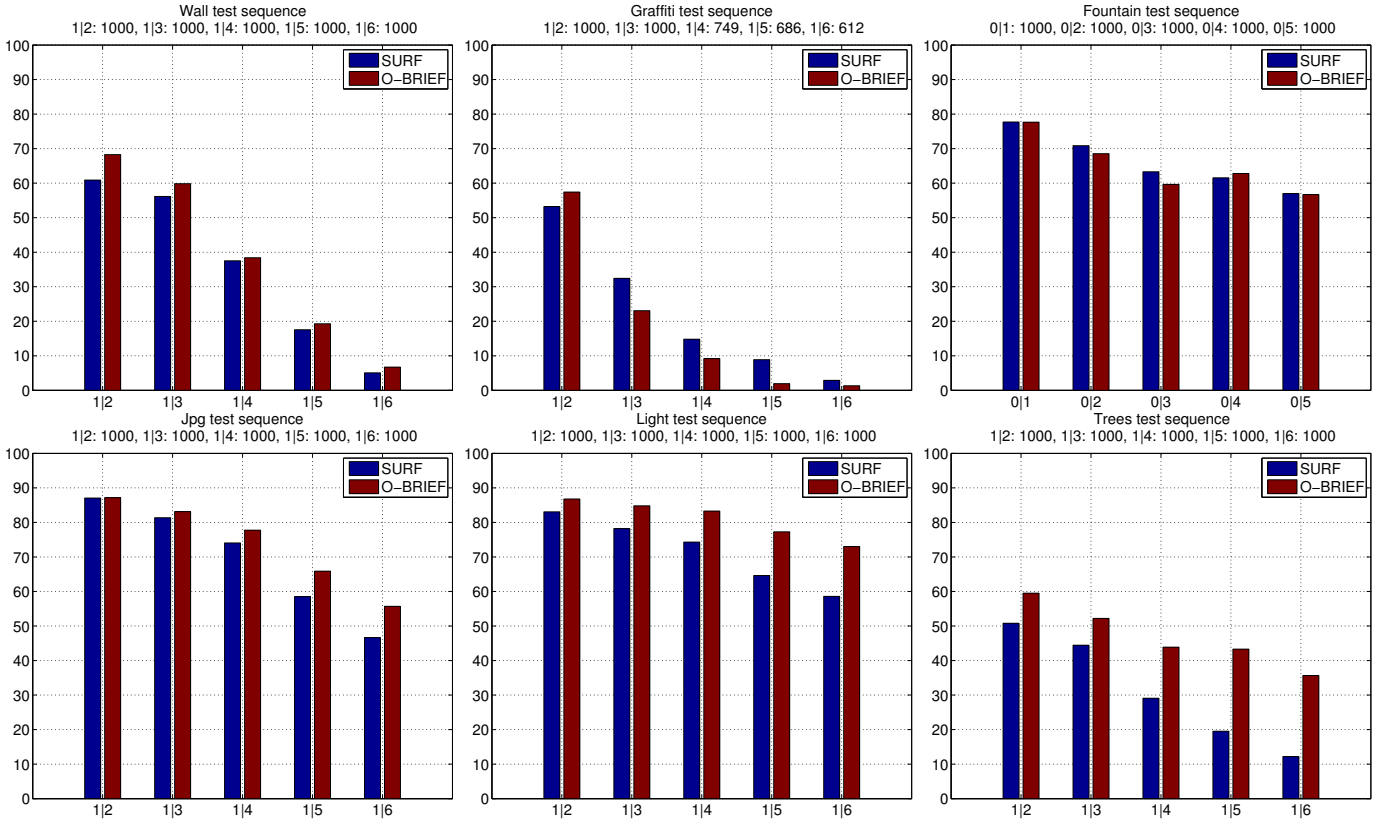



Fig. 12: Recognition rate vs. image pairs of all sequences. Comparing to SURF.

### C. *Influence of the Number of Keypoints to be Matched*

To demonstrate what happens when the number of keypoints to be matched increases, we use the Liberty dataset that contains over 400k patches and the corresponding ground-truth. We extracted subsets of corresponding patches of cardinalities ranging from 1 to 7000. To use the same metric as in Section V-B, we report recognition rates as proportions of the descriptors for which the nearest-neighbor in descriptor space corresponds to the same real-world point.

Since the Libertydataset consists of patches extracted around interest points no feature detection is needed and descriptors are computed for the central point of the patch. Since both SIFT and SURF perform sampling at a certain scale, we optimized this scaling parameter for optimal performance. For a fair comparison, we did the same for BRIEF by adjusting the size of the patch that was used to create the descriptor.

