B-SIFT:A Binary SIFT Based Local Image Feature Descriptor *

Ni Zhen-sheng [1,2]

[1]National Engineering Research Center of Digital Life, State-Province Joint Laboratory of Digital Home Interactive Applications, School of Information Science & Technology, Sun Yat-sen University, Guangzhou 510006,China
nzhsh@mail2.sysu.edu.cn

[2]Shenzhen Digital Home Key Technology Engineering Laboratory,Research Institute of Sun Yat-sen University in Shenzhen, Shenzhen 518057, China

***Abstract*— Local feature point detection and description are the basis for Computer Vision. SIFT is one of the most efficient local image descriptors and have been well studied in recent years. In this paper, we introduce B-SIFT, a novel binary local image descriptor which is based with SIFT. The method is compared with SIFT, and is shown that B-SIFT is better both in accuracy and efficiency.**

Keywords-SIFT descriptor ; image matching; binary image descriptor.

I. INTRODUCTION

Local image interest point detectors and descriptors are widely applied in many Computer Vision areas including visual tracking, image registration and image recognition. Visual tracking is a key factor for many applications and most of these are computed in real time [3]. Image registration is the process of overlaying two or more images of the same scene taken at different times, from different sensors or from different viewpoint [4]. Image recognition as image retrieval is to create "visual words" for searching most similar images from large databases. All of these areas require image feature points to be robust, accurate and fast computed. To achieve this requirement, modern researches focus on making image feature points both in high accuracy and low complexity.

More and more local image descriptor algorithms have been designed in recent years. One of the most state-of-the-art algorithm is Lowe's SIFT[1]. SIFT provides scale-invariant and rotate-invariant descriptors which are highly discriminative for similar image comparison. Many algorithms has been proposed based in SIFT. PCA-SIFT[2] reduced SIFT feature dimension by using Principal Components Analysis. CSIFT[5] extended the SIFT algorithm to extract colored local invariant feather. Rank-SIFT[7] use learning-to-rank to improve the stability of detected SIFT keypoints.

Besides improving the descriptor feature quality, some other works are focus in speed. Bay et al proposed SURF[6] in 2008. SURF is also scale and rotation invariant and speed up computation by using integral image and box filters. BRIEF[8] is a binary local descriptor. It is popular in these days for its binary form of vectors can easily be matched by computer. BRISK[9] is another binary local descriptor following BRIEF. This algorithm used FAST[10] to detect keypoints and improve DAISY[11] to construct the binary feature vectors, which makes BRISK descriptor get more invariant properties and faster computational ability than BRIEF.

Usually, faster descriptor has less accuracy and stability. Binary feature vectors are hard to describe the local keypoints precisely. In this paper, we consider accuracy and efficiency trade-off of an descriptor. We construct SIFT descriptor in binary form to avoid complex computation in descriptor matching and reduce feature size in descriptor storage. Meanwhile, we inprove matching accuracy by realigning SIFT orientation features.

The remainder of this paper is organized as follows. Section II reviews the key steps of SIFT algorithm. Section III introduces our algorithm in details. Experiment results are discussed in Section IV. And we summarize total works in Section V.

II. THE SIFT DESCRIPTOR REVIEW

SIFT algorithm is based on three steps: (1) keypoint detection; (2) orientation assignment and (3) keypoint description.

A. Scale-Space Keypoint Detection

In the first step, a Gaussian pyramid is constructed as in figure 1. Keypoint detection is to find local extremum in a series of difference-of-Gaussian (DoG) images. The DoG function $D(x, y, \sigma)$ is computed by the difference of two adjacent scaled images.

Then some unstable points of these local extrema are filtered and the rest are localized to sub-pixel accuracy .

*This research is supported by NSFC-Guangdong Joint Fund (U0935004, U1135003), the National Key Technology R&D Program (2011BAH27B01, 2011BHA16B08), and the Industry-academy-research Project of Guangdong (2011A091000032).

