Analysis and Evaluation of Keypoint Descriptors for Image Matching

M. Hassaballah, Hammam A. Alshazly and Abdelmgeid A. Ali

Abstract Feature keypoint descriptors have become indispensable tools and have been widely utilized in a large number of computer vision applications. Many descriptors have been proposed in the literature to describe regions of interest around each keypoint and each claims distinctiveness and robustness against certain types of image distortions. Among these are the conventional floating-point descriptors and their binary competitors that require less storage capacity and perform at a fraction of the matching times compared with the floating-point descriptors. This chapter gives a brief description to the most frequently used keypoint descriptors from each category. Also, it provides a general framework to analyze and evaluate the performance of these feature keypoint descriptors, particularly when they are used for image matching under various imaging distortions such as blur, scale and illumination changes, and image rotations. Moreover, it presents a detailed explanation and analysis of the experimental results and findings where several important observations are derived from the conducted experiments.

M. Hassaballah (✉)
Faculty of Computers and Information, Computer Science Department,
South Valley University, Luxor, Egypt
e-mail: m.hassaballah@svu.edu.eg

H. A. Alshazly
Faculty of Science, Mathematics Department, South Valley University,
Qena 83523, Egypt
e-mail: hammam.alshazly@sci.svu.edu.eg

A. A. Ali
Faculty of Computers and Information, Computer Science Department,
Minia University, Minia, Egypt

M. Hassaballah and K. M. Hosny (eds.), *Recent Advances in Computer Vision*, Studies in Computational Intelligence 804, https://doi.org/10.1007/978-3-030-03000-1_5

1 Introduction

The decomposition of an image into local regions of interest (ROI) or features, which allows to exploit the local appearance properties of images is a widely used technique in almost all computer vision applications [1]. Detecting and describing these features have been an active research area for decades because of their promising performance [2]. These features can simply represent the description of such local regions of the image by their characteristics or properties [3]. Vision-based applications such as image matching [4], image registration [5], 3D reconstruction [6], image retrieval [7], biometric systems [8, 9], identification systems [10], object recognition [11], and many others all rely on the presence of stable and representative features [12]. Local feature descriptors which are distinctive and yet invariant to many kinds of geometric and photometric transformations are key factors in the performance of these applications [13].

Generally, the core issue of designing a good local feature descriptor lies in how to represent the local ROIs discriminatingly and efficiently. In this regards, the local features can be hand-crafted or learnt. Normally, a processing pipeline for all hand-crafted based descriptors is to detect a set of repeatable interest points also known as keypoints from images and represent the neighborhood around each keypoint in an efficient and discriminative feature vector known as a keypoint feature descriptor [14]. The design of hand-crafted features often involves finding the right trade-off between accuracy and computational efficiency [15]. While, learnt features can be extracted from images via training some algorithms such as convolutional neural networks (CNN) and deep learning to understand keypoints and descriptors by themselves. The features extracted by the learning-based methods (e.g., lower levels of CNN) are strongly depend on the training set. Thus, special care must be considered in the selection of the representative training dataset (i.e., size and content).

With respect to the hand-crafted based descriptors, there are two main categories in the literature: floating-point and binary descriptors. The floating-point feature descriptors such as SIFT [16] and SURF [17] are computed and stored as real-valued feature vectors. These descriptors can be divided into distribution-based descriptors, differential descriptors, and spatial-frequency descriptors. However, the most frequently used floating-point descriptors are the distribution-based ones, which are based on counting the occurrence of gradient orientation in the local ROIs of the image because the local object appearance and shape within images can be described by the distribution of intensity gradients or edge directions. Unfortunately, these descriptors are based on histogram statistics of gradient information (i.e., magnitude and orientation) which are more expensive to compute and hence having high computational complexity. Besides, they produce high dimensional feature vectors, which become a significant concern to run on devices with limited computational and storage capability. Since each dimension of the descriptor is stored as a decimal number making the descriptor requires more memory. These factors indicate the difficulty in using these type of descriptors for some applications and on devices with low-power, low-memory, and resource-constrained. On the other side, the binary descriptors

such as BRIEF [18] and ORB [19] are obtained through a specific design process: the operations used to compute the descriptor are quite different from the operations involved in the floating-point descriptors. That is, the binary descriptors are compact representation of ROIs in the form of a binary string resulting from pixel intensity comparisons at sampled predefined locations in the region. The binary descriptors require lower memory compared with gradient-based descriptors, and the similarity of descriptors can be evaluated by computing the Hamming distance, which can be computed very efficiently. Additionally, they use a fixed shape of pattern (i.e., only circular pattern and rectangle pattern) that can not be changed. As a result, they have no affine invariance and the high false matching rate between different viewpoints images is obvious [20].

Evaluating the performance of all these descriptors pairs is considered an important step towards choosing suitable combinations for developing robust computer vision systems. In 2005, Mikolajczyk and Schmid [3] evaluated the performance of various descriptors paired with several region detectors in a stereo matching scenario. They concluded that SIFT-based descriptors perform the best. Johansson et al. [21] presented a comparative evaluation of various detectors and descriptors on infrared (IR) images and outlined useful insights. With the development of binary descriptors, studies have evolved to compare the performance of binary features with non-binary features descriptors. For instance, Heinly et al. [22] conducted a performance evaluation of some binary descriptors in image matching using SIFT and SURF as a baseline. They concluded that the performance of these descriptors do not always meet the authors' recommendations and that binary descriptors suffer when taking into consideration any transformations not present in the data. Bekele et al. [23] extended the evaluation in [22] by adding the FREAK descriptor using the Stanford Mobile Visual Search (SMVS) dataset, which simulates many characteristic transformations for mobile devices. Figat et al. [24] evaluated the performance of binary descriptors for utilization in real-time object recognition by service robots and concluded that a combination of ORB detector and FREAK descriptor gives the best result. Mukherjee et al. [25] conducted an experimental study on evaluating different combinations of various detectors and descriptors highlighting several performance characteristics. In [20], an extension to the existing evaluations is presented by adding the recently proposed binary descriptors (i.e., LATCH) and separated the effect of each transformation on the descriptors' performance. Some insightful implications and highlights are summarized based on the obtained experimental results. Other comparative studies are carried out for different applications in [26–29].

The rest of the chapter is organized as follows. Section 2 presents a detailed explanation of the state-of-the-art keypoint descriptors including floating-point and binary descriptors and highlights their characteristics. Section 3 discusses the important steps in image matching. Section 4 introduces the performance evaluation framework and discusses the benchmark datasets, matching criteria, and performance metrics involved in the evaluation process. The obtained experimental results under each image distortion are given in Sect. 5. Finally, the chapter is concluded in Sect. 6.

2 Keypoint Descriptors

Before computing the keypoint descriptors, a set of interest points need to be detected using any of the keypoint detectors [30]. Once the interest points have been detected from the image at specific locations, scale, and orientation, the local descriptor can be constructed based on these parameters around each keypoint in a scale and rotation-invariant manner. The descriptor should be aligned with the keypoint orientation and be proportional to its detected scale. The following subsections discuss the most frequently used keypoint descriptors from each category.

2.1 Floating-Point Descriptors

2.1.1 Scale Invariant Feature Transform

Scale invariant feature transform (SIFT) [16] is one of the most successful keypoint descriptors and the most widely used in many vision tasks such as image registration, image classification, and object recognition to name a few. Basically, SIFT includes both keypoint detection and feature description. The set of image features are obtained through four major stages. First, detecting the scale-space extrema by searching for a set of stable features across all possible scales using a difference-of-Gaussian (DoG) function. In order to obtain the local extrema, each sample point is compared with its eight neighbors in its current scale and nine neighbors in the above and below scales. The point is selected if it is larger than all these neighbors or smaller than all of them. Second, the accurate keypoint localization, where a detailed model is fit to determine the location, scale, and ratio of principal curvatures. Keypoints with low contrast or the ones that are poorly localized on an edge are rejected. In other words, keypoints are selected based on their stability. Third, the orientation assignment, where each keypoint is assigned one or more orientations based on the local image gradient directions. Then, an orientation histogram is constructed from the gradient orientations in a region around the keypoint. The orientation histogram has 36 bins covering the 360° range of orientations. Finally, a descriptor is computed for each detected point, based on the local image information at the characteristic scale. The SIFT descriptor builds a histogram of gradient orientations from sample points in a region around the keypoint, finds the highest orientation value and any other values that are within 80% of the highest, and uses these orientations as the dominant orientation of the keypoint.

