Compressed Representation of Feature Vectors Using a Bloomier Filter and Its Application to Specific Object Recognition

Katsufumi Inoue and Koichi Kise
Graduate School of Engineering, Osaka Prefecture University
1-1 Gakuencho, Naka, Sakai, Osaka, 599-8531 Japan
inoue@m.cs.osakafu-u.ac.jp, kise@cs.osakafu-u.ac.jp

Abstract

Nearest neighbor search of feature vectors representing local features is often employed for specific object recognition. In such a method, it is required to store many feature vectors to match them by distance calculation. The number of feature vectors is, in general, so large that a huge amount of memory is needed for their storage. A way to solve this problem is to skip the distance calculation because no feature vectors need to be stored if there is no need to calculate the distance. In this paper, we propose a method of object recognition without distance calculation. The characteristic point of the proposed method is to use a Bloomier filter, which is far memory efficient than hash tables, for storage and matching of feature vectors. From experiments of planar and 3D specific object recognition, the proposed method is evaluated in comparison to a method with a hash table.

1. Introduction

Object recognition has been spotlighted as a key technology to deal with a large volume of image data effectively and efficiently. Tasks of object recognition can be classified into two categories: generic [20] and specific [17, 18, 15]. Generic object recognition is to recognize *classes* of objects such as "a chair" and " a car". Specific object recognition, on the other hand, is for identifying object *instances* such as a specific type of chair and car, in other words "the chair" and "the car". This paper focuses on the latter, especially methods which employ local features such as SIFT (Scale-Invaliant Feature Transform) [13].

A technology of specific object recognition which allows us to recognize a large number of objects is required to provide a practical service because there are a lot of objects around us. Local features are multidimensional and real-valued vectors and hundreds to thousands of them are generally extracted from a single image. Since they have high discrimination power, we utilize them for object recognition. As a specific object recognition method, a fundamental method is based on voting by matching feature vectors [17, 12]. This method employs nearest neighbor search of feature vectors for matching. Although even such a simple method offers high recognition rates, it poses the following problems caused by a large number of feature vectors. First, matching of feature vectors (finding their nearest neighbors) requires a long processing time. Second, many feature vectors need an immense amount of storage.

For the first problem, it is necessary to speed up the process of nearest neighbor search for matching. Fortunately, many methods have already been proposed. Approximate nearest neighbor search [1, 2] utilized in these methods enables us to improve the efficiency dramatically while keeping the accuracy of recognition [9].

For the second problem, on the other hand, it is not easy to achieve the breakthrough. Some methods to reduce the amount of memory have already been proposed. For example, one of them utilizes the "bag of features" model, which is based on vector quantized feature vectors called "*visual words*" [14, 16, 19] for reduction of the memory. Another method is to reduce it by scalar quantization which limits the number of bits for each dimension [10]. However, these methods need the amount of memory proportional to the number of feature vectors because of the distance calculation. Therefore there is a limit of memory reduction with these methods.

A possible approach to solve this problem is to skip the distance calculation. If the distance calculation is not necessary for recognition, there is no need to store feature vectors. From this viewpoint, a hash-based method has been proposed [9]. In this method, the similarity of feature vectors is identified whether they have the same hash values. However, this method still has a problem about the amount of memory: most bins of the hash table are empty. Accordingly, there is still a room of improving the space efficiency of this method.

To achieve the breakthrough, we propose a new memory

2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops
978-1-4244-4441-0/09/$25.00 ©2009 IEEE

reduction method with a Bloomier filter [4]. Bloomier filter is probabilistic data structures that is more space efficient than hash tables. Similar to hash tables, Bloomier filter records whether feature vectors are stored in the memory. Bloomier filter differs from hash tables in that Bloomier filter allows false positives of feature vectors. The false positive means that a feature vector is erroneously identified as a member of a data set. The objective of this research is to reduce the amount of memory for object recognition by using a Bloomier filter, while keeping the recognition rate as high as possible. In this paper, the proposed method is evaluated based on experiments of planar and 3D specific object recognition in comparison to a method with a hash table.

2. Related Work

An important issue of specific object recognition based on local features is how to reduce the amount of memory for recognition. Some conventional memory reduction methods, especially based either on vector quantization [14, 16, 19] or on scalar quantization [10], have been proposed. The former employs not *raw* feature vectors, which means feature vectors as they are, but their vector quantized version called "*visual words*" to reduce the amount of memory. For large scale object recognition, high accuracy of object recognition is achieved only with a large number of visual words, each of which represents a few feature vectors. Since the relation between recognition accuracy and the required memory is a trade-off, it is difficult to reduce the amount of memory with this method while keeping the recognition as high as possible. On the other hand, the latter is to reduce it by scalar quantization which limits the number of bits for each dimension. This method has been known that it allows us to reduce the memory without affecting the accuracy of object recognition. However, the amount of memory which depends on the number of feature vectors is also required with this method. From these reasons, it is necessary to improve memory reduction.

