Effective and Efficient Image Copy Detection Based on GPU

Hongtao Xie[1,2], Ke Gao[1], Yongdong Zhang[1], Jintao Li[1], Yizhi Liu[1], and Huamin Ren[1]

[1] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190, China
[2] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, 100093, China
{xiehongtao,kegao,zhyd,jtl,liuyizhi,renhuamin}@ict.ac.cn

Abstract. To improve the accuracy and efficiency of image copy detection, a novel system is proposed based on Graphics Processing Units (GPU). We combine two complementary local features, Harris-Laplace and SURF, to provide a compact representation of an image. By using complementary features, the image is better covered and the detection accuracy becomes less dependent on the actual image content. Moreover, ordinal measure (OM) is applied as semilocal spatial coherent verification. To improve time performance, the process of local features generation and OM calculating are implemented on the GPU through NVIDIA CUDA. Experiments show that our system achieves a 15% precision improvement over the baseline Hamming embedding approach. Compared to the CPU-based method, the GPU realization reaches up to a 30-40x speedup, having real-time performance.

Keywords: image copy detection, CUDA, GPU, local feature, semilocal spatial coherent verification.

1 Introduction

The goal of image copy detection, given a query copy image, is to locate its original image in the database. The copy image is obtained by editing the original image through photometric and geometric changes. It is useful in many applications, such as copyright protection and redundant image filtering.

State-of-the-art image copy detection systems [1, 2, 3, 4] are based on a bag-of-features (BOF). BOF image retrieval systems first extract a set of local descriptors for each image, such as the popular SIFT descriptor [5]. Combined with effective region detectors [6, 7], these descriptors are invariant to local deformations [8]. Then, the detection systems quantize the descriptors into visual words and apply textual indexing and retrieval methods. The commonly used quantizer is k-means clustering. By adopting an inverted file index of visual words the retrieval systems avoid storing and comparing high-dimensional descriptors sequentially.

While critical for scalability, quantization has two major shortcomings. First, quantization reduces the discriminative capacity of local descriptors, since different descriptors quantized to the same visual word are considered to match with each other. Second, it is sensitive to transformations. The slight modifications to an image patch

K.N. Kutulakos (Ed.): ECCV 2010 Workshops, Part II, LNCS 6554, pp. 338–349, 2012.

can lead to its descriptor being quantized to different visual words. Soft-quantization [4] has been proposed to solve these two problems by quantizing a descriptor to several neighboring visual words, but it increases the index file size and still ignores the spatial information of feature points. Global spatial coherent verification [1, 2, 3] has been proposed as post-processing to reject mismatches, but it is computationally expensive. Besides, the extracting of local regions and descriptors is so time-consuming that most of the existing systems only apply one kind of local feature [1, 3, 4]. As different local features contain different characteristics of an image, adopting one kind of local feature cannot represent an image comprehensively [8].

In the past few years, the progress of GPU is tremendous. The computational capability of GPU today is much higher than that of the CPU. Due to its powerful computing capability, the GPU nowadays serves not only for graphics display, but also for general-purpose computation [9], such as molecular dynamics and image processing. To promote the use of GPU in the field of parallel computing, NVIDIA announced a powerful GPU architecture called "Compute Unified Device Architecture" (CUDA) [6]. CUDA provides two main modifications to effectively improve the programmability of GPU: unified shaders and shared memory. CUDA is basically a single instruction and multiple data architecture and can let programmers efficiently map a computing problem onto the GPU [11, 12, 13].

In this paper we propose a novel scheme which combines two local features, Harris-Laplace (with SIFT descriptor) [5, 6] and SURF [7], to design an effective and efficient image copy detection system. These two local features are complementary to each other and can provide highly compact representation of an image. We also employ OM [14, 15] to represent the spatial configuration around the interest point, supplying semilocal spatial coherent verification. OM is easy to calculate and has great distinguishability. Furthermore, the processes of interest point extraction, descriptor generation and OM computing are all accomplished on GPU, which improve the time performance significantly. Experiments show that our scheme achieves a 15% improvement over the baseline approach [1] and has real- time performance.

