Large Scale Specific Object Recognition by Using GIFTS Image Feature

Hiroki Nakano[1(✉)], Yumi Mori[2], Chiaki Morita[1], and Shingo Nagai[1]

[1] Tokyo Laboratory, IBM Japan Ltd., Tokyo, Japan
{hnakano,cmorita,snagai}@jp.ibm.com
[2] Yokohama National University, Yokohama, Japan
moriyumi@ynu.ac.jp

Abstract. We propose GIFTS (Goods Image Features for Tree Search) which uses image local features for large-scale object recognition. Each GIFTS is a kind of keypoint feature. The feature vector consists of intensity deltas for 128 selected pixel pairs around the keypoint. By generating a KD-Tree from the GIFTS feature vectors of the training images and using the KD-Tree to search for nearest neighbor feature vectors of a query image, query times are on the order of log *N* for specific object recognition. We used the proposed method for book cover queries with 100,000 training images, had recognition accuracy over 99% with query times within one second.

Keywords: Object recognition · Local feature · KD-Tree · Augmented reality

1 Introduction

As performance and object recognition algorithms improve in computers, identification of various types of goods is possible using pattern recognition techniques. In bookstores, there is a strong desire to improve the efficiency of store inventory management by overlaying useful information (e.g. sales rankings and numbers of copies in stock) directly on book cover images by using tablet computers and without using barcode readers.

The authors propose a large-scale book cover image query system by using new image local features named GIFTS (Goods Image Features for Tree Search) to achieve query accuracy over 99% and response times within one second with more than 100,000 training images of books. By generating a KD-Tree (k-dimensional tree) from the GIFTS feature vectors of the training images of the books, searching for similar feature vectors in the KD-Tree, we confirmed that order of log *N* query time can be achieved even for huge numbers of feature vectors.

The paper is organized as follows: in Section 2 we describe the GIFTS image local features. Section 3 is about the large-scale goods image query system using GIFTS with the KD-Tree. In Section 4 we present and discuss our experimental results.

1.1 Related Work

After the invention of SIFT image local features for specific object recognition by David G. Lowe [1], many image local features with high recognition accuracy were developed such as SURF with better computational time and PCA-SIFT with fewer dimensions in the feature vectors [2,3,4]. In contrast to SIFT-like image features using DoG (Difference of Gaussians), the ORB image local feature approach uses local corner characteristics, and the method drastically improved the computational times while maintaining the recognition accuracy [5]. However, the query times were still proportional to the number of training images. With more than 10,000 training images, computational times for query become impractical.

There are two query methods with query time that are not proportional to the number of training images. One is KD-Tree and the other is LSH (Locality Sensitive Hash), which uses a hash table [6,7]. The KD-Tree is suitable for the feature vectors with real number components, and LSH is suitable for the feature vectors with binary bit pattern components [8]. The proposed GIFTS feature vector has 128-dimension real number components, and so we used KD-Tree for the mapping of the GIFTS feature vectors. The average computational cost for queries with KD-Tree is on the order of Log N, where N is the number of feature vectors of the training images.

We found some published work using approximate nearest neighbor search methods for large-scale specific object recognition [9], but we couldn't find any prior work with recognition accuracy over 99% and response times under one second for 100,000 training images.

2 GIFTS Image Local Feature and KD-Tree

2.1 GIFTS Image Feature

A GIFTS image local feature has three components: keypoints, feature vectors, and distances for keypoint pairs. GIFTS uses oFAST (Oriented FAST) keypoints that are rotation- and scale-invariant local image features [5]. For scale invariance, we first prepared the images at eight scales, each separated by a scaling factor of $\sqrt{2}$. Then we classify 16 pixels on a circular ring of radius R around the center pixel. These 16 pixels are sorted into darker, similar, and brighter classes based on the differences in the gray levels between the center pixel and the surrounding 16 pixels. If the gray level difference is more than an experimentally determined threshold (i.e. 25), then the corresponding pixel is classified in the darker or brighter class. If 9 consecutive darker pixels or brighter pixels are found among the 16 pixels, then the center pixel is identified as a keypoint (see Fig. 1).

