Rotated Image Based Photomosaic Using Combination of Principal Component Hashing

Hideaki Uchiyama and Hideo Saito

Keio University, 3-14-1 Hiyoshi, Kohoku-ku 223-8522, Japan
{uchiyama,saito}@ozawa.ics.keio.ac.jp

Abstract. This paper introduces a new method of Photomosaic. In this method, we propose to use tiled images that can be rotated in a restricted range. The tiled images are selected from a database. The selection of an image is done by a hashing method based on principal component analysis of a database. After computing the principal components of the database, various kinds of hash tables based on the linear combination of the principal component are prepared beforehand. Using our hashing method, we can reduce the computation time for selecting the tiled images based on the approximated nearest neighbor searching in consideration of a distribution of data in a database. We demonstrate the effectiveness of our hashing method by using a huge number of data in high dimensional space and better looking results of our tiling in experimental results.

1 Introduction

Mosaic is one of the traditional arts in which a large image is generated by tiling small pieces of colored glass, stone and so on. The tiling ways are determined depending on a shape and a texture of the image and pieces. Recently, computer-generated mosaics are studied as a non-photorealistic rendering [1,2].

Photomosaic is an image that has packed with many smaller images called tiles by using a reference image [3]. In a Photomosaic image, pixel colors are replaced with small tiled images that approximate pixel colors in local regions in the reference image. Then small tiled images can be seen by observing in close-up, while the approximated reference image can be seen by observing from the distance. The applications of Photomosaic have already been available in [4,5].

In Photomosaic, research topics can be divided into two aspects as follows:

- To find optimized arrangement of tiled images that provides better approximation of the reference image
- To select tiled images that represent local regions in a good approximation from a database of tiled images

In the first research topic, previous tiling methods replace a pixel or local square region in a reference image with an image without overlapping and rotating [3,4,5]. As a result, some parts of the Photomosaic image are not matched

T. Wada, F. Huang, and S. Lin (Eds.): PSIVT 2009, LNCS 5414, pp. 668–679, 2009.

with the reference image because the color of the image in a database is not always matched with the color of the local region in the reference image.

In the second research topic, the selection of tiled images from a database is addressed as an approximate nearest neighbor searching problem. Previous researches described the method for evaluating color difference between a square region in a reference image and every image in a database to find the best matching image. However, the searching problem is also very important in Photomosaic for reducing the computation for generating a Photomosaic image.

As an extended approach of Photomosaic, Videomosaic which generates a video composed of many smaller videos has been proposed [6]. A method called Jigsaw Image Mosaics (JIM) is another tiling method by using an arbitrary shape image. The images can be deformed for packing into an arbitrary area [7]. Puzzle Image Mosaics (PIM) is an improved method of JIM for reducing the computation cost and generating better result visually by enhancing edges of shapes [8]. However, JIM and PIM are applied to only less textured images. In addition, the small pieces cannot be sometimes observed in close-up because they may be deformed and upside down [7,8].

In this paper, we propose a Photomosaic method with two following contributions:

- Tiling method using rotated images
- Approximate nearest neighbor searching method based on hashing by combinations of principal components

In our tiling method, images for tiling are not deformed because deformed images cannot be sometimes recognized as a small image due to loss of the original appearance. Instead of deforming, our method allows images to be rotated and overlapped. The use of rotated images increases the possibility of a region in a reference image to match with an image in a database. Our method has templates for several rotated images beforehand. In a local region, the best template matched with the region is selected. As a result, a Photomosaic image with tiled images is generated.

Approximate nearest neighbor searching is one of the important techniques for example based pattern recognition. In Photomosaic, the searching tiled images from a database needs much computation cost. As mentioned above, the quick searching is important. Many works discussed in Section 2 have already proposed for reducing computation cost. However there is still a problem for accelerating the search of nearest data from huge number of data in high dimensional space. We adopt a hashing method based on principal component analysis for considering the distribution of data in a database. In our method, various kinds of hash tables are generated by combination of principal components of a database for clustering neighbors. In the retrieval phase, the candidates of the nearest neighbor of a query are collected from every hash table and the nearest neighbor is selected from the candidates by distance computation.