The paper is organized as follows. Section 2 describes the new image indexing strategy. Section 3 presents the details of GPU implementation. Finally, section 4 shows the experiment results and section 5 concludes the paper.

2 Image Indexing Strategy

In this section, we propose a novel image copy detection system. Instead of using a single local feature, we make use of two local features, which represent different parts of an image, to increase the detection accuracy of our system. Meanwhile, to avoid the complex global spatial coherent verification, we adopt OM [14] as semilocal spatial coherent verification. Then an effective image copy detection system is proposed.

2.1 Combination of Local Features

Local features have been widely used in image copy detection and other applications [1 - 8]. But the existing systems [1 - 4] usually adopt only one kind of local feature.

As a local feature just represents partial information of an image, such as corner, blob and region, it is not representative to distinguish an image in a large corpus of images [8]. Among all the local features, we pay special attention to Harris-Laplace [6] and SURF [7].

Fig. 1. Left: Interest points detected by Harris-Laplace. Right: Interest points detected by SURF. The red circles represent the detected features.

The Harris-Laplace detector [6] is based on the second moment matrix. The second moment matrix is also called the auto-correlation matrix, which is often used for feature detection and describing local image structures. This matrix describes the gradient distribution in a local neighborhood of a point.

$$M = \sigma_D^2 g(\sigma_I) * \begin{bmatrix} I_x^2(X,\sigma_D) & I_x I_y(X,\sigma_D) \\ I_x I_y(X,\sigma_D) & I_y^2(X,\sigma_D) \end{bmatrix} . \tag{1}$$

Where σ_I is the integration scale, σ_D is the differentiation scale; I_x and I_y are the derivatives computed in the *x* and *y* direction. The eigenvalues of matrix *M* represent two principal signal changes in the neighborhood of point *X*. This property enables the extraction of points, for which both curvatures are significant, that is the signal change is significant in the orthogonal directions i.e. corners, junctions etc [6], as the left image of Fig.1 shows.

The SURF detector is based on Hessian matrix [7], which can also be applied to describe the properties of local image structures. The Hessian matrix of an image is built with second order derivatives.

$$H(X,\sigma_D) = \begin{bmatrix} I_{xx}(X,\sigma_D) & I_{xy}(X,\sigma_D) \\ I_{xy}(X,\sigma_D) & I_{yy}(X,\sigma_D) \end{bmatrix} . \tag{2}$$

Where σ_D is the differentiation scale; I_{xx}, I_{yy} and I_{xy} are the second order derivatives. These derivatives encode the shape information by providing the description of how the normal to an isosurface changes, that is the signal change is conspicuous in all the directions around the point *X*. Based on this property, blob-like structures can be found in the image [7], as the right image of Fig.1 shows.

The theoretical analysis of Harris-Laplace and SURF shows that SURF is in a sense complementary to Harris-Laplace [8]. As an example shown in Fig.1, for an image of sunflower field, Harris-Laplace [6] detects "corner" like structures and the detected points are near object boundaries. For the same image, SURF [7] detects "blob" like structures and the detected points are localized in the object plane than corners. By using them together, the image is better covered and the detection performance becomes less dependent on the actual image content. In this motivation, we combine these two detectors to realize effective image copy detection.

2.2 Semilocal Spatial Coherent Verification

To improve the discriminative power of local features, global spatial coherent verification has been introduced to image copy detection [1-5]. By estimating the affine transformation between the query image and the candidate images, it can filter out images that do not arise from valid 2D geometric transforms of the query image. Global spatial coherent verification is effective, but it has a high degree of computational complexity. Local spatial consistency from k (k = 15) nearest neighbors, a weaker but computationally more feasible geometric constraint, is proposed in [2] to filter false matches. However it is sensitive to image transformations.