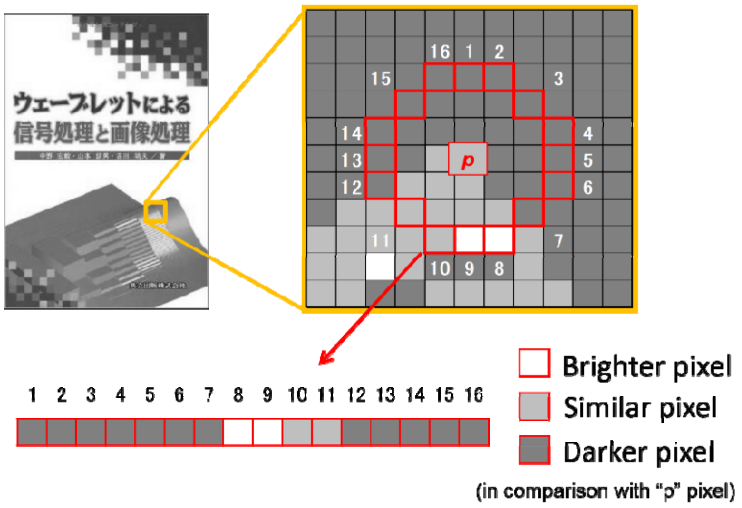

Fig. 1. GIFTS keypoints

To obtain the orientation of each keypoints, we calculate a gray level centroid for all the pixels within radius R. The offset of the coordinates of the centroid and that of the center pixel indicates the orientation of the keypoint. The intensity of the keypoint is defined as the sum of absolute values of the gray level deltas between the center pixel and the brighter or darker pixels in radius R. The intensity is used for selection of the strongest keypoints. We search the keypoints across 8 multiple resolution images for scale-invariance.

For the feature vector of each keypoint, we choose 128 pixel pairs in the rectangle of 31 ×31 pixels around the keypoint, and each component of the feature vector is the delta of the gray level of one of these pixel pairs. Fig. 2 shows the pixel pairs for generating the GIFTS feature vectors. The selection method is the greedy search algorithm described in [5].

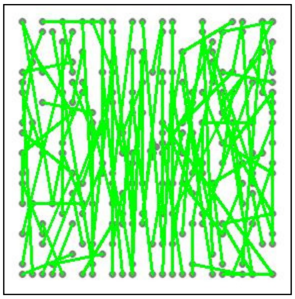

Fig. 2. 128 pixel pairs for generating feature vectors of GIFTS keypoints

Here is how we select the 128 pixel pairs with the greedy search algorithm:

Step 1:

Select 100,000 keypoints at random from the training image set. Calculate the variances of the differences of the gray levels of all of the pixel pairs in the 31×31 pixel patch around the keypoint. Select the pixel pair with the largest variance.

Step 2:

Among the remaining pixel pairs, choose another pixel pair with the next largest variance and the absolute correlation with the already selected pixel pairs is less than a threshold.

Step 3:

Repeat Step 2 until 128 pixel pairs have been selected.

By selecting pixel pairs using the test set of training images, the proper pixel pairs can be found for the object image set. In the paper [5], 256 pixel pairs were selected by using the binary differences between pairs of pixels. In our approach, 128 pixel pairs are selected by using the gray level differences of the pixel pairs. Thus we expect more accurate recognition performance with lower calculation cost compared to the approach in paper [5].

Each component of a feature vector is a 128-dimension real number, and the feature vector is invariant to in-plane rotation because we rotate the coordinates of the pixel pairs with respect to the orientation of the keypoint. For any coordinate set of 128 pixel pairs (x_i, y_i), we define the 2×128 matrix

$$\mathrm{S} = \begin{pmatrix} x_0, \cdots, x_{127} \\ y_0, \cdots, y_{127} \end{pmatrix} .$$

Using the corresponding rotation matrix R_θ, we construct a rotated version S_θ of S:

$$\mathrm{S}_\theta = \mathrm{R}_\theta \mathrm{S}.$$

The major parameter set for the keypoints and feature vector calculations we used (e.g. using surrounding 16 pixels for keypoint search and patch size of 31×31 pixels for feature vector calculation) are the same as the recommended values for the oFast approach in paper [5]. For the distance of two keypoints, we use the Euclidian distance of the two feature vectors after normalizing the magnitude of each feature vectors to the value one.

2.2 KD-Tree

The KD-Tree is the k-dimensional extension of a binary tree. By generating the tree structure from k-dimensional vectors, we can search for the nearest-neighbor vector of the query vector in the tree. If we use a KD-Tree, the average search time for a query image in the training images is of order of $M \times \log N$, where N is the number of keypoints of the training image set and *M* is the number of keypoints of the query image [6]. When the number of training images is 10,000 and the average number of keypoints for each training image is 1,000, the KD-Tree will contain 10^7 feature vectors. In the search phase, each keypoint of a query image is voted to the training image which has the nearest neighbor keypoint in the feature vector space. We take the training image with the highest rating that exceeds the predetermined threshold number as the search result. We use the FLANN library to generate and search the KD-Trees [10].