978-0-7695-4899-9/12 $26.00 © 2012 IEEE
DOI 10.1109/ICDH.2012.69

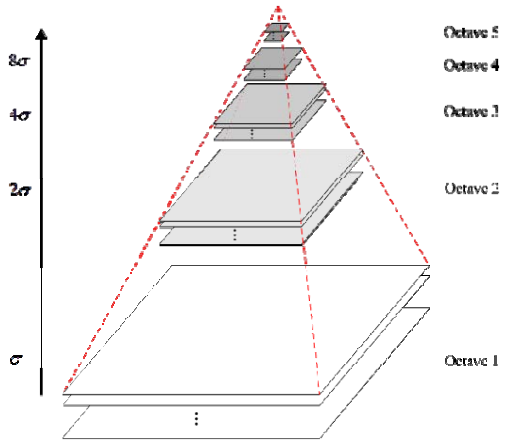


Figure 1. Gaussian pyramid

$$D(x,y,\sigma)=\left[G(x,y,k\sigma)-G(x,y,\sigma)\right]*I(x,y) \\ =L(x,y,k\sigma)-L(x,y,\sigma) \quad (1)$$

B. Keypoint Orientation Assignment

A keypoint is assigned an orientation to make the descriptor rotate-invariant. For each keypoint, dominant orientation is identified based on the closest scaled image. An orientation histogram of local image gradients is constructed in figure 2.

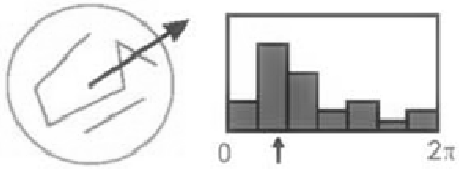


Figure 2. Local gradient histogram

A pixel $p(x, y)$ gradient magnitude $m(x, y)$ and orientation $\theta(x, y)$ is computed as follows:

$$m(x,y)=\sqrt{\left(L(x+1,y)-L(x-1,y)\right)^2+\left(L(x,y+1)-L(x,y-1)\right)^2} \quad (2)$$

$$\theta(x,y)=\tan^{-1}\left[\frac{L(x,y+1)-L(x,y-1)}{L(x+1,y)-L(x-1,y)}\right] \quad (3)$$

The magnitude of each pixel then would be weighted by a Gaussian filter as described in [1]. Then every weighted magnitude is add into the histogram which has 36 bins. Each bin covers 10 degree so that every pixel can be filled in. The dominant peak in the histogram is assigned as the orientation of the keypoint.

C. Keypoint Description

After step B, all the keypoints are detected and each keypoint contains three properties: scale, orientation and location, which make a keypoint scale-invariant and rotate-invariant. This step is to find a feature vector to describe the keypoint. SIFT uses the local gradient data to create the descriptor.

Figure 3 present a sample of Lowe's SIFT descriptor. Lowe used 16 orientation histograms aligned in a 4x4 grid. Each histogram has 8 orientation bins. This 4x4x8 elements is combined as the SIFT feature vector. After been constructed, this 128-element vector is then normalized to unit length. Finally, a normalized 128-element float type vector is created as the SIFT keypoint descriptor.

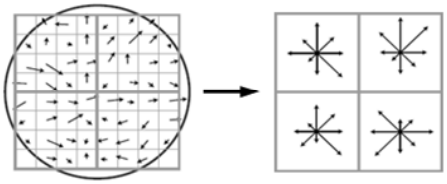


Image gradients Keypoint descriptor

Figure 3. Each orientation histogram calculated from a 4x4 pixel window and divided to 8 orientation bins[1].

In the next section, we use this 4x4x8 format as a typical calculation input to introduce B-SIFT descriptor.

III. B-SIFT DESCRIPTOR

In this section we describe our descriptor (termed B-SIFT) composition algorithm. As the feature detection and the orientation estimation procedures are just the same as SIFT, we focus our goal on transforming SIFT float type descriptor into our binary type descriptor. Our algorithm modified the SIFT keypoint description module to construct a binary type descriptor instead of the SIFT 128-length float numbers. We use the same module from SIFT algorithm to detect the image keypoints and to compute the keypoint location, scale and orientation.

A. Keypoint Description

First of all, when the gradient direction has been computed we rotate the image to the keypoint p main direction, and sample a new orientation histogram for each of the 4x4 sub-regions with 8 different directions compared with p. Instead of normalizing these 128 histogram values to build the standard SIFT descriptor, B-SIFT considers the differences of the 8 aggregated values from one single orientation histogram and the relation between the adjacent sub-regions to construct a 384-dimension binary vector. Therefore, B-SIFT descriptor contains three different parts: inner values, horizontal values and vertical values. Each part has 128 bit binary values.

Inner Values

Inner values present the difference between the 8 directions' aggregated values in every gradient histogram.