The description stage of the SIFT algorithm starts by sampling the image gradient magnitudes and orientations in a 16×16 region around each keypoint. Then, a set of orientation histograms is created where each histogram contains sample points from a 4×4 subregion and having eight orientations bins in each. A Gaussian weighting function with σ equal to half the region size is used to assign weight to the magnitude of each sample point and gives higher weights to gradients closer to the center of

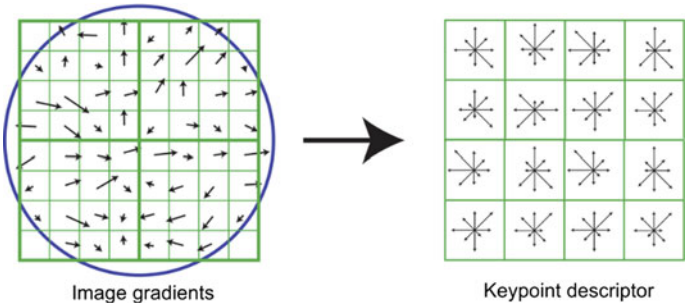


Fig. 1 A schematic representation for computing SIFT descriptor using 16 × 16 pixel patch and 4 × 4 descriptor array. Image adapted from [16]

the region, which are less affected by positional changes. The descriptor is then constructed as a vector containing the values of all the orientation histograms entries. Since there are 4 × 4 histograms each with 8 bins, the feature vector has $4 \times 4 \times 8 = 128$ elements for each keypoint. Finally, the feature vector is normalized to unit length in order to get invariance against illumination variations. Figure 1 illustrates the schematic representation of the SIFT algorithm; where the gradient orientations and magnitudes are computed at each pixel and then weighted by a Gaussian window (indicated by overlaid circle). A weighted gradient orientation histogram is then computed for each subregion.

The standard SIFT descriptor representation is noteworthy in several aspects. First, the representation is carefully designed to avoid problems due to boundary changes in location, orientation and scale. Second, it is fairly compact, expressing the patch of pixels using 128 element feature vector for each keypoint. Third, the representation is surprisingly resilient to deformations such as those caused by the perspective effects. These characteristics are evidenced in excellent matching performance against competing algorithms under different scales, rotations and lighting variations. On the other hand, the construction of the standard SIFT feature vector is complicated and the choices behind its specific design are not clear. Additionally, the SIFT's high dimensionality is a common problem which affects significantly the computational time of the descriptor.

2.1.2 Speeded-up Robust Features

Speeded-up robust features (SURF) [17] is a scale and rotation-invariant detection and description algorithm. It was developed to approximate the well-established SIFT technique in terms of repeatability, distinctiveness, and robustness while being faster to compute and match. These characteristics are achieved by exploiting the strengths of the Hessian matrix-based measure for detection and the distribution-based methods for description. In the detection stage, a set of blob-like structures (i.e., interest regions) are detected based on the determinant of Hessian matrix. For

instance, given a point P at (x, y) location in an image I, the Hessian matrix H(P, σ) for P at scale σ is defined by

$$H(P,\sigma) = \begin{bmatrix} I_{xx}(P,\sigma) & I_{xy}(P,\sigma) \\ I_{xy}(P,\sigma) & I_{yy}(P,\sigma) \end{bmatrix} \tag{1}$$

where I_{xx}, I_{xy}, and I_{yy} are the second-order image derivatives computed using a Gaussian function of standard deviation σ. The detector searches for a subset of points where the derivatives responses are high in two orthogonal directions or where the determinant of the Hessian matrix has a local maxima. SURF approximates the second order Gaussian derivatives in an efficient way with the help of integral images using a set of box filters. These approximations are denoted by D_{xx}, D_{yy}, and D_{xy}. Thus, the approximated determinant of Hessian can be expressed as

$$\det(H_{approx}) = D_{xx}D_{yy} - (wD_{xy})^2 \tag{2}$$

where w is a relative weight of the filter responses and it is used to balance the expression for the Hessian's determinant. The approximated determinant responses are stored in a blob response map over different scales where local maxima are detected. Finally, performing a non-maximum suppression in a $3 \times 3 \times 3$ neighborhood to get the steady interest points.

After detecting the set of interest points, the SURF's description stage starts by constructing a square window centered around the detected interest point and oriented along its main orientation. The size of this window is $20s$, where s is the scale at which the interest point is detected. Then, the interest region is further divided into smaller 4×4 sub-regions and for each sub-region the Haar wavelet responses in the vertical and horizontal directions (denoted d_x and d_y, respectively) are computed at a 5×5 sampled points as illustrated in Fig. 2. These responses are weighted with a Gaussian window centered at the interest point to increase the robustness against geometric deformations and localization errors. The wavelet responses d_x and d_y are summed up for each sub-region and entered in a feature vector v, where

$$v = \left(\sum d_x, \sum |d_x|, \sum d_y, \sum |d_y| \right). \tag{3}$$

By computing this for all the 4×4 sub-regions resulting a feature vector of length $4 \times 4 \times 4 = 64$ dimensions. Finally, the feature descriptor is normalized to a unit vector in order to reduce the illumination effects.

There are two obvious advantages of the SURF descriptor compared to SIFT. First is the processing speed, SIFT computes an image pyramid by convolving the image several times with large Gaussian kernels, while SURF accomplishes an approximation of that using integral images. Second, less memory requirements as SURF uses 64 dimensional feature vector to describe local image features while SIFT uses 128 dimensions. However, the SIFT descriptor is more suitable to describe images affected by translation, rotation, scaling, and other illumination deformations [16].

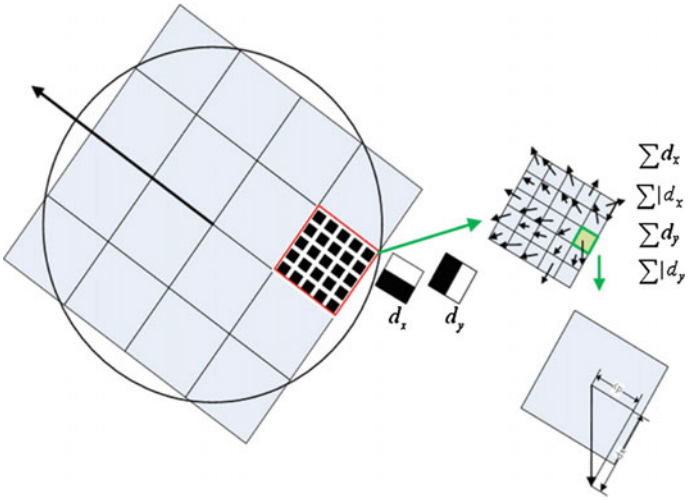


Fig. 2 Computing SURF descriptor using oriented grid with 4×4 square sub-regions. Only 2×2 sub-divisions are shown for illustration purpose [17]

2.1.3 KAZE Feature

The term KAZE is a Japanese word that means wind and the KAZE algorithm is introduced in [31] as a multi-scale feature detection and description scheme that operates completely in nonlinear scale spaces. The idea behind its creation is to detect and describe 2D image features at different scale levels and to obtain a better localization accuracy and distinctiveness. The previous techniques such as SIFT and SURF find multi-scale features through building a Gaussian scale space by filtering the original image with a Gaussian kernel of increasing standard deviation and then searching for a set of stable features across all possible scales. However, Gaussian blurring does not respect the natural boundaries of objects and smooths in the same degree details and noise when evolving the original image through the scale space and hence reducing the localization accuracy and distinctiveness. In contrast, KAZE makes the blurring locally adaptive to the image data by means of nonlinear diffusion filtering so that noise will be blurred but important image details and object boundaries will remain unaffected.

Given an input image, KAZE follows the same four steps as SIFT but with small differences in each step. It starts by building the nonlinear scale space by means of Additive Operator Splitting (AOS) techniques and variable conductance diffusion which is one of the simplest cases of nonlinear diffusion. Then, a set of interest points are detected which exhibit a maxima of the scale-normalized determinant of Hessian response at multiple scale levels. Finally, computing the dominant orientation of each keypoint and then constructing the feature descriptor in a scale and rotation invariant manner. A brief description of each step can be summarized as follows:

- **Nonlinear scale space computation**: KAZE follows a similar approach to SIFT in building the scale space using a certain number of octaves and sub-levels. However, instead of using Gaussian scale space kernel, it uses a nonlinear diffusion filtering combined with a conductivity function which makes the diffusion adaptive to local image structures. The conductivity function is defined by

$$c(x, y, t) = g(|\nabla L_\sigma(x, y, t)|), \tag{4}$$

  where ∇L_σ is the gradient of a Gaussian smoothed version of the original image L. In contrary to SIFT, which use the difference of Gaussian (DoG) to process the filtered images, KAZE uses AOS schemes to process them as the filtered images build a system of partial differential equations (PDEs).
- **Keypoint localization**: In order to detect a set of keypoints at multi-scale, KAZE computes the response of the scale-normalized determinant of Hessian at multiple scale levels using the following formula:

$$L_{Hessian} = \sigma^2(L_{xx}L_{yy} - L_{xy}^2), \tag{5}$$

  where L_{xx} and L_{yy} are the second order horizontal and vertical derivatives and L_{xy} is the second order cross derivative. After getting the set of filtered images from the nonlinear scale space, the detector response is analyzed at different scale levels to find the maxima in scale and spatial locations. Searching for extrema is done in all filtered images and is speeded up by first checking responses over a window of a 3×3 pixels to discard the non-maxima responses quickly. Finally, the keypoint position is estimated by sub-pixel accuracy similar to the method proposed in [32].
- **Orientation assignment**: The dominant orientation for a local neighborhood centered around the keypoint is estimated in a similar way to SURF using a circular area of radius $6\sigma_i$ and a sampling step σ_i. For each sample the first order derivatives L_x and L_y are weighted by a Gaussian kernel centered around the keypoint. Then, the derivative responses are represented as points in a vector space where the dominant orientation is obtained by summing the responses in a sliding circle covering an angle of $\pi/3$ and selecting the orientation of the longest vector.
- **Descriptor computation**: For building the feature descriptor, a modified version of the SURF descriptor is adapted to fit the non-linear scale-space model. The first order derivatives L_x and L_y of size σ_i are computed for each detected keypoint over a rectangular grid of size $24\sigma_i \times 24\sigma_i$. The grid is divided into overlapping 4×4 subregions of size $9\sigma_i \times 9\sigma_i$ and for each subregion the derivative responses are weighted by a Gaussian kernel centered on the subregion center and summed into a descriptor vector $d_v = (\sum L_x, \sum L_y, \sum |L_x|, \sum |L_y|)$. By doing this for all the 4×4 sub-regions, the resulting feature vector would have length of $4 \times 4 \times 4 = 64$ dimensions. Finally, the 64 feature vector is normalized into a unit vector to gain robustness against illumination variations.

2.2 Binary Descriptors

A binary descriptor is a compact representation of an image patch or region in the form of a binary string resulting from pixel intensity comparisons at sampled predefined locations in the patch [20]. Binary descriptors are of a particular interest as they speed-up the feature extraction time and minimize the storage capacity for the extracted local images features compared with their floating-point competitors. For example, given a keypoint and a patch of size $s \times s$, the task is to build a binary descriptor of size n by selecting n pairs, (P_i, P_j), from the patch using a predefined mechanism or through learning the optimal set of pairs. When comparing the pairs of pixels, if the pixel intensity of P_i is greater than P_j, then a binary value 1 is set in the binary string and 0 otherwise. Generally, a binary descriptor is mainly composed of three parts:

- A sampling pattern: where to sample the neighborhood points in an $N \times N$ image patch centered around the keypoint.
- An orientation mechanism: measuring the patch's dominant orientation and rotating it accordingly to gain robustness against rotation changes.
- Sampling pairs: refer to the set of pixel pairs to be compared in order to obtain the binary string representing the image patch.

On the other side, the binary descriptors provide efficient alternative to their floating-point competitors. Recently, it has been a topic of great interest in the area of local image description and hence various binary descriptors have been proposed. The main characteristics of these descriptors can be summarized as follows:

- They provide comparable performance in feature matching similar to their floating-point competitors such as SIFT, SURF and KAZE.
- They have lower computation complexity by simply comparing smoothed pixel intensity values and no gradients computations are involved; thus, faster extraction time is achieved.
- Less memory requirements to store the extracted features, gradient based descriptors usually require 64 or 128 real values to store while binary descriptors require only 256 or 512 bits which is four to eight times less.
- Efficient in time-constrained applications and suitable for mobile devices which have limited storage and computational power.
- The Hamming distance is used for matching the descriptors, which can be executed very fast on modern CPUs using a single XOR operation followed by a bit count.
- Easy proliferation on camera-enabled and mobile devices.

2.2.1 Binary Robust Independent Elementary Feature

Binary robust independent elementary features (BRIEF) [18, 33] is the first binary feature point descriptor computed directly from the pixel intensity comparisons to avoid the high computational time consumed in computing gradients. Given a

detected keypoint and a window w of predetermined size around it. A binary descriptor B_w is constructed from a set S of n pixel pairs defined as: $S = \{s_i\}_{i=1,..,n} = \{P_{i,1}, P_{i,2}\}_{i=1,..,n}$. Where, $P_{i,1} = (x_{i,1}, y_{i,1})$ and $P_{i,2} = (x_{i,2}, y_{i,2})$ refer to a pair of sampling coordinates in the window w. After smoothing the patch using Gaussian kernel, for each pair, $(P_{i,1}, P_{i,2})$, the smoothed intensity of $P_{i,1}$ is compared with $P_{i,2}$ to result a single bit according to the following test:

$$T(w, s_i) = \begin{cases} 1, & \text{if } P_{i,1} > P_{i,2} \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

Then, the binary descriptor B_w of n-dimensional bit-string is obtained by

$$B_w = \sum_{1 \le i \le n} 2^{i-1}\, T(w, s_i) \tag{7}$$

In order to compromise between speed, memory storage, and recognition rate, the value of n can be set to be 128, 256, and 512 bits as described in [18]. However, n is usually set to 256 and increasing it to 512 will slightly improve performance, but the memory storage and matching time will also increase. Moreover, when increasing n value above 512, no further improvement will be achieved. BRIEF has no fixed sampling pattern and the sampling pairs of the binary tests are chosen randomly using an isotropic Gaussian distribution.

2.2.2 Oriented FAST and Rotated BRIEF

The **O**riented FAST and **R**otated **B**RIEF (ORB) descriptor is proposed in [19] to provide a fast and efficient alternative to SIFT and SURF. In contrast to SIFT and SURF, ORB is free from licensing restrictions and can provide similar matching performance while being computationally efficient. It consists of a keypoint detector and a feature descriptor which are build upon the well-known (Features from Accelerated Segment Test) FAST keypoint detector [34] and the BRIEF descriptor [18], respectively. Even though it exploits the attractive performance and low computational cost for both techniques, it addresses the lack of rotation invariance in FAST and BRIEF.

ORB starts by detecting the set of keypoints from the image using FAST detector and employs the Harris corner measure to order the keypoints to exclude the non-corner points. In order to produce multi-scale features, a scale pyramid of the image is employed and a set of FAST features filtered by Harris are detected at each level in the pyramid. Since FAST does not provide a measure of corner orientation, the intensity centroid measure is used to compute the keypoint orientation.

The ORB keypoint descriptor is build upon a rotated version of the BRIEF binary descriptor. Even though BRIEF provides a fast and efficient way to compute and store the extracted features, its matching performance falls significantly for in-plane image rotations. A possible solution to overcome this limitation is to steer BRIEF

(sBRIEF) according to the orientation of the keypoints by multiplying the matrix of binary tests with their orientation angle. Once BRIEF is oriented along the keypoint direction, the intensity relationship between the rotated pairs will move toward some fixed pattern and increases the correlations among the pairs used for computing the binary descriptor. Such correlation affects the discriminative ability of the descriptor. For more discriminative features there are two desirable properties for the sampling pairs to have. One is uncorrelation between the sampling pairs which allows each pair to contribute and bring new information to the descriptor and hence increasing the information carried by the descriptor. The other is the high variance of the pairs which makes the features respond differentially to inputs. In order to achieve these properties, a learning method is used to search among all the possible sampling pairs and learn the optimal set of 256 sampling pairs having these properties. The resulting binary descriptor is referred to as rotated BRIEF (rBRIEF) and has significant improvement in the variance and correlation that steered BRIEF.

2.2.3 Binary Robust Invariant Scalable Keypoints

Binary robust invariant scalable keypoints (BRISK) is a feature detection and description scheme proposed in [35]. BRISK provides a fast keypoint detector inspired by the AGAST detection methodology [36] with some modifications in order to detect high quality keypoints and gain robustness to scale changes. To this end, the BRISK detector searches for the local maxima in the image plane as well as in scale-space using the FAST score as a measure of saliency. On the other hand, the BRISK descriptor has a predefined sampling pattern in which points are equally spaced on scaled concentric circles around the keypoint. For each sampling point, a small patch is considered around it and is smoothed using Gaussian smoothing. The sampling pairs are divided into long-distance and short-distance pairing subsets. The long-distance pairs are used to estimate the orientation of the patch while the short-distance ones are used to construct the final descriptor. Based on the sampling pattern and distance thresholds the descriptor is limited to a bit-string of length 512 bits. While the BRISK descriptor is considered a pure binary descriptor assembled via smoothed pixel intensity comparisons similar to BRIEF and ORB, it has some fundamental differences. First, it uses a fixed handcrafted sampling pattern resulting in a uniform sampling-point density at a given radius around the keypoint. Second, it uses fewer number of sampling points that pairwise comparisons which in turn limits the complexity of looking up intensity values. Third, the comparisons in BRISK are spatially restricted such that the intensity variations are required to be locally consistent.