Another approach to the reduction is to use not main memory but auxiliary storage [5]. The strategy is to store only the pointers to feature vectors. Although this method is useful to reduce the memory, it requires a longer processing time owing to the longer accessing time of auxiliary storage compared with the main memory.

Besides the methods mentioned above, there is a memory reduction method by sampling feature vectors [7]. In this method, feature vectors stored in the database are selected by evaluation of their effectiveness for recognition. Even with this strategy, the amount of memory is still proportional to the number of feature vectors stored in the database.

Further memory reduction can be achieved by giving up storing feature vectors. This means that no distance calculation is employed for matching feature vectors. A hash-based method with this approach has been proposed [9]. In the hash-based method, only the existence of feature vectors is marked in the hash table, and matching of feature vectors is done by checking the mark. Since there is no feature vectors stored in the database, a drastic reduction can be achieved. However, we still have a problem that distribution of the marks are quite uneven; almost all bins in the hash table are typically empty. Thus there is still a room of improving the space efficiency.

3. Conventional Method Based on a Hash Table

In this section, we explain a conventional method with a hash table [9]. This method is a base of the proposed method. The processing of this method is divided two steps: the step of database construction, i.e., storage of feature vectors in a hash table, and the step of object recognition using the strategy of voting. In the following section, we explain the concrete steps of this method.

3.1. Database Construction

The conventional method utilizes feature vectors calculated by SIFT [13] as the detector of local regions and PCA-SIFT [8] as their descriptor. The number of dimensions of feature vectors is 36. For fast access to feature vectors, this method employs a hash function as follows. Let $\boldsymbol{p}$ be an original feature vector $\boldsymbol{p} = (p_1, p_2, \ldots, p_{36})$ obtained by PCA-SIFT from one of the images called "model images" for construction of the database. First, in order to index $\boldsymbol{p}$ in the hash talbe, it is required to convert real-valued feature vectors into integers. It is achieved by binarizing each dimension [1]:

$$u_j = \begin{cases} 1 & \text{if } p_j \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

to produce a bit vector $\boldsymbol{u} = (u_1, u_2, \ldots, u_d)$ from which the first $d(\leq 36)$ elements are employed for the indexing. Next the following hash function

$$H_{\text{index}} = \left(\sum_{j=1}^{d} u_j 2^{(j-1)} \right) \mod H_{\text{size}} \tag{2}$$

where H_{size} is the size of the hash table is applied to obtain the hash value of $\boldsymbol{p}$. Then the ID of the object from which $\boldsymbol{p}$ is extracted is stored in the bin indicated by the hash value. Multiple vectors with the same hash value are stored by chaining. If the length of the chain exceeds a predetermined threshold c, all entries with the same hash value are deleted. This is the strategy called "stopword elimination". Because these vectors tend to be very similar with

[1] The median of each dimension of PCA-SIFT is close to 0.

one another, we consider these vectors have little contribution to object recognition. In iterating the above process, all feature vectors extracted from model images are stored in the hash table.

3.2. Object Recognition

Let $\boldsymbol{q}$ be a feature vector called a query vector extracted a query image. In the conventional method, the elementary task is to retrieve a set of feature vectors $\{\boldsymbol{p}\}$ close to $\boldsymbol{q}$ from the database. Then this method simply votes for the objects to which all vectors in the set $\{\boldsymbol{p}\}$ belong. The object with the maximum number of votes is the result of recognition.

The most important step here is to find the set $\{\boldsymbol{p}\}$. A way is to access the hash table based on the hash value obtained from the query vector. In this case feature vectors having the same hash value are retrieved. Unfortunately it is unsatisfactory since variation of the query vector may change its bit vector, which results in accessing the bin that may not contain the correct object ID.

In order to ease this problem, the query vector is expanded into several bit vectors to find its corresponding object ID. Because the query vector is transformed into the bit vector using the threshold of 0 for each dimension, the query vector having values close to 0 at some dimensions can be converted into bit vectors different from its original one. The conventional method simply utilizes the error range e as a parameter for generating different bit vectors. For the dimension q_j satisfying $|q_j| \leq e$ in the query vector $\boldsymbol{q} = (q_1, q_2, \ldots, q_d)$, the other bit value $u'_j = 1 - u_j$ is also utilized to generate bit vectors. This means that the error of the value is estimated within the range defined by e. For example, for the query vector $\boldsymbol{q} = (3, 59, -12)$ and $e = 5$, two bit vectors (1, 1, 0) and (0, 1, 0) are employed for the access of the hash table.