We compare each direction value clockwise with the next one. Each histogram gets 8 bits of inner values. To sum up all the 4x4 grids, we get a 128-bit long binary vector.

Figure 4 is a 2x2 histograms sample. Let S_{mn} denotes the sub-region of the *m*th column and *n*th row position of the 4x4 grids. Let $\mathbf{v}_i$ denotes the *i*th direction value of the histogram. For each sub-region, the *i*th bit $\mathbf{b_i}$ of the inner value corresponds to:

$$b_i = \begin{cases} 1 & v_i \geq v_{i+1} \\ 0 & v_i < v_{i+1} \end{cases} \quad \text{i=1 to 7}$$
$$b_8 = \begin{cases} 1 & v_8 \geq v_1 \\ 0 & v_8 < v_1 \end{cases} \quad (4)$$

The byte(8 bits) of one histogram is combined from:

$$S_{mn} = [b_1, b_2, ..., b_8] \quad m\text{=1 to 4; } n\text{=1 to 4} \quad (5)$$

The 128 bits of inner values V_I is to combine the 4x4 grids values together:

$$V_I = [S_{11}, S_{12}, ..., S_{mn}] \quad (6)$$

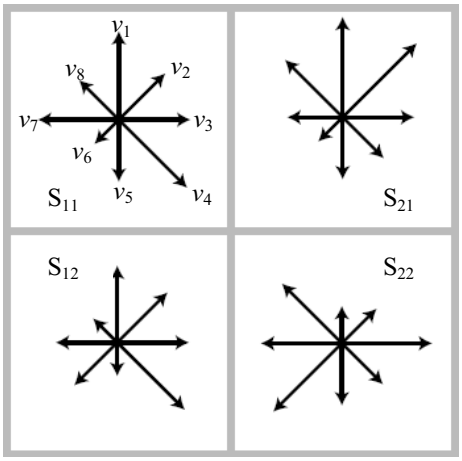


Figure 4. Sample of a 2x2 grid gradient histograms.

Horizontal Values

Horizontal values present the difference of the horizontally adjacent histograms' values at the same direction. As shown in Figure 4, we compare the 4 histograms of each row of the 4x4 grid. Each histogram returns a 8-bit result S_{mn}. Totally we combine all the histograms and get another 128-bit result. Formula (7) and (8) show the calculating process. Note that it is just a little different from formula (4) and (5).

$$b_i \mid S_{mn} = \begin{cases} 1 & v_i \mid S_{mn} \geq v_i \mid S_{(m+1)n} \\ 0 & v_i \mid S_{mn} < v_i \mid S_{(m+1)n} \end{cases} \quad m\text{=1 to 3} \quad (7)$$

$$b_i \mid S_{4n} = \begin{cases} 1 & v_i \mid S_{4n} \geq vi \mid S_{1n} \\ 0 & v_i \mid S_{4n} < vi \mid S_{1n} \end{cases} \quad n\text{=1 to 4; } i\text{=1 to 8}$$

$$S_{mn} = [b_1, b_2, ..., b_8] \quad m\text{=1 to 4; } n\text{=1 to 4} \quad (8)$$

The 128 bits of horizontal values V_H is:

$$V_H = [S_{11}, S_{12}, ..., S_{mn}] \quad (9)$$

Vertical Values

Vertical values consider the vertical different of the gradient histograms. It is very similar to the horizontal values. We combine every column of the 4x4 grid values together to get the last 128-bit feature vector.

$$b_i \mid S_{mn} = \begin{cases} 1 & v_i \mid S_{mn} \geq v_i \mid S_{m(n+1)} \\ 0 & v_i \mid S_{mn} < v_i \mid S_{m(n+1)} \end{cases} \quad n\text{=1 to 3}$$
$$b_i \mid S_{m4} = \begin{cases} 1 & v_i \mid S_{m4} \geq vi \mid S_{m1} \\ 0 & v_i \mid S_{m4} \geq vi \mid S_{m1} \end{cases} \quad m\text{=1 to 4; } i\text{=1 to 8} \quad (10)$$

$$S_{mn} = [b_1, b_2, ..., b_8] \quad m\text{=1 to 4; } n\text{=1 to 4} \quad (11)$$