2.2.4 Fast Retina Keypoint

Fast retina keypoint (FREAK) [37] is another binary keypoint descriptor similar to the BRISK in using a hand-crafted sampling pattern and in obtaining the binary descriptor by comparing intensities between pairs of sampling points after Gaussian

smoothing. The difference lies in the design of the sampling pattern where FREAK uses a sampling pattern inspired by the retinal sampling grid with more sampling points near the center. In addition, to reduce noise sensitivity FREAK uses Gaussian kernels of different size to smooth each sampling point and uses overlapping circles that change exponentially in size where each circle represents the standard deviations of the Gaussian kernel applied to the sampling point. The reasons behind changing the kernels' size and overlapping the circles were experimented by the authors and were found to increase the performance [37].

The binary descriptor is constructed by thresholding the difference of sampling pairs with their corresponding Gaussian kernel according to

$$B = \sum_{i=0}^{n-1} T(P_i) \times 2^i, \tag{8}$$

where P_i is a pair of sampling points, n is the required size of the descriptor, and

$$T(P_i) = \begin{cases} 1, & \text{if } I(P_i^{r1}) > I(P_i^{r2}) \\ 0, & \text{otherwise,} \end{cases} \tag{9}$$

with $I(P_i^{r1})$ is the smoothed intensity of the first sampling point of the pair P_i. In order to obtain a set of uncorrelated and distinctive sampling pairs, a learning method similar to ORB is used to find sampling pairs over keypoints in standard datasets and then extract the most discriminative pairs. The authors experimented with a predetermined set of 512 sampling pairs and observed that adding more pairs does not increase performance. To achieve rotation invariance, FREAK uses a method to measure orientation similar to BRISK but instead of using long pairs, it selects a set of predefined 45 symmetric pairs. The orientation weights over these pairs are summed and the sampling window is rotated by this orientation to some canonical orientation.

2.2.5 Accelerated-KAZE Feature

Accelerated KAZE (A-KAZE) [38] is a multi-scale feature detection and description approach that utilizes the benefits of nonlinear spaces. Since, constructing the scale space using non-linear diffusion filtering such as in KAZE [31] is highly time consuming and computationally demanding. AKAZE uses a faster method called Fast Explicit Diffusion (FED) embedded in a pyramidal framework to significantly speed-up feature detection in the non-linear scale space. Further, FED schemes are easy to implement and provide more accuracy than the AOS schemes used in KAZE [38]. For detecting the set of interest points, the determinant of Hessian is computed for each filtered image L^i in the nonlinear scale space.

$$L^i_{Hessian} = \sigma^2_{i,norm}(L^i_{xx}L^i_{yy} - L^i_{xy}L^i_{xy}), \tag{10}$$

The set of differential multi-scale operators are normalized with respect to scale using a normalized scale factor taking into account the octave of each particular image. The set of keypoints are found by searching for the maxima of the detector response in scale and spatial location. The feature description stage is achieved by using a modified version of the local difference binary (LDB) descriptor [39] and exploits the gradient and intensity information from the nonlinear scale space. The LDB descriptor works with the same principle of the BRIEF [18] but computes the binary tests between the average of areas instead of single pixels to increase robustness. Rotation invariance of AKAZE is achieved by utilizing the A-KAZE detector's estimation of orientation and rotating the LDB grid accordingly. Also, A-KAZE uses the A-KAZE detectors estimation of a patch scale to sub-sample the grid in steps that are a function of the patch scale.

2.2.6 Learned Arrangements of Three Patch Codes

Learned arrangements of three patch codes (LATCH) [40] is proposed to alleviate the sensitivity of a binary descriptor to noise and local appearance variations. The sensitivity arises from the nature of binary descriptors which compare pixel pairs; thus changing either of the pixels leads to changes in the descriptor values and hence affecting their performance. To overcome this limitation, the authors proposed to compare triplets of pixel patches instead of pairs of pixel values. However, LATCH still belongs to the family of pure-binary descriptors which are built from simple pixel intensity comparisons but with few differences in the way it is constructed. First, it uses small patch comparisons instead of single pixel comparisons. Second, using triplets of small patches rather than pairs of sampling points. Third, using supervised learning, linear Support Vector Machines (SVM), to learn the optimal set of sampling triplets. The binary descriptor for a detection window w of predefined size is constructed from an ordered set S of patch pixel triplets defined by $S = \{s_i\}_{i=1,\ldots,n} = \{[P_{i,a}, P_{i,1}, P_{i,2}]\}_{i=1,\ldots,n}$; with $P_{i,a}$, $P_{i,1}$, and $P_{i,2}$ refer to three $k \times k$ patches denoted as the anchor and the companions, respectively. The single binary value is obtained via comparing the Frobenius norm between the anchor patch and the companions according to the following test:

$$T(w, s_i) = \begin{cases} 1, & \text{if } ||P_{i,a} - P_{i,1}||^2 > ||P_{i,a} - P_{i,2}||^2 \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

The LATCH descriptor uses a window size of 48×48 pixels centered around the keypoint and compares the intensity of three 7×7 mini-patches in order to produce a single bit in the final binary string representing the patch. Therefore, for producing a binary descriptor of length 512 bits, 512 triplets are needed, where each triplet defines the location of the three mini-patches. LATCH can be computed efficiently and provides rotation-invariant mechanism.

Fig. 3 An illustrative example of matching keypoints based on their local feature descriptors

3 Image Matching

Image matching or feature matching is a part of many computer vision applications such as image registration and object recognition as the task is to establish correspondences between two images of the same scene or object as seen in Fig. 3. Where, image matching can be considered as the prime operation for obtaining semantic information. A common approach to image matching consists of detecting a set of keypoints from images and represent the local neighborhood around each keypoint by a feature descriptor. Then, establish feature correspondences by finding the nearest neighbor in the descriptor space. That is comparing the features and making decision about the quality of the matches. For example, given a feature f_1 in an image I_1, how to find the best candidate match in image I_2. First, define a distance function that compares two feature descriptors. Second, compute the distance to all the features in image I_2 and identify the nearest neighbor which is the one with the minimum distance.

A simple and straightforward similarity measure is to find the Euclidean norm or the sum of square differences (SSD) between the two feature descriptors. However, one drawback of these method is the difficulty to discard ambiguous matches as many features from an image may have no correct match in the other image. Figure 4 illustrates the incorrect location of the best candidate match f_2 to feature f_1 using minimum distance. Therefore, we need a more precise way to discard the features that do not have any good matches. A more efficient measure is to use the ratio distance between the best match and the second best match as illustrated in Fig. 5. Thus, for a correct match the ratio distance is required to be less than a specific threshold, while an ambiguous or a bad match will have distance ratio close to 1.

Fig. 4 Matching two features from two images using SSD, where f_2 is the nearest neighbor (i.e., best match) to f_1

Fig. 5 Matching two features using the ratio distance where f_2 and f_2' are the best and second best SSD matches to f_1 in I_2

This measure performs well as the correct matches need to have the closest match significantly closer than the closest incorrect match to achieve reliability. It should be noted that, the nearest-neighbor matching in the feature space of image descriptors using Euclidean norm or SSD can be used for matching the vector-based features. However, in practice, the optimal nearest-neighbor algorithm and its parameters depend on the dataset characteristics. On the other hand, these algorithms are not suitable for binary features such as BRIEF, ORB, FREAK, BRISK, and LATCH features. Binary features are compared using the Hamming distance calculated by performing a bitwise XOR operation followed by a bit count on the result. This involves only bit manipulation operations that can be performed very fast.

Generally, the performance of methods that use keypoints depends on the properties of the underlying keypoint detectors and the choice of associated image descriptors. Thus, choosing the appropriate detectors and descriptors for the images contents in the target applications is important. Statistically robust methods like

random sample consensus (RANSAC) [41] can be used to filter outliers in the matched feature sets while estimating the geometric transformation or fundamental matrix, which is useful in feature matching for image registration and object recognition applications.

4 Performance Evaluation

In this section, we analyze the performance characteristics of state-of-the-art and well-established floating-point (SIFT, SURF, and KAZE) and binary (BRIEF, ORB, BRISK, FREAK, AKAZE, and LATCH) keypoint descriptors. In order to keep the evaluation of the descriptors independent from the number of detected features we limit the number of keypoints to 1000 features. We use the detector/descriptor pairings for SIFT, SURF, KAZE, ORB, and AKAZE while using the SURF detector for BRIEF, BRISK, FREAK, and LATCH descriptors. The reason behind SURF is its repeatability and speed as well as its invariance to image rotation and scale changes. After detecting the set of keypoint and describing their neighborhood using the specified descriptor, matching the descriptors is performed by a brute force matcher using the Euclidean distance for SIFT, SURF, and KAZE while the Hamming distance is used for all the binary descriptors. Table 1 reports the main characteristics of the used keypoint descriptors regarding selecting the set of intensity comparison tests and their robustness to rotation and scale changes.