Fig. 15 and Table III depict the resulting recognition rates. All descriptors perform less and less well with increasing dataset size and, even for small cardinalities, performances are consistently worse than those reported in Section V-B because the data set is much more challenging.

On this dataset, SIFT performs best. BRIEF-32 does better than U-SURF for cardinalities up to $N = 1000$ and worse for larger ones. To outperform U-SURF it becomes necessary to make BRIEF more

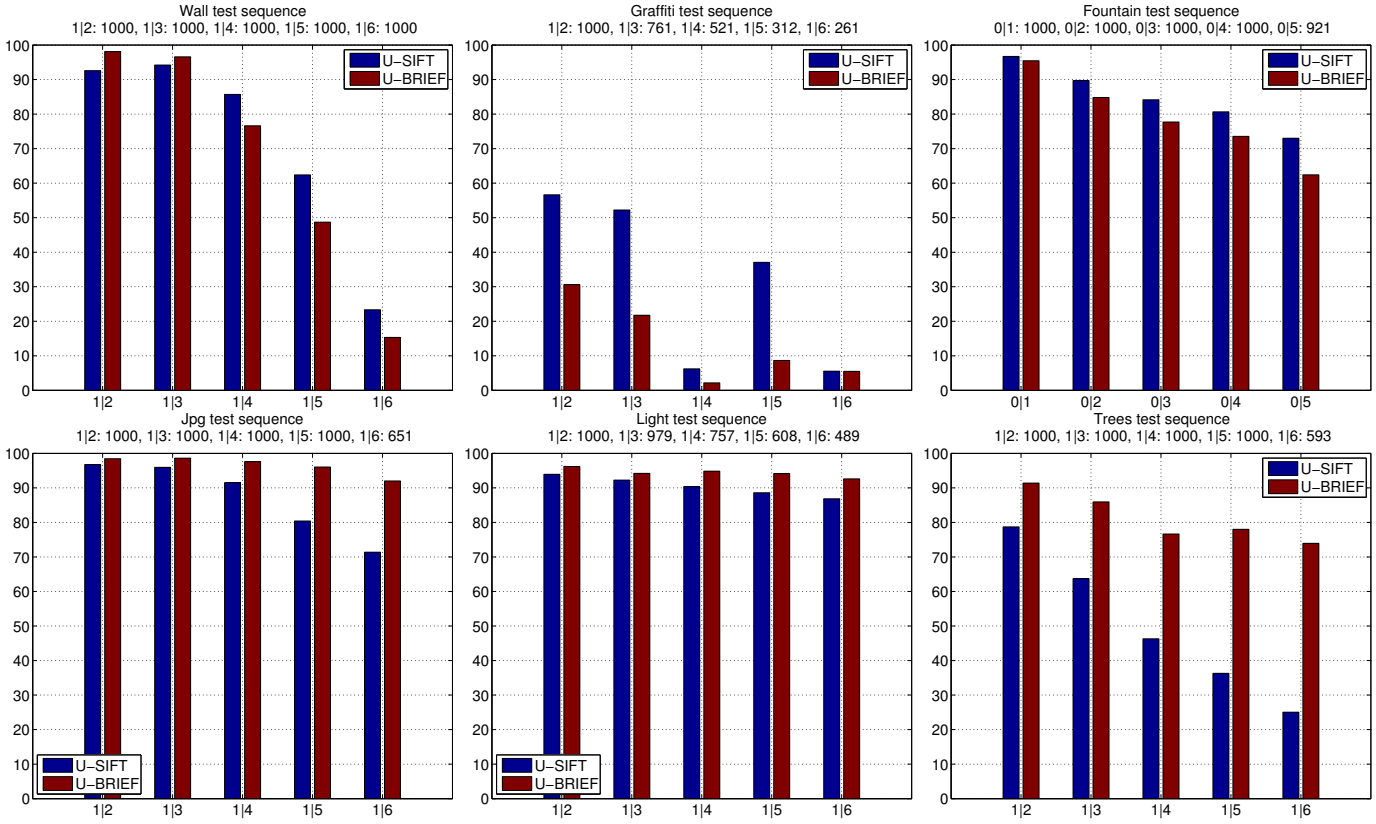



Fig. 13: Recognition rate vs. image pairs of all sequences. Comparing to U-SIFT.

discriminative by using more bits and using BRIEF-64 or BRIEF-128, which are still smaller and, even more importantly, much faster to compute as will be discussed below. Furthermore, BRIEF-128 still only requires 1024 bits, which is half of the 2048 bits required to store the 64 floats of U-SURF.