3 Large Scale Book Image Query System

In Japan, around one million distinct books are in print and on sale in bookstores. We generate several KD-Trees for the parallel processing of the divided training images. As the number of training images increase, we can add processor cores for the additional sets of training image. The design allows the target query time to be satisfied even though the number of training images increases. In our experiment, each KD-Tree handles 10,000 book cover images, and accessed by one processor core. In the query phase, we search the KD-Trees in all of the cores in parallel, and the training image with the highest score is regarded as the query result. We select the top n candidates because some of the images of book covers are quite similar to each other. Fig. 3 shows the system configuration. We assume the group of book cover images is G1, G2, ···, Gx, and each group consists of around 10,000 books in the same category from the same publisher. Thus we can select the search categories by enabling or disabling groups.

The computational steps are as follows:

(a) In the query server, the GIFTS feature vectors for each training image is calculated.
(b) KD-Trees for each training image groups is generated and stored in the memory of respective processor core.
(c) Users take photos of book covers with a tablet PC, and send them to the query server.
(d) In the query server, the GIFTS feature vectors are extracted from the query images, and matched against the feature vectors in the KD-Trees so candidate training images can be extracted.
(e) The results are sent to the tablet PC and displayed.

In this paper, the time from sending an image to the query server and receiving a query result from the server is defined as the book search time. This is from Step (c) thorough Step (e). The time between receipt of an image by the query server and return of the query result is defined as the query time, corresponding to Step (d). Fig. 4 shows an example of query results displayed on a tablet PC. The results include book names, ISBN numbers, number of copies available, and instructions for replacement in bookshelves. One of the advantages of using a tablet PC is the capability of displaying the results for several books at one time. Conventional barcode-reader-based systems don't have so much versatility.