The rest of this paper is organized as follows: Section 2 discusses the related works about nearest neighbor searching problems. Section 3 presents our approximate nearest neighbor searching method based on combination of principal

component hashing. Section 4 presents Photomosaic method allowing rotation and overlap of images. In Section 5, our methods are evaluated for proving the effectiveness and Section 6 concludes this paper with discussions and possible future works.

2 Nearest Neighor Searching

In nearest neighbor searching problem, we divided the solutions into two categories, tree based approach [9,10] and hash based approach [11,12,13,14].

2.1 Approximate Nearest Neighbor

In tree based approach, Approximate Nearest Neighbor (ANN) is a method using a binary tree [9]. Each node of the tree represents a cell generated by subdividing space. Each leaf is associated with a single point lying within the bounding rectangle for the cell. In the searching, ANN gets candidates from tree search first. Then, the nearest data of the query is selected from the candidates by computing the distances between a query and each candidate.

In ANN, there is an important parameter ϵ for representing degree of approximation. $\epsilon = 0$ means the nearest neighbor searching. The larger ϵ provides smaller computation cost. However, the nearest neighbor may not be found instead of reducing the computation cost.

2.2 Locality Sensitive Hashing

Locality Sensitive Hashing (LSH) is one of approximate nearest neighbor searching methods using a hash [12,13].

In the registration phase, a d-dimensional input vector is converted into L sets of k-dimensional vector by L transform matrices. L is the number of hash tables. Each k-dimensional vector is registered in a list of each hash tables by computing a hash value from k-dimensional vector. In the retrieval phase, a d-dimensional query vector is converted in the same way of the registration phase. Candidates of nearest neighbor vectors are selected from L hash tables. Then, the nearest data of the query is selected from the candidates by computing the distances between a query and each candidate.

In LSH, L transform matrices should be prepared such as $g_1(\boldsymbol{v}), \ldots, g_L(\boldsymbol{v})$. $g_i(\boldsymbol{v})$ has k transform matrices for converting a d-dimensional vector into a natural number such as $g_i(\boldsymbol{v}) = (h_1(\boldsymbol{v}), \ldots, h_k(\boldsymbol{v}))$. $h_j(\boldsymbol{v}))$ is

$$h_{a,b}(\boldsymbol{v}) = \left\lfloor \frac{\boldsymbol{a} \cdot \boldsymbol{v} + b}{w} \right\rfloor \tag{1}$$

where $\boldsymbol{a}$ is a d-dimensional vector, b adjusts bias in $[0, w]$ and $\lfloor \cdot \rfloor$ is a floor function. Each element of $\boldsymbol{a}$ is determined from normal random number.

The relationship between the relative error ratio and complexity of calculation is clearly defined in LSH. However, the searching of the nearest neighbor may not work because the transform matrices are prepared without considering the distribution of data in a database.

2.3 Principal Component Hashing

Principal Component Hashing (PCH) is inspired from LSH [14]. Compared with LSH, $\boldsymbol{a}$ in Eq.2 is an eigenvector of a database in PCH. Each eigenvector is segmented into several buckets by making the density of each bucket be equivalent for clustering neighbors. The candidates are selected by sum of bucket sets on several eigenvectors. In the distance computation, the approximated nearest neighbor can be selected by cutting off the candidates effectively from first eigenvector's bucket.

In PCH, the nearest neighbor searching of a query starts from the buckets of the first eigenvector by computing a hash value of the query. The query is searched on a range of buckets on each eigenvector. However, the size of the range influence the accuracy of the nearest neighbor.

3 Combination of Principal Component Hashing

In Photomosaic, the selection of tiled images from a database should be addressed. A tiled image may be more than thousand dimension vector. In addition, the database may have more than ten thousand images. For this reason, we propose a quick searching method.

Our method is inspired by PCH and LSH. Our hash function is extended from that of PCH. This means that we adapt principal component analysis for converting a input space into lower dimensional space. The registration data by a hash and data retrieval are close to those of LSH. The transform matrices in Section 2.2 is composed of eigenvectors by PCA.

3.1 Registration

In LSH, a data is transformed into a hash key by using a uniform discretization step w in Eq.2. On the other hand, $\boldsymbol{a} \cdot \boldsymbol{v}$ is discretized based on distribution of data in PCH. We adopt the method of PCH.