Since SIFT performs better than BRIEF, whatever its size, we also investigated what part of the loss of discriminative power simply comes from the fact that we are using a binary descriptor, as opposed to a floating point one. To this end, we used a method we recently introduced to binarize SIFT descriptors and showed to be state-of-the-art [9] and as short as quantized descriptors like DAISY [13]. It involves first computing a projection matrix that is designed to jointly minimize the in-class covariance of the descriptors and maximize the covariance across classes, which can be achieved in closed-form. This being done, optimal thresholds that turn the projections into binary vectors are computed so as to maximize recognition rates. This amounts to performing Linear Discriminant Analysis on the descriptors before binarization. Because it is coded on 16 bytes, it is denoted as LDA-16 in the graph of Fig. 15. It performs better than BRIEF, but at the cost of computing much more since one has to first compute the full SIFT descriptor and then binarize it. In applications where time is of the essence and memory requirements are less critical, such as establishing correspondences for augmented reality purposes, it therefore makes sense to use BRIEF-32, -64, or -128 as required by the difficulty of the scenes to be matched.

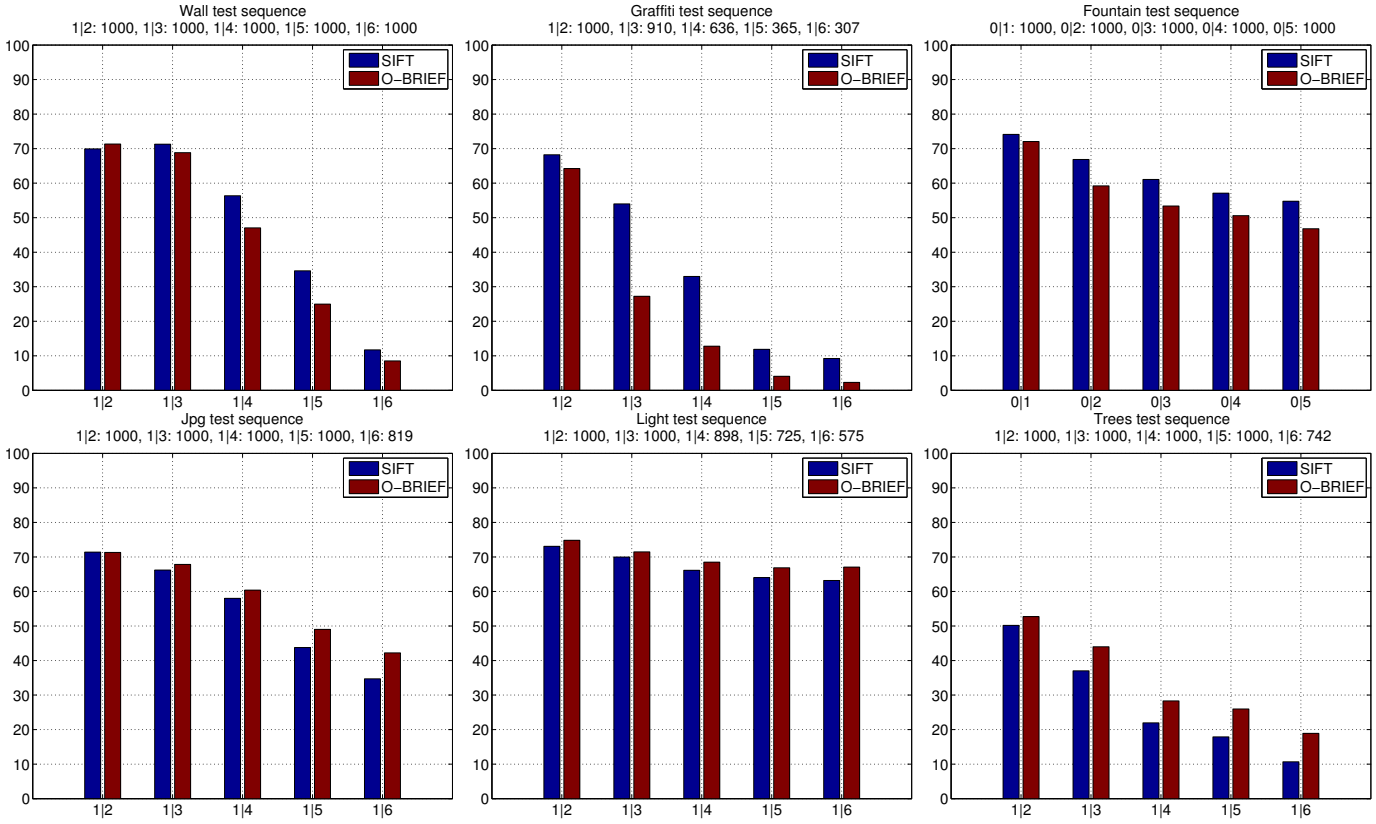



Fig. 14: Recognition rate vs. image pairs of all sequences. Comparing to SIFT.

