Camera Based Document Image Retrieval with More Time and Memory Efficient LLAH

Tomohiro Nakai, Koichi Kise, Masakazu Iwamura
Graduate School of Engineering, Osaka Prefecture University
1-1 Gakuen-cho, Naka, Sakai, Osaka, 599-8531 Japan
nakai@m.cs.osakafu-u.ac.jp, {kise, masa}@cs.osakafu-u.ac.jp

Abstract

In this paper, we propose improvements of our camera-based document image retrieval method with Locally Likely Arrangement Hashing (LLAH). While LLAH has high accuracy, efficiency and robustness, it requires a large amount of memory. It is also required to speed up the retrieval of LLAH for applications to real-time document image retrieval. For these purposes, we introduce the following two improvements. The first one is reduction of the required amount of memory by removing less important features for indexing from the database and simplifying structure of the database. The second improvement is to reduce exploring alternatives during the retrieval process. From the experimental results, we have confirmed that the proposed improvements realize reduction of the required amount of memory by about 80% and that of processing time by about 60%.

1. Introduction

Document image retrieval is a task of finding document images relevant to a given query from a database of a large number of document images. Various types of queries have been employed in document image retrieval [1]. Camera-based version of document image retrieval is characterized by its queries obtained by capturing documents with cameras. It has excellence that it enables linking paper documents to various services. In other words, paper documents can be viewed as media for providing services to the user. For example, the user can access relevant web sites by capturing a paper document when their URLs are related to the document images in the database.

We have already proposed a camera-based document image retrieval method based on a hashing technique called Locally Likely Arrangement Hashing (LLAH). LLAH is characterized by its accuracy, efficiency and robustness. It has been shown that more than 95% accuracy is realized with about 100 ms retrieval time on a 10,000 pages database[2]. Such accuracy and efficiency are realized by stable and discriminative features of LLAH. It has also been confirmed that LLAH is robust to *perspective distortion*, *occlusion* and *non-linear surface deformation* of pages which are typical problems for camera-captured documents [3, 4]. Features based on geometric invariants defined in local areas realize robustness to those problems.

In exchange for the accuracy and the robustness, LLAH requires a large amount of memory. For example, in order to realize accurate retrieval on a 10,000 pages database, 2.6GB memory is needed. Such heavy consumption of memory limits the scalability of LLAH. For the retrieval of 100,000 pages, for instance, the memory space is beyond what can be easily prepared. In addition, since real-time document image processing using cameras has significant usability [5], an application of LLAH to real-time document image retrieval is desired. In order for LLAH to be used in a real-time processing, further speeding up of its retrieval process is necessary.

In this paper, we propose some improvements of LLAH that solve the above problems. The basic idea of the improvement for the memory consumption is to remove unreliable features for indexing and simplify the structure of the database. As for the speeding up of processing, we introduce a feature-based method of reducing the number of combinations that need to be explored during the retrieval. From the experimental results using 10,000 document images, we have confirmed that the required amount of memory and the processing time are 1/5 and 2/5 of the original, respectively. We also show that the improved version of LLAH scales quite well: the memory consumption and the processing time is almost constant up to the database of size 10,000 images.

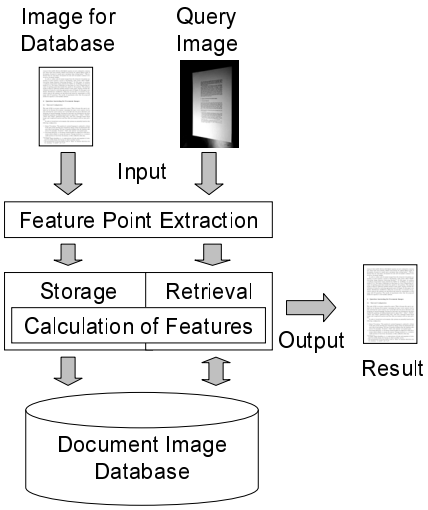


Figure 1. Overview of processing.

