10.2478/msr-2013-0022

CGCI-SIFT: A More Efficient and Compact Representation of Local Descriptor

Dongliang Su[1], Jian Wu[1], Zhiming Cui[1], Victor S. Sheng[2], Shengrong Gong[1]

[1]The Institute of Intelligent Information Processing and Application, Soochow University, Suzhou 215006, China
[2]Department of Computer Science, University of Central Arkansas, Conway 72035, USA,
jianwu@suda.edu.cn

This paper proposes a novel invariant local descriptor, a combination of gradient histograms with contrast intensity (CGCI), for image matching and object recognition. Considering the different contributions of sub-regions inside a local interest region to an interest point, we divide the local interest region around the interest point into two main sub-regions: an inner region and a peripheral region. Then we describe the divided regions with gradient histogram information for the inner region and contrast intensity information for the peripheral region respectively. The contrast intensity information is defined as intensity difference between an interest point and other pixels in the local region. Our experimental results demonstrate that the proposed descriptor performs better than SIFT and its variants PCA-SIFT and SURF with various optical and geometric transformations. It also has better matching efficiency than SIFT and its variants PCA-SIFT and SURF, and has the potential to be used in a variety of real-time applications.

Keywords: Image matching, descriptor, SIFT, CGCI-SIFT, real-time

1. INTRODUCTION

IMAGE MATCHING is a primary technology in computer vision and image processing. Among the image matching algorithms, local descriptor algorithms [1] are more stable. Local descriptors are discriminative, and robust to partial occlusion. They do not require segmentation preprocessing, and are invariant under a variety of transformations. All these properties make local descriptor algorithms be widely applied in many fields, such as, content-based large-scale retrieval [2], video analysis, copy detection, object recognition, photo tourism, and 3D reconstruction [3].

A good local descriptor algorithm should have following characteristics: no necessity of pre-segmenting images [4], high repeatability of feature detector, low dimension of feature descriptor, robustness to partial occlusion, and invariance against image transformations, such as, illumination, rotation, scale, blur, and affine.

Local descriptors have received considerable attention in recent years. Harris proposed the Harris corner detector [5], based on the eigenvalues of the second-moment matrix, but it is not scale-invariant. Lowe introduced a scale invariant feature transformation (SIFT) [1]. It is invariant under a variety of transformations, such as scale and viewpoint changes, rotation, and illumination transformations. Mikolajczyk and Schmid [6] showed that SIFT is one of the most effective image matching algorithms against viewpoint and scale transformations. However, the dimensionality of a SIFT descriptor is high. This results in inefficiency in real-time applications. In order to improve the matching accuracy and reduce the matching time, various extensions of SIFT have been proposed. For example, Ke and Sukthankar proposed PCA–SIFT [8], which uses image gradient patch, and applies principal component analysis (PCA) to replace the smoothed weighted histograms in SIFT to reduce the size of a descriptor. It performs better on artificially generated data. E.N. Mortensen proposed GSIFT [9], which combines SIFT with global texture information. H. Bay proposed SURF [7], which has similar steps with SIFT. But SURF adopts a new processing method for each step. Its computing speed is faster. E. Tola proposed a descriptor DAISY [10], which computes dense depth and occlusion maps from wide-baseline image pairs on the basis of the EM algorithm. It is very efficient for intensive computing. Yang and Sluzek [11] proposed a low dimension descriptor combined with shape features and location information.

Local descriptor algorithms consist of three primary steps. First, interest points are detected at distinctive locations in an image, such as corners. Second, the local region of the interest point is represented by a feature vector. The descriptor has to be distinctive, robust to noise and detection errors, and invariant against geometric and photometric transformations. Finally, vectors of descriptors are matched between different images. Many extensions of SIFT are mainly related to the construction of the SIFT descriptor. The algorithm proposed in this paper is also related to the improvement of the SIFT descriptor.

In this paper, we propose a novel invariant local descriptor, a combination of gradient histograms with contrast intensity (CGCI), for image matching and object recognition. It exploits contrast intensity information by evaluating intensity difference between an interest point and other pixels in the local region. It is one of the extensions of a standard descriptor SIFT, called CGCI-SIFT in following paragraphs. It is more efficient than SIFT and its two variants (PCA-SIFT and SURF), since it can require less data to represent a local region. Our experimental results in Section 4 show that it not only achieves significantly better performance, but also uses less time in both feature extraction and image matching, comparing with SIFT and its two variants.

The remainder of this paper is organized as follows. Section 2 reviews the relevant aspects of the SIFT algorithm. Section 3 details the local feature construction of our CGCI-SIFT descriptor. Section 4 presents our evaluation methodology, performance metrics, and experimental results. Finally, we conclude our work in Section 5.

2. REVIEW OF THE SIFT ALGORITHM

SIFT consists of four major stages of computation used to generate a set of image features: (1) scale-space extreme detection; (2) keypoint localization; (3) orientation assignment; (4) keypoint descriptor.

In the first stage, SIFT searches over all image locations and scales to find potential interest points as keypoints. It is implemented efficiently by constructing a Gaussian pyramid and searching for extreme in a series of difference-of-Gaussian (DoG) [12] images. It is proved that under a series of reasonable hypotheses a Gaussian function is the only possible scale-space kernel [1].

The scale-space of an image is defined as a function $L(x,y,\sigma)$, which is the convolution of an original image $I(x,y)$ with a variable-scale Gaussian $G(x,y,\sigma)$. That is:

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y) \tag{1}$$

where $*$ is the convolution operation in x and y, and

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{2}$$

where σ is a scale-space factor. The size of σ is the determiner of a smoothing degree of an image. Large scale represents its general information, and small scale for its detailed characteristics [13].

The SIFT algorithm detects stable interest point locations by a DOG (difference-of-Gaussian) function, which can be computed from the difference of two nearby scales:

$$\begin{aligned} D(x,y,\sigma) &= (G(x,y,k\sigma) - G(x,y,\sigma)) \times I(x,y) \\ &= L(x,y,k\sigma) - L(x,y,\sigma) \end{aligned} \tag{3}$$

where k is a constant multiplicative factor.

In the second stage, interest point candidates are localized to sub-pixel accuracy and eliminated if found to be low-contrast keypoints and not robust to small amounts of noise.

In the third step, each keypoint is assigned one or two dominant orientations based on its local region. This step makes SIFT invariant to rotation. The Gaussian-smoothed image $L(x,y,\sigma)$ at the scale σ of an interest point is taken, so that all computation is performed in a scale-invariant manner. For an image sample $L(x,y)$ at a scale σ, its gradient magnitude $m(x,y)$ and gradient orientation $\theta(x,y)$ are computed using pixel differences. Their mathematical definitions are:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \tag{4}$$

$$\theta(x,y) = \arctan \frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)} \tag{5}$$

The final stage of SIFT constructs a representation for each keypoint based on a patch of pixels in its local region. A 4×4 array of orientation histograms with 8 bins are computed in a 16×16 region around a keypoint. These histograms are computed from magnitude and orientation values of samples in the local region. Each histogram contains samples from 4×4 = 16 sub-regions divided from an original local region. There are 4×4 = 16 histograms and each with 8 bins, so the vector has 128 elements. The 128-dimension vector is then normalized to a unit length in order to reduce the impact of non-linear illumination.