TABLE III: Recognition rates of descriptors for increasing numbers $N$ of corresponding pairs from the Liberty dataset.

| N | 500 | 1000 | 2000 | 3000 | 5000 | 7000 |
|---|---|---|---|---|---|---|
| **SIFT** | 59.8% | 51.33% | 46.87% | 40.7% | 40.26% | 33.26% |
| **SURF** | 41.47% | 35.03% | 31.23% | 26.9% | 25.88% | 21.16% |
| **BRIEF-32** | 43.93% | 35.33% | 29.97% | 26.08% | 23.96% | 19.29% |
| **BRIEF-64** | 45.8% | 38.17% | 33.37% | 28.98% | 27.46% | 22.33% |
| **BRIEF-128** | 48% | 40.43% | 35.37% | 30.53% | 29.16% | 23.49% |
| **LDA-16** | 53.27% | 45.57% | 40.28% | 35.13% | 34.56% | 28.19% |

### *D. Comparing Computation Times*

In this section we compare the timings of the different methods. Fig. 16 shows the total time required for detecting, describing and matching 512 feature points. BRIEF is almost two orders of magnitude faster than its fastest competitor, U-SURF. In particular, the standard version of BRIEF, BRIEF-32, appears particularly efficient as it exploits integral images for smoothing and the `POPCNT` instruction for computing the Hamming distance.

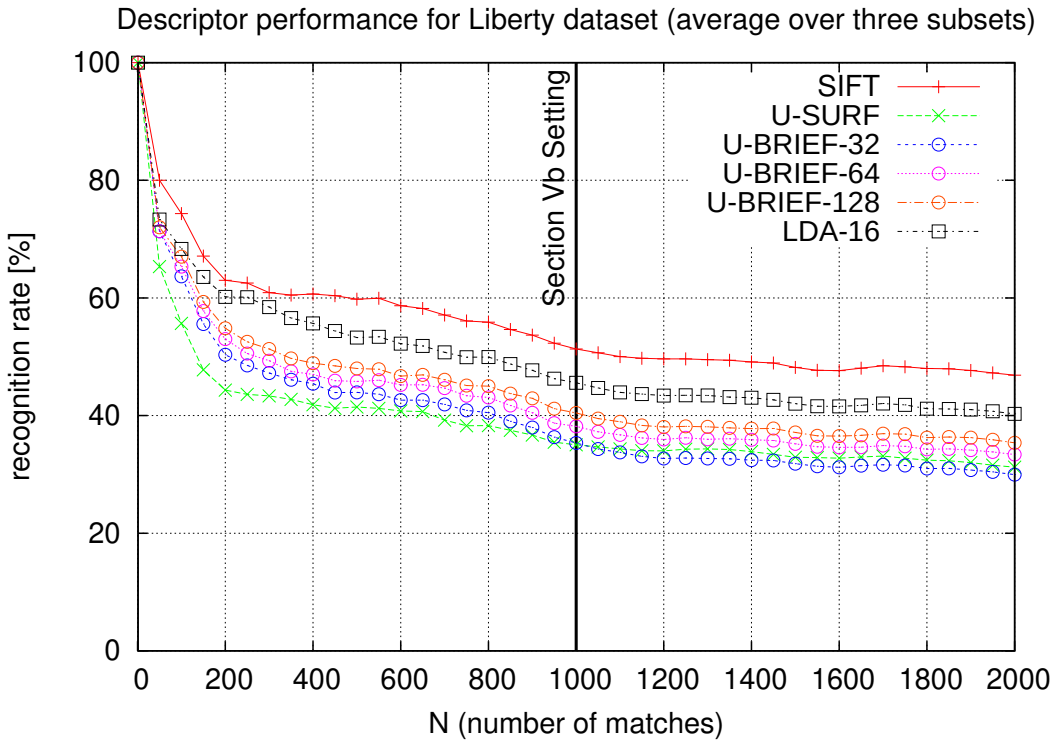



Fig. 15: Descriptors performance as the function of the number $N$ of corresponding pairs on the Liberty dataset.

