A Review on Image Feature Detection and Description

Mai Thanh Nhat Truong, Sanghoon Kim*
Department of Electrical, Electronic, and Control Engineering, Hankyong National University

Abstract

In computer vision and image processing, feature detection and description are essential parts of many applications which require a representation for objects of interest. Applications like object recognition or motion tracking will not produce high accuracy results without good features. Due to its importance, research on image feature has attracted a significant attention and several techniques have been introduced. This paper provides a review on well-known image feature detection and description techniques. Moreover, two experiments are conducted for the purpose of evaluating the performance of mentioned techniques.

1. Introduction

Recently, image feature detectors and descriptors have become an important algorithms in the computer vision and image processing. They have been applied widely in many vision-based applications, such as image representation [1], image classification [2], object recognition [3], 3D modeling [4], tracking [5], and biometrics systems [6]. Such applications require robust features in the image, which are not only representative but also invariant to noise, scale, or illumination. Therefore, detecting and extracting features from images are essential parts for these applications.

In digital images, the concept of feature in computer vision refers to a piece of information, which represents characteristics of the image. This concept is generally the same as feature in machine learning and pattern recognition, even though image data has a very sophisticated collection of features. Feature can be considered as interesting parts of an image, and features are used as a starting point for many computer vision algorithms. Since features are used as the starting point and main primitives for subsequent algorithms, the overall algorithm will often only be as good as its generated feature. In general, image features can be categorized as edges, corners blobs, and ridges.

Feature detection is a method to compute abstractions of image information and making local decisions at every image point whether there is an image feature of a given type at that point or not. Feature detection is a low-level image processing operation. That is, it is usually performed as the first operation on an image, and examines every pixel to see if there is a feature present at that pixel. If this is part of a larger algorithm, then the algorithm will typically only examine the image in the region of the features. After detecting keypoints in an image, we need a method to describe local properties of the image at those points, hence the name feature description. These algorithms extract interesting information from the image data at detected keypoints. A common practice to organize the information provided by these feature description algorithms are encoding them as the elements of one single vector, which is commonly referred to as a feature vector. The set of all possible feature vectors constitutes a feature space.

In the following sections, we will provide a review on well-known image feature detection and feature description techniques. We also conducted two experiments for the purpose of comparing performance of mentioned algorithms.

2. Feature detection and description techniques

2.1 SIFT

SIFT (Scale-Invariant Feature Transform) [7] algorithm was published by David Lowe in 1999. Its applications include object recognition, robotic mapping and navigation, image stitching, 3D modeling, gesture recognition, video tracking, individual identification of wildlife and match moving. The algorithm is patented in the US; the owner is the University of British Columbia. However, SIFT is allowed to be used in academic research.

SIFT is both feature detector and feature descriptor. SIFT transforms an image into a large collection of local feature vectors, each of which is invariant to image translation, scaling, and rotation, and partially invariant to illumination changes and affine or 3D projection. Previous approaches to local feature generation lacked invariance to scale and were more sensitive to projective distortion and illumination change. The SIFT features share a number of properties in common with the responses of neurons in inferior temporal (IT) cortex in primate vision. SIFT author also describes improved approaches to indexing and model verification. The scale-invariant features are efficiently identified by using a staged filtering approach. The first stage identifies key locations in scale space by looking for locations that are maxima or minima of a difference-of-Gaussian function. Each point is used to generate a feature vector that describes the local image region sampled relative to its scale-space coordinate frame. The features achieve partial invariance to local variations, such as affine or 3D projections, by blurring image gradient locations. This approach is based on a model of the behavior of complex cells in the cerebral cortex of mammalian vision. The resulting feature vectors are called SIFT keys. Due to complex computation, the execution time of SIFT is usually high. SIFT does not work well when dealing with affine transformation. SIFT features exhibit the highest matching accuracies for an affine transformation of 50 degrees. After this transformation limit, results start to become unreliable.

2.2 SURF

SURF (Speeded Up Robust Features) [8] is partly inspired by SIFT descriptor. The standard version of SURF is several times faster than SIFT and claimed by its authors to be more robust against different image transformations than SIFT. SURF was first presented by Herbert Bay, et al., at the 2006

European Conference on Computer Vision. An application of the algorithm is patented in the United States.