The construction of the SIFT keypoint descriptor is complicated, and the dimension of the SIFT descriptor is high. Our initial motivation is to explore simpler alternatives, which are faster for computation, more distinctive and compact.

3. THE CGCI-SIFT DESCRIPTOR

The main idea of all invariant local descriptors is on how to divide the region of the neighborhood of a keypoint and represent the region effectively and discriminatively. In this section, we first discuss our fundamental ideas, and then explain the description of a keypoint of our CGCI-SIFT.

3.1. Fundamental Ideas

The first step of establishing local invariant descriptor is to effectively choose a patch of pixels as a region in the local neighborhood of a keypoint. SIFT chooses an adjacent 16×16 region, using the keypoint as the center. After the region is chosen, SIFT divides this region into 4×4 sub-regions, and calculate the gradient histogram in each sub-region. Our CGCI-SIFT follows this approach to choose a 16×16 local region for each keypoint.

After a local region is chosen for a keypoint, one of the main issues is how to represent the local region effectively and discriminatively. As we know, the intensity of each pixel in the local region is different. With in-depth investigations and analyses, we conjecture that the intensity of the pixels near to the keypoint is similar to that of the keypoint. The intensity of the pixels far away from the keypoint could be significantly different from that of the keypoint. For example, Fig.1. shows a typical gray-scale intensity distribution of its neighborhood pixels of an interest point p_k. p_k is the keypoint in its local region. As we can see from the figure, the intensity of each pixel is different. The intensity difference between each pixel and the keypoint is varied. The intensity difference between each pixel (indicated by white dots in the figure) and p_k is small, and these pixels are closest to the keypoint. However, the intensity difference between each pixel (indicated by black dots or stars) and p_k is great, and these pixels are far away from the keypoint. This observation motives us to divide the 16×16 local region into two non-overlapping sub-regions: an inner region R_{inner} and a peripheral region $R_{peripheral}$, as shown in Fig.1.

The log-polar coordinate division system has been proved that pixels near the interest point are more susceptive than pixels far away [15]. It is also proved that the log-polar coordinate division system is more effective [6], and is used by many algorithms. For example, GLOH [6] changes a rectangular grid division system of SIFT into a log-polar location grid [14]. Thus, in this paper, we use a *log-polar coordinate grid* with three bins in the radial direction and eight bins in the angular direction. (Details are explained in next subsection.) The direction of 0 degrees in the log-polar coordinate system is set to coincide with the dominant orientations of the interest point based on its local region.

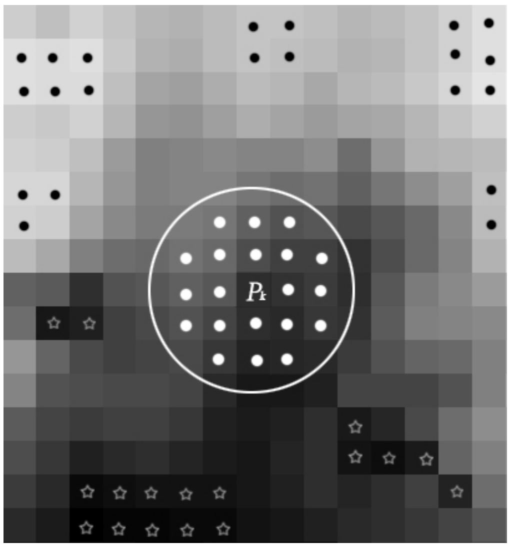


Fig.1. A typical gray-scale intensity distribution of neighboring pixels.

3.2. *Description of Local Feature Points*

Our algorithm CGCI-SIFT is one of local descriptors. It receives the same input as the standard SIFT descriptor: the scale, location, and dominant orientations of an interest point. As we said before, it extracts a 16×16 region around the keypoint at the scale, and rotates to its dominant orientation, in the same manner as SIFT does, shown in Fig.1. Each interest point p_k is in the center of a 16×16 local region R.

Considering different effects of the pixels in the local region of an interest point, CGCI-SIFT makes use of two known methods to construct its descriptor, instead of storing the gradient orientations of all pixels in a local region like the SIFT approach. Besides, in contrast to classical approaches like SIFT (PCA-SIFT and SURF) that exploit the rectangular grid as a division system, CGCI-SIFT applies a log-polar coordinate system to divide the local region. It uses a log-polar coordinate grid with three bins in the radial direction (the radiuses of the three bins are set to 2, 5, and 8 respectively). It divides the 16×16 local region R into two non-overlapping sub-regions: an inner region, R_{inner} and a peripheral region $R_{peripheral}$. The region within radius 2 is the inner region. Both the region (outside the circle with radius 2 and inside the circle with radius 5) and the region (outside the circle with radius 5 and inside the circle with radius 8) are the two components of the peripheral region. The following paragraph provides the details of how the log-polar coordinate system is used in our CGCI-SIFT.

CGCI-SIFT further uses a log-polar coordinate system (r,θ) to divide R_{inner} into η +1 non-overlapping sub-regions, denoted as: $R_0, R_1 \ldots R_\eta$ with $r = 2$, and to divide $R_{peripheral}$ into t - η non-overlapping sub-regions, denoted as: $R_{\eta+1,}\ R_{\eta+2} \ldots R_t$. Note that η and t are the two parameters of our CGCI-SIFT, which determines the number of sub-regions of the inner region R_{inner} and the number of sub-regions of the peripheral region $R_{peripheral}$ of a keypoint. Thus, the two parameters directly impact the dimensions of a keypoint description.

For R_{inner}, we adopt the gradient weighted histograms, which are defined in (4) and (5), to construct descriptors. We calculate the magnitudes and orientations of an image gradient in a sub-region R_i which is belong to the R_{inner}, and build smoothed orientation histograms to catch the representative information. The gradient orientations are quantized in 8 bins. Each bin of a sub-region R_i in R_{inner} can be represented as $G_{R_i} d_j (i \in \{0,1,2,\eta\},\ j \in \{0,1,2,3 \ldots 7\})$.

For $R_{peripheral}$, we consider a technique that represents the contrast values of pixels within a region with respect to a keypoint. A contrast value is defined as the intensity difference between a pixel and the keypoint [16]. In the following, we introduce how to compute contrast values.