Unlimited application of this strategy increases the expanded bit vectors exponentially. In order to avoid this problem, the conventional method utilizes the limit b of the number of dimensions for the application. The method applies the strategy for q_j from $j = d$. If the number of dimensions that satisfies the threshold e exceeds the limit b, those with larger indices are adopted up to the limit. In other words, the number of expanded bit vectors for a query vector is at most 2^b. For the above example of $\boldsymbol{q}$ with $e = 20$ and $b = 1$, two bit vectors (1, 1, 0) and (1, 1, 1) are obtained.

In order to improve efficiency of the recognition process, the conventional method employs cascaded recognizers [11] as shown in Fig. 1. This is based on the observation that the degree of difficulty for correct recognition depends on images to be recognized. The step s recognizer, which is depicted in Fig. 1 as a square with a number from 1 to N, is the above mentioned recognizer with $b = s - 1$. In the first step, a query image is recognized by using the bit vector calculated from $\boldsymbol{q}$ with $b = 0$. If the maximum number of votes is greater enough than the second one, the object with the maximum number of votes is the result of recognition. If not, the recognizer at the next step (with $b = 1$) is applied. The cascaded recognizers are characterized by the property called difference accessibility. This strategy guarantees that, even if the final recognizer is employed, it takes almost the same processing time as the single recognizer with the value b.

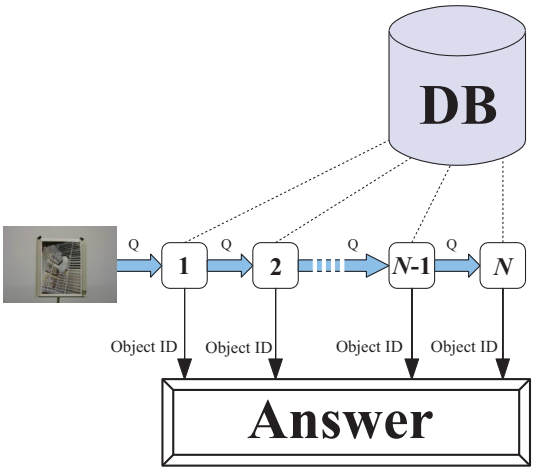


Figure 1. Cascade of recognizers.

3.3. Problem of the Conventional Method

In the conventional method, it is required that the dimension d of bit vectors is large enough to achieve high accuracy. A larger d causes the exponential growth of the size of hash table. In the preliminary experiment with a hash table whose size is $H_{\text{size}} = 2^d$ and 10 million feature vectors calculated by PCA-SIFT, more than 65% bins of the hash table with $d = 24$ and more than 96% of them with $d = 28$ were empty. From the preliminary experiment, it is found that this method has a problem of space inefficiency. To solve this problem, we propose a new memory reduction method using the Bloomier filter whose space efficiency is better than that of a hash table.

4. Bloom and Bloomier Filters

In this section, we explain the Bloomier filter [4] and the Bloom filter [3] which is a base of the Bloomier filter.

4.1. Bloom Filter

The Bloom filter is a space efficient probabilistic data structure. This is used to memorize whether an element is a member of a data set. The Bloom filter has some problems as follows. First, it has a risk of false positives which are a type of error that an element is erroneously recognized as a member of a data set. Second, once an element is added to the set, it cannot be removed. While having these problems,

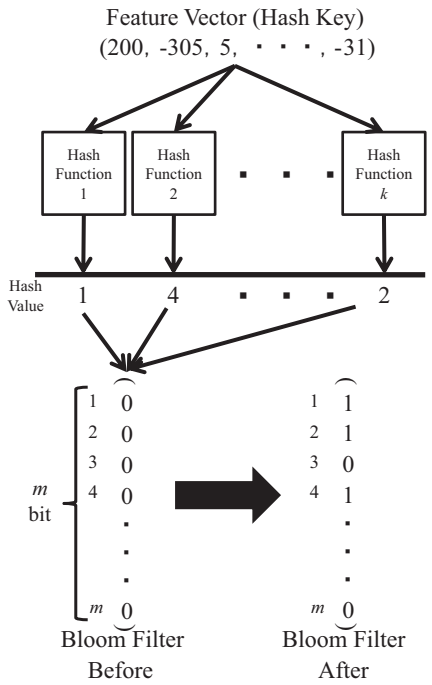


Figure 2. Overview of the storage of feature vectors to a Bloom filter.