So far, little attention has been paid to using the information of the interest point's spatial neighborhood to improve its distinguishability [15]. In this paper, instead of using global spatial coherent verification, we adopt OM [14] to represent the semilocal spatial relation of the neighborhood around the detected interest point, providing semilocal spatial coherent verification. As shown in Fig.2, the red dot is the interest point p; suppose the characteristic scale of p is σ, the side length of the local region and the spatial neighborhood are $k_1\sigma$ and $k_2\sigma$ respectively. In experiment, k_1 is 10 and k_2 equals to 20. We extract the descriptor of p from its local region and the OM of p from the spatial neighborhood.

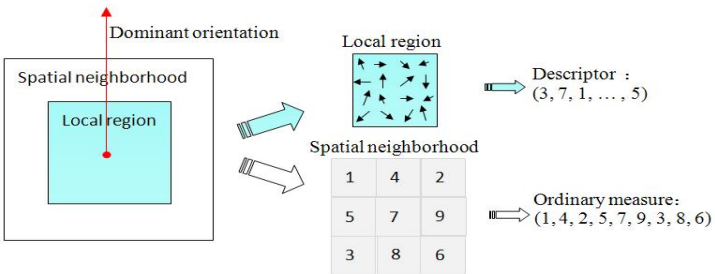


Fig. 2. Feature generation areas and corresponding features

Let $om_x = (x_1, x_2, \ldots, x_9)$ and $om_y = (y_1, y_2, \ldots, y_9)$ are the two OMs of interest points *X* and *Y*, the similarity of om_x and om_y is defined by:

$$S(om_x, om_y) = \sum_{i=1}^{9} d(x_i, y_i), \tag{3}$$

where $d(\bullet)$ is the L_1 distance.

2.3 Image Copy Detection Strategy

Our image copy detection system is illustrated in Fig.3. To improve the detection accuracy, we apply approximate nearest neighbors (ANN) indexing structure [16] for feature storage and search.

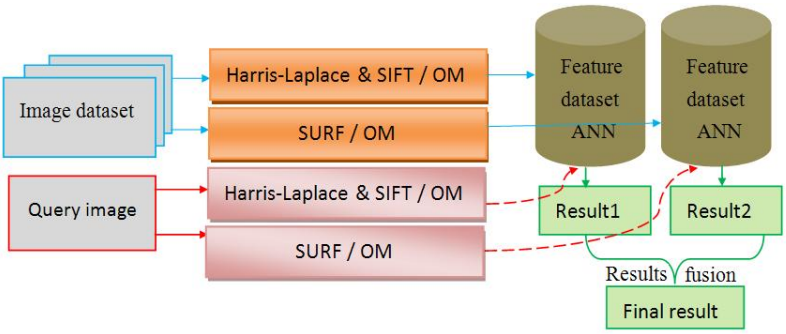


Fig. 3. The image copy detection system

The system consists of the following three steps:

1. Construct feature datasets: For each image in the image database we extract its local features, Harris-Laplace (with SIFT descriptor) [5, 6] and SURF [7]. The OMs are computed for all the interest points. Then the ANN algorithm [16] builds feature datasets to store and index the features, one dataset for a kind of feature. The OMs are stored in the document.
2. Query and filter: As we apply two kinds of features, there are twice query processes for a query image Q. For each interest point q of Q, we use approximate k (k = 10) near neighbor search to query the corresponding feature dataset, getting the initial candidate point set S'. Then we filter out the point p in S', if $S(om_q, om_p) < 5$. The score of the candidate image P is calculated by:

$$score(P) = \frac{match(Q,P)}{min(Q,P)} \quad . \tag{4}$$

   Where $match(Q,P)$ is the number of matched points between image Q and P, and $min(Q,P)$ is the minimum point number of the two images. So Q have two result lists and P may appear in both lists and have two scores, $score_1(P)$ and $score_2(P)$.
3. Results fusion: We fuse the result lists returned by the queries of each kind of feature. For each candidate image in "Result 1" and "Result 2", the fusion is defined by:

$$final\ score(P) = max(score_1(P)\ ,\ score_2(P)) \quad . \tag{5}$$

It means to take its maximum score. The final returned images are sorted according to their scores, and the top-ranked images are the copy images of the query.