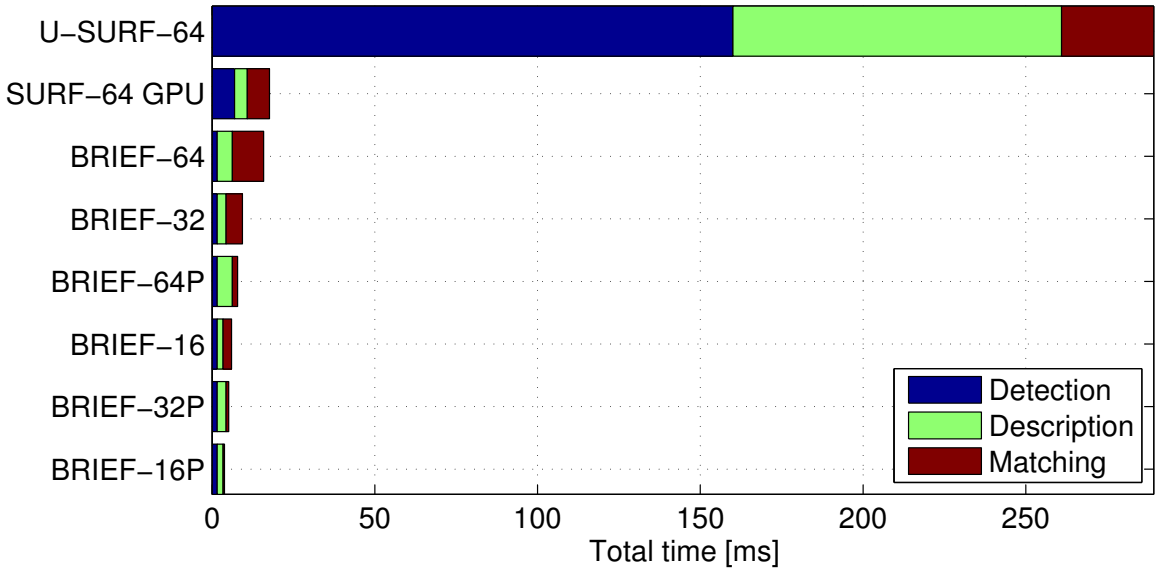



Fig. 16: Timings for a detection–describe–match cycle. Ordered by decreasing total time. The methods shown include three versions of BRIEF and U-SURF-64 implemented on both the CPU and the GPU. The P suffix for BRIEF indicates that in the matching step, the POPCNT instruction has been employed.

All timings are the medians over 10 instances of the task at hand. We measure the CPU wall clock time programatically, that is using the system's tick count value rather than counting cycles. When the