For a pixel p in the local region around the interest point p_k, we compute the intensity difference $D(p)$ defined as $D(p) = I(p) - I(p_k)$, where $I(p)$ and $I(p_k)$ represent the intensity value of p and p_k respectively. For each p in a sub-region R_i, we define the positive intensity information (6) and negative intensity information (7) [16] with respect to p_k respectively:

$$H_{R_i}P = \frac{\Sigma\{(I(p) - I(p_k)) \mid p \in R_i \text{ and } D(p) \geq 0\}}{NumR_{i+}} \tag{6}$$

$$H_{R_i}N = \frac{\Sigma\{(I(p_k) - I(p)) \mid p \in R_i \text{ and } D(p) < 0\}}{NumR_{i-}} \tag{7}$$

where $NumR_{i+}$ represents the number of pixels whose intensity is greater than the interest point in the sub-region R_i. In contrast, $NumR_{i-}$ represents the number of pixels whose intensity is smaller than that of the interest point.

Next, we normalize the descriptor to a unit vector to deal with illumination changes. Finally, we combine the all histogram entries from all the sub-regions into a single vector. Thus, the CGCI-SIFT descriptor of p_k with its local region R can be defined as:

$$CGCI = (G_{R_0} d_0, \ldots, G_{R_0} d_7 \ \ldots \ G_{R_\eta} d_0, \ldots, G_{R_\eta} d_7, \\ H_{R_{(\eta+1)}} P, H_{R_{(\eta+1)}} N \ \ldots \ H_{R_t} P, H_{R_t} N) \tag{8}$$

From the above explanation, the number of dimensions of a descriptor provided by our CGCI-SIFT can be calculated using:

$$Dimen = 8 \times (\eta + 1) + 2 \times (\mathrm{t} - \eta)\,. \tag{9}$$

Our CGCI-SIFT is a general framework. It could have many different versions by setting its two parameters η and t. Each version has a certain dimension, which can be calculated by (9).

As we know, the dimension of the descriptor has a direct impact on its computation time. The dimension of CGCI-SIFT can be much lower than that of SIFT with proper parameter settings. Thus, CGCI-SIFT would also be more efficient than SIFT.

In our experiments, we set the radius of the log-polar coordinate to 2, 5, and 8 respectively for both the 64-dimension CGCI-SIFT (CGCI-64) and the 40-dimension CGCI-SIFT (CGCI-40). Please note that the parameters are set as $\eta = 3, t = 19$ in CGCI-64. For CGCI-40, its parameters are set as $\eta = 0, t = 16$.

The structures of the CGCI-64 descriptor and the CGCI-40 are shown in Fig.2. From the left sub-figure of Fig.2., we can see that its inner region is separated into four non-overlapping sub-regions R_0, R_1, R_2, R_3, and its peripheral region is separated into sixteen non-overlapping sub-regions $R_4, R_5, R_6 \ldots R_{19}$. Hence, there are 4×8=32 entries in the inner descriptor, and 2×16=32 entries in the peripheral descriptor. We combine the two parts together. Thus, the dimension of the CGCI-64 descriptor is 32+32 = 64.

It is easy to follow the above explanation to find out the dimension of the CGCI-40 descriptor is 32+8 = 40. We do not repeat the explanation here.

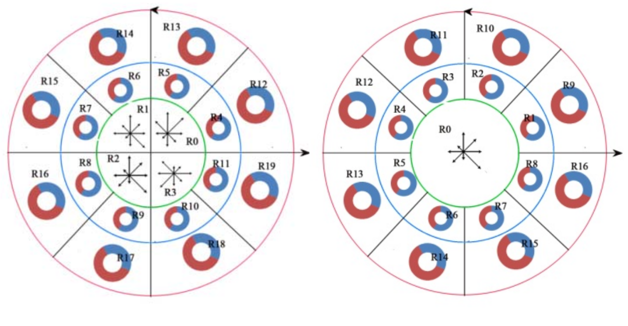


Fig.2. The region structures of CGCI-64 (left) and CGCI-40 (right).

4. EXPERIMENTS

In this section, we conduct experiments to investigate the performance of our CGCI-SIFT, and make comparisons with SIFT and its two variants (PCA-SIFT and SURF) under different situations: scale and rotation change, blur change, illumination change, JPEG compression, and affine change. We also investigate its time consumption, by comparing with SIFT and its two variants.

4.1. Data Sets

We evaluate our CGCI-SIFT on five image data sets used in [6] and [17]. These data sets consist of real images with different geometric and photometric transformations and for different scene types. Within the data sets, a set of *boat* images are used to evaluate the scale and rotation invariance of CGCI-SIFT, a set of *bike* images used to evaluate the blur invariance of CGCI-SIFT, a set of *Leuven* images used to evaluate the illumination invariance of CGCI-SIFT, a set of *house* images used to evaluate the JPEG compression invariance of CGCI-SIFT, and a set of *graffiti* images used to evaluate the affine invariance of CGCI-SIFT.

4.2. Evaluation Measures

We use two measures, the number of correct matches and false matches obtained from an image pair, to evaluate the performance of our CGCI-SIFT. The adopted matching strategy is the nearest neighbor distance ratio matching. The nearest neighbor distance ratio matching can be explained as follows.

Considering a pair of images (a reference image and a test image), p_a denotes a keypoint in the reference image, and p_b and p_c denote two keypoints in the test image, where p_b and p_c are the nearest point and second nearest point, respectively. We define that the keypoints p_a and p_b match if $dist(p_a, p_b) < \beta \times dist(p_a, p_c)$, where $dist()$ is a Euclidean distance between the descriptors of two keypoints, and β is a threshold. Otherwise, we define that the keypoints p_a and p_b mismatch. Every keypoint descriptor in the reference image is compared with every keypoint descriptor in the test image.

The threshold β is varied to obtain the curves to present our experimental results (*recall* versus *1-precision*, which is used in [6], [7] and [17]) in Section 4.3. Recall is defined as the number of correct matches with respect to the number of corresponding matches between two images of the same scene:

$$recall = \frac{\#correctMatches}{\#correspondences} \tag{10}$$

We determine the number of correct matches and correspondences using an overlap error ε. The overlap error measures how well the regions correspond under a particular transformation, and that is a homography in our case. It is defined by the ratio of the intersection over the union of the regions:

$$\varepsilon = 1 - \frac{A \cap H^T B H}{A \cup H^T B H} \tag{11}$$

where A and B are the regions and H is the homography between the two images. Details can be found in [17]. The correct match that we define is satisfied with the condition $\varepsilon < 0.5$, which means the overlap error in the image region covered by two corresponding regions is less than 0.5 of the region union.