In Figure 1, discretized value $h_i(\boldsymbol{v})$ from a data $\boldsymbol{v}$ on i-th eigenvector is shown. First, a discretization level d is determined beforehand (in this case, $d = 3, 0 \leq h_i(\boldsymbol{v}) \leq 2$). Thresholds for segmenting the eigenvector into several buckets are determined by making the number of elements in each bucket be same. In retrieval phase, a query is discretized by using the same thresholds. This process is done for the selected number of eigenvectors.

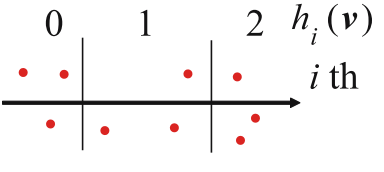


Fig. 1. Discretization

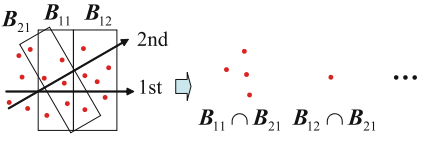


Fig. 2. Product Sets of Buckets

In PCH, an eigenvector corresponds to a hash table as $h_i(\boldsymbol{v})$ is a hash value. Candidates for the nearest neighbors of a query are collected from union of buckets on each eigenvector. The union sometimes provides the huge number of the candidates. Distance computation between a query and the candidates takes most time in approximated nearest neighbor searching. To reduce the number of the candidates, our method generates new buckets by product sets of buckets on several eigenvectors.

Figure 2 briefly shows our method in 2 dimension, where $\boldsymbol{B}_{ij}$ is a bucket of discretized value j on i-th eigenvector. New buckets are generated by every product set of 2 buckets such as $\boldsymbol{B}_{11} \cap \boldsymbol{B}_{21}$. In PCH, one bucket is put on an eigenvector. On the other hand, one bucket is composed of several eigenvectors by product set.

Next, we explain the details in n dimension. Important parameters in our method are as follows: n is the number of selected eigenvectors, m is the number of eigenvectors for computing a product set (less than n) and d is a discretization level on each eigenvector. n and m define the number of hash tables because the number is a combination of m out of n ($_nC_m$). The case of 2 dimension as mentioned above has one hash table because of $n = 2$ and $m = 2$. In the hash table, a bucket is inserted into a list by computing a hash value from discretized values on each eigenvector. The number of lists in each hash table is d^m.

Figure 3 explains the case of $n = 4$ and $m = 3$. The number of generated hash tables is 4. A product set by each bucket of m eigenvectors is stored in a list. A data $\boldsymbol{v}$ is registered in x-th hash table by computing a hash value H_x:

$$H_x = \sum_{y=0}^{m-1} h_{c_{xy}}(\boldsymbol{v}) d^y \tag{2}$$

where c_{xy} represents the eigenvector number in Figure 4. One data is registerd in $_nC_m$ hash tables by computing each hash value.

Our method is merged with PCH and LSH. The discretized value $h_i(\boldsymbol{v})$ comes from PCH. L and k in LSH correspond to $_nC_m$ and m of our method respectively. Our main contribution is the way of making hash tables based on combination of principal components. Therefore, we call our method Combination of Principal Component Hashing (CPCH). In Section 5, the influence by the sizes of n, m and d for searching results is evaluated.

3.2 Retrieval

In retrieval phase, the candidates of a query $\boldsymbol{q}$ are collected from hash tables generated in Section 3.2. The nearest neighbor is selected from the candidates by distance computation.

First, a query $\boldsymbol{q}$ is transformed into $_nC_m$ hash values by using Eq.2. Data in each list are collected by the hash values as candidates. Each data may be counted several times because the date is stored in several lists. After candidates are collected, the candidates are sorted depending on the counts. Since the best count's candidate is not always the nearest neighbor of a query, the distance

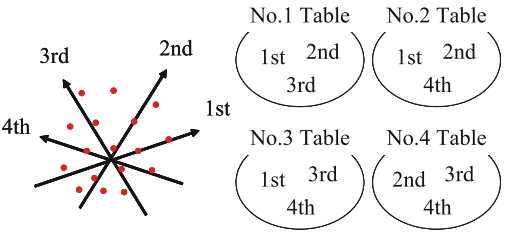


Fig. 3. Generation of Hash Tables

Vector Number

Table Number x \ y	0	1	2
0	1	2	3
1	1	2	4
2	1	3	4
3	2	3	4

Fig. 4. Combination Patterns (c_{xy} in Eq.2)

computation is done against several candidates. We use top b % in the distance computation as PCH used for reducing the number of candidates. The number of candidates and the influence of b are discussed in Section 5.