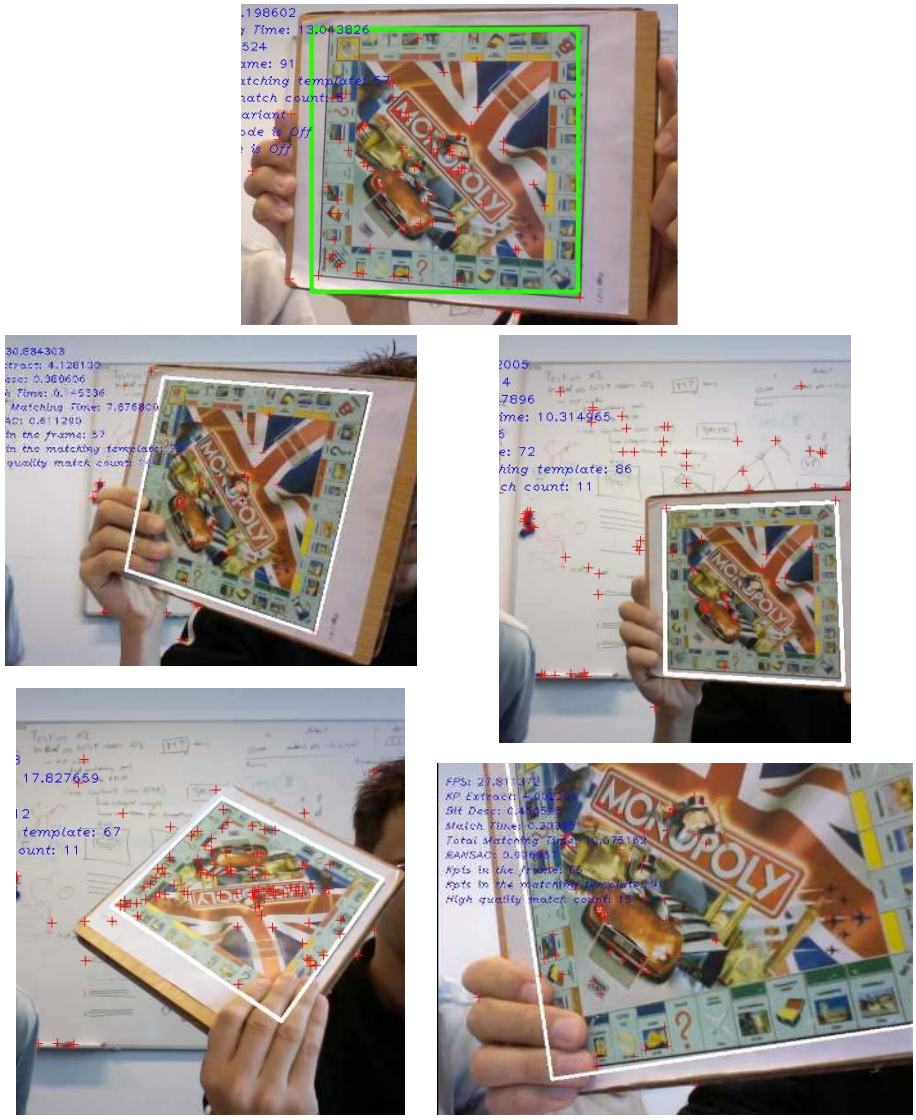



Fig. 17: Screenshots from the object detection application. Top image: Initialization by drawing a square around the detection target. Remaining 4 images: The system at run-time. Note that it tolerates substantial scale changes and is fully invariant to in-plane rotation while running at 27 FPS on the CPU of a simple laptop.

system load is low, the two values should be close, though from a practical point of view, the wall clock time is the one that truly counts.

### *E. On-the-Fly Object Detection*

To demonstrate the value of D-BRIEF, we present a system designed for real-time object detection where the object to be detected can be learned *instantaneously*. In other words, unlike earlier systems, such as our own real-time mouse pad detection application presented in [42], this one does not require a training phase, which may take several minutes. Using the same implementation using SURF features,

for example, is impossible unless resorting to the GPU. By contrast, the current application builds on D-BRIEF-32P and runs on a single CPU while maintaining frame-rate performance. We show results in a few frames in Fig. 17.

In such an application, unlike those presented earlier, full rotational invariance is clearly a desirable feature. This can be achieved thanks to BRIEF's extremely efficient processing pipeline, even without resorting to any trick for speeding things up. To this end, when the target image is acquired, we build a template database that contains 18 rotated views at 3 scales of the target, totaling up to 54 views. The additional descriptors are computed on synthetic images obtained by warping the target image. Then each incoming frame is matched against the database and the one with the highest number of matches to features-in-template score is selected. These matches are still noisy and hence fed to RANSAC, which robustly estimates a homography between the views that is used to re-project the template target's corners into the image frame.

While this basic system works at 25–30 Hz, its computation time and memory requirements increase linearly with the number of templates. Its performance would be further enhanced by i) enabling tracking and/or ii) using a more efficient feature point search as was done in [37] to achieve real-time performance using SIFT and Ferns. Furthermore, to achieve a more efficient feature point search, a tree resembling much that of a Vocabulary Tree [43] could be used.

## VI. 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 [44], [17] that involves moving from the Euclidean to the Hamming distance for matching purposes.

Future work will aim at developing data structures that allow for sub-linear time look-up of BRIEF descriptors.

## References

[1] K. Mikolajczyk and C. Schmid, "A Performance Evaluation of Local Descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, 2004.

[4]The code is publicly available: http://cvlab.epfl.ch/software/brief/

[2] G. Hua, M. Brown, and S. Winder, "Discriminant Embedding for Local Image Descriptors," in *International Conference on Computer Vision*, 2007.

[3] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 20, no. 2, pp. 91–110, 2004.

[4] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "SURF: Speeded Up Robust Features," *Computer Vision and Image Understanding*, vol. 10, no. 3, pp. 346–359, 2008.

[5] T. Tuytelaars and C. Schmid, "Vector Quantizing Feature Space With a Regular Lattice," *International Conference on Computer Vision*, 2007.

[6] S. Winder, G. Hua, and M. Brown, "Picking the Best Daisy," in *Conference on Computer Vision and Pattern Recognition*, June 2009.

[7] M. Calonder, V. Lepetit, K. Konolige, J. Bowman, P. Mihelich, and P. Fua, "Compact Signatures for High-Speed Interest Point Description and Matching," in *International Conference on Computer Vision*, September 2009.

[8] G. Shakhnarovich, "Learning Task-Specific Similarity," Ph.D. dissertation, Massachusetts Institute of Technology, 2005.

[9] C. Strecha, A. Bronstein, M. Bronstein, and P. Fua, "LDAHash: Improved Matching With Smaller Descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011, in press.