The 1-precision represents the number of false matches relative to the total number of matches, defined as:

$$1 - precision = \frac{\#\ falseMatches}{\#correctMatches + \#falseMatches} \tag{12}$$

The recall and 1-precision are independent terms. The results in *recall versus 1-precision* show the performance of an algorithm on only a pair of images. We need to display the performance of an algorithm on between an image and a sequence of its transformations. Thus, we add another evaluation measure, correct match rate (CMR). Correct match rate is defined as follows.

$$CMR = \frac{\#correctMatchs}{\#correctMatchs + \#\ falseMatchs} \tag{13}$$

where $\#correctMatchs$ represents the number of correct matches, and $\#\ falseMatchs$ represents the number of false matches. In our experiments of evaluating the performance of each algorithm in term of correct matching rate (CMR), we set the threshold β=0.49 as the matching condition.

4.3. Experiment Results

In the experiments of this paper, we use CGCI-64 (η =3 and *t=19*) and CGCI-40 (η =0 and *t=16*) as the two representatives of our CGCI-SIFT, and compare their experimental results with SIFT, SURF and PCA-SIFT. Again, CGCI-64 denotes the CGCI-SIFT descriptor with 64 dimensions, while CGCI-40 denotes the CGCI-SIFT descriptor with 40 dimensions. Thus, we can see the performance of CGCI-SIFT under different dimensions. Note that CGCI-SIFT is a general framework. It could have many different versions by setting its two parameters η and t. Each version has a certain dimension, which can be calculated by (9). SURF and PCA-SIFT are extensions of SIFT. Both of them have reduced the dimension of SIFT, so we choose them to make comparisons.

All methods are implemented in Matlab 2010a and executed on a Dell PC computer. It has a Pentium(R) Dual-Core CPU E5300@2.60 GHz, and 4G RAM, running Windows 7.

Image Rotation and Scale. We conduct the first set of experiments on a set of boat images [17] to evaluate the robustness of our CGCI-SIFT against image rotation and scale change. The set of images is shown in Fig.3. This set has six images in total. Each represents different scale and rotation respectively. Note that the range of rotations of the images is from 30 to 50 degrees, and the range of their scales is from 2 to 2.5. We also compare our CGCI-SIFT with SIFT and its two variants (PCA-SIFT and SURF). Our experiments match the image A1 with others (A2-A6) respectively. The experimental results are shown in Fig.4.

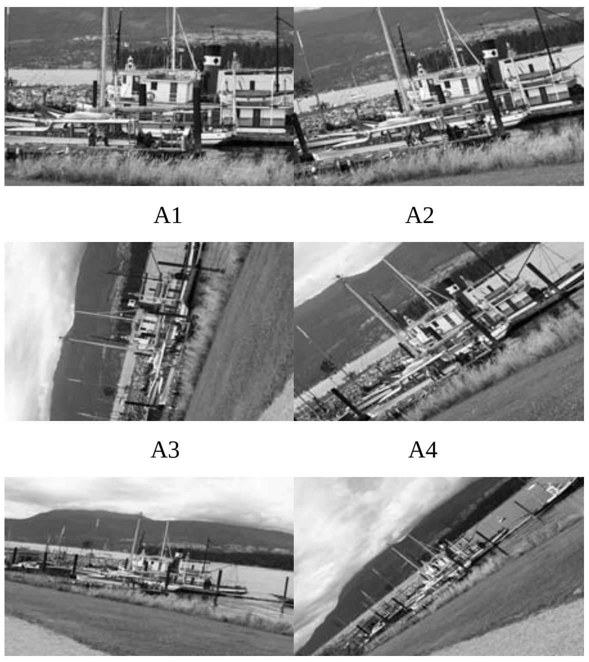
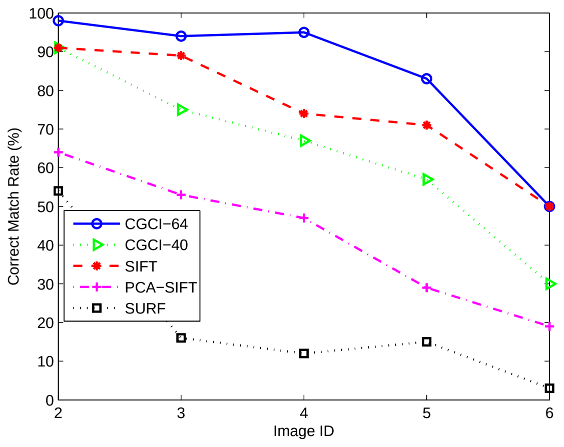
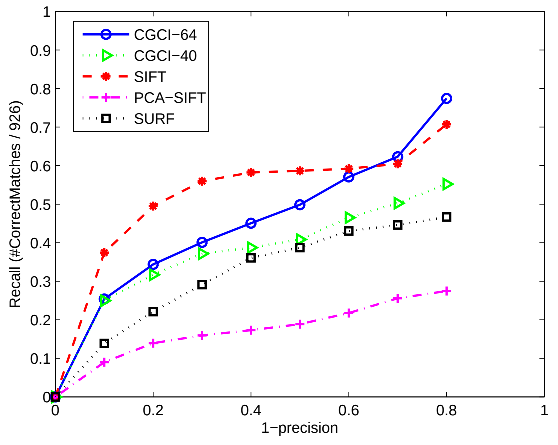


Fig.3. Images used for evaluation under rotation and scale changes.



(a) Correct Match Rate



(b) Recall versus 1-precision

Fig.4. Evaluation under rotation and scale changes.

Fig.4.(a). shows the matching correct rate of each algorithm between the image A1 and the others (A2-A6) respectively. Note that we also compare the performance of the two versions of CGCI-SIFT (CGCI-64 and CGCI-40). Fig.4.(a). clearly shows the relationships among the compared algorithms. From Fig.4.(a)., we can see that CGCI-64 has the highest matching correct rate under all comparing pairs. It is followed by SIFT. CGCI-40 performs the third, which is better than both PCA-SIFT and SURF. Between PCA-SIFT and SURF, SURF has the lowest matching correct rate under all image pairs.

Fig.4.(b). shows the specific experimental results for matching the image A1 with A6 for our CGCI-SIFT (CGCI-64 and CGCI-40), SIFT and its variants (PCA-SIFT and SURF). First, we can see that the recall of every algorithm increase with the increment of 1-precision, as expected. Among these algorithms, the recall of SIFT increases quickly when 1-precision is small. Our CGCI-64 increases more quickly than SIFT does when 1-precision is greater than 0.3. Thus, after 1-precision is greater than 0.7, our CGCI-64 has higher recall than SIFT. Except SIFT, both CGCI-64 and CGCI-40 always have higher recall than SURF and PCA-SIFT. Compared with CGCI-40, CGCI-64 is better.

Image Blur: The second set of experiments is conducted on a set of bike images to evaluate the performance of our CGCI-SIFT against blur invariance. The set of images is shown in Fig.5. It has six images in total. The image B1 is an original one from the image set [17]. We use it to produce 10 different blur images with different fuzzy radiuses (from 1 to 10, refer to the horizontal axis in Fig.6.(a).), which represent different degrees of blur. Fig.5. shows the original image and five produced images with their corresponding blur radius. The experimental results are shown in Fig.6.