Bloom filters have a strong advantage that some number of elements are added to the set without increasing the amount of memory for storing them. Due to this advantage, the number of elements which can be stored in Bloom filters is generally greater than that with other data structures, such as self-balancing binary search trees and hash tables, with the same amount of memory. Obviously, the probability of false positives depends on the number of elements that are added to the set. Although the Bloom filter has the risk of false positives, we utilize the space efficiency of the Bloom filter in order to reduce the amount of memory for storing feature vectors. In the following, let us explain the way to store feature vectors to the Bloom filter and to retrieve feature vectors from it.

Figure 2 shows the overview of the storage. First, an empty Bloom filter which is a bit array of m bits is prepared and all bits are set to 0. In the following, we call m (the size of bit array) as "Table Size". Next, a feature vector is converted to k hash values using k different hash functions. At this moment, these hash values obtained by the hash functions are m integers ranging from 1 to m. Each hash value indicates one of the m array positions. Then, to add a feature vector, set the bits at all positions indicated by the hash values to 1.

In the retrieval process, in order to test whether a query feature vector is a member of the data set, feed it to the k hash functions to get k array positions. If all bits at these positions are 1, the query feature vector is regarded as a member of the data set. If not, in other words some bits at these positions are 0, the feature vector is not in the data set. The false positive occurs when all bits are 1 for a query feature vector which is not a member of the set. The reason of this false positive is that all bits have accidentally been set to 1 during the storage of some other feature vectors in the data set.

4.2. Bloomier Filter

The Bloomier filter is an associative array which can associate a value with a feature vector by using multiple Bloom filters. This data structure has the same problems and advantages that the Bloom filter has. Let us explain how to associate a value with a feature vector.

In order to make the story simple, let us consider the case that the values associated by the Bloomier filter are 0 and 1. In this case, we create a pair of Bloom filters X and Y. The feature vectors whose associated value is 0 are added to X, and those with 1 are added to Y. To retrieve the value associated with a query feature vector, we check both Bloom filters. If the query feature vector is contained in X and not in Y, the probability that the associated value is 0 is high and vice versa.

We utilize the Bloomier filter to associate object IDs. Suppose an object ID is represented by n bits, we prepare the Bloomier filter consisting of $2n$ Bloom filters. In other words, we distinguish 2^n objects with the above Bloomier filter. If we employ the Bloom filter to identify N objects, the required number of Bloom filters is $\log_2 N$.

5. Proposed Method

In this section, we propose a specific object recognition method using the Bloomier filter. Like the conventional method mentioned above, we utilize feature vectors calculated by PCA-SIFT. In the following, first let us show how to construct the database with the Bloomier filter. Then, we explain the process of object recognition using them.

5.1. Database Construction

Let $X_1, X_2, \ldots, X_n$ be the Bloom filters whose associated value is 0 and $Y_1, Y_2, \ldots, Y_n$ be those whose associated value is 1. The *Table Size* of each Bloom filter is calculated as:

$$a \times M_f^g \quad [\text{bit}] \tag{3}$$

where $M_f^g (f \in \{1, 2, \ldots, n\}, g \in \{0, 1\})$ is the total number of feature vectors whose fth bit of object ID is g. a means how many bits are employed for storing one feature vector in the database. In the case of X_i, for example, M_i^0 means the total number of feature vectors obtained from objects whose ith bit of object ID is 0. Hence, the *Table Size* of X_i is $a \times M_i^0$.

The next step of database construction is to store feature vectors to n Bloom filters in order to associate the object ID. To make the story simple, we consider the following example. Suppose that the object ID 3 is represented as "10". Because the first bit of the object ID 3 is 1 and the second bit is 0, the feature vectors extracted from the object whose ID is 3 are stored to the Bloom filers Y_1 and X_2. We explain more details in the following. First, we convert the feature vector $\boldsymbol{p}$ into the bit vector $\boldsymbol{u}$ by using the first $d(\leq 36)$ elements of the feature vector. If the number of feature vectors converted into the same bit vector exceeds the threshold c, these feature vectors are rejected to be utilized for database construction. When the number is less than or equal to c, k hash functions are applied to obtain k hash values indicating the array positions. In the proposed method, we employ 8 hash functions produced by [6].