SURF is a fast and performant scale and rotation invariant interest point detector and descriptor. It relies on integral images for image convolutions to reduce computation time, builds on the strengths of the leading existing detectors and descriptors (using a fast Hessian matrix-based measure for the detector and a distribution-based descriptor). In feature description, it describes a distribution of Haar wavelet responses within the interest point neighborhood. Integral images are used for speed and only 64 dimensions are used reducing the time for feature computation and matching. The indexing step is based on the sign of the Laplacian, which increases the matching speed and the robustness of the descriptor. The important speed gain is due to the use of integral images, which drastically reduce the number of operations for simple box convolutions, independent of the chosen scale. Even without any dedicated optimizations, an almost real-time computation without loss in performance is possible, which represents an important advantage for many on-line computer vision applications. Experiments showed that the performance of Hessian approximation is comparable and sometimes even better than the SIFT. The high repeatability is advantageous for camera self-calibration, where an accurate interest point detection has a direct impact on the accuracy of the camera self-calibration and therefore on the quality of the resulting 3D model. SURF is faster than SIFT in term of execution time. However, when speed is not critical, SIFT outperforms SURF.

2.3 FAST

FAST (Features from Accelerated Segment Test) [9] is a corner detection method, which could be used to extract feature points and later used to track and map objects in many computer vision tasks. FAST corner detector was originally developed by Edward Rosten and Tom Drummond, and published in 2006. The most promising advantage of the FAST corner detector is its computational efficiency. FAST is not a feature descriptor, hence it must be combined with other descriptors in specific applications.

FAST corner detector uses a circle of 16 pixels (a Bresenham circle of radius 3) to classify whether a candidate point p is actually a corner. Each pixel in the circle is labeled from integer number 1 to 16 clockwise. If a set of N contiguous pixels in the circle are all brighter than the intensity of candidate pixel p (denoted by I_p) plus a threshold value t or all darker than the intensity of candidate pixel p minus threshold value t, then p is classified as corner. FAST need less execution time that many other well-known feature extraction methods. It is suitable for real-time video processing application because of high-speed performance. This feature detector has a special characteristics, that is it can be improved by using machine learning to select optimal value for N. Without machine learning, N is usually chosen as 12, which may affect the overall performance. FAST has high levels of repeatability under large aspect changes and for different kinds of feature. It is many times faster than other existing corner detectors when it was announced. This is, however, also its weakness. Since high speed is achieved by analyzing the fewest pixels possible, the detector's ability to average out noise is reduced.

2.4 BRISK

BRISK (Binary Robust Invariant Scalable Keypoints) [10] is a point-feature detector and descriptor, recently developed by Autonomous Systems Lab (ETH Zurich, Switzerland). BRISK achieved low computational complexity thanks to the application of a novel scale-space FAST-based detector, in combination with the assembly of a bit-string descriptor from intensity comparisons retrieved by dedicated sampling of each keypoint neighborhood.

In BRISK, points of interest are identified across both the image and scale dimensions using a saliency criterion. In order to boost efficiency of computation, keypoints are detected in octave layers of the image pyramid as well as in layers in-between. The location and the scale of each keypoint are obtained in the continuous domain via quadratic function fitting. For feature description, a sampling pattern consisting of points lying on appropriately scaled concentric circles is applied at the neighborhood of each keypoint to retrieve gray values: processing local intensity gradients, the feature characteristic direction is determined. Finally, the oriented BRISK sampling pattern is used to obtain pairwise brightness comparison results which are assembled into the binary BRISK descriptor. Once generated, the BRISK keypoints can be matched very efficiently thanks to the binary nature of the descriptor. With a strong focus on efficiency of computation, BRISK also exploits the speed savings offered in the SSE instruction set widely supported on today's architectures. BRISK is faster than SIFT and SURF, while using less computational resource.