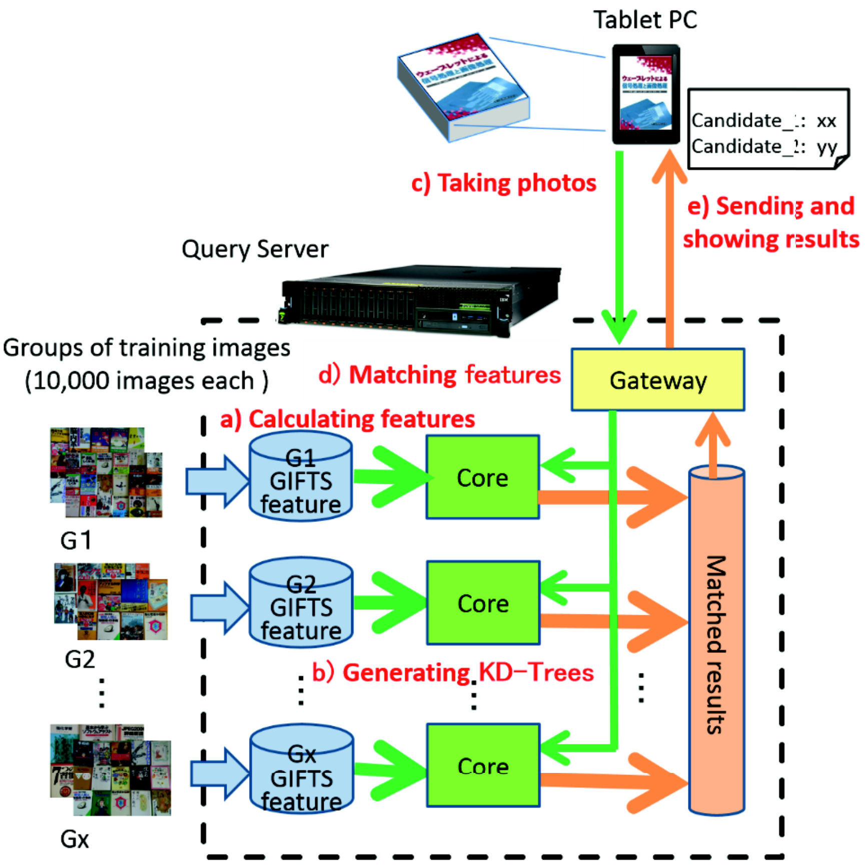

Fig. 3. System configuration

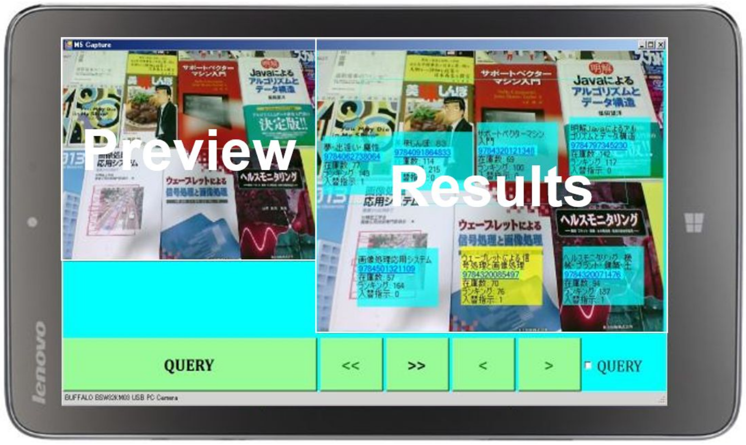

Fig. 4. An example of query results on a tablet PC

4 Experimental Results and Considerations

We measured the recognition accuracy and query times of the proposed method using GIFTS features. We gathered 100,000 book cover images for the training images. The sizes of the training images are between 120×180 to 300×450 pixels. We used an IBM Power8 server (3.72 GHz, 20 cores, 512 GB of memory, running RHEL) for the experiment. We measured the query times by assigning 10,000 book images per processor core. We prepared 1,000 query books, and each book cover has 4 images with tilt angles of 0, 15, 22.5, and 30º (See Fig. 5). Thus a total of 4,000 images for the book cover queries were prepared.

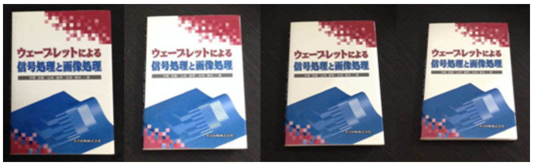

Fig. 5. Examples of query images with 4 tilt angles

We registered the 100,000 training images and prepared the 4,000 query images, and compared the recognition accuracies of GIFTS, GIFTS-2, and SIFT. The GIFTS-2 version reduced the maximum number of keypoints per image (Kmax) to 300, which was the only difference between GIFTS and GIFTS-2.

Here is our definition of percentage recognition accuracy:

$$\frac{\text{Correct queries}}{\text{Query images}} \times 100.$$

The query results consist of GIFTS (Maximum number of keypoints per image Kmax: 1,000), GIFTS-2 (Kmax: 300), and SIFT (Kmax: 1,000) are shown in Table 1 and Fig. 6. GIFTS obtained over 99% of recognition accuracy when the tilt angle between training image and query image is less than 22.5º. GIFTS obtained better recognition accuracy than SIFT, but the average query time is inferior. In general, GIFTS generates more keypoints than SIFT, so the query times are longer in proportion to the number of keypoints. With GIFTS-2 the number of total keypoints for the training images is comparable to SIFT, and this results in query times about the half of the SIFT times, while the recognition accuracy remains better as long as the tilt angle is less than 30º. We believe that SIFT is less robust than GIFTS when a query image undergoes homographic transforms, though SIFT is highly robust for in-plane rotations and scaling [11].

Fig. 7 shows the average query time per each number of training images. The maximum number of keypoints is 1,000 for both the training and query images. We used 1 core for each 10,000 training images, so the 1000,000 images used 10 cores, resulting in query times around 0.2 seconds. Fig. 7 indicates that the average query time of KD-Tree is of order $M \times \log N$.

Table 1. Comparison of GIFTS, GIFTS-2, and SIFT (10,000 training images)

	GIFTS	GIFTS-2	SIFT
Max keypoints/image (Kmax)	1,000	300	1,000
Total keypoints	7,185,731	2,674,683	2,491,024
Average query time [sec]	0.148	0.044	0.112

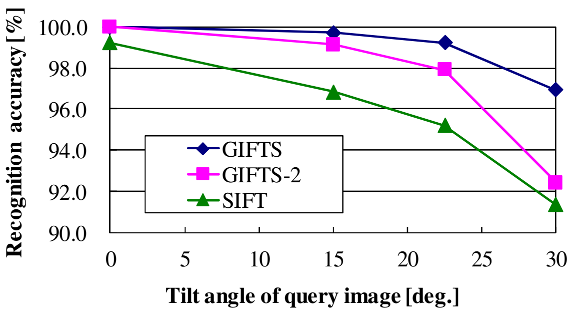

Fig. 6. Recognition accuracy against tilt angles of the query images

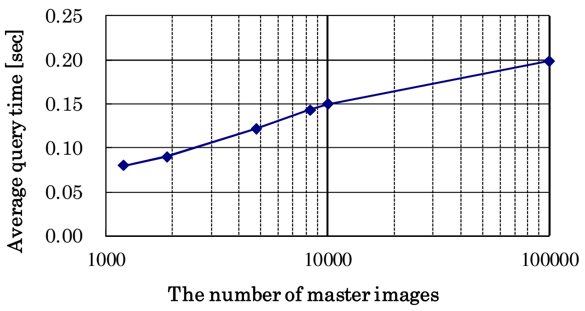

Fig. 7. Average query time by the number of training images (GIFTS)