The database is constructed by applying the process mentioned above to all feature vectors. The amount of memory for storing feature vectors is less than that with the conventional method because of the property of the Bloomier filter.

5.2. Object Recognition

Let us explain the recognition process with the Bloomier filter. First, in order to decide whether the ith bit$(i = 1, 2, \ldots, n)$ of object ID is 0 or 1, both Bloom filters X_i and Y_i are applied to a query feature vector $\boldsymbol{q}$ for testing whether they contain it. If X_i contains q, the ith bit of object ID is 0. On the other hand, if $\boldsymbol{q}$ is contained in Y_i, the ith bit of object ID is 1. From this process applied to all bits of object ID, we can obtain the object ID for voting. Finally, the object having the maximum number of votes is regarded as the result. In the following, let us give a detailed explanation.

The most important step here is to determine the Bloom filters which contain a feature vector $\boldsymbol{p}$ close to $\boldsymbol{q}$. Recall that a feature vector is converted into a bit vector. Unfortunately, since variation of a query vector may change its bit vector, Bloom filters which contain $\boldsymbol{p}$ close to $\boldsymbol{q}$ might not be obtained. To solve this problem, the proposed method utilizes the error range e as a parameter for generating different bit vectors like the conventional method. In the proposed method, the limit b of the number of dimensions is also utilized, and the number of dimensions that satisfies the threshold e exceeds the limit b, those with larger bits are adopted up to the limit. Then, the proposed method determines whether the $i(i = 1, 2, \ldots, n)$th bit of object ID is 0 or 1 with the set of bit vectors calculated by applying the strategy of the expansion of bit vectors. If only $X_i(Y_i)$ contains a query bit vector, the ith bit of object ID is 0(1). In the case that a query bit vector is not contained in both X_i and Y_i, the proposed method discards the query vector. On the other hand, there is also a case that a query vector is contained in both X_i and Y_i. To handle this case, the proposed method votes for both objects whose ith bit of object ID is 0 and 1. However, unlimited application of this strategy increases the number of votes exponentially. To solve this problem, the proposed method utilizes the limit t of the number of bits for the application. After all query vectors extracted from a query image are evaluated by the above processing. the object having the maximum number of votes becomes the result of recognition.

In the proposed method, cascaded recognizers are also employed in order to improve the efficiency of the recognition process.

Figure 3. Examples of 55 3D objects.

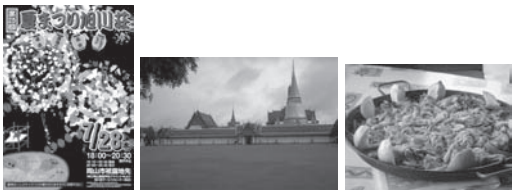


Figure 4. Examples of 5,000 planar objects.

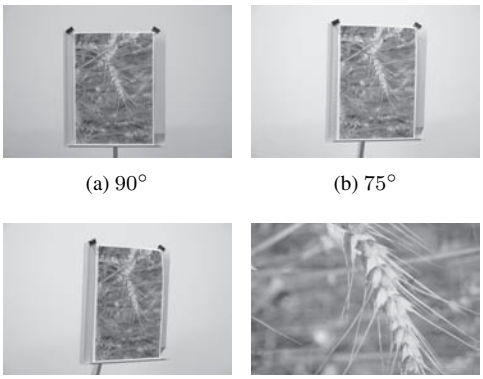
6. Experiments

We have evaluated the proposed method using two datasets: 55 3D objects and 5,000 planar objects.

6.1. Experimental Setting

The dataset of 55 3D objects was prepared by ourselves by taking images of 55 objects. Figure 3 shows some examples. The images were captured by rotating each object on a turn table in increments of 5 ° from frontal view and the above diagonal 15 °, 30 ° using the web camera whose resolution is 640×480. In these images, the images in increments of 10 °$(0\,°, 10\,°, \ldots, 350\,°)$ were utilized as model images. The rest were utilized as query images. The numbers of model and query images were both 108 per object. In total 1.2 million feature vectors were extracted from all model images.

Next, let us explain the dataset of 5,000 planar objects. As model images, we have prepared in total 5,000 images collected using Google image search (the left image of Fig. 4) and Flickr (the right image of Fig. 4), as well as images that had been available at the site of PCA-SIFT (the center of Fig. 4). Images were resized to make their



(a) 90° (b) 75°

(c) 60° (d) Part

Figure 5. Example of query images.

longest side less than 640 pixels. The average number of feature vectors extracted from an image was about 2,000. The query images were prepared as follows. First, we chose at random 500 images from the model images and printed out onto A4 paper. Then these were converted into images by taking their pictures in four different ways as shown in Fig. 5. The size of images were reduced to 512×341 pixels and then PCA-SIFT was applied to extract feature vectors. About 600 feature vectors were obtained on an average from an image.