2. Document image retrieval with original LLAH

We first explain the original LLAH and the retrieval process with it.

2.1. Overview of processing

Figure 1 shows the overview of processing. At the step of feature point extraction, a document image is transformed into a set of feature points. Then the feature points are inputted into the storage step or the retrieval step. These steps share the step of calculation of features. In the storage step, every feature point in the image is stored independently into the document image database using its feature. In other words, a document image is indexed by using each feature point. In the retrieval step, the document image database is accessed with features to retrieve images by voting. We explain each step in the following.

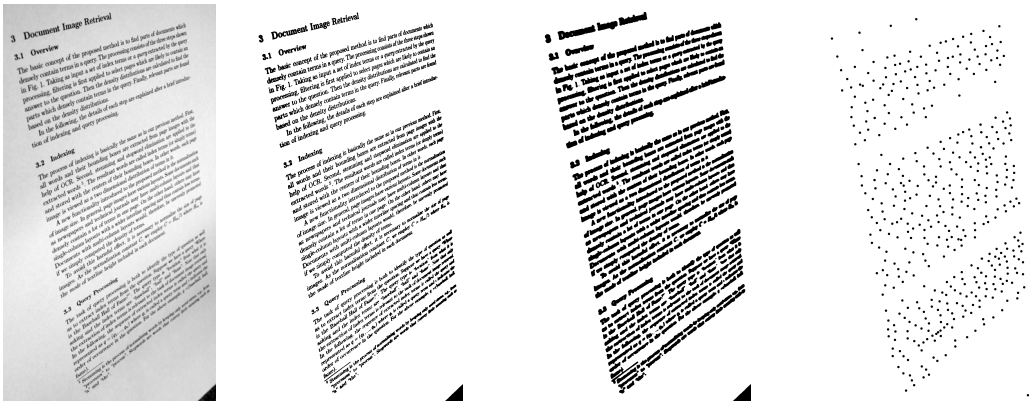
2.2. Feature point extraction

An important requirement of feature point extraction is that feature points should be obtained identically even under perspective distortion, noise, and low resolution. To satisfy this requirement, we employ centroids of word regions as feature points.

The processing is as follows. First, the input image (Fig. 2(a)) is adaptively thresholded into the binary image (Fig. 2(b)). Next, the binary image is blurred using the Gaussian filter. Then, the blurred image is adaptively thresholded again (Fig. 2(c)). Finally, centroids of word regions (Fig. 2(d)) are extracted as feature points.

2.3. Calculation of features

The feature is a value which represents a feature point of a document image. In order to realize successful retrieval, the feature should satisfy the following two requirements. One is that the same feature should be obtained from the same feature point even under various distortions. If different features are obtained from the same feature point at storage and retrieval processes, the corresponding document image cannot be retrieved. We call this requirement "stability of the feature". The other requirement is that different features should be obtained from different feature points. If the same feature is obtained from different feature points, not only the corresponding document image but also other document images are retrieved. We call this requirement "discrimination power of the feature". Both two require-



(a) Input image. (b) Binarized image. (c) Connected components. (d) Feature points.

Figure 2. Feature point extraction.

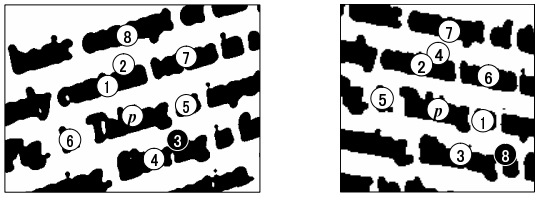


Figure 3. Eight nearest points of p in two images captured from different viewpoints. A circled number indicates its order of distance from p. Different points shown as black circles come up due to perspective distortion.

ments, the stability and the discrimination power, have to be satisfied for successful retrieval.