Fig.6.(a). shows the matching correct rate of each algorithm between the original image B1 and the 10 generated images respectively. Note that we also compare the performance of the two versions of CGCI-SIFT (CGCI-64 and CGCI-40). Fig.6.(a). clearly shows the relationships among the compared algorithms. From Fig.6.(a)., we can see that both versions of our CGCI-SIFT (CGCI-64 and CGCI-40) perform significantly better than all others (SIFT, PCA-SIFT, and SURF). Between two versions themselves, CGCI-64 performs slightly better when the bur degree is greater than 6. Among the rest three algorithms, SIFT performs better than PCA-SIFT. SURF performs the worst.

Fig.6.(b). shows the specific experimental results for matching the original image B1 and the most blurring image B6 shown in Fig.5. for our CGCI-SIFT (CGCI-64 and CGCI-40), SIFT and its variants (PCA-SIFT and SURF). First, we can see that the recall of every algorithm increase with the increment of 1-precision, as expected. Among these algorithms, the recall of our CGCI-SIFT increases quickly when 1-precision is small. Both versions of our CGCI-SIFT always dominates all other algorithms (SIFT, PCA-SIFT, and SURF). Between the two versions of our CGCI-SIFT, CGCI-64 performs slightly better than CGCI-40. Among the three other algorithms (SIFT, PCA-SIFT, and SURF), the recall of SIFT is higher than that of SURF. The recall of SURF is higher than that of PCA-SIFT.

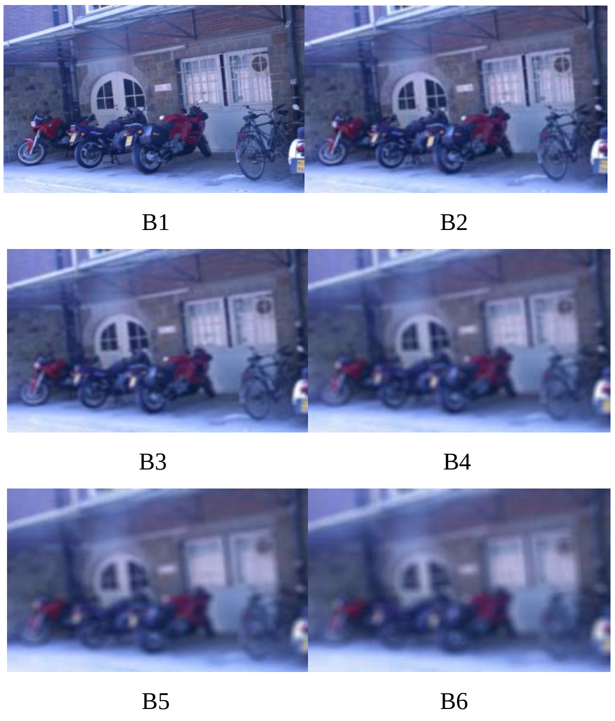
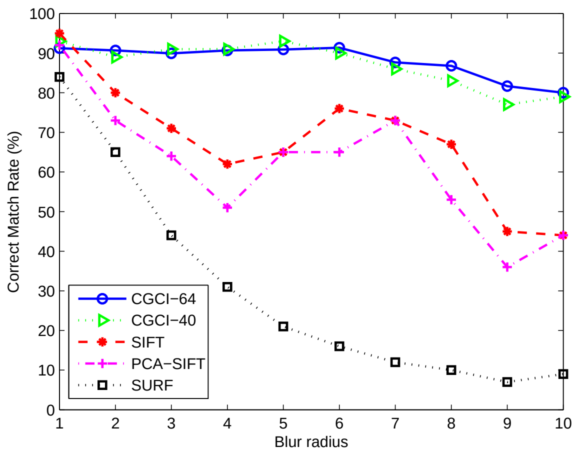
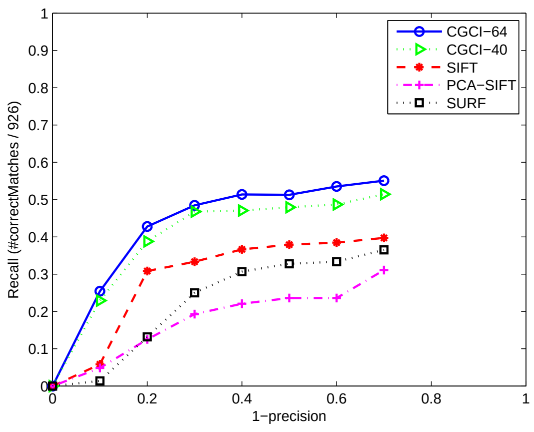


Fig.5. Images used for evaluation under image blurring.



(a) Correct Match Rate



(b) Recall versus 1-precision

Fig.6. Evaluation under blur change.

Illumination Changes. The third set of experiments is conducted on a set of Leuven images to evaluate the performance of our CGCI-SIFT against illumination invariance. The set of images are shown in Fig.7. Please note that the image C3 is the original image [17]. We use the original image C3 to produce 10 different illumination images by decreasing and increasing brightness intensity (from -150 to 150, refer to the horizontal axis in Fig.8.(a)). The brightness intensity represents a degree of illumination. Fig.7. shows the original image C3 and five produced images (C1-C2, and C4-C5). Note that we order the images according their brightness. Images C1 and C2 are darker than the original one C3. The images C4-C5 are brighter than the original one. The experimental results are shown in Fig.8.

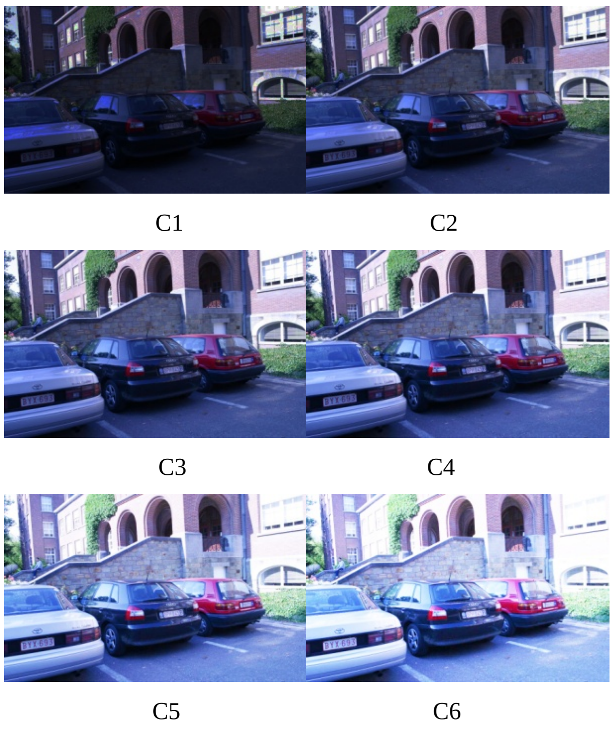
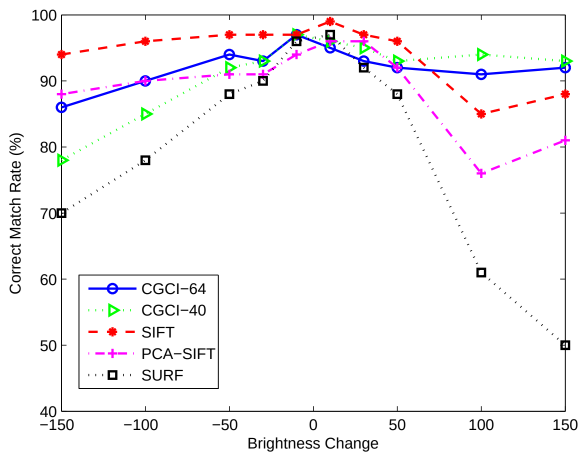