The proposed method was compared to the conventional method proposed in [9]. Parameters for each method were set as follows. For both methods, we had four parameters, i.e., the limit b of the number of dimensions for expansion of the original bit vector, the length c of hash chains for deleting feature vectors, the number d of dimensions for indexing feature vectors, and the error range e of a feature value. The tested ranges were as follows. $b = 0, 1, \ldots, 10$, $c = 1, 2, \ldots, 10$, $d = 24, 28$, $e = 200$. We employed a computer with AMD Opteron8378 2.4GHz CPU and 128 GB RAM. In the following results, the processing time indicates the average time required for recognition of a single query image excluding the time for extracting feature vectors.

6.2. Experimental Results for 55 3D objects

First we evaluated the accuracy of recognition and the processing time required for recognition. We tested the proposed method by combining the number of bits $a (a = 8, 16, 24, 32)$ and the limit $t (t = 1, 2, 3, 4, 5)$ of the number of bits of object ID in addition to the parameters mentioned above. For the conventional method, the size of hash table was $H_{\text{size}} = 2^d$. Figures 6 and 7 show the experimental results with $d = 24$ and $d = 28$, respectively. The horizontal and the vertical axes indicate the recognition rate and the processing time, respectively. From the experimental

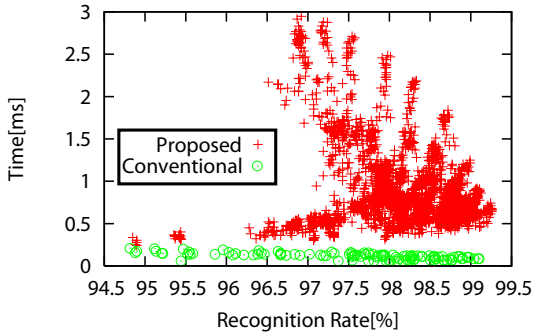


Figure 6. Recognition rate and processing time for 55 3D objects ($d = 24$).

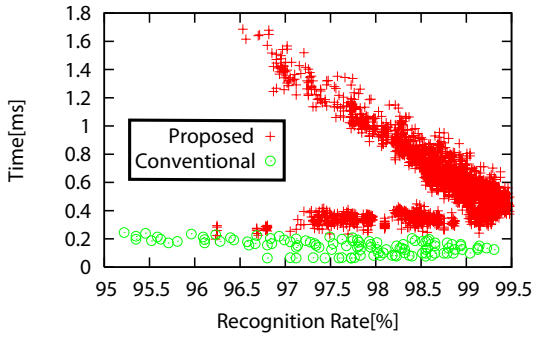


Figure 7. Recognition rate and precessing time for 55 3D objects ($d = 28$).

results, we confirmed that the longer processing time was necessary for the proposed method compared to the conventional method when the same recognition rate was achieved. This is because the number of times of accessing the hash tables to decide the object IDs with the proposed method is larger than that with the conventional method. Object IDs for voting are decided by accessing the hash table only once in the conventional method. On the other hand, it is necessary for the proposed method to access $2n$ hash tables in order to determine them.

Next we investigated the relation among the recognition rate, the processing time and the required memory by changing the parameter c and d which affect the amount of memory in both the proposed method and the conventional method. Table 1 shows the experimental results of the maximum recognition rate for each d with the proposed method $a = 1$ and those with the conventional method. From the experimental results, it is shown that the space efficiency with the proposed method was better than that with the conventional method when the similar recognition rates were obtained.

6.3. Experimental Results for 5,000 Planar Objects

Like the experiment for 55 3D objects, we evaluated the recognition rate and the processing time. The same tested

Table 1. Recognition rate, processing time and required memory for 55 3D objects.

Method	c	d	Other parameters	Recognition rate[%]	Required memory[MB]	Processing time[ms]
Proposed method	7	24	$b = 2, e = 200, t = 1$	99.21	72	0.50
	10	28	$b = 3, e = 200, t = 3$	99.45	72	0.50
Conventional method	3	24	$b = 2, e = 200$	99.11	231	0.09
	2	28	$b = 3, e = 200$	99.31	2199	0.12

Table 2. Recognition rate, processing time and required memory for 5,000 planar objects.