Next, we explain the details of collecting the candidates in Figure 5. In t-th list of s-th hash table, the new bucket B'_{st} generated by a product set of several buckets B_{ij} is stored. In our retrieval phase, the candidates of the nearest neighbors of a query $\boldsymbol{C}(\boldsymbol{q})$ are collected by

$$\boldsymbol{C} = \bigcup_{s=0}^{{}_nC_m-1} B'_{sH_s} \tag{3}$$

The equation represents that the candidates are collected from union of product sets of buckets. On the other hand, the candidates are collected from union of buckets in PCH.

After collection of the candidates, the distance computation between a query and each candidate is done by

$$D = \sqrt{\sum_{i=1}^{d}(x_i - y_i)} \tag{4}$$

where x_i and y_i are d-dimensional vector and D is a distance. Finally, the approximate nearest neighbor of a query is found.

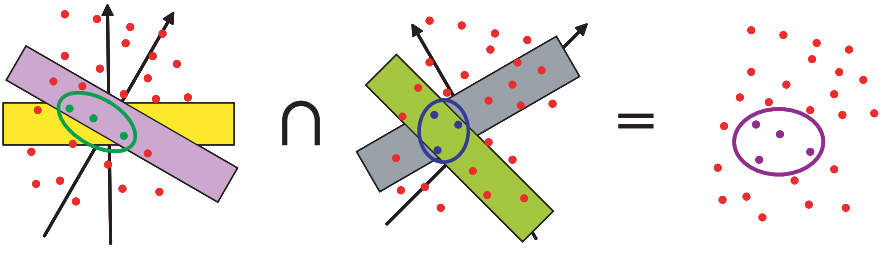


Fig. 5. Collection of Candidates

4 Photomosaic with Rotated Images

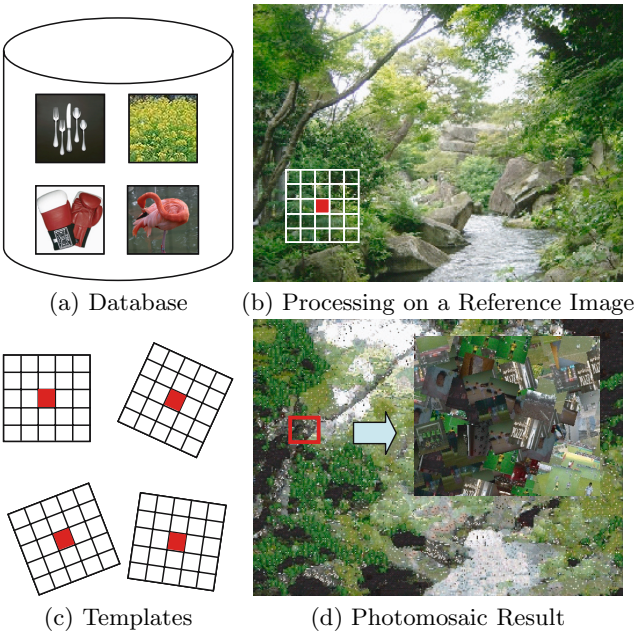
In previous Photomosaics, a tile corresponds to a pixel or a small square region. Our method can treat a rotated square region as a tile. In a database, square images are stored, which are collected from the Internet. Their size is transformed into same size in Figure 6(a).

In Photomosaic processing, a square region represented in Figure 6(c) are extracted at a target pixel as shown in Figure 6(b). Our method has several templates including rotated regions for making rotated squares be a tile. By preparing many templates, the range of the rotation is changeable. In addition, many sizes of images can be tiled by preparing their templates.