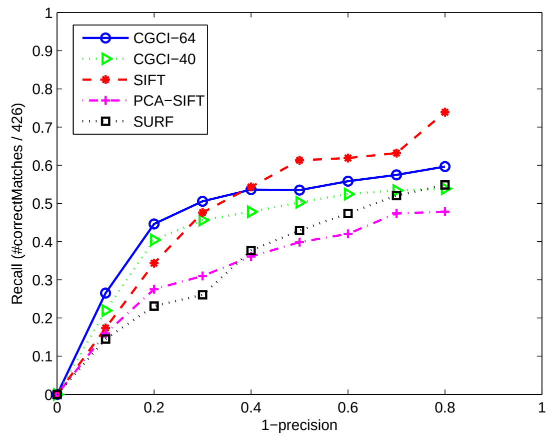
Fig.7. Images used for evaluation under Illumination changes.

Fig.8.(a). shows the matching correct rate of each algorithm between the original image C3 and the 10 generated images respectively. From Fig.8.(a)., we can see that our CGCI-SIFT performs consistently well under different brightness intensities. When the brightness intensity is very high (greater than 100 in the horizontal axis in Fig.8.(a).), our CGCI-SIFT performs the best, although it performs a little worse than SIFT (the best one among all algorithms) when the brightness intensity is negative. From Fig.8.(a)., we can also see that PCA-SIFT performs well, and SURF performs the worst.

Fig.8.(b). shows the specific experimental results for matching the original image C3 and the brightest image C6 among the images shown in Fig.7. First, we can see that the recall of every algorithm increase with the increment of 1-precision, as expected. Among these algorithms, the recall of our CGCI-SIFT increases quickly when 1-precision is less than 0.4. Both versions of our CGCI-SIFT have better performance when the value 1-precision is less than 0.4. When the value 1-precision is greater than 0.4, SIFT performs the best, followed by CGCI-64 and CGCI-40. Both CGCI-64 and CGCI-40 perform better than SURF and PCA-SIFT. Between SURF and PCA-SIFT, PCA-SIFT performs better when 1-precision is smaller than 0.4, while it performs worse when 1-precision is greater.



(a) Correct Match Rate



(b) Recall versus 1-precision

Fig.8. Evaluation under illumination changes.

JPEG Compression. JPEG compression is widely used to reduce the size of images in real-time applications with specific purposes, such as network traffic reduction. Matching descriptors with JPEG images indicate whether the descriptors can still represent image regions under compressions.

The forth set of experiments is conducted on a set of house images to evaluate the performance of our CGCI-SIFT against JPEG invariance. The set of images are shown in Fig.9. Please note that the image D1 is the original image [6]. We use the original image D1 to produce 5 different JPEG compression images by increasing the compression percentage (from 60 to 100, refer to the horizontal axis in Fig.10.(a).). The experimental results are shown in Fig.10.

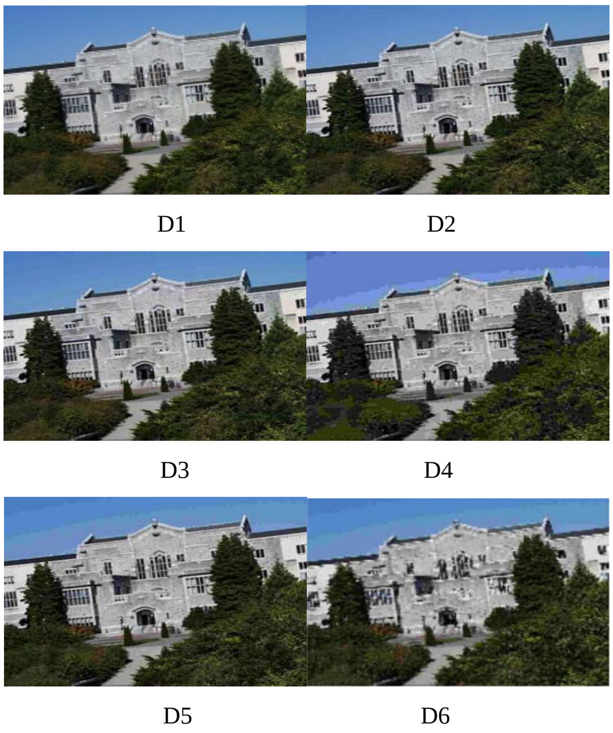
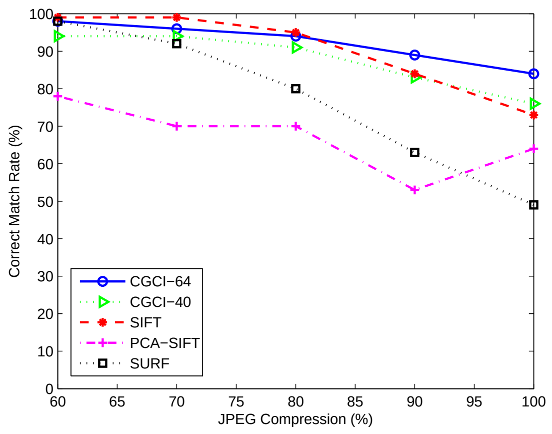
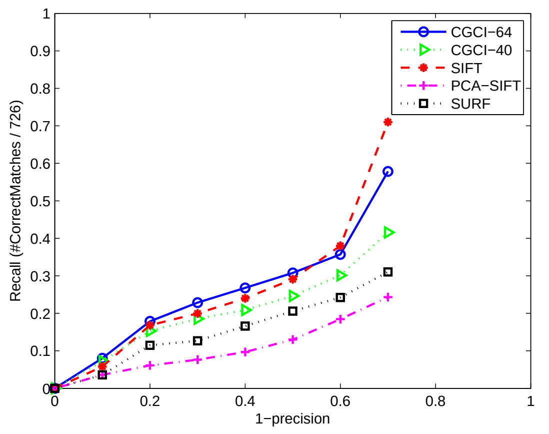


Fig.9. Images used for evaluation under JPEG compression.