3 Implementation on the GPU

This section presents the implementation of our image copy detection system on the GPU. Limited to the architecture of GPU [10] and the characteristics of the algorithms, we cannot transplant the whole system to the GPU. So far, we can only realize interest point extraction, descriptor generation and OM computing on the GPU, while the rest of the system is still carried out on the CPU. The CPU and GPU co-working model of our system is illustrated in Fig.4.

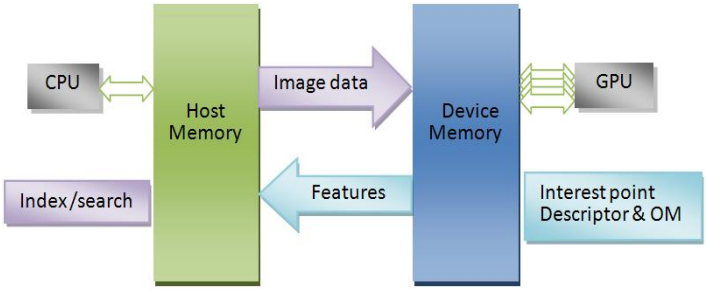


Fig. 4. The CPU and GPU co-working model

3.1 GPU- Based Harris-Laplace

Harris-Laplace algorithm only involves convolution and derivation operations, which have the inherent nature of parallelism. So, we partition the image data equally into data blocks and distribute them among the thread blocks, as shown in Fig.5. In the implementation, there are 16×16 threads in a thread block and the total number of thread block is:

$$Block\_num = \frac{image\_width}{16} \times \frac{image\_height}{16} \quad . \tag{6}$$

Each thread is responsible for the processing of a pixel, and the intermediate results are stored in the shared memory to reduce the time consuming caused by data transition. Furthermore, to speed up convolution operation, we approximate Gaussian with box filters, which is beneficial to GPU acceleration too. As box filters are constant, we put them in the constant memory when the program starts.

During the convolution calculation, the problem of "boundary cases" will be confronted. As shown in Fig.5, when calculating convolution for data blocks 1 and 6, the required data are represented by the red and blue dashed box. If we set "conditional

check" for each pixel, speedup gain will be reduced [10]. So we copy image data to texture memory, which handles the "boundary cases" automatically.

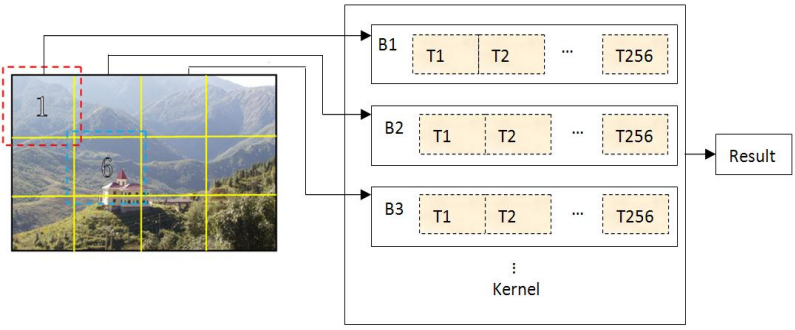


Fig. 5. Data partition and distribution

The data layout of our GPU-based Harris-Laplace follows the method of [11]. Four scales, which are four Gaussian filtered versions, are being calculated simultaneously. The scale-level parallelism can get further speedup.

3.2 GPU- Based SURF, SIFT and OM

We implement SURF and SIFT on GPU as [12] and [11] do, except for histogram computation, which can also be applied for OM calculation. As a commonly used analysis tool, histogram is quite difficult to compute efficiently on the GPU [10]. CUDA SDK takes advantages of atomic shared memory operations and designs an efficient histogram calculation method [13]. But atomic functions operating on shared memory are only available for devices of compute capability 1.2 and above.

Based on the compute capability of our device (NVIDIA GeForce 9800GTX+, 1.1), we use shared memory to calculate histogram step by step, as illustrated in Fig.6. To calculate the histogram of a 8×8 pixel block, we divide it into four 4×4 pixel blocks; then, four threads compute the four histograms of these blocks, one for each; the intermediate results are stored in the shared memory; finally, one thread combines the four histograms into one, getting the final result.