In our method, selection of a tiled image at each pixel is done first. At each pixel, the regions corresponding to each template are extracted. The similarities between the regions and images in a database are computed for determining the best matched template. After the similarity computation at each pixel finished, the tiling starts from the least similarity image.

For the region extracted by each template in a reference image, the nearest image is searched from a database by using CPCH described in Figure 3. In Figure 6(c), the number of templates is 4 and 4 images are searched at each pixel. For selecting one nearest image from 4 images, the similarities R_{SSD} between the regions of each template T and searched images I are computed from

$$R_{SSD} = \sum_i \sum_j \sum_c (T(i,j,c) - I(i,j,c))^2 \tag{5}$$



(a) Database (b) Processing on a Reference Image

(c) Templates (d) Photomosaic Result

Fig. 6. Tiling Overview

where (i, j) is a pixel and c is color channel. As a result, best similarity image is selected at each pixel.

This process is not performed in every pixel because many overlaps are occurred. For reducing the overlaps, we empirically set the interval of target pixels for this process. In case that the size of a square image in a database is M $\times$M and the maximum degree of the rotation is θ, the interval is M $\times \sin\theta$.

In the previous process, one best similarity's image is selected from several templates at each target pixel. The order of tiling images starts from the least similarity's image. This causes less possibilities of higher similarity's image to be overlapped by another image.

Usually, a Photomosaic image has higher resolution than the reference image because the larger size of the square image in a database is tiled. The size of Figure 6(d) is five times larger than that of Figure 6(b).

5 Experimental Results

All parts of our algorithm are implemented in C++ and following experiments are carried out on Intel Core 2 Duo 2.2 GHz and 3GB RAM with Windows XP.

5.1 Perfomance Evaluation of CPCH

Our method has four parameters as n, m, d and b. We report the influence of these parameters for the searching results.