Method	c	d	Other parameters	Recognition rate[%]	Required memory[MB]	Processing time[ms]
Proposed method	10	24	$b = 4, e = 200, t = 1$	86.50	353	10.94
	2	28	$b = 6, e = 200, t = 3$	94.80	352	5.75
Conventional method	6	24	$b = 3, e = 200$	91.35	479	0.45
	1	28	$b = 6, e = 200$	94.90	2418	0.59

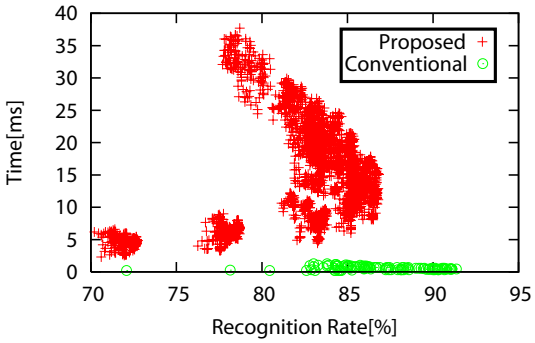


Figure 8. Recognition rate and processing time for 5,000 planar objects ($d = 24$).

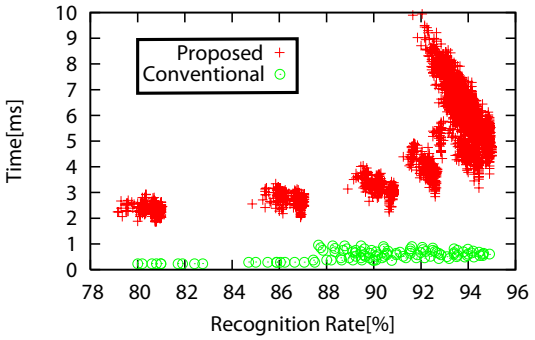


Figure 9. Recognition rate and processing time for 5,000 planar objects ($d = 28$).

ranges of the parameters and the same size of the hash table as the previous experiments were employed. Figures 8 and 9 show experimental results with $d = 24$ and $d = 28$, respectively. From the experimental results, we obtained the similar relation between them. In particular with respect to the results of $d = 24$, the proposed method was inferior to the conventional method in the accuracy of recognition. The reason is as follows. Bit vectors with $d = 24$ caused many false positives that deteriorated the recognition accuracy of the proposed method.

Next we researched the relation among the three measures like the experiment mentioned in the previous section. Table 2 shows the results. From the experimental results with $d = 28$, the proposed method excelled the conventional method in the space efficiency. The required memory with $d = 24$ of the proposed method came close to that with the conventional method. In the case of $d = 24$, the proposed method is inferior to the conventional method in the sense that the recognition rate is lower though a similar amount of memory is needed. In the case of $d = 28$, however, the advantage of the proposed method is clear. A similar recognition rate is achieved by using much less memory. Since the recognition rate is higher for both methods with $d = 28$, it can be said that the proposed method is superior to the conventional method with respect to the memory requirement, if higher recognition rates are necessary.

Finally, we evaluated the accuracy and the processing time when the required memory of the proposed method was almost the same as that of the conventional method. The proposed method with $d = 28, a = 8, 16$ was compared with the conventional method with $d = 24, H_{\text{size}} = 2^{24}$ and with $d = 28, H_{\text{size}} = 2^{24} - 1$. For these methods, the parameter $c = 1, 2, \ldots, 10$ were employed. Additionally, for the proposed method, the parameter $t = 1, 2, \ldots, 5$ were utilized. Figure 10 shows the experimental result. From the results, we achieved a higher accuracy using the proposed method. Consequently it can be said that the proposed method is successful to compress the representation of feature vectors. However, the longer processing time of the proposed method was required. Thus, the improvement of the proposed method is necessary to speed up the processing time.

Let us summarize the experimental results. We confirmed that the processing time of the proposed method is longer than that of the conventional method. If the recognition rate of the proposed method is almost the same as that of the conventional method, we achieve the smaller required memory using the proposed method. If the required memories of the proposed method and the conventional method are almost the same, the proposed method allows us the higher accuracy of recognition.

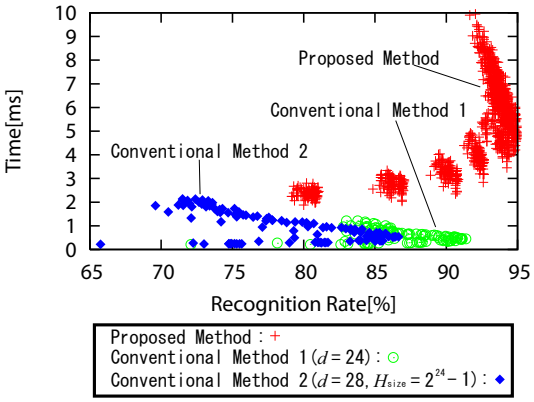


Figure 10. Experimental result with 3 methods.