(a) Correct Match Rate



(b) Recall versus 1-precision

Fig.10. Evaluation under JPEG Compression.

Fig.10.(a). shows the matching correct rate of each algorithm between the original image D1 and the five generated images respectively. From Fig.10.(a)., we can see that our CGCI-SIFT performs well under different JPEG compression. When the compression rate is greater than 80 in the horizontal axis in Fig.10.(a).), our CGCI-64 performs the best, although it performs a little worse than SIFT (the best one among all algorithms) when the compression rate is less than 80. From Fig.10.(a)., we can also see that our CGCI-40 performs better than PCA-SIFT and SURF. Between PCA-SIFT and SURF, PCA-SIFT performs better.

Fig.10.(b). shows the specific experimental results for matching the image D1 with D6 for our CGCI-SIFT (CGCI-64 and CGCI-40), SIFT and its variants (PCA-SIFT and SURF). First, we can see that the recall of every algorithm increase with the increment of 1-precision, as expected. Among these algorithms, the recall of our CGCI-64 increases quickly when 1-precision is smaller than 0.6, although SIFT performs better after that. Our CGCI-40 performs the best among the rest algorithms (PCA-SIFT and SURF), although it performs worse than CGCI-64 and SIFT. Between PCA-SIFT and SURF, SURF performs better.

Affine Transformation. The fifth set of experiments is conducted on the set of Graffiti images [17] to evaluate the performance of our CGCI-SIFT against affine invariance. This set has six images in total, shown in Fig.11. Each image has a different viewpoint (from 20 to 60 degrees, refer to the horizontal axis in Fig.12.(a).). The experimental results are shown in Fig.12.

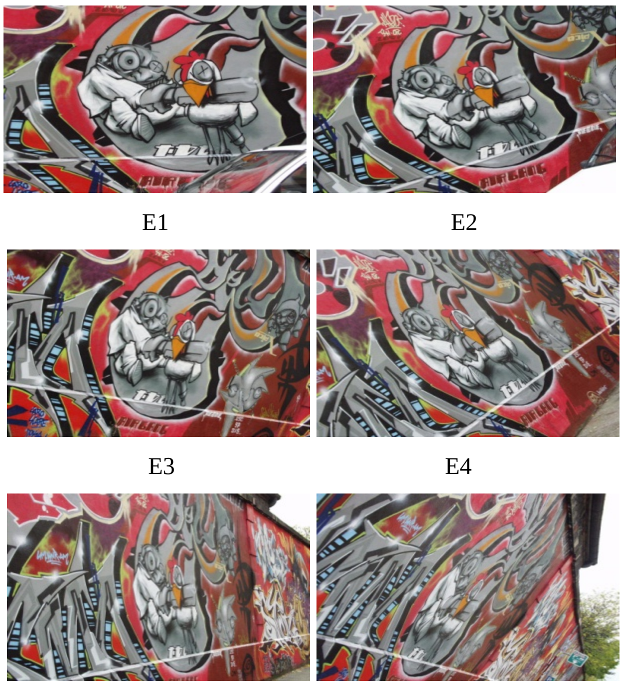
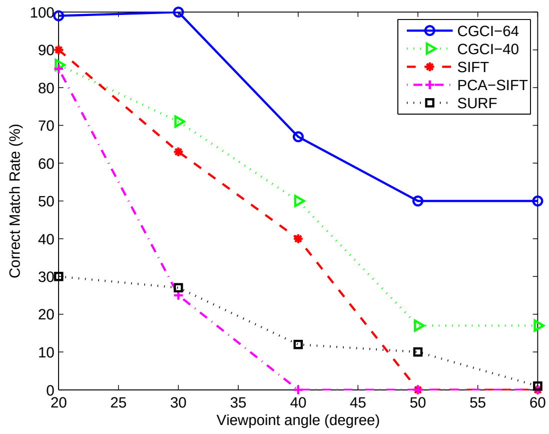
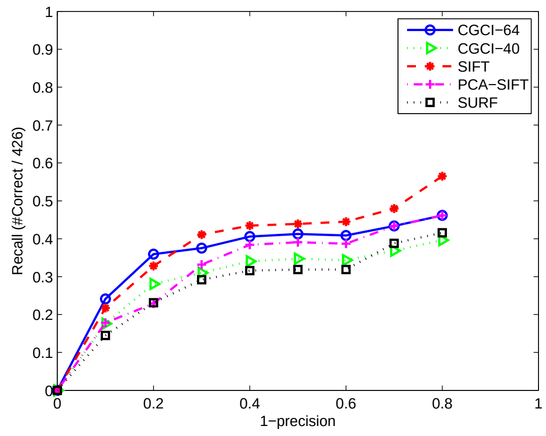


Fig.11. Images used for evaluation under viewpoint changes.

Fig.12.(a). shows the matching correct rate of each algorithm between the image E1 and the others (E2-E6) respectively. Note that we also compare the performance of the two versions of CGCI-SIFT (CGCI-64 and CGCI-40). From Fig.12.(a)., we can also see that the correct matching rate of every algorithm decreases when the viewpoint angle increases. Fig.12.(a). also clearly shows the relationships among the compared algorithms. From Fig.12.(a)., we can see that both versions of our CGCI-SIFT (CGCI-64 and CGCI-40) perform significantly better than all others (SIFT, PCA-SIFT, and SURF) under different angles of viewpoints. Between two versions themselves, CGCI-64 always performs much better than CGCI-40 under different viewpoints. It also maintains a very high matching correct rate. Among the rest three algorithms, SIFT performs better than PCA-SIFT and SURF. PCA-SIFT performs better than SURF when the angle of the viewpoint is smaller than 30 degrees. Otherwise, SURF performs slightly better than PCA-SIFT.



(a) Correct Match Rate



(b) Recall

Fig.12. Evaluation under viewpoint changes.

Fig.12.(b). shows the specific experimental results for matching the image E1 with E6 for our CGCI-SIFT (CGCI-64 and CGCI-40), SIFT and its variants (PCA-SIFT and SURF). First, we can see that the recall of every algorithm increase with the increment of 1-precision, as expected. Among these algorithms, the recall of our CGCI-64 increases quickly when 1-precision is small, although SIFT performs better when 1-precision is greater or equal than 0.3. Overall, SIFT and our CGCI-64 perform better than the rest algorithms (CGCI-40, PCA-SIFT, and SURF). Among the three algorithms, PCA-SIFT performs the best, followed by our CGCI-40. SURF performs the worst.