The 128 bits of vertical values V_I is:

$$V_V = [S_{11}, S_{12}, ..., S_{mn}] \quad (12)$$

Combine these three values we get B-SIFT descriptor, a 384 length binary vector $D_{B\text{-}SIFT}$. It only costs 48 8-bit bytes in computer. This can save 90% of the image feature storage at least.

$$D_{B-SIFT} = [V_I, V_H, V_V] \quad (13)$$

As these 128x3 bits have no ordinal relation, organizing sequences of calculating these binary values is no order restriction . Which means constructing an B-SIFT descriptor can be easily parallelized in computer implementation. The cost of reforming SIFT descriptors is comparatively very small. After local descriptor binaryzation, the keypoints matching time declines obviously which will be verified in the next section. With above deduction, the whole processing time of our method would be much less than original SIFT descriptor.

B. Keypoints matching

Judging two B-SIFT descriptors is similar or not by computing the Hamming Distance of the two 384 binary vectors, which can be easily done by modern computer

architectures. Just as in BRIEF[6], we define a certain threshold $T = 40$ for the matching B-SIFT keypoint pairs. How to determine this threshold will be further discussed in next section.

IV. Experiment Results

As our B-SIFT is based on original SIFT algorithm to detect the keypoints, its keypoint repeatability is same as SIFT. In this section, we discuss on B-SIFT's matching accuracy and efficiency.

A. Threshold Determining & Precision Recall Trade-off

As other local descriptor matching procedures, the first of all is to determine matching threshold. A high threshold may cause high precision rate but low recall rate and vise versa. Threshold makes an important role especially in image retrieval system. Different from video tracking which is another local descriptor widely used field, image retrieval usually matches thousands of images for one query picture to fetch the best match result. An excessive low threshold can make matching time growing extremely.

B-SIFT descriptor matching is to calculate the Hamming distance: the number of bits different in the two descriptors. As the descriptor is a 384-dimension vector, threshold over 100 would make it lost the distinctiveness. So we test several images changing threshold from 10 to 100 and figure out that the threshold about 40 makes best performance considering precision recall trade-off.

B. Keypoint Matching Accuracy

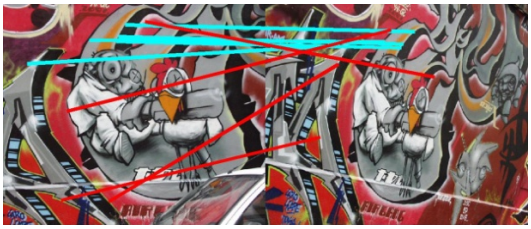
For accuracy test, we compare our method with SIFT on some real-world images taken from different viewpoints. We choose the Graffti[1] images (contain affine, scale and angle changing situation), and the test images (a set of buildings screened in different position) offered in VLFEAT[2] .

(a)

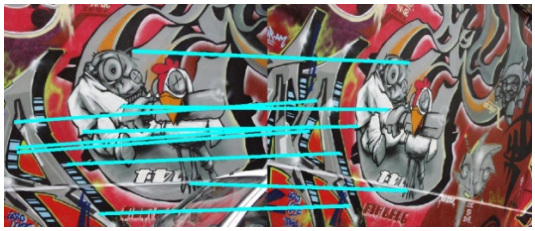
(b)

Figure 5. Images of building scenes. (a) SIFT (b) B-SIFT.

Figure 5 shows the matching accuracy of SIFT and B-SIFT in two corresponding building scenes. We draw the top 30 matches of each method in the combined image. The correct matches are lined in cyan color. And the red lines point out the mismatches. SIFT has 3 errors of the 30 matches, 2 errors at the corner of a window and 1 error at the edge of the chimney. Besides, no error occurred in B-SIFT top 30 matches.



(a)



(b)

Figure 6. Images of Graffiti. (a) SIFT (b) B-SIFT.

Figure 6 presents SIFT and B-SIFT keypoints matching accuracy of the Graffiti images. We line the top 10 matches to show the results clearly. As Graffiti images have affine transform effect, SIFT get worse matching result 6 correct and 4 errors of the top 10 matches. Meanwhile, B-SIFT get 10 correct matches, which implies our method can efficiently short this lack of affine transformation.