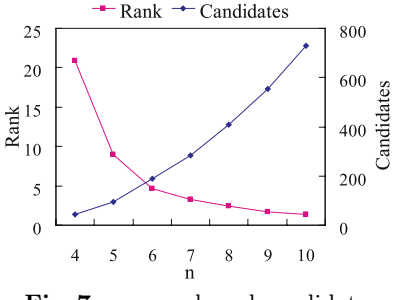


Fig. 7. n vs rank and candidates

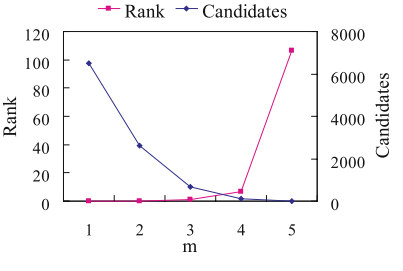


Fig. 8. m vs rank and candidates

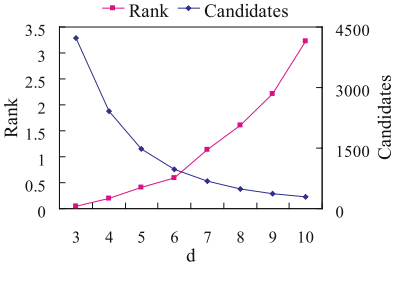


Fig. 9. d vs rank and candidates

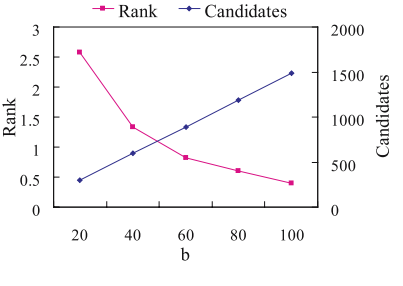


Fig. 10. b vs rank and candidates

(a) Lenna (b) Mandrill

Fig. 11. Reference Images

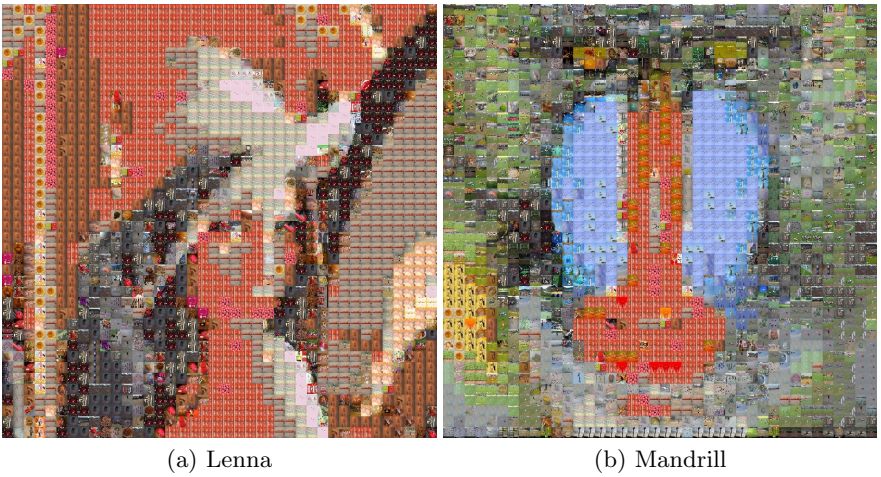
Table 1. Averages of absolute difference

	b	g	r
Fig.12(a)	27	35	35
Fig.13(a)	24	31	31
Fig.12(b)	34	35	34
Fig.13(b)	32	32	31

In a database, 10000 color images which size is 20×20 are stored. We handle an image as a 1200 dimensional vector. 500 images which are not included in the database are prepared as queries. In each query image, the distances with 10000 images are computed for making the rank order of the nearest as a ground truth beforehand.

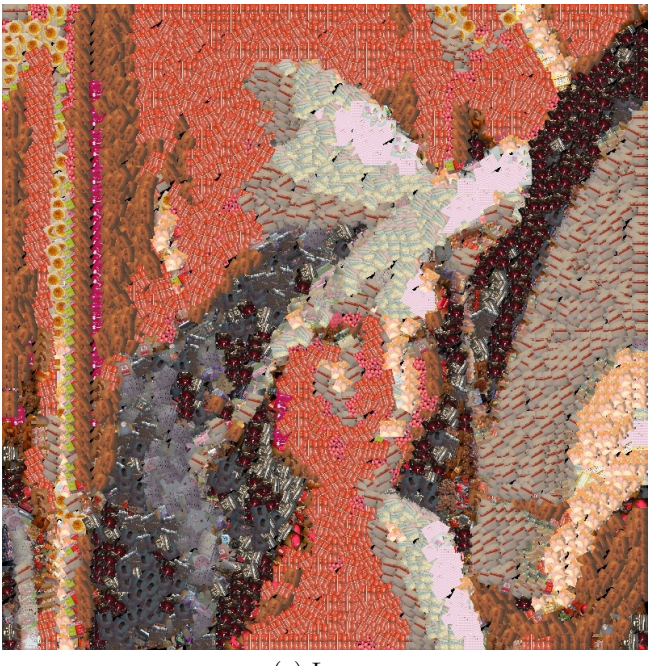
In this experiment, the nearest neighbor image of each query image is searched from 10000 images by our method. For evaluating the accuracy of the searched image, the rank of the image is drawn out from the pre-computed rank order. If the rank is 0, the searched image is the nearest neighbor image. After the searching is done for every query image, the average rank is computed (Rank in Figure.7-10). In addition, the average number of candidates described in Section 3.2 is evaluated (Candidates in Figure.7-10) because the number influence computation time. The parameter in Figure.7-10 are as follows: $m = 3$, $d = 10$, $b = 100$ (in Figure 7). $n = 7$, $d = 7$, $b = 100$ (in Figure 8). $n = 7$, $m = 3$, $b = 100$ (in Figure 9). $n = 7$, $m = 3$, $d = 5$ (in Figure 10).

Since the number of hash tables is ${}_nC_m$, the number of candidates increases when n increases and m decreases in Figure 7 and 8. n, m should be determined



(a) Lenna (b) Mandrill

Fig. 12. Tiling without rotated images



(a) Lenna

(b) Mandrill

Fig. 13. Tiling with rotated images

by considering ${}_nC_m$ for reducing the candidates. d influences the number of lists and the number of elements in each list of a hash table. In Figure 9, the average number of candidates is 284 and the average rank is 3.2 at $d = 10$. In this case, the searched image can be the approximate nearest neighbor image because the average rank is still 3. From this result, larger d may provide better approximate nearest neighbor searching. In Figure 10, the average rank is still 2.5 even if the top 20% candidates are used for the nearest neighbor searching. This represents that neighbor images of a query get more counts. d can be small for reducing the candidates.