(1) Stability

In the cases of occlusion, the whole image of a document is not captured. In order to realize stability against occlusion, a feature has to be calculated from a part of document image. In LLAH, each feature point has its feature calculated from an arrangement of its neighboring points. Since features are calculated in a local part of a document image, the same feature can be obtained as long as the same part is captured.

Camera-captured images generally suffer from perspective distortion. In order to calculate features stable against perspective distortion, a geometric invariant is used. For this purpose, one may think the cross-ratio which is invariant to perspective transformation is appropriate. However, it is confirmed that an affine invariant gives higher accuracy than the cross-ratio [2]. This is because perspective transformation in a local area can be approximated as affine transformation and the affine invariant is more robust to change of feature points than the cross-ratio.

In this paper we utilize an affine invariant defined using four coplanar points ABCD as follows:

$$\frac{P(A,C,D)}{P(A,B,C)} \tag{1}$$

where P(A,B,C) is the area of a triangle with apexes A, B, and C.

The simplest definition of the feature of a feature point p is to use 4 nearest feature points from p. However, nearest feature points can change by the effect of perspective distortion as shown in Fig. 3. Hence the invariant from 4 nearest points is not stable. In order to solve this problem, we utilize feature points in a broader local area. In Fig. 3, it is shown that 6 points out of 7 nearest points are common. In general, we assume that common m points exist in n nearest neighbors under some extent of perspective distortion. Based on this assumption, we use common m points to calculate a stable feature. As shown in Fig. 4, common m points are obtained by examining all possible combinations $P_{m(0)}, P_{m(1)}, \cdots, P_{m({}_nC_m-1)}$ of m points from n nearest points. As long as the assumption holds, at least one combination of m points is common. Thus a stable feature can be obtained.

(2) Discrimination power

The simplest way of calculating the feature from m points is to set $m = 4$ and calculate the affine invariant from these 4 points. However, such a simple feature lacks the discrimination power because it is often the case that similar arrangements of 4 points are obtained from different feature points. In order to increase the discrimination power, we utilize again feature points of a broader area. It is performed by increasing the number $m(> 4)$. As m increases, the probability that different feature points have similar arrangement of m points decreases. As shown in Fig. 5, an arrangement of m points is described as a sequence of discretized invariants $(r_{(0)}, r_{(1)}, \cdots, r_{({}_mC_4-1)})$ calculated from all possible combinations of 4 feature points taken from m feature points.

2.4. Storage

Figure 6 shows the algorithm of storage of document images to the database. In this algorithm, the document ID is the identification number of a document, and the point ID is that of a point.

Next, the index H_{index} of the hash table is calculated by

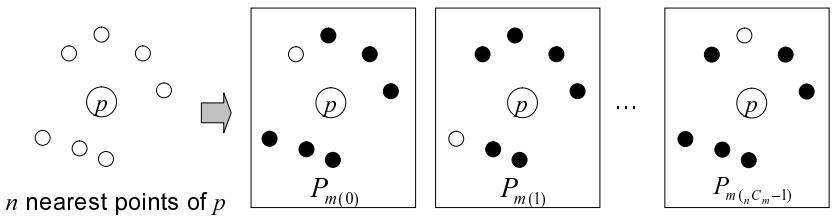


Figure 4. All possible combinations of $m(= 6)$ points from $n(= 7)$ nearest points are examined.

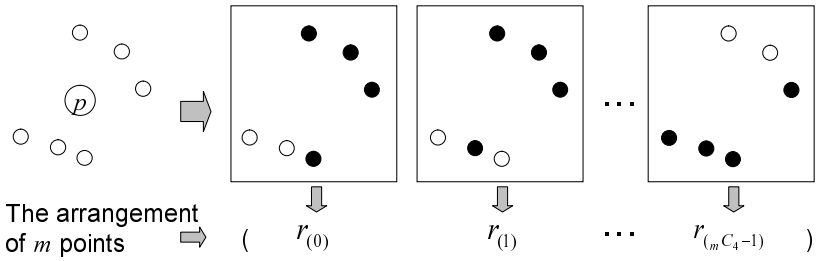


Figure 5. The arrangement of $m(=6)$ points is described as a sequence of invariants calculated from all possible combinations of 4 points.