4.1 Image Datasets

In order to evaluate the feature descriptors on real images and making our work compatible with other existing ones, we use the Oxford dataset provided and described by Mikolajczyk et al. [3]. The dataset contains image sequences reflecting

Table 1 A summary of the keypoint descriptors and their main characteristics

Descriptor	Type	Detector	Sampling pairs	Rotation invariant	Scale invariant
SIFT [16]	Float-type	SIFT	NA	Yes	Yes
SURF [17]	Float-type	SURF	NA	Yes	Yes
KAZE [31]	Float-type	KAZE	NA	Yes	Yes
BRIEF [33]	Binary	SURF	Random	No	No
ORB [19]	Binary	ORB	Learned	Yes	No
BRISK [35]	Binary	SURF	Manual	Yes	Yes
FREAK [37]	Binary	SURF	Learned	Yes	Yes
AKAZE [38]	Binary	AKAZE	Learned	Yes	Yes
LATCH [40]	Binary	SURF	Learned	Yes	No

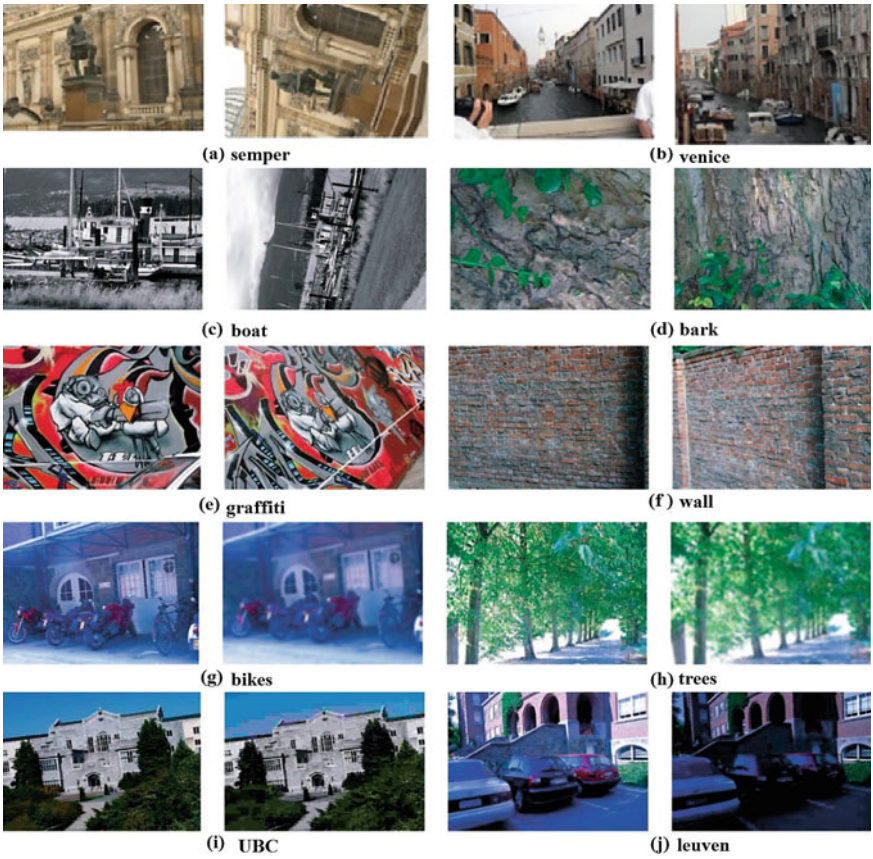


Fig. 6 The different image distortions and some sample images from the benchmark datasets used in the experiments: **a** pure rotation, **b** pure scaling, **c** and **d** combined scale and rotation, **e** and **f** viewpoint changes, **g** and **h** blur, **i** JPEG compression, and **j** illumination changes

different degrees of geometric and photometric distortions. Eight image sequences each with six images where the first image works as a reference image and the remaining are transformed images. Also, the ground truth homographies between the reference image and each of the transformed images are also given. However, the Oxford dataset lacks for certain transformations such as pure in-plane rotation and pure scaling. For isolating the effect of each distortion on the descriptors' performance, two datasets from Heinly's evaluation datasets [22] are considered in this evaluation; namely, the semper dataset which contains nine images for pure in-plane rotation and venice dataset with seven images for pure scaling. Sample images of these datasets are shown in Fig. 6.

4.2 Performance Metrics

There are several metrics to evaluate the performance of feature descriptors and each metric concentrates on specific characteristic of the feature's performance. However, the most commonly used metrics are based on the number of correct and false matches between a pair of images. In this evaluation, three standard performance metrics are used which are recall and precision in percentages and the average number of best matches. A brief definition of each metric is given as follows:

- **Recall**: is the ratio between the number of correct matches to the number of matches before passing the ratio test.

$$recall = \frac{\#\ Correct\ \ Matches}{\#\ Correspondences} \tag{12}$$

- **Precision**: is the ratio between the number of correct matches to the number of putative matches or the matches after passing the ratio test.

$$precision = \frac{\#\ Correct\ \ Matches}{\#\ Putative\ \ Matches} \tag{13}$$

- **Average number of best matches**: is the total number of correct matches from all images in a dataset divided by the number of images.

4.3 Matching Criteria

In order to obtain the putative matches, we adopted the nearest neighbor distance ratio (NNDR) matching strategy, which has proven to be effective as stated in [16, 42]. For each descriptor from the reference image, the two nearest neighbors descriptors in the distorted image are returned as best and second best matches and then finding the distance ratio between them. If the distance ratio is less than a specific threshold value 0.8 (as suggested in [16]), the best match is selected as a putative match, otherwise both matches are rejected. A threshold value of 0.8 eliminates 90% of the false matches while discarding less than 5% of the correct matches as mentioned in [16, 22, 23].

The next step in the evaluation process is to find the homographies between the reference image and each transformed one if they are not given. One effective method is RANSAC [41] or its fuzzified version [43] where the homographies are estimated between images by finding feature correspondences between those images. Then, the homographies are used to re-project keypoints from the reference image to their positions in the transformed images and if the difference between the transformed and the reprojected keypoints is within 2.5 pixels, the putative match is considered to be a correct or valid match. The threshold value (2.5) is chosen empirically as

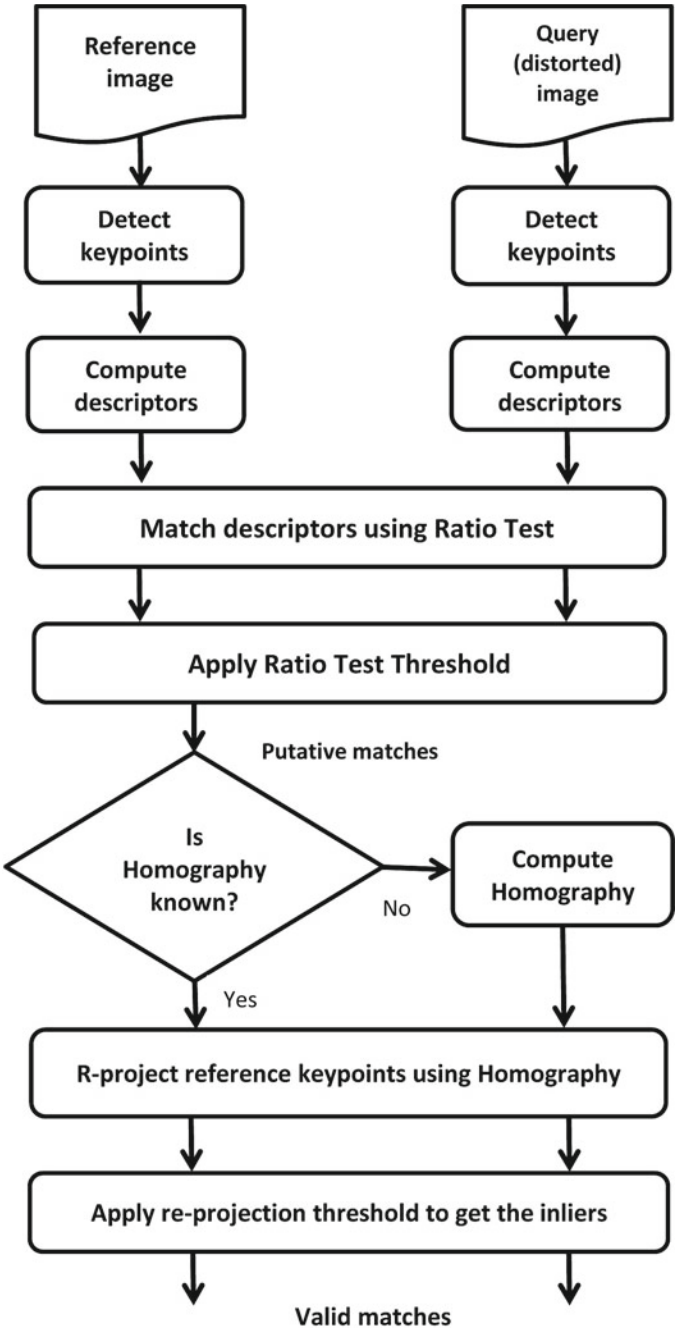


Fig. 7 The validation framework illustrating the steps followed to obtain the set of correct matches

suggested in [22] because it provides a good balance between the Harris's corner detectors in FAST [34] and the centers of blobs in the blob detectors (SIFT [16] and SURF [17]). The entire evaluation process is illustrated in Fig. 7.