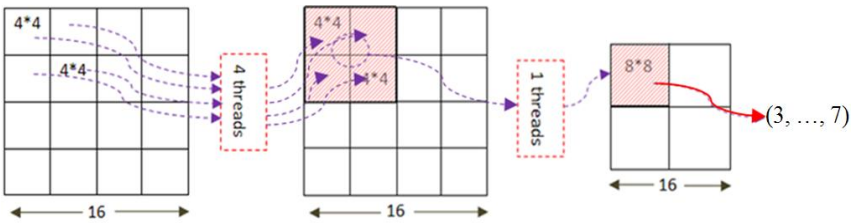


Fig. 6. Parallel histogram computation

4 Experiments

4.1 Experiment Data and Environment

We take the INRIA Copydays dataset as evaluation dataset[1].The dataset contains 157 original images. To represent typical transformations performed on images in a copy detection application, each image of the dataset has been transformed with three kinds of transformations:

- Image resizing (by a factor of 4 in dimension and 16 in total surface), followed by JPEG compression ranging from JPEG3 (low quality) to JPEG75 (high quality).
- Cropping ranging from 5% to 80% of the image surface.
- Strong transformations: print and scan, perspective effect, blur, paint, contrast change, etc.

The transformed images are illustrated in Fig.7. The goal of this dataset is to evaluate the behavior of indexing algorithms for most common image copies.

We also have 100 thousand images as "distracting images", which are crawled from Flickr[2]. The distracters include nature scenes, people, buildings and cartoons. Their size ranges from 256×364 to 1024×1024.

Fig. 7. Sample images from INRIA Copydays and corresponding transformed images

In the evaluation, we use the 157 original images as queries. Following the standard evaluation measure [1, 3], we use mean average precision (mAP) as our

[1] http://lear.inrialpes.fr/people/jegou/data.php
[2] http://www.flickr.com/

evaluation metric. For each query image we calculate its precision-recall curve, from which we obtain its average precision and then take the mean value over all queries.

The experiment environment is: Intel Core E8400 3.0GHz with 2048MB memory, NVIDIA GeForce 9800GTX+ with 512MB DRAM, Microsoft Windows XP sp2, CUDA Toolkit 2.1 and CUDA Driver (181.20).

4.2 Time Performance Analysis

Fig.8 and Fig.9 show the time used by CPU and GPU to extract Harris_Laplace, SIFT, SURF and OM for images of different size. From these tables, we observe that local feature extraction speed can get significant improvement with GPU. It only takes 67.4 ms to compute a 600×600 image, and the speedup for high resolution images is much more salient. This saves time for feature querying process and is the basis for real-time image copy detection.