1: **for each** $p \in$ {All feature points in a database image} **do**
2: $P_n \leftarrow$ A set of the n nearest points of p.
3: **for each** $P_m \in$ { All combinations of m points from P_n } **do**
4: $L_m \leftarrow (p_0, \cdots, p_i, \cdots, p_{m-1})$ where p_i is an ordered point of P_m based on the angle from p to p_i with an arbitrary selected starting point p_0.
5: $(L_4(0), \cdots, L_4(i), \cdots, L_4({}_mC_4 - 1)) \leftarrow$ A lexicographically ordered list of all possible $L_4(i)$ that is a subsequence consisting 4 points from L_m.
6: **for** $i = 0$ to ${}_mC_4 - 1$ **do**
7: $r_{(i)} \leftarrow$ a discretized affine invariant calculated from $L_4(i)$.
8: **end for**
9: $H_{\mathrm{index}} \leftarrow$ The hash index calculated by Eq. (2).
10: Store the item (document ID, point ID, $r_{(0)}, \cdots, r_{({}_mC_4-1)}$) using H_{index}.
11: **end for**
12: **end for**

Figure 6. Storage algorithm.

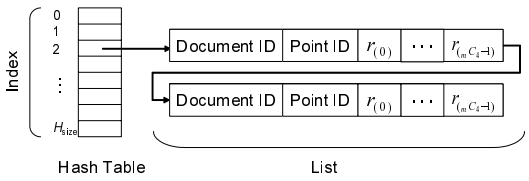


Figure 7. Configuration of the hash table.

the following hash function:

$$H_{\mathrm{index}} = \left(\sum_{i=0}^{{}_mC_4-1} r_{(i)} k^i \right) \bmod H_{\mathrm{size}} \qquad (2)$$

where $r_{(i)}$ is a discrete value of the invariant, k is the level of quantization of the invariant, and H_{size} is the size of the hash table.

The item (document ID, point ID, $r_{(0)}, \cdots, r_{({}_mC_4-1)}$) is stored into the hash table as shown in Fig. 7 where chaining is employed for collision resolution.

1: **for each** $p \in$ {All feature points in a query image} **do**
2: $P_n \leftarrow$ A set of the n nearest points of p.
3: **for each** $P_m \in$ { All combinations of m points from P_n } **do**
4: **for each** $p_0 \in P_m$ **do**
5: $L_m \leftarrow (p_0, \cdots, p_i, \cdots, p_{m-1})$ where p_i is an ordered point of P_m based on the angle from p to p_i with a starting point p_0.
6: $(L_4(0), \cdots, L_4(i), \cdots, L_4({}_mC_4 - 1)) \leftarrow$ A lexicographically ordered list of all possible $L_4(i)$ that is a subsequence consisting 4 points from L_m.
7: **for** $i = 0$ to ${}_mC_4 - 1$ **do**
8: $r_{(i)} \leftarrow$ a discretized affine invariant calculated from $L_4(i)$.
9: **end for**
10: $H_{\mathrm{index}} \leftarrow$ The hash index calculated by Eq. (2).
11: Look up the hash table using H_{index} and obtain the list.
12: **for each** item of the list **do**
13: **if** Conditions to prevent erroneous votes [2] are satisfied **then**
14: Vote for the document ID in the voting table.
15: **end if**
16: **end for**
17: **end for**
18: **end for**
19: **end for**
20: Return the document image with the maximum votes.

Figure 8. Retrieval algorithm.

2.5. Retrieval

The retrieval algorithm is shown in Fig. 8. In LLAH, retrieval results are determined by voting on documents represented as cells in the voting table.