4.4. *Time Consumption Evaluation*

Many real-world applications of image matching are real-time. It is import to investigate the time consumption of each algorithm. In this section, we investigate the time consumption of our CGCI-SIFT, by comparing with SIFT and its two variants (PCA-SIFT and SURF). We have conducted experiments to investigate the time consumption of each algorithm under different situations (i.e., scale and rotation change, blur change, illumination change, JPEG compression, and affine change), on the whole data sets (six images for scale and rotation change, 11 images for image blur change and illumination change respectively, six images for JPEG impression and affine change respectively, 40 images in total) [17].

Table 1. shows the time consumption of each algorithm running on the 40 images. In order to see the details of the time consumed by each algorithm, we also show feature extraction time and matching time separately in the table. The feature extraction time is the total time of interest point selection and descriptor construction for all the 40 images. The matching time is the total time required to find the corresponding pairs between the image A1 and other five images in its group (A2-A6) for scale and rotation invariance, between the original image B1 and other 10 generated images for blur invariance, between the original image C3 and other 10 generated images for illumination invariance, between the image D1 and other five images in its group (D2-D6) for JPEG impression, and between the image E1 and other five images in its group (E2-E6). We also provide the summation of the time consumption of stages in the last column of Table 1.

As we can see from Table 1., the running time of CGCI-SIFT is shorter than SIFT. For PCA-SIFT, the matching time is the shortest, however, mapping the 3024-dimensional vector to the 36-dimensional vector still requires much time. Therefore, the descriptor construction of PCA-SIFT is the most time consuming. The SURF has more advantages on detection and description parts, but the detected points of SURF are more than others. Therefore, the matching time is longer than CGCI-SIFT. The descriptor generating time of CGCI-SIFT is much less than that of SIFT because only a simple operation is required to construct CGCI-SIFT. In contrast, SIFT needs to compute the magnitudes and orientations of all the pixels in a local region. The matching time of CGCI-SIFT is also shorter because the dimensions of CGCI-SIFT are smaller than those of SIFT.

Table.1. Total computation time for each algorithm on all 40 images (in Seconds).

Name	Feature extraction(s)	Matching (s)	Total(s)
CGCI-64	1.375	1.080	2.458
CGCI-40	1.187	0.854	2.042
SIFT	2.132	1.846	3.913
PCA-SIFT	3.141	0.729	3.875
SURF	1.390	1.187	2.580

5. Conclusion

This paper proposed a novel invariant local descriptor CGCI-SIFT, an extension of the standard local descriptor SIFT. Considering different effects of different sub-regions inside the local region of an interest point, CGCI-SIFT divides the local interest region around the interest point into two main sub-regions: an inner region and a peripheral region. Then, it makes use of two known methods to construct its descriptor. It uses the gradient histogram information for the inner region, and the contrast intensity information for the peripheral region, instead of only storing the gradient orientations of all pixels in a local region (the SIFT approach). Besides, in contrast to classical approaches like SIFT (PCA-SIFT and SURF) that exploit the rectangular grid as division systems, CGCI-SIFT applies a log-polar coordinate system to divide the local region, which is more sensitive to the pixels that are near to the interest point than those farther away.

Our experimental results demonstrate that the proposed descriptor CGCI-SIFT performs better than SIFT and its variants PCA-SIFT and SURF with various optical and geometric transformations, such as scale and rotation change, blur change, and viewpoint change. It is competitive under illumination change and JPEG compression. Most importantly, it also has better matching efficiency than SIFT and its variants PCA-SIFT and SURF. Because of its high matching accuracy and efficient computation, CGCI-SIFT can to be used in a variety of real-time applications.

In the future, we will continue to improve the detector of CGCI-SIFT. At the same time, we will transform the gray-value-based CGCI-SIFT to a color-based version, so that more discriminative descriptors can be applied to color images.

Acknowledgment

This research was partially supported by the Natural Science Foundation of China under grant No. 61003054, 61170020, and 61170124, the Program for Postgraduates Research Innovation in Jiangsu Province in 2011 under grant No. CXLX11_0072, the Beforehand Research Foundation of Soochow University, and the National Science Foundation (IIS-1115417).

References

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

[2] Strecha, Ch., Bronstein, A.M. (2012). LDAHash: Improved matching with smaller descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34 (1), 66-78.

[3] Senchenko, E.S., Chugui, Yu.V. (2011). Shadow inspection of 3D objects in partially coherent light. *Measurement Science Review*, 11 (4), 104-107.

[4] Jin Guofeng, Zhang Wei, Yang Zhengwei et al. (2012). Image segmentation of thermal waving inspection based on particle swarm optimization fuzzy clustering algorithm. *Measurement Science Review*, 12 (6), 296-301.

[5] Harris, Ch., Stephens, M. (1988). A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, 147-151.

[6] Mikolajczyk, K., Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27 (10), 1615-1630.

[7] Bay, H., Ess, A., Tuytelaars, T., Gool, L.V. (2008). SURF: Speeded up robust features. *Computer Vision and Image Understanding*, 10 (3), 346-359.

[8] Yan Ke, Sukthankar, R. (2004). PCA-SIFT: A more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition (CVPR 2004)*, 27 June- 2 July 2004. IEEE, Vol. 2, 506-513.

[9] Mortensen, E.N., Deng Hongli, Shapiro, L. (2005). A SIFT descriptor with global context. In *Computer Vision and Pattern Recognition (CVPR 2005)*, 20-25 June 2005. IEEE, Vol. 1, 184-190.

[10] Tola, E., Lepetit, V., Fua, P. (2010). Daisy: An efficient dense descriptor applied to wide baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32 (5), 815-830.

[11] Yang Duanduan, Sluzek, A. (2010). A low-dimensional local descriptor incorporating TPS warping for image matching. *Image & Vision Computing*, 28 (8), 1184-1195.

[12] Tuytelaars, T., Mikolajczyk, K. (2008). Local invariant feature detectors: A survey. *Computer Graphics and Vision*, 3 (3), 177-280.

[13] Rabbani, H. (2011). Statistical modeling of low SNR magnetic resonance images in wavelet domain using laplacian prior and two-sided rayleigh noise for visual quality improvement. *Measurement Science Review*, 11 (4), 125-130.

[14] Xianqing Lei, Chunyang Zhang, Yujun Xue, Jishun Li. (2011). Roundness error evaluation algorithm based on polar coordinate transform. *Measurement*, 44 (2), 345-350.

[15] Bai Cong, Kpalma Kidiyo, Ronsin Joseph. (2011). Analysis of histogram descriptor for image retrieval in DCT domain. In *Intelligent Interactive Multimedia Systems and Services*. Springer, Vol. 11, 227-235.

[16] Huang, C.R., Chen, C.R., Chung, P.C. (2008). Contrast context histogram an efcient discriminating local descriptor for object recognition and image matching. *Pattern Recognition*, 41 (10), 3071-3077.

[17] Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A. (2005). A comparison of affine region detectors. *International Journal of Computer Vision*, 65 (1/2), 43-72.

Received December 7, 2012.
Accepted June 10, 2013.