5 Experimental Results and Discussion

This section presents and discusses the obtained experimental results of this evaluation. We tested the individual effects of various geometric and photometric distortions as well as some of their combinations in order to gain a clear understanding of the descriptors' performance. Figures 8, 9, and 10 summarize the experimental results for all the assessed keypoint descriptors under each image distortion. Furthermore, several illustrative examples for feature matching under each image transform are depicted in Fig. 11; while the analysis of the experimental results under each transform is mentioned in the following subsections.

5.1 Image Rotations

In order to evaluate the descriptors' performance under pure in-plane image rotation, we used the semper dataset (Fig. 6a) which contains images with different rotation angles upto 180°. Overall, KAZE obtains the highest recall, precision and number

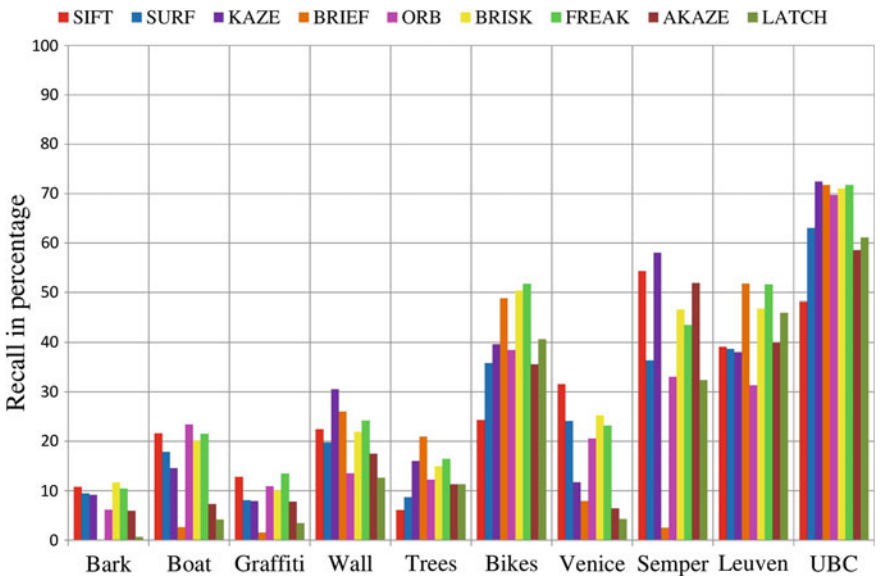


Fig. 8 Recall value in percentage obtained by each of the descriptors on each dataset

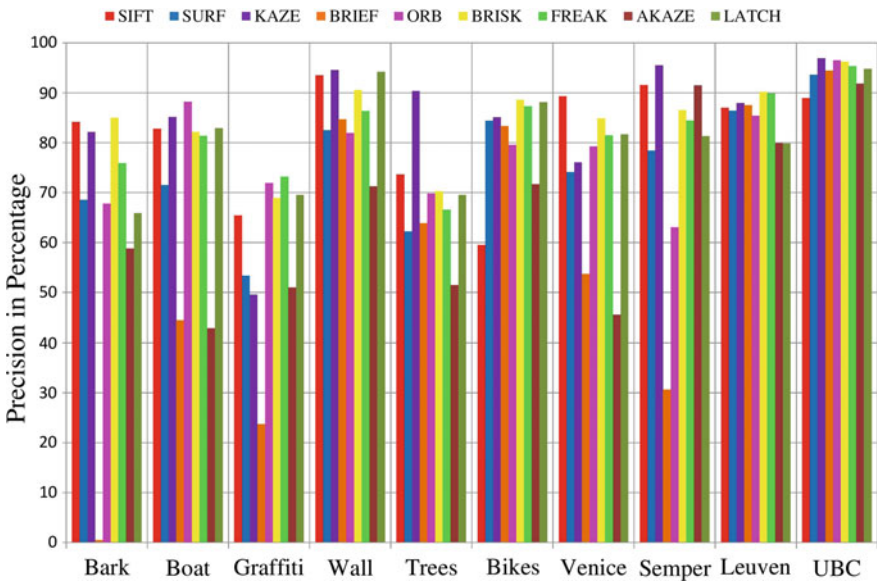


Fig. 9 Precision value in percentage given by the descriptors on each dataset

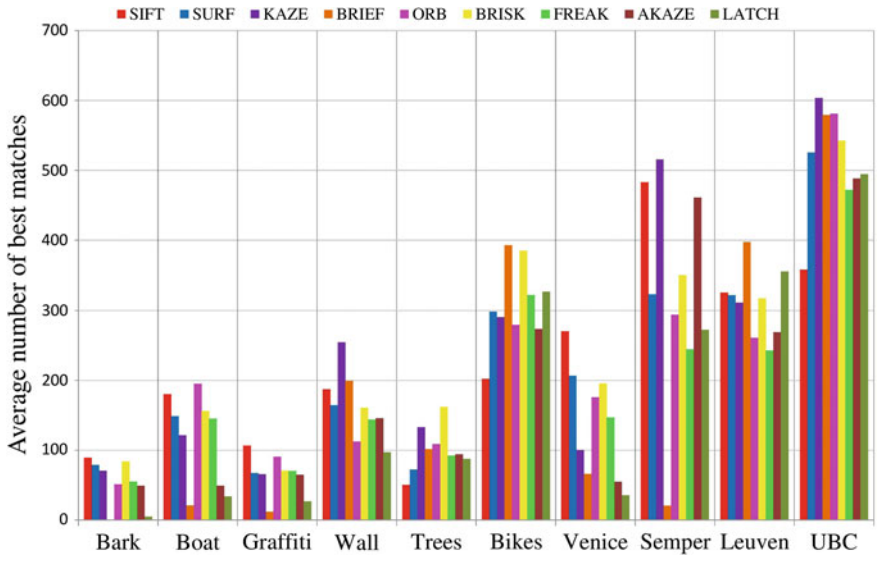
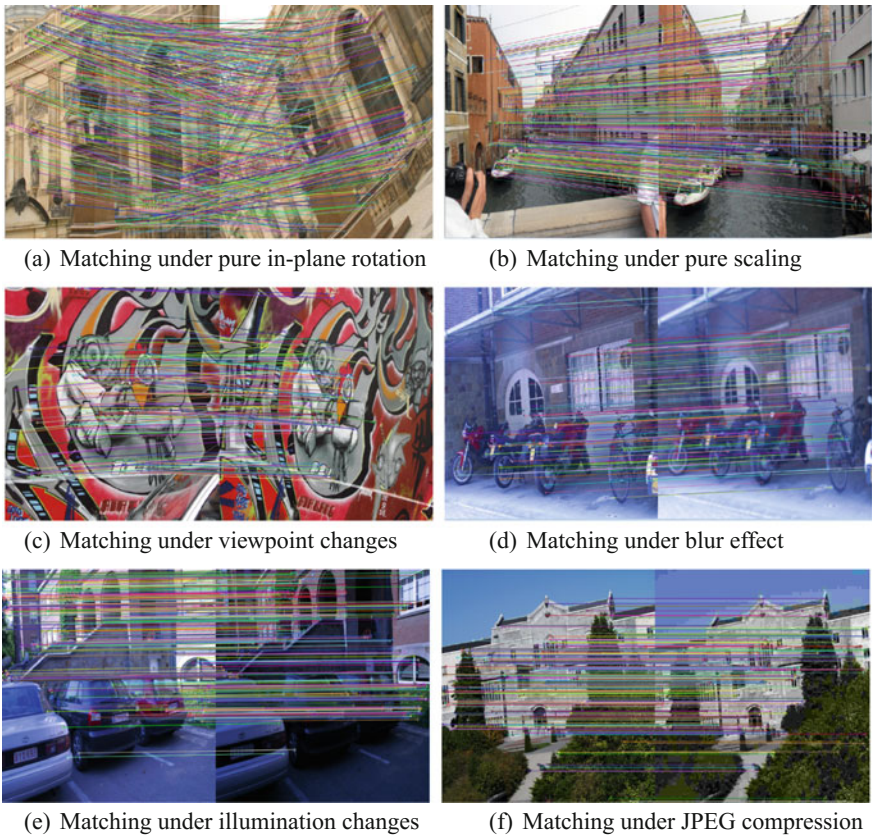