First, the hash index is calculated at the lines 7 to 10 in the same way as in the storage step. At the line 11, the list

shown in Fig. 7 is obtained by looking up the hash table. For each item of the list, a cell of the corresponding document ID in the voting table is incremented if it has the same feature $(r_{(0)},\cdots,r_{({}_mC_4-1)})$. Finally, the document which obtains the maximum votes is returned as the retrieval result.

3. Reduction of the required amount of memory

In this section, we introduce a method of reduction of the required amount of memory. In LLAH, many features are calculated in order to realize the stability of features. All features are stored in the hash table in the form of linked lists regardless to their importance. We reduce memory consumption by removing less important features and changing data structure of the database.

Let us show an example. In the following condition,

- $n = 7$, $m = 6$, $H_{\rm size} = 1.28 \times 10^8$
- The number of document images in the database is 10,000
- Average number of feature points in a document image is 630
- Document ID, point ID and $r_{(i)}$ are stored in 2 bytes, 2bytes and 1byte variables respectively
- A pointer variable requires 8 bytes

the hash table requires $1.28 \times 10^8 \times 8 = 1.0{\rm GB}$ and linked lists require $10,000 \times 630 \times 7 \times (2+2+1 \times 15+8) = 1.2{\rm GB}$. Therefore the total required amount of memory is $2.2{\rm GB}$.

Features which cause collisions in the hash table are less important because such features are shared by other points and thus likely to lack discrimination power. They also increase the processing time of retrieval since they frequently come up and increase the number of votes. From an experimental result, it is confirmed that the number of hash table entries with collisions is 28% of the number of hash table entries with at least one item. Since the number of entries with collisions is minor, removing features with collisions will not cause fatal effect on accuracy of retrieval. Thus we have decided to remove linked lists with collisions.

Removal of features with collisions enables further reduction of memory consumption since it allows to simplify the data structure of the hash table. Features (invariants $r_{(i)}$) are stored in order to find the appropriate item from the linked list in the case of collisions. Because we have eliminated all collisions, we can also remove the records of features. Moreover document IDs and point IDs are not needed to be stored in the form of linked list because only one item is stored at an index of the hash table. Therefore we adopt a simple hash table as shown in Fig. 9 as the structure of the database. Changing the data structure into a simple hash table enables further reduction of the memory consumption. For example, the required amount of memory is 512MB under the condition of the above example ($H_{\rm size} = 1.28 \times 10^8$, a document ID and a point ID are stored in 2 bytes variables). Therefore 77% reduction of the required amount of memory can be realized.

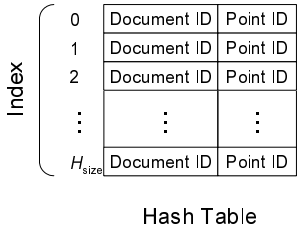


Figure 9. Configuration of the simplified hash table.

Another way to reduce the required amount of memory is to reduce the size of the hash table $H_{\rm size}$. However, $H_{\rm size}$ significantly affects performance of retrieval. Especially for the simplified hash table, the size of the hash table is fatal. If the hash table has an insufficient number of index space to store items, many entries will be invalidated due to collisions.

4. Speeding up retrieval

We also introduce an improvement of the storage and the retrieval algorithm to speed up the retrieval process.

In the retrieval algorithm shown in Fig. 8, all points of P_m is used as a starting point p_0 to examine all cyclic permutations of L_m at the lines 4 and 5. This is because L_m of the retrieval algorithm does not necessarily match L_m of the storage algorithm due to rotations of camera-captured images.

However, the examination of all cyclic permutations can be omitted if the same p_0 is always selected both at the storage and the retrieval processes. We have introduced a selection rule of the starting point to the storage and retrieval algorithm.