[10] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua, "Fast Keypoint Recognition Using Random Ferns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 3, pp. 448–461, 2010.

[11] Intel, "SSE4 Programming Reference: software.intel.com/file/18187." Intel Corporation, Denver, CO 80217-9808, April 2007.

[12] ARM, "RealView Compilation Tools," 2010.

[13] M. Brown, G. Hua, and S. Winder, "Discriminative Learning of Local Image Descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 43–57, 2010.

[14] E. Tola, V. Lepetit, and P. Fua, "Daisy: an Efficient Dense Descriptor Applied to Wide Baseline Stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 5, pp. 815–830, 2010.

[15] H. Jégou, M. Douze, and C. Schmid, "Product Quantization for Nearest Neighbor Search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 117–128, January 2011.

[16] A. Gionis, P. Indik, and R. Motwani, "Similarity Search in High Dimensions Via Hashing," in *International Conference on Very Large Databases*, 2004.

[17] A. Torralba, R. Fergus, and Y. Weiss, "Small Codes and Large Databases for Recognition," in *Conference on Computer Vision and Pattern Recognition*, June 2008.

[18] H. Jégou, M. Douze, and C. Schmid, "Improving Bag-Of-Features for Large Scale Image Search," *International Journal of Computer Vision*, vol. 87, no. 3, pp. 316–336, 2010.

[19] R. Salakhutdinov and G. Hinton, "Learning a Nonlinear Embedding by Preserving Class Neighbourhood Structure," in *International Conference on Artificial Intelligence and Statistics*, 2007.

[20] S. Taylor, E. Rosten, and T. Drummond, "Robust Feature Matching in $2.3\mu S$," in *Conference on Computer Vision and Pattern Recognition*, 2009.

[21] R. Zabih and J. Woodfill, "Non Parametric Local Transforms for Computing Visual Correspondences," in *European Conference on Computer Vision*, May 1994, pp. 151–158.

[22] B. Froba and A. Ernst, "Face Detection With the Modified Census Transform," in *International Conference on Automatic Face and Gesture Recognition*, 17-19 2004, pp. 91–96.

[23] F. Stein, "Efficient Computation of Optical Flow Using the Census Transform," in *Pattern Recognition*, C. Rasmussen, H. Bülthoff, B. Schälkopf, and M. Giese, Eds. Springer Berlin / Heidelberg, 2004, pp. 79–86.

[24] T. Ojala, M. Pietikäinen, and D. Harwood, "A Comparative Study of Texture Measures With Classification Based on Feature Distributions," *Journal of Machine Learning Research*, vol. 29, pp. 51–59, 1996.

[25] X. Wang, T. Han, and S. Yan, "An HoG-LBP Human Detector With Partial Occlusion Handling," in *International Conference on Computer Vision*, 2009.

[26] M. Heikkila, M. Pietikainen, and C. Schmid, "Description of Interest Regions With Local Binary Patterns," *Pattern Recognition*, vol. 42, no. 3, pp. 425–436, March 2009.

[27] G. Zhao and M. Pietikainen, "Local Binary Pattern Descriptors for Dynamic Texture Recognition," in *International Conference on Pattern Recognition*, 2006, pp. 211–214.

[28] B. Brahnam and L. Nanni, "High Performance Set of Features for Human Action Classification," in *International Conference on Image Processing*, 2009.

[29] S. Marcel, Y. Rodriguez, and G. Heusch, "On the Recent Use of Local Binary Patterns for Face Authentication," *International Journal on Image and Video Processing Special Issue on Facial Image Processing*, pp. 469–481, 2007.

[30] L. Nanni and A. Lumini, "Local Binary Patterns for a Hybrid Fingerprint Matcher," *Pattern Recognition*, vol. 41, no. 11, pp. 3461–3466, 2008.

[31] V. Lepetit and P. Fua, "Keypoint Recognition Using Randomized Trees," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 9, pp. 1465–1479, September 2006.

[32] J. Koenderink, "The Structure of Images," *Biological Cybernetics*, vol. 50, no. 5, pp. 363–370, August 1984.

[33] T. Lindeberg, "Scale-Space for Discrete Signals," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 3, pp. 234–254, 1990.

[34] M. Agrawal, K. Konolige, and M. Blas, "Censure: Center Surround Extremas for Realtime Feature Detection and Matching," in *European Conference on Computer Vision*, 2008.

[35] E. Rosten and T. Drummond, "Machine Learning for High-Speed Corner Detection," in *European Conference on Computer Vision*, 2006.

[36] E. Rosten, R. Porter, and T. Drummond, "Faster and Better: a Machine Learning Approach to Corner Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 105–119, 2010.