Fig. 10 Average number of best matches obtained by each descriptor on each dataset

of correct matches under pure in-plane rotations followed by SIFT in all metrics. This indicates the robustness of the gradient-based descriptors over binary ones under rotation changes. For the binary descriptors, AKAZE and BRISK are the top



(a) Matching under pure in-plane rotation (b) Matching under pure scaling

(c) Matching under viewpoint changes (d) Matching under blur effect

(e) Matching under illumination changes (f) Matching under JPEG compression

Fig. 11 Illustrative examples for matching pairs of images and locations of the matched keypoints

performers in all metrics respectively. As expected, BRIEF has the lowest values in all metrics as its performance drops off significantly after small rotations as it is not designed to handle rotation changes; while all other binary descriptors provide resistance to some extent for rotation changes. As it can be noted, the SURF-BRISK combination outperforms SURF detector/descriptor pairing in all metrics reflecting the importance of combining various detectors and descriptors in order to improve performance and enrich the applicability of using binary descriptors as alternatives for floating-point ones such as KAZE or SIFT which are time and memory consuming.

5.2 Scale Changes

For pure scale changes reflected by the venice dataset (Fig. 6b), SIFT is the best performer beating all other descriptors in all metrics. Among the binary descriptors,

BRISK obtains the best performance while ORB and FREAK are still performing well. ORB performs better than BRIEF and LATCH although all of them do not handle scale changes. This limited robustness against scale changes can be attributed to the scale pyramid employed by the ORB detector allowing it to detect FAST features at each level in the pyramid. The performance of LATCH and BRIEF decreases sharply when increasing the scale factor as they are not designed to handle the scaling effect.

5.3 Combined Rotation and Scale Changes

When analyzing the combined effects of rotation and scale changes as depicted in the boat and bark datasets (Fig. 6c, d), which is considered the most challenging distortions in the images. Starting by the bark dataset, BRISK takes the lead of binary descriptors as it performs well outperforming SIFT in recall and precision while the later gives more correct matches. SURF and KAZE obtain a good performance followed by FREAK as all these descriptors are rotation and scale invariant. While, the remaining other descriptors are ranked as ORB, AKAZE LATCH, and BRIEF; respectively. On the other hand, for the boat dataset, even though it combines the same rotation and scale changes, the ranking of the descriptors is different. ORB is ranked first beating all other descriptors in all metrics while SIFT is ranked second. FREAK exceeds BRISK in recall while the later provides higher precision and number of valid matches. The performance of BRIEF is the worst as it is not designed to handle any of these effects.

5.4 Perspective Changes

This part evaluates the descriptors' performance under changes in viewpoint of approximately 50 degrees as in the graffiti and wall datasets (Fig. 6e, f). For the graffiti dataset, SIFT gives the highest recall and number of inliers while FREAK obtains the highest precision. ORB performs well as the second top performer in precision and number of correct matches while the third in recall after FREAK and SIFT respectively. BRISK shows a moderate performance and outperforms SURF, LATCH and the lowest performance is obtained by BRIEF due to its limited complexity. While for the wall dataset, KAZE takes the lead of all the evaluated descriptors in all metrics proving its robustness to changes in perspective. While BRIEF leads the binary descriptors in recall and number of correct matches but obtains a slightly low precision compared with other binary descriptors. BRIEF's higher performance can be attributed to the fact that it has a fixed pattern and there are no significant changes in scale and the orientation of images is still the same. However, the low precision value is due to a less restrictive matching criteria or less distinctive image features. SIFT performs well and obtains recall and number of inliers following BRIEF, while

FREAK and BRISK vary interchangeably in recall and precision. AKAZE obtains the lowest precision and a moderate recall and number of inliers while LATCH obtains higher precision but the lowest in recall and number of inliers.

5.5 Blur Effect

Here, the performance of the descriptors is investigated under the blur effects resulting from changing the camera focus as in the bikes and trees datasets (Fig. 6g, h). It is noted that the values of all metrics are much higher in case of bikes dataset. Further, under this effect KAZE performs better than other gradient-based such as SIFT and SURF. Similarly, the binary descriptors outperform both SIFT and SURF in all cases, which can be attributed to the fact that all binary descriptors tend to smooth the pixel pairs independently or the entire patch before computing the final descriptor. Also, it indicates that the effect of blur results in less distinctive image gradients.

5.6 JPEG Compression

The UBC dataset (Fig. 6i) is used to evaluate the performance under JPEG compression effect. Generally, all descriptors perform well and the highest recall, precision, and number of valid matches are obtained under this effect. The KAZE descriptor is the best performer with respect to recall, precision and number of inliers. the ORB descriptor performs well achieving a high recall value and more number of correct matches; while all other descriptor perform more or less in the same level. Surprisingly, SIFT's performance is the lowest and is exceeded by all other descriptors. This can be attributed to the changes made to the relative difference between pixels and their influence on the image gradients.

5.7 Illumination Changes

For matching the keypoints under the effect of illumination changes introduced by varying the camera aperture, the the leuven dataset (Fig. 6j) is used. The performance of BRIEF is still favorable giving the highest recall value and number of correct matches; however, the highest precision is obtained by BRISK. LATCH has the lowest precision but the second best for recall and number of correct matches. SIFT followed by SURF obtain a moderate performance while FREAK gives higher precision values but obtains the lowest recall value and number of valid matches.

Table 2 Time and memory space consumed by each descriptor for a single feature

Descriptor	SIFT	SURF	KAZE	BRIEF	ORB	BRISK	FREAK	AKAZE	LATCH
Time (ms)	1.782	1.052	2.701	0.205	0.211	0.090	0.476	0.645	0.863
Memory (bytes)	128 (512)	64 (256)	64 (256)	32	32	64	64	61	32

5.8 Extraction Time and Memory Space

The extraction time and storage capacity are two important factors in measuring the efficiency of a feature extraction method or a feature descriptor; hence, we report them here for all the evaluated feature descriptors. The time and memory space required by each descriptor to extract and store a single feature are given in Table 2. The extraction time is averaged over 32K keypoints extracted from various images with different geometric and photometric transformations. All the experiments are performed using implementation coming from OpenCV (version 3.2.0) running on a PC with Intel Pentium 2.16 GHz processor, 4GB RAM, and Windows 8.1 Operating System. From Table 2, it is clear that the extraction time of all binary descriptors is an order of magnitude faster than their floating-point alternatives. The memory storage given in Table 2 for each feature descriptor is based on the assumption that the floating-point descriptors are stored in a quantized form, i.e., one byte per dimension. However, if the descriptors are stored as reals, the required number of bytes increases as the values in the parenthesis for SIFT, SURF, and KAZE descriptors. The conducted experiments and results highlight the important and motivating properties of binary descriptors in terms of their efficiency and compactness and suggest that BRISK descriptor is the fastest one with more memory space required (64 bytes) among the binary descriptors.

5.9 Discussion

From the analysis of the experimental results, we highlighted several observations that are derived from the conducted experiments and gained some insights about which descriptor is suitable for specific image features and/or distortions. These observations are summarized as follows:

- The claim from the authors of KAZE that it could surpass SIFT is supported by the test results in this study as KAZE outperforms SIFT and all other methods under pure rotation, perspective changes and JPEG compression; however, SIFT takes the lead over all methods under pure scaling.
- BRIEF performance is favorable under perspective transformations and non-geometric distortions such as blur, JPEG compression, and changes in illumination. This is attributed to the fact that, the image sequences under these transformations

have similar scales and orientations. This highlights the impact of scale and rotation changes on the performance of binary descriptors.
- The binary descriptors suffer dramatically in performance if they consider any transformations that are not present in the given images.
- The gradient-based features such as KAZE and SIFT outperform in the existence of geometric transformations such as rotation, scale and their combination.
- The performance of BRISK is not always the best among the binary descriptors and other descriptors such as ORB can be used alternatively.
- A combination of a repeatable and fast detector with a binary descriptor offers suitable alternatives for using floating-point methods which are time and memory demanding.
- The extraction time of descriptors is not consistent in many evaluations and varies according to the utilized feature detectors.
- There is no universal and optimal descriptor for all image datasets and setup should be custom-made according to the existing type of features and distortions need to be handled in the given images.