The selection rule is shown in Fig. 10. For each point i of m points, an affine invariant $s_{(i)}$ is calculated by combining it with the following three points. If $s_{(j)}$ has the maximum value in $(s_{(0)} \cdots s_{(m-1)})$, the point j is selected as the starting point. In the example of Fig. 10, point 1 is the starting point. If there are two or more equivalent maximum values, the succeeding value $s_{((i+1) \bmod m)}$ is used to select one of them. For example, if $s_{(i)} = s_{(j)}$

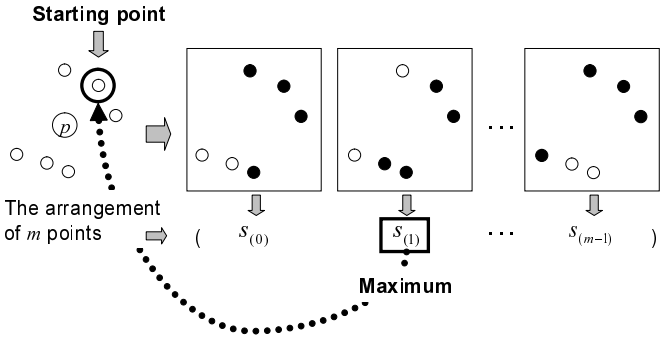


Figure 10. The point which gives maximum $s_{(i)}$ is selected as the starting point.

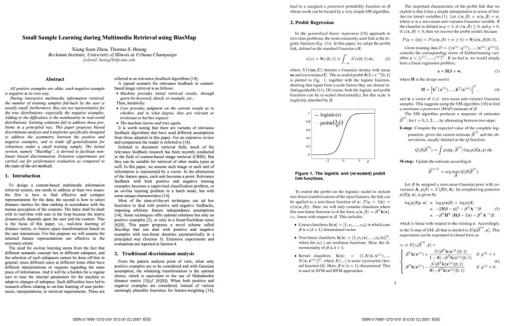


Figure 11. Examples of images in the database.

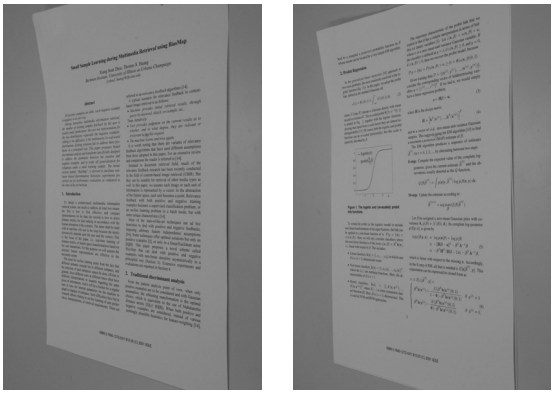


Figure 12. Examples of query images.

and both are the maximum, then the value of $s_{((i+1)\bmod m)}$ and $s_{((j+1)\bmod m)}$ are compared. If $s_{((i+1)\bmod m)}$ is larger, the point i is selected as the starting point. In the case that $s_{((i+1)\bmod m)} = s_{((j+1)\bmod m)}$ the succeeding values $s_{((i+2)\bmod m)}$ and $s_{((j+2)\bmod m)}$ are likewise examined.

5. Experimental results

In order to examine effectiveness of improvements introduced in this paper, we investigated performances of the original and improved versions of LLAH. To clarify the effect of memory reduction stated in Sect. 3. and that of speeding up stated in Sect. 4., we measured the required amount of memory, processing time and accuracy of three versions of LLAH: the first one is the original LLAH, the second one is the memory reduced, and the third one is the memory reduced and speeded up version. The third version is the proposed method in this paper.

Document images stored in the database were images converted with 200 dpi from PDF files of single- and double-column English papers collected mainly from CD-ROM proceedings. Examples of images in the database are shown in Fig. 11. Query images were captured using a digital camera with 6.3 million pixels. As shown in Fig. 12, they were captured from a skew angle (about 45 degree). Since the angle with which query images are captured (45 degree) is different from that of the images in the database (90 degree), experiments performed with these query images and the database would demonstrate robustness of the proposed method to perspective distortion. Note that the query images suffer from severer distortion than those of [2]. Experiments were performed on a workstation with AMD Opteron 2.8GHz CPUs and 16GB memory. Parameters[1] were set to $n = 7, m = 6, k = 15$. H_{size} was set to 1.28×10^8.