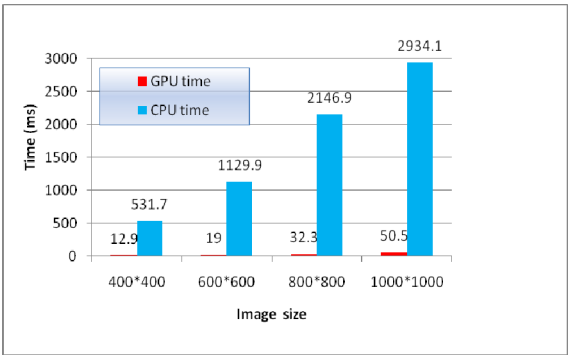


Fig. 8. Time cost of CPU and GPU to calculate SURF and OM

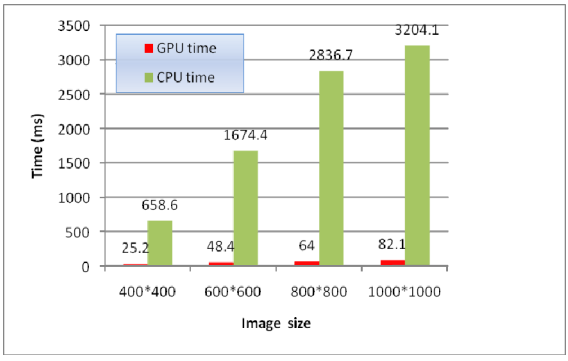


Fig. 9. Time cost of CPU and GPU to calculate Harris-Laplace, SIFT and OM

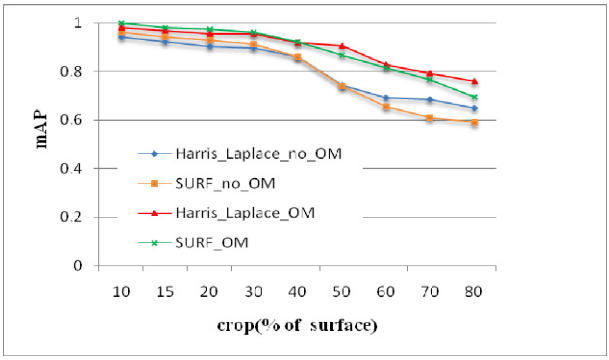
Table 1 illustrates the time used by the different parts of our system for detecting a 640×480 image. The matching process is implemented on CPU, so the time cost is identical. Compared to the CPU-based method, the GPU realization achieves up to a 30-40x speedup. With the powerful parallel computing capability of GPU, our system has real-time performance.

Table 1. Time (ms) used by the different parts of our system for detecting a 640×480 image

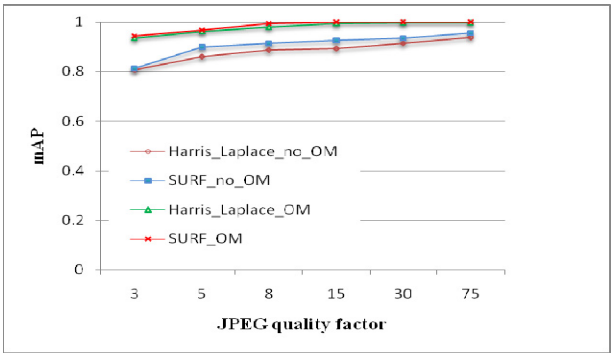
	Feature extraction	Matching	Total	Speedup
GPU	52.4	17.5	69.9	37.4
CPU	2602.5		2620	

4.3 Accuracy Performance Analysis

To prove the effectiveness of semilocal spatial coherent verification played by OM, we do queries with and without OM and compare the corresponding mAP values, as illustrated in Fig.10 (a) and (b). "Harris_Laplace_no_OM" means we only use Harris_Laplace, without applying OM. "Harris_Laplace_OM" means we not only use Harris_Laplace, but also adopt OM for semilocal spatial coherent verification. The same hold for "SURF_no_OM" and "SURF_OM".



(a) CROP



(b) SCALE (1/16) +JPEG

Fig. 10. The performance verification of OM as spatial coherent verification

From Fig.10, we can see that using OM as semilocal spatial coherent verification can obviously improve detection accuracy. The mAP values are increased by 10 – 20% for CROP and SCALE + JPEG attacks; the effect is more obvious when the attacks getting stronger. This is because that using the spatial neighborhood information can improve the distinguish ability of local features, and reduce mismatches.

To evaluate the detection accuracy of our system, we take Hamming embedding (HE) [1] as the "baseline" approach, which is one of the best methods in state-of-the-art [1, 3]. The vocabulary has 2000 visual words, which gives best performance when we experiment with different sizes. As our system has excellent performance in dealing with JPEG attack, we only show the comparison of handling cropping attack, as illustrated in Fig.11. "FUSE" is fusing the result lists returned by "Harris_Laplace_OM" and "SURF_OM", as described in 2.3.