[37] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg, "Pose Tracking from Natural Features on Mobile Phones," in *International Symposium on Mixed and Augmented Reality*, September 2008.

[38] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool, "A Comparison of Affine Region Detectors," *International Journal of Computer Vision*, vol. 65, no. 1/2, pp. 43–72, 2005.

[39] C. Strecha, W. Hansen, L. V. Gool, P. Fua, and U. Thoennessen, "On Benchmarking Camera Calibration and Multi-View Stereo for High Resolution Imagery," in *Conference on Computer Vision and Pattern Recognition*, 2008.

[40] A. Vedaldi, "An Open Implementation of the Sift Detector and Descriptor," UCLA CSD, Tech. Rep., 2007.

[41] G. Bradski and A. Kaehler, *Learning OpenCV*. O'Reilly Media Inc., 2008.

[42] M. Ozuysal, P. Fua, and V. Lepetit, "Fast Keypoint Recognition in Ten Lines of Code," in *Conference on Computer Vision and Pattern Recognition*, June 2007.

[43] D. Nister and H. Stewenius, "Scalable Recognition With a Vocabulary Tree," in *Conference on Computer Vision and Pattern Recognition*, 2006.

[44] G. Shakhnarovich, P. Viola, and T. Darrell, "Fast Pose Estimation With Parameter-Sensitive Hashing," in *International Conference on Computer Vision*, 2003.

**Michael Calonder** received his Ph.D. degree in Computer Vision from the Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland, in 2010. His research focused on high-speed methods for interest point matching and its applications to real-time robotics, especially to SLAM. In 2011 he left academia for industry and is now an Associate Director in a Swiss bank.

**Vincent Lepetit** is a Senior Researcher at the Computer Vision Laboratory, EPFL. He received the engineering and master degrees in Computer Science from the ESIAL in 1996. He received the Ph.D. degree in Computer Vision in 2001 from the University of Nancy, France, after working in the ISA INRIA team. He then joined the Virtual Reality Lab at EPFL (Swiss Federal Institute of Technology) as a post-doctoral fellow and became a founding member of the Computer Vision Laboratory. His research interests include vision-based Augmented Reality, 3D camera tracking, object recognition and 3D reconstruction.

**Mustafa Özuysal** received his B.Sc. and M.Sc. degrees in Electrical and Electronics Engineering from Middle East Technical University (METU), Ankara, Turkey. He completed his Ph.D. studies in the Computer Vision Laboratory (CVLab) at the Swiss Federal Institute of Technology in Lausanne (EPFL) in 2010. His thesis work concentrates on learning features from image and video sequences for fast and reliable object detection. His research interests include object tracking-by-detection, camera egomotion estimation and augmented reality.

**Tomasz Trzcinski** received his M.Sc. degree in Research on Information and Communication Technologies from Universitat Politècnica de Catalunya (Barcelona, Spain) and M.Sc. degree in Electronics Engineering from Politecnico di Torino (Turin, Italy) in 2010. He joined EPFL in 2010 where he is currently pursuing his Ph.D. in computer vision in the field of binary local feature descriptors and their application. His research interests include Local Feature Descriptors, Visual Search and Augmented Reality. He worked with Telefonica R&D in Barcelona in 2010. His work there focused on adapting a visual search engine for vision-based geo-localisation.

**Christoph Strecha** received an degree in physics from the university of Leipzig (Germany) and the Ph.D. degree from the Catholic University of Leuven (Belgium) in 2008. He did his Ph.D. thesis in computer vision in the field of multi-view stereo. He joined EPFL (Swiss Federal Institute of Technology) in 2008 where he works a a post-doc in the computer vision group. His research interests include photogrammetry, structure from motion techniques, city modeling, multi-view stereo and optimization-based techniques for image analysis and synthesis. He is co-chair of Commission III/1 of the International Society for Photogrammetry and Remote Sensing and founder of Pix4D.

**Pascal Fua** received an engineering degree from Ecole Polytechnique, Paris, in 1984 and the Ph.D. degree in Computer Science from the University of Orsay in 1989. He joined EPFL (Swiss Federal Institute of Technology) in 1996 where he is now a Professor in the School of Computer and Communication Science. Before that, he worked at SRI International and at INRIA Sophia-Antipolis as a Computer Scientist. His research interests include shape modeling and motion recovery from images, analysis of microscopy images, and Augmented Reality. He has (co)authored over 150 publications in refereed journals and conferences. He has been an associate editor of IEEE journal Transactions for Pattern Analysis and Machine Intelligence and has often been a program committee member, area chair, and program chair of major vision conferences.