5.1. Required amount of memory

Figure 13 shows the amount of required memory of the three version of LLAH with 100, 1,000 and 10,000 pages databases. The original version of LLAH required 5 times larger amount of memory than improved ones. Moreover,

[1] Some experimental results with different n and m are found in [2].

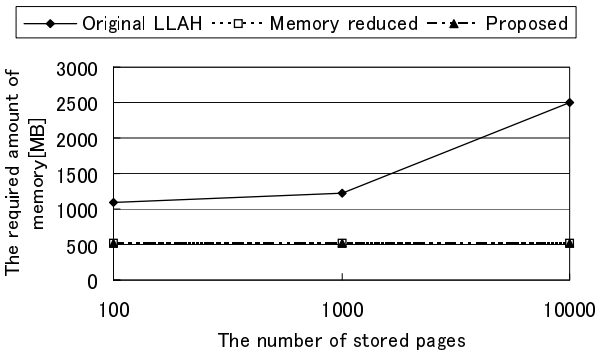


Figure 13. The relationship between the number of stored pages and the required amount of memory.

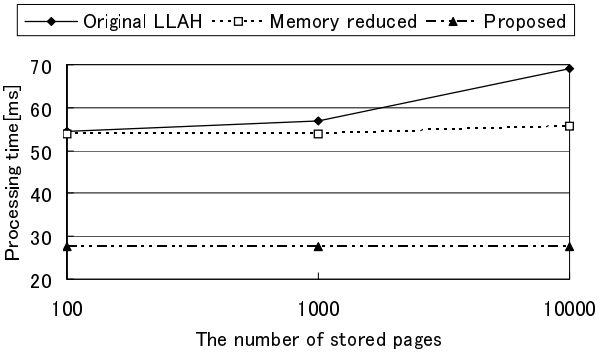


Figure 14. The relationship between the number of stored pages and processing time of retrieval.

the amount increased with increasing the number of stored pages. Since the original LLAH adopts the linked list as the form of the database, more stored pages result in more amount of memory. On the other hand, in the memory reduced versions, the required amount of memory was constant regardless of the number of stored pages. Since these versions adopt a simple hash table as the form of the database, required memory is that for a hash table of a fixed size.

5.2. Processing time of retrieval

Figure 14 shows processing time by each version of LLAH. The proposed version of LLAH realizes reduction of the processing time by about 60%. This is because the number of computed invariants and access to the database have been reduced by skipping the calculation of all possible cyclic permutations.

It is also confirmed that processing time of the memory reduced version was almost constant regardless of the number of stored pages as contrasted to the original version. This is because access to a simple hash table requires constant computations while access to linked lists requires computations in proportion to the number of stored items.

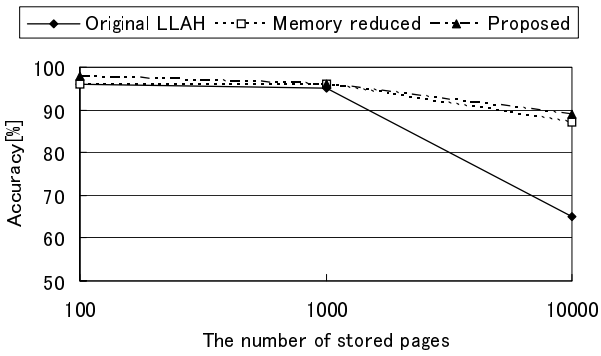


Figure 15. The relationship between the number of stored pages and accuracy.

5.3. Accuracy

Figure 15 shows accuracy of retrieval of each version of LLAH. Improved versions demonstrated higher performance in terms of accuracy in addition to time and memory efficiency. In contrast to the original LLAH which showed less accuracy with a larger number of stored pages, improved versions showed higher accuracy although they also had a decrease in accuracy with 10,000 stored pages.