6 Conclusion

This chapter gives an overview of the well-established and recent methods in feature keypoint description. A special attention is given to floating point-based and binary-based descriptors, which represent the state-of-the-art and the most effective so far. The fundamental issues of feature correspondence and performance evaluation of various feature descriptors are presented. Also, the necessity of standard measures defined in a unified framework for determining what is a correctly matched pair of features is discussed. As the performance of a feature descriptor is affected by a wide range of factors such as the feature type and distortions exist in the given images as well as in real-time applications, not only the accurate and robust representation but also the speed in feature extraction may be crucial. Therefore, this chapter provides a validation framework to conduct an experimental evaluation of the most recent and prominent feature descriptors under various image transformations. Besides, the computational time and memory storage required for each feature are analyzed. Finally, the chapter highlights several observations that are derived from the conducted experiments.

References

1. Li, J., Allinson, N.M.: A comprehensive review of current local features for computer vision. Neurocomputing **71**(10–12), 1771–1787 (2008)
2. Hassaballah, M., Awad, A.I.: Detection and description of image features: an introduction. Image Feature Detectors and Descriptors, pp. 1–8. Springer (2016)

3. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE Trans. Pattern Anal. Mach. Intell. **27**(10), 1615–1630 (2005)
4. Duo, J., Chen, P., Zhao, L.: LCO: a robust and efficient local descriptor for image matching. AEU Int. J. Electron. Commun. **72**, 234–242 (2017)
5. Bouchiha, R., Besbes, K.: Comparison of local descriptors for automatic remote sensing image registration. Signal Image Video Process. **9**(2), 463–469 (2015)
6. Guo, Y., Bennamoun, M., Sohel, F., Lu, M., Wan, J., Kwok, N.M.: A comprehensive performance evaluation of 3D local feature descriptors. Int. J. Comput. Vis. **116**(1), 66–89 (2016)
7. Xu, P., Zhang, L., Yang, K., Yao, H.: Nested-SIFT for efficient image matching and retrieval. IEEE Multimed. **20**(3), 34–46 (2013)
8. Hassaballah, M., Aly, S.: Face recognition: challenges, achievements and future directions. IET Comput. Vis. **9**(4), 614–626 (2015)
9. Alshazly, H.A., Hassaballah, M., Ahmed, M., Ali, A.A.: Ear biometric recognition using gradient-based feature descriptors. In: International Conference on Advanced Intelligent Systems and Informatics, pp. 435–445. Springer (2018)
10. Otero, B., Rodriguez, E., Ventura, J.: SURF-based mammalian species identification system. Multimed. Tools Appl. **76**(7), 10133–10147 (2017)
11. Altun, O., Albayrak, S.: An evaluation of local interest regions for non-rigid object class recognition. Expert Syst. Appl. **39**(3), 2335–2340 (2012)
12. Kang, T.K., Choi, I.H., Lim, M.T.: MDGHM-SURF: a robust local image descriptor based on modified discrete Gaussian-Hermite moment. Pattern Recognit. **48**(3), 670–684 (2015)
13. Chen, J., Patel, V.M., Liu, L., Kellokumpu, V., Zhao, G., Pietikäinen, M., Chellappa, R.: Robust local features for remote face recognition. Image Vis. Comput. **64**, 34–46 (2017)
14. Hassaballah, M., Aly, A.A., Alshazly, H.A.: Image features detection, description and matching. Image Feature Detectors and Descriptors: Foundations and Applications, vol. 630, pp. 11–45 (2016)
15. Nanni, L., Ghidoni, S., Brahnam, S.: Handcrafted vs. non-handcrafted features for computer vision classification. Pattern Recognit. **71**, 158–172 (2017)
16. Lowe, D.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**(2), 91–110 (2004)
17. Bay, H., Ess, A., Tuytelaars, T., Gool, L.: Speeded-up robust features (SURF). Comput. Vis. Image Underst. **110**(3), 346–359 (2008)
18. Calonder, M., Lepetit, V., Ozuysal, M., Trzcinski, T., Strecha, C., Fua, P.: BRIEF: computing a local binary descriptor very fast. IEEE Trans. Pattern Anal. Mach. Intell. **34**(7), 1281–1298 (2012)
19. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: an efficient alternative to SIFT or SURF. In: International Conference on Computer Vision, pp. 2564–2571 (2011)
20. Alshazly, H.A., Hassaballah, M., Ali, A.A., Wang, G.: An experimental evaluation of binary feature descriptors. In: International Conference on Advanced Intelligent Systems and Informatics, pp. 181–191 (2017)
21. Johansson, J., Solli, M., Maki, A.: An evaluation of local feature detectors and descriptors for infrared images. In: European Conference on Computer Vision Workshops, pp. 711–723 (2016)
22. Heinly, J., Dunn, E., Frahm, J.M.: Comparative evaluation of binary features. In: European Conference on Computer Vision, pp. 759–773 (2012)
23. Bekele, D., Teutsch, M., Schuchert, T.: Evaluation of binary keypoint descriptors. In: IEEE International Conference on Image Processing, pp. 3652–3656 (2013)
24. Figat, J., Kornuta, T., Kasprzak, W.: Performance evaluation of binary descriptors of local features. In: International Conference on Computer Vision and Graphics, pp. 187–194 (2014)
25. Mukherjee, D., Wu, Q.J., Wang, G.: A comparative experimental study of image feature detectors and descriptors. Mach. Vis. Appl. **26**(4), 443–466 (2015)
26. Miksik, O., Mikolajczyk, K.: Evaluation of local detectors and descriptors for fast feature matching. In: International Conference on Pattern Recognition, pp. 2681–2684 (2012)
27. Khan, N., McCane, B., Mills, S.: Better than SIFT? Mach. Vis. Appl. **26**(6), 819–836 (2015)

28. Madeo, S., Bober, M.: Fast, compact, and discriminative: evaluation of binary descriptors for mobile applications. IEEE Trans. Multimed. **19**(2), 221–235 (2017)
29. Kanwal, N., Bostanci, E., Clark, A.F.: Evaluation method, dataset size or dataset content: how to evaluate algorithms for image matching? J. Math. Imaging Vis. **55**(3), 378–400 (2016)
30. Awad, A.I., Hassaballah, M.: Image feature detectors and descriptors: foundations and applications. Studies in Computational Intelligence, vol. 630. Springer (2016). ISSN 1860-949X
31. Alcantarilla, P.F., Bartoli, A., Davison, A.J.: KAZE features. In: European Conference on Computer Vision, pp. 214–227 (2012)
32. Brown, M., Lowe, D.G.: Invariant features from interest point groups. In: British Machine Vision Conference (2002)
33. Calonder, M., Lepetit, V., Ozuysal, M., Trzcinski, T., Strecha, C., Fua, P.: BRIEF: binary robust independent elementary features. In: European Conference on Computer Vision, vol. 34, no. 7, pp. 778–792 (2010)
34. Rosten, E., Porter, R., Drummond, T.: Faster and better: a machine learning approach to corner detection. IEEE Trans. Pattern Anal. Mach. Intell. **32**(1), 105–119 (2010)
35. Leutenegger, S., Chli, M., Siegwart, R.Y.: BRISK: binary robust invariant scalable keypoints. In: International Conference on Computer Vision, pp. 2548–2555 (2011)
36. Mair, E., Hager, G.D., Burschka, D., Suppa, M., Hirzinger, G.: Adaptive and generic corner detection based on the accelerated segment test. In: European conference on Computer vision, pp. 183–196 (2010)
37. Alahi, A., Ortiz, R., Vandergheynst, P.: Freak: fast retina keypoint. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 510–517 (2012)
38. Alcantarilla, P.F., Nuevo, J., Bartoli, A.: Fast explicit diffusion for accelerated features in nonlinear scale spaces. In: British Machine Vision Conference (2013)
39. Yang, X., Cheng, K.T.: LDB: An ultra-fast feature for scalable augmented reality on mobile devices. In: IEEE International Symposium on Mixed and Augmented Reality, pp. 49–57 (2012)
40. Levi, G., Hassner, T.: LATCH: learned arrangements of three patch codes. In: IEEE Winter Conference on Applications of Computer Vision, pp. 1–9 (2016)
41. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM **24**(6), 381–395 (1981)
42. Moreels, P., Perona, P.: Evaluation of features detectors and descriptors based on 3D objects. Int. J. Comput. Vis. **73**(3), 263–284 (2007)
43. Lee, J.J., Kim, G.: Robust estimation of camera homography using fuzzy RANSAC. In: International Conference on Computational Science and Its Applications, pp. 992–1002 (2007)