4.1 Comparison with ORB

ORB is a well-known keypoint approach with high recognition accuracy and two orders of magnitude faster than SIFT [5]. Each feature vector of ORB is a bit pattern (256 bits), which it is suitable for LSH (Locality Sensitive Hashing). We also used the FLANN library for LSH. We measured the recognition accuracy by using 10,000 training images and 4,000 query images (Fig. 8). ORB showed over 98% recognition accuracy when the tilt angle is less than 22.5°, but the accuracy of GIFTS was superior to ORB. This may be because ORB creates its binary bit patterns for the feature vectors by binarizing the intensity deltas of pixel pairs. This means the intensity information is lost in the binary transformation. Therefore, ORB's recognition performance with similar book covers seems to be inferior to GIFTS.

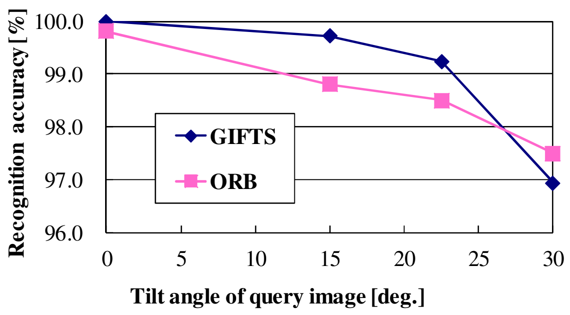

Fig. 8. Recognition accuracy against tilt angles of the query images (GIFTS vs. ORB)

5 Conclusions and Future Work

In the paper, we proposed a new image local feature named GIFTS for a large-scale book cover image query system. By combining GIFTS with KD-Trees, we could recognize images accurately and quickly. By testing the proposed system on book cover images queries with 100,000 training images, we correctly recognized more than 99% of the query images within one second. Our future work will include improvements of the performance in real-life operations in bookstores and applying the system to merchandise other than books. Also, this technology seems likely to be useful in such applications as shopping by people with weak eyesight [12].

References

1. Lowe, D.: Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision **60**(2), 91–110 (2004)
2. Ke, Y., Sukthankar, R.: PCA-SIFT: a more distinctive representation for local image descriptors. In: CVPR, vol. 2, pp. 506–513 (2004
3. Bay, H., Tuytelaars, T., Van Gool, L.: Surf: speeded up robust features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006, Part I. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006)
4. Juan, L., Gwun, O.: A Comparison of SIFTPCA-SIFT and SURF. International Journal of Image Processing **3**(4), 143–152 (2009)
5. Rublee, E., et. al.: ORB: an efficient alternative to SIFT or SURF. In: ICCV2011, pp. 2564–2571 (2011)
6. Arya, S., et al.: An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions. Journal of the ACM **45**(6), 891–923 (1998)
7. Andoni, A.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In: 47th Annual IEEE Symposium on Foundations of Computer Science, pp. 459–468 (2006)
8. Muja, M., Lowe, D.: Fast matching of binary features. In: Conference On Computer And Robot Vision (CRV 2012), pp. 404–410 (2012)

9. Kise, K., Noguchi, K., Iwamura, M.: Simple representation and approximate search of feature vectors for large-scale object recognition. In: Proceedings of BMVC 2007, vol. 1, pp. 182–191 (2007)
10. Muja, M., Lowe, D.: FLANN: Fast library for approximate nearest neighbors. http://www.cs.ubc.ca/research/flann
11. Bekel, D., Teutschy, M., Schucherty, T.: Evaluation of binary keypoint descriptors. In: ICIP 2013, pp. 3652–3656 (2013)
12. Do, A.T., Ilango, K., Ramasamy, D., Kalidasan, S., Balakrinan, V., Chang, R.T.: Effectiveness of low vision services in improving patient quality of life at Aravind Eye Hospital. Indian J. Ophthalmol **62**(12), 1125–1131 (2014)