2.5 ORB

SIFT uses 128-dim vector for descriptors. Since it is using floating point numbers, it takes basically 512 bytes. Similarly SURF also takes minimum of 256 bytes (for 64-dim). Creating such a vector for thousands of features takes a lot of memory which are not feasible for resource-constraint applications especially for embedded systems. Larger the memory, longer the time it takes for matching. BRIEF (Binary Robust Independent Elementary Features) [11] helps reduce resource consumption. ORB (Oriented FAST and Rotated BRIEF) [12] is a combination of FAST and BRIEF, with some improvements. Hence ORB is both detector and descriptor. The main goal of ORB is to reduce resource consumption. The contribution of ORB is the addition of a fast and accurate orientation component to FAST, and an efficient computation of oriented BRIEF features. Many keypoint detectors include an orientation operator (SIFT and SURF are two prominent examples), but FAST does not. There are various ways to describe the orientation of a keypoint; many of these involve histograms of gradient computations. However, ORB use centroid technique to calculate orientation for FAST. ORB authors also propose a learning method for de-correlating BRIEF features under rotational invariance, leading to better performance in nearest-neighbor applications.

2.6 FREAK

FREAK (Fast Retina Keypoint) [13] is inspired by the human visual system and more precisely the retina. A cascade of binary strings is computed by efficiently comparing image intensities over a retinal sampling pattern.

The authors claim that FREAKs are in general faster to compute with lower memory load and also more robust than SIFT, SURF or BRISK. They are competitive alternatives to existing keypoints in particular for embedded applications. FREAK is not a feature detector, it can only be applied to keypoints which are already detected by other feature detection algorithms. In this algorithm, a cascade of binary strings is computed by efficiently comparing pairs of image intensities over a retinal sampling pattern. It select pairs to reduce the dimensionality of the descriptor yields a highly structured pattern that mimics the saccadic search of the human eyes.

3. Experiments

In this section, we compare the performance of feature detection and feature description techniques. The algorithms are implemented in C++ under Microsoft Windows 7, using OpenCV library for processing image data. The system has 4GB of RAM and a quad-core Intel CPU running at 3.0GHz. The performance of algorithms is evaluated by two experiments. In the first experiment, five feature detection techniques are used to detect interest points in three images, each image contains a single object. The performance of each technique is evaluated by number of detected features, the goodness of features, and execution time. In the second experiment, five feature description techniques are used to locate three objects in the first experiment, which are now placed in a cluttered scene.

3.1 Feature detection evaluation

In this experiment, five feature detectors are taken into comparison. The selected algorithms are SIFT, SURF, FAST, BRISK, and ORB. Selected detectors are applied to three images for locating keypoints. Each image contains a single objects. These three sample image are shown in Figure 1.

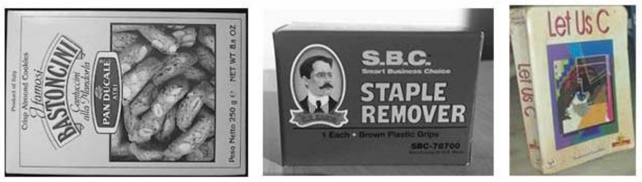

Figure 1. Three sample image containing single object.

The results of this experiment is shown in Figure 2, Table 1 and Table 2. As shown in Figure 2, keypoints detected by SIFT have the best distribution, in other words, SIFT keypoints cover important parts of the object with reasonable density. As shown in Table 1, SURF and BRISK detect more keypoint than other methods, and their distributions of keypoint have high density. FAST and ORB detect keypoints mainly in the text regions. Despite giving the best results, SIFT requires highest execution time. On average, considering both detected keypoints and execution time, SURF produces most reasonable results.

Table 1. Number of detected features

	Sample 1	Sample 2	Sample 3
SIFT	604	442	389
SURF	786	573	245
FAST	409	207	26
BRISK	1662	991	258
ORB	454	483	367

Table 2. Execution time.

	Sample 1	Sample 2	Sample 3
SIFT	0.2523s	0.1747s	0.1088s
SURF	0.0420s	0.0398s	0.0191s
FAST	0.0010s	0.0007s	0.0002s
BRISK	0.0690s	0.0426s	0.0165s
ORB	0.0222s	0.0142s	0.0088s

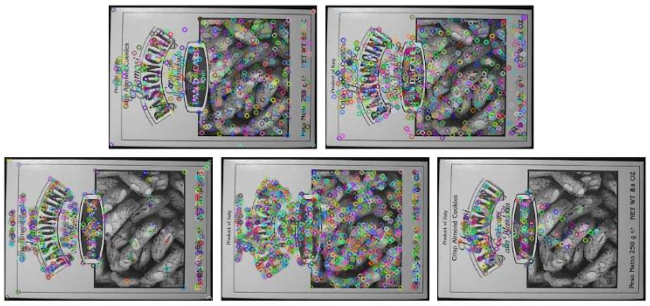

Figure 2. (From left to right, from top to bottom) Feature detection results of first sample from SIFT, SURF, FAST, BRISK, ORB.