By determining n, m, d and b appropriately, the candidates are reduced with keeping the better searching results. The way of determining n, m, d and b automatically will be a next research topic.

5.2 Tiling Results

We report comparisons between a method with rotated images and without rotated images. 3500 square images in our database are collected by using Google Image Search and their size is transformed into 10×10. Two images (200×200) are prepared as shown in Figure 11. We set the parameters of CPCH as $n = 10$, $m = 3$, $d = 5$ and $b = 100$.

In our tiling method, shapes of tiled images depend on prepared templates as described in Section 4. Our method can be applied to the previous method with no rotated image as shown in Figure 12. Figure 13 is generated by using 5 templates which rotation degrees are $-30°$, $-15°$, $0°$, $15°$, $30°$. When searched images are tiled, their size is converted from 10×10 to 40×40 for making the result be 800×800.

For comparing the result of two methods, average of absolute difference between a reference image and a generated image is computed. The size of the generated images is converted into 200×200 to be matched with that of the reference images. Table 1 represents that the result of rotated images provided less average of absolute difference in each reference image. From this result, the use of rotated images provides a better visual result.

6 Conclusions and Future Works

This paper presents Photomosaic method with rotated images and approximate nearest neighbor searching method called combination of principal component hashing. In our Photomosaic method, the templates for rotated images are prepared. At each pixel, the best template is selected. The order of the tiling starts from the least similarity images. The experimental results represent that Photomosaic with rotated images provides better visual result than that without rotated images. In our nearest neighbor searching, a hashing by combination of principal component is proposed. By clustering neighbors with a product set of eigenvectors, various hash tables are generated. In our retrieval, the candidates are collected from the hash tables. In the experimental results, the influence of n, m, d and b for searching result is presented.

In our tiling method, we didn't consider salient area in a tiled image and the area was sometimes overlapped by another image. For remaining the salient area steadily, the salient area is extracted using visual attention model [15] and the method for avoiding the overlap should be considered. In searching method, we will discuss how to determine appropriate n, m, d and b automatically in the next research topic. After that, comparison between CPCH and other nearest neighbor searching method should be done on the same environment.

References

1. Hausner, A.: Simulating decorative mosaics. In: Proc. ACM SIGGRAPH, pp. 573–580 (2001)
2. Elber, G., Wolberg, G.: Rendering traditional mosaics. The Visual Computer 19, 67–78 (2003)
3. Blasi, G.D., Petralia, M.: Fast photomosaic. In: Proc. ACM Winter School on Computer Graphics (2005)
4. Andreamosaic, `http://www.andreaplanet.com/andreamosaic/`
5. Easy photo mosaic maker, `http://www.wyy2001.50megs.com/epmm/index.htm`
6. Klein, A.W., et al.: Video mosaics. In: Proc. NPAR, pp. 21–28 (2002)
7. Kim, J., Pellacini, F.: Jigsaw image mosaics. In: Proc. ACM SIGGRAPH, pp. 657–664 (2002)
8. Di Blasi, G., Gallo, G., Petralia, M.: Puzzle image mosaic. In: Proc. VIIP (2005)
9. Arya, S., et al.: An optimal algorithm for approximate nearest neighbor searching fixed dimensions. Journal of the ACM 45, 891–923 (1998)
10. Nister, D., Stewenius, H.: Scalable recognition with a vocabulary tree. In: Proc. CVPR, pp. 2161–2168 (2006)
11. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. In: Proc. VLDB, pp. 518–529 (1999)
12. Datar, M., et al.: Locality-sensitive hashing scheme based on p-stable distributions. In: Proc. SCG, pp. 253–262 (2004)
13. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Communications of the ACM 51, 117–122 (2008)
14. Matsushita, Y., Wada, T.: Principal component hashing for general distributions. In: Proc. IPSJ SIG Technical Report, 283–288 (2008) (in Japanese)
15. Itti, L., Koch, C., Niebur, E.: A model of saliency-based visual attention for rapid scene analysis. IEEE Trans. PAMI 20, 1254–1259 (1998)