C. Image Retrieval Test

Image Retrieval Systems usually store image features as "visual words". Local image descriptor is a well-formed feature vector so can be easily fitted in. As in real world, query images are various and hard to find a extremely same picture in the image databases. So we choose the PCA-SIFT[2] small dataset[3] (10 common household items, 3 different viewpoints images for each) to test B-SIFT performance in image retrieval usage. PCA-SIFT formulated local descriptors in image retrieval as follows[2]. First, to extract the corresponding feature vectors of the images. Then to compare each feature vector of query image

1 http://www.robots.ox.ac.uk/~vgg/research/affine/
2 http://www.vlfeat.org
3 http://www.cs.cmu.edu/~yke/pcasift/

against all feature vectors in other images and count the number of features that are within a threshold distance. Unlike the PCA-SIFT counting the number of matches in a certain threshold as a similarity between images, we use the average distances value of top five matches to avoid the bad usability: return no result in a query system.

For this experiment, we use the same judgment as PCA-SIFT. Each image was used as a query into the dataset. If both of the other two images of the corresponding object were returned in the top three positions, this image query test awarded 2 points. If only one of the two corresponding pictures returned would get 1 point. Otherwise, no point earned. The total point is 60 and we calculate the percentage of the correct retrieval. The result is shown in Table 1. B-SIFT has a 67% retrieval correct rate better than SIFT.

TABLE I. CORRECT RETRIEVAL

	SIFT	**B-SIFT**
Correct rate	48%	67%

D. Matching Time

To compare B-SIFT's speed performance with SIFT, we calculate the precise computing time with an Intel quad-core i7 3.4GHz processor and 4GBx2 DDR3-1333 memory, running Ubuntu 11.10 64-bit OS. We do not optimize the Euclidean distance computing instructions in SIFT matching, and the Hamming distance in B-SIFT matching. Table 2 presents the time costs of comparing 1000 vectors distances.

TABLE II. MATCH TIME

	SIFT	**B-SIFT**
Matching time(ns)	359.8	239

B-SIFT get nearly 50% faster than SIFT in matching efficiency. And this advantage level would be higher for some processor architectures which is no float computing optimization.

V. CONCLUSIONS

We have proposed a SIFT-like local keypoints descriptor B-SIFT. It represents SIFT in binary form, and get more advantage both in matching accuracy and efficiency. We believe that B-SIFT works well in real-world image retrieval application and can be easily fitted with other type of image features.

REFERENCES

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," International Journal of Computer Vision (IJCV), 60(2):91-110, 2004.

[2] Y. Ke and R. Sukthankar, "PCA-SIFT: A More Distinctive Representation for Local Image Descriptors," 2004.2.

[3] S. Gauglitz, T. Höllerer and M. Turk, "Evaluation of interest point detectors and feature descriptors for visual tracking," International Journal of Computer Vision (IJCV), 94(3):335-360,2011.

[4] B. Zitova and J. Flusser, "Image registration methods: A survey," Image and Vision Computing, 21, 997-1000, 2003.

[5] A. E. Abdel-Hakim and A. A. Farag, "CSIFT: A SIFT descriptor with color invariant characteristics," In Proceedings of the IEEE Conference on Computer Vision and Pattern Recogniztion (CVPR), 2006.7.

[6] H. Bay, A. Ess, T. Tuytelaars and L. V. Gool, "SURF: Speeded up robust features," Computer Vision and Image Understanding (CVIU), 110(3):346-359, 2008.

[7] B. Li, R. Xiao, Z. Li, R. Cai, B. L. Lu and L, Zhang, "Rank-SIFT: Learning to Rank Repeatable Local Interest Points," In Proceedings of the IEEE Conference on Computer Vision and Pattern Recogniztion (CVPR), 2011.

[8] M. Calonder, V. Lepetit, C. Strecha and P. Fua, "BRIEF: Binary Robust Independent Elementary Features," In Proceedings of the European Conference on Computer Vision (ECCV), 2008. 1.

[9] S. Leutenegger, M. Chli and R. Y. Siegwar, "BRISK: Binary Robust Invariant Scalable Keypoints," In Proceedings of the International Conference on Computer Vision (ICCV), 2011.

[10] E. Rosten and T. Drummond, "Machine learning for highspeed corner detection," In Proceedings of the European Conference on Computer Vision (ECCV), 2006. 1,2.

[11] E.Tola, V. Lepetit and P. Fua, "Daisy: an Efficient Dense Descriptor Applied to Wide Baseline Stereo," IEEE Trans. On Pattern Analysis and Machine Intelligence (PAMI), 32(5): 815-830, 2010.4.