3.2 Feature description evaluation

In this experiment, five feature description techniques are used to locate three objects in the first experiment, now placed in cluttered scenes, among other objects with arbitrary positions. The selected algorithms are SIFT, SURF, BRISK, ORB. A combination of FAST and FREAK is also used in this experiment, because FAST is detector-only and FREAK is descriptor-only, they cannot work separately for object localization test.

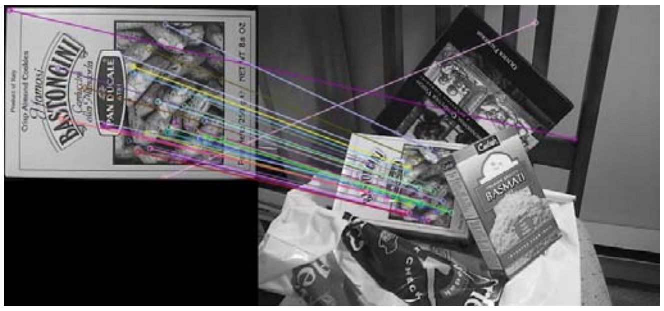

Figure 3. Matching result from SIFT.

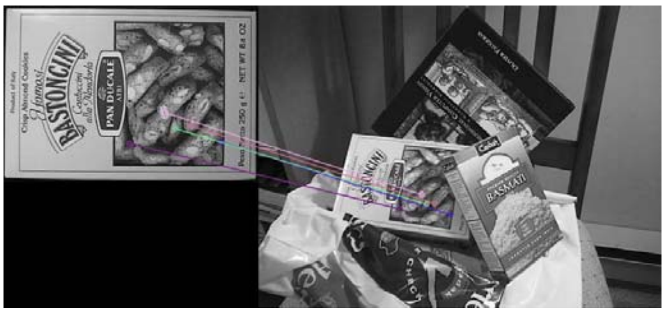

Figure 4. Matching result from SURF.

Table 3. Error rate comparison

	Sample 1	Sample 2	Sample 3
SIFT	6.06%	15.78%	21.05%
SURF	100%	0%	64.28%
FAST+FREAK	86.85%	63.39%	63.63%
BRISK	100%	9.37%	30.59%
ORB	100%	93.82%	37.12%

Keypoints will be matched by using FLANN (Fast Library for Approximate Nearest Neighbors) [14]. Only good matches, which have low distance, are kept. Table 3 shows error rate of five algorithms, the lower the better. Error rate is calculated as the percentage of mismatched keypoints over the total number of matches. Figure 3 shows a sample matching results from SIFT. As we can see, there are 2 incorrect matches out of 33 matches, hence the error rate is 6.06%. Figure 4 shows 100% accuracy results from SURF. However, there are only 5 matches, the object may not be recognized if we choose high threshold. Figure 5 and 6 shows the results of SIFT and SURF in another test. This time SIFT retains reasonable results while SURF produce results with zero-accuracy. In this experiments, SIFT produces best overall results.

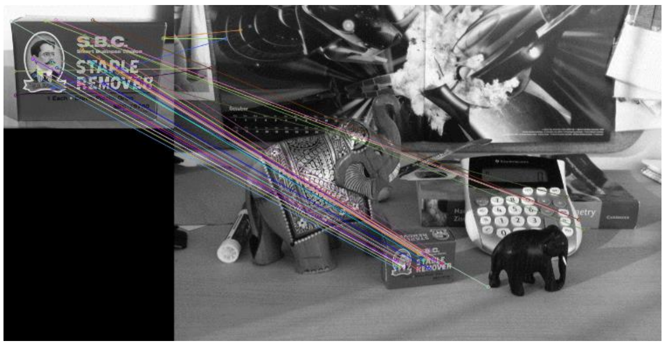
Figure 5. Matching results from SIFT in second sample.