7. Conclusion

In this paper we have proposed a new memory reduction method for specific object recognition by using the Bloomier filter. From experimental results for 55 3D objects, we achieved the recognition rate over 99% with about 1/3 of the required memory with the conventional method. Experimental results with 5,000 planar objects show that about 3/4 of the required memory with the conventional method allowed us the recognition rate over 94%. From these results, we confirmed the effectiveness of the proposed method.

Future work is to evaluate the proposed method with more objects, to logically analyze the required memory of the proposed method and to compare the required memory with the method of "bag of features".

Acknowledgment

This work was supported in part by the Grant-in-Aid for Scientific Research (B) (19300062) from Japan Society for the Promotion of Science (JSPS).

References

[1] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, 2008. 1

[2] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, 45(6):891–923, 1998. 1

[3] B. H. Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Commun. ACM, Vol. 13, No. 7*, pages 422–426, 1970. 3

[4] B. Chazelle, J. Kilian, R. Rubinfeld, and A. Tal. The Bloomier Filter: An Efficient Data Structure for Static Support Lookup Table. In *Proc. 15th Annual ACM-SIAM SODA*, pages 30–39, 2004. 2, 3

[5] F. Fraundorfer, H. Stewènius, and D. Nistèr. A Binning Scheme for Fast Hard Drive Based Image Search. *Proc. of CVPR2007*, pages 1–6, 2007. 2

[6] General Purpose Hash Function Algorithms. *http://www.partow.net/programming/hashfunctions/index.html#RSHashFunction*. 5

[7] K. Inoue, H. Miyake, and K. Kise. A memory reduction method for 3d object recognition based on selection of local features. *Proceedings of Third Korea-Japan Joint Workshop on Pattern Recognition (KJPR2008)*, pages 7–8, Nov. 2008. 2

[8] Y. Ke and R. Sukthankar. PCA-SIFT:A more distinctive representation for local image descriptors. In *Proc. of CVPR2004, Vol. 2*, pages 506–513, 2004. 2

[9] K. Kise, K. Noguchi, and M. Iwamura. Simple representation and approximate search of feature vectors for large-scale object recognition. *Proc. 18th British Machine Vision Conference (BMVC2007)*, 1:182–191, Sept. 2007. 1, 2, 6

[10] K. Kise, K. Noguchi, and M. Iwamura. Memory Efficient Recognition of Specific Objects with Local Features. *Proc. of the 19th International Conference of Pattern Recognition (ICPR2008) WeAT3.1*, 2008. 1, 2

[11] K. Kise, K. Noguchi, and M. Iwamura. Robust and Efficient Recognition of Low-Quality Images by Cascaded Recognizers with Massive Local Features. In *Proc. of 1st IEEE Workshop on Emergent Issues in Large Amount of Visual Data*, AW-18-12, 2009. 3

[12] D. Lowe. Local Feature View Clustering for 3D Object Recognition. In *Proc. of CVPR2001*, volume 1, pages 682–688, 2001. 1

[13] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. In *International Journal of Computer Vision, Vol. 60, No. 2*, pages 91–110, 2004. 1, 2

[14] D. Nistér and H. Stewénius. Scalable Recognition with a Vocabulary Tree. In *Proc. CVPR2006*, pages 775–781, 2006. 1, 2

[15] M. Özuysal, M. Calonder, V. Lepetit, and P. Fua. Fast Keypoint Recognition using Random Ferns. *IEEE Transctions on Pattern Analysis and Machine Intelligence*, 2009. 1

[16] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *Proc. of CVPR2008*, 2008. 1, 2

[17] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:530–535, 1997. 1

[18] J. Sivic and A. Zisserman. Video Google: A Text Retrieval Approach to Object Matching in Videos. In *Proc. of ICCV2003*, pages 1470–1477, 2003. 1

[19] Š. Obdržálek and J. Matas. Sub-linear Indexing for Large Scale Object Recognition. In *British Machine Vision Conference (BMVC)*, pages 1–10, 2005. 1, 2

[20] L. Yang, R. Jin, R. Sukthankar, and F. Jurie. Unifying Discriminative Visual Codebook Generation with Classifier Training for Object Category Recognition. In *Proc. of CVPR2008*, 2008. 1