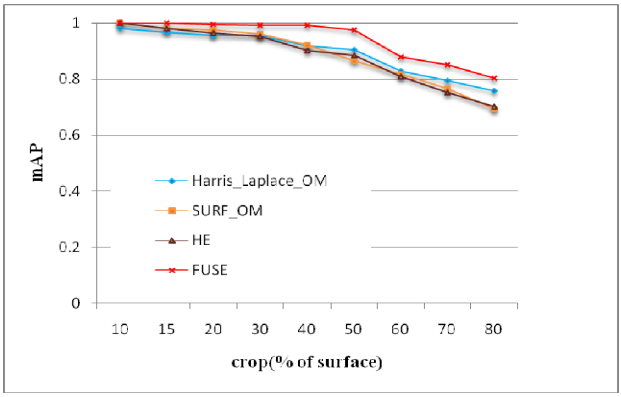


Fig. 11. The performance verification of our system

From Fig.11, we can draw two major observations. First, result fusion improves the mAP value remarkably, as can be seen by comparing the results for "FUSE" to "Harris_Laplace_OM" and "SURF_OM". This shows the benefit of using complementary features. As complementary local features has a much more compact representation of an image, and can deal with images of different content. Second, compared to HE [1], the mAP of "FUSE" gets significant improvement. When cropping rate is 50%, the mAP is increased from 0.83 to 0.96, about 15% improvement. It demonstrates the effectiveness of our system.

5 Conclusions

We have introduced an effective and efficient image copy detection system based on GPU. We combine two complementary local features together and use OM as semilocal spatial coherent verification. To speed up detection, the process of local features generation and OM computing are implemented on the GPU. The combination of complementary local features can represent the information of an image

comprehensively. OM makes the features more discriminative and reduces mismatches. Experiments show that our system outperforms the current state-of-the-art and has excellent time performance.

Features combination and using GPU are two general and powerful frameworks. Our future work is to combine features in an advanced way and port the other parts of copy detection, such as indexing and matching on the GPU.

Acknowledgements. This work is supported by the National Basic Research Program of China (973 Program, 2007CB311100); National Nature Science Foundation of China (60873165 60802028); Beijing New Star Project on Science & Technology (2007B071); Co-building Program of Beijing Municipal Education Commission.

References

1. Jegou, H., Douze, M., Schmid, C.: Hamming Embedding and Weak Geometric Consistency for Large Scale Image Search. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part I. LNCS, vol. 5302, pp. 304–317. Springer, Heidelberg (2008)
2. Sivic, J.: Video Google: A text retrieval approach to object matching in videos. In: ICCV (2003)
3. Perd'och, M., Chum, O., Matas, J.: Efficient Representation of Local Geometry for Large Scale Object Retrieval. In: CVPR (2009)
4. Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Lost in quantization: Improving particular object retrieval in large scale image databases. In: CVPR (2008)
5. Lowe, D.: Distinctive Image Features from Scale In-variant Keypoints. IJCV, 91–110 (2004)
6. Mikolajczyk, K., Schmid, C.: Scale and affine invariant interest point detectors. IJCV, 63–86 (2004)
7. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: Speeded Up Robust Features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006)
8. Tuytelaars, T., Mikolajczyk, K.: A survey on local invariant features. In: FTCGV, pp. 177–280 (2008)
9. Owens, J.D., Houston, M.: GPU computing. Proceedings of the IEEE 96(5) (May 2008)
10. NVIDIA. NVIDIA CUDA Programming Guide version2.0, `http://www.nvidia.com/object/cuda_get.html`
11. Cornelis, N., Van Gool, L.: Fast scale invariant feature detection and matching on programmable graphics hardware. In: CVPR (2008)
12. Wu, C.C.: SiftGPU: A GPU Implementation of Scale Invariant Feature Transform, `http://www.cs.unc.edu/`
13. Podlozhnyuk, V.: Histogram calculation in CUDA, `http://www.nvidia.com/object/cuda_get.html`
14. Bhat, D.N., Nayar, S.K.: Ordinal measures for image correspondence. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(4), 415–423 (1998)
15. Zhang, Y.D.: Content-Based Copy Detection by MCG-ICT-CAS. In: TRECVID Workshop (2008)
16. Lin, K.I.: The ANN-Tree: An Index for Efficient Approximate Nearest-Neighbor Search. In: CDSAA (2001)