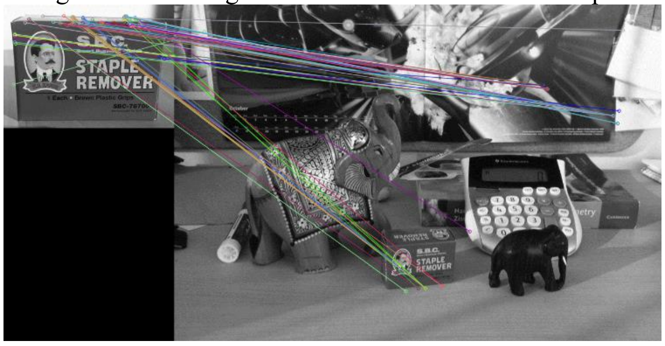
Figure 6. Bad matching results from SURF.

4. Conclusions

Recently, image feature detectors and descriptors have become an important algorithms in the computer vision and image processing. This study we reviewed several feature detection and feature description techniques. Two experiments have been conducted to compare the performance of algorithms. From experimental results, in general SURF produce best performance in feature detection, while SIFT dominates in object localization tests. For future works, we are trying to improve the accuracy of point feature detector and descriptor for object recognition in various environments.

5. Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(2015R1D1A1A01057518).

References

[1] Yap, T., Jiang, X. and Kot, A.C.: Two-dimensional polar harmonic transforms for invariant image representation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(7):1259–1270, 2010.
[2] Liu, S. and Bai, X.: Discriminative features for image classification and retrieval. Pattern Recognition Letter 33(6):744–751, 2012.
[3] Andreopoulos, A. and Tsotsos, J.: 50 years of object recognition: directions forward. Computer Vision and Image Understanding, 117(8):827–891, 2013.
[4] Moreels, P. and Perona, P.: Evaluation of features detectors and descriptors based on 3D objects. International Journal of Computer Vision, 73(3):263–284, 2007.
[5] Takacs, G., Chandrasekhar, V., Tsai, S., Chen, D., Grzeszczuk, R. and Girod, B.: Rotation-invariant fast features for large-scale recognition and real-time tracking. Signal Processing: Image Communication, 28(4):334–344, 2013.
[6] Mian, A., Bennamoun, M. and Owens, R.: Keypoint detection and local feature matching for textured 3D face recognition. International Journal of Computer Vision, 79(1):1–12, 2008.
[7] Lowe, D. G.: Object recognition from local scale-invariant features. In Proceedings of the Seventh IEEE International Conference on Computer Vision, volume 2, pages 1150–1157, 1999.
[8] Bay, H., Ess, A., Tuytelaars, T. and Gool, L. V.: Speeded-Up Robust Features (SURF). Computer Vision and Image Understanding, 110(3):346–359, 2008.
[9] Rosten, E. and Drummond T.: Machine learning for high-speed corner detection. In Proceedings of 9th European Conference on Computer Vision, volume 2, pages 430–443, 2006.
[10] Leutenegger, S., Chli, M. and Siegwart, R. Y.: BRISK: Binary Robust Invariant Scalable Keypoints. In Proceedings of the 2011 International Conference on Computer Vision, volume 1, pages 2548–2555, 2011.
[11] Calonder, M., Lepetit, V., Strecha, C. and Fua, P.: BRIEF: Binary Robust Independent Elementary Features. In Proceedings of 11th European Conference on Computer Vision, volume 1, pages 778–792, 2010.
[12] Rublee, E., Rabaud, V., Konolige, K. and Bradski, G: ORB: an efficient alternative to SIFT or SURF. In Proceedings of IEEE International Conference on Computer Vision, volume 1, pages 2564–2571 , 2011.
[13] Ortiz, R.: FREAK: Fast Retina Keypoint. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition, volume 1, pages 510–517, 2012.
[14] Muja, M. and Lowe, D. G.: Fast approximate nearest neighbors with automatic algorithm configuration. In Proceedings of VISAPP International Conference on Computer Vision Theory and Applications, volume 1, pages 331–340, 2009.

* Corresponding author