Although the improvements proposed in this paper were not intended to improve accuracy, they realized higher accuracy. We consider the reason is that erroneous votes are decreased as a result of removal of less important features. In the improved versions, features which cause collisions are removed from the database. Since such features tend to cause erroneous votes, removal of them results in higher accuracy.

6. Related work

In LLAH, document images are retrieved based on local arrangements of feature points. Therefore it can be classified into an image retrieval method using local features. There have been various types of image retrieval methods

using local features. They can be classified into two types: one based on complex features such as SIFT and the other based on simple features such as feature points.

Video Google [6] is one of the image retrieval methods using complex local features. In Video Google, a codebook is created by clustering SIFT features extracted from images prior to retrieval. Retrieval is performed based on vector quantized SIFT features of query images using the codebook. In order to realize accurate retrieval, a large codebook is needed. However use of a large codebook results in long processing time since nearest neighbor search takes much time. It is also a problem that calculation of SIFT features needs much computation.

Geometric Hashing(GH) [7] is well known as an image retrieval method based only on feature points. In GH, features are calculated by combining feature points in order to realize stability of features. For example, O(N^4) computation is required for the retrieval under affine distortion where N is the number of feature points. Therefore the number of combinations becomes enormous when images have many feature points. Since document images have many feature points, it is prohibitive to apply GH to retrieval of document images. For more details, see [8].

7. Conclusion

In this paper, we have introduced improvements to the LLAH. The required amount of memory was decreased by removal of unnecessary features and simplification of structure of the hash table. Processing time of retrieval was shortened by the improvement of the retrieval algorithm. From the experimental results, we have confirmed reduction of the required amount of memory by 80% and shortening of the processing time of retrieval by 60%. It is also confirmed that the improvements bring higher accuracy. From these results, we can conclude the proposed improvements realize better scalability and extensibility to applications which require hi-speed retrieval.

Our future tasks include improvements of feature point extraction process. The feature point extraction process can become a bottleneck of the whole image retrieval process using LLAH since it requires heavy image processing. Since LLAH currently covers only English document images, it is also necessary to add other objects in the target of LLAH.

Acknowledgement

This research is supported in part by Grant-in-Aid for Scientific Research from Japan Society for the Promotion of Science (19300062, 19·7621).

References

[1] D. Doermann, “The indexing and retrieval of document images: a survey”, Computer Vision and Image Understanding, vol. 70, no. 3, pp.287–298, 1998.

[2] T. Nakai, K. Kise, and M. Iwamura, “Use of affine invariants in locally likely arrangement hashing for camera-based document image retrieval”, Lecture Notes in Computer Science (7th International Workshop DAS2006), vol. 3872, pp.541–552, 2006.

[3] J. Liang, D. Doermann, and H. Li, “Camera-based analysis of text and documents: a survey”, IJDAR, vol. 7, pp.84–104, 2005.

[4] P. Clark, and M. Mirmehdi, “Recognising text in real scenes”, IJDAR, vol. 4, pp. 243–257, 2002.

[5] C. H. Lampert, T. Braun, A. Ulges, D. Keysers, and T. M. Breuel, “Oblivious document capture and real-time retrieval”, Proc. CBDAR2005, pp.79–86, 2005.

[6] J. Sivic and A. Zisserman, “Video google: a text retrieval approach to object matching in videos”, Proc. ICCV2003, vol. 2, pp.1470–1477, 2003.

[7] H. J. Wolfson and I. Rigoutsos, “Geometric hashing: an overview”, IEEE Computational Science & Engineering, vol. 4, no. 4, pp.10–21, 1997.

[8] M. Iwamura, T. Nakai and K. Kise, “Improvement of retrieval speed and required amount of memory for geometric hashing by combining local invariants”, Proc. BMVC2007, 2007 [to appear].