Multiple Inverted Array Index Structure for Asymmetric Similarity Measure

Noburou Taniguchi, Hiroki Akama, and Masashi Yamamuro
NTT Information and Communication Systems Laboratories
Yokosuka, Kanagawa, Japan

Abstract

Due to the growing volume of digital image data in recent years, the demand and necessity for a system that can manage large image sets are rising. It is vital to create sophisticated retrieval functions in order to enhance the user's convenience. A lot of attention is now being placed on feature-based similarity retrieval which employs features extracted automatically from images.

We have been carrying out research on the image retrieval system called *ExSight* which makes use of feature-based similarity retrieval. In this paper, we propose a similarity measure for evaluating image shape similarity, and an access method for the proposed similarity measure. We also discuss our experimental observations.

1 Introduction

Because of the continued expansion in the volume of digital image data in recent years, a management system that can well handle large image collections is becoming an urgent necessity. Such a system needs a retrieval function that is both powerful and easy to use.

There are two major techniques for image retrieval. One is the conventional keyword matching technique and the other is the feature-based similarity retrieval (also called 'content-based retrieval') technique. The former is still useful in many cases but has problems - costly construction and perceptional inconsistency- because it needs so much human work. The latter, on the other hand, uses the features extracted automatically from images so it can avoid these problems.

We have been researching an image retrieval system, called *ExSight*, that uses feature-based similarity retrieval.

ExSight uses objects, which we call "sub-images", for retrieval; the shape of an object is important. We use the periphery of each object as a shape feature. For this feature, we introduce a new similarity evaluation method called "Asymmetric Similarity Measure" (ASM).

From the viewpoint of query processing, fast access methods for nearest-neighbor searches in multi-dimensional space are also important to provide quick response to users. Because it is difficult to use any multi-dimensional tree index based on ASM, we have developed a new access method that uses an index structure called "Multiple Inverted Arrays" (MIA).

2 System Overview

The overview of our system is presented in Fig. 1. It works as follows:

<**Preprocess**>

1. Every image in the given image collection is fragmented automatically into a number of sub-images. Each sub-image contains one object.
2. Feature vectors are extracted from each sub-image.

3. Images, sub-images, and feature vectors are stored in a database, where the connections between an image and its sub-images and the sub-images to their feature vectors are preserved.

<**Interactive process**>

4. A user gives an image as the key.
5. The system calculates the similarity between the key image and each sub-image in the database.
6. The system returns a certain number of images that contain the most similar sub-images.

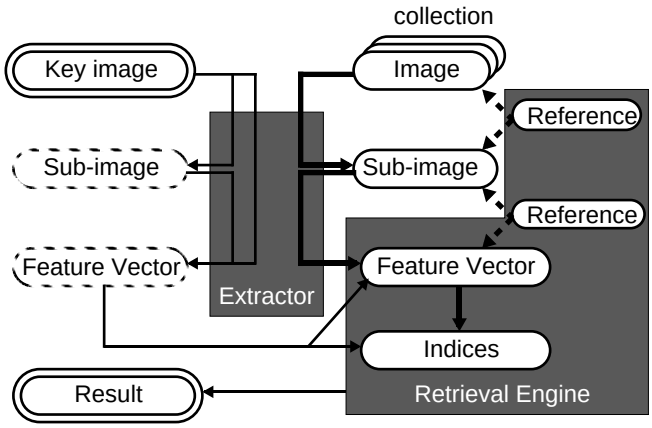

Figure 1: System overview

Figure 2 is a snapshot of the user-interface window. To run a query, a key sub-image must be given (it appears in the upper-left corner area) and the 'search' button pushed. Within a few seconds, images containing the most similar sub-images (each one is highlighted with a rectangle) appear in the lower area of the window. The user also can indicate how many similar sub-images are needed, and which features and similarity measures are to be used for similarity evaluation, by using pop-up windows.

Figure 2: Snapshot of user-interface window

3 Asymmetric Similarity Measure

3.1 Outershape feature

There are many features that can be used to represent image characteristics. For example, RGB color histogram, HSI color histogram, and L*a*b color histogram [1] are often used as color features.

There is even more variety in shape features. For example, 33 features are reported in [2]. We took a very simple approach to use the periphery of a sub-image as the shape feature. We call it "outershape". The outershape feature is derived as follows: (see also Fig. 3)

1. Find the center of gravity (G) of a sub-image (i.e. object). Draw the circumcircle whose center is G.
2. Measure the distance between the circle and the object's outer-edge along the radius direction for all 360 degrees.
3. Make the point with the shortest distance the start.
4. Reduce the dimensionality to a certain number (e.g., 24), by using the median method.

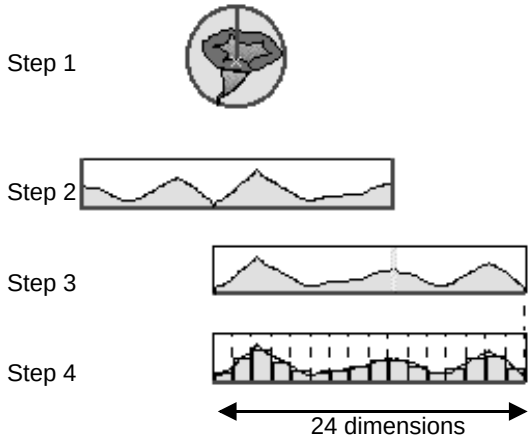

Figure 3: Generation of outer-shape vector

3.2 Asymmetric Similarity Measure

For the outershape feature, we developed a similarity evaluation method called "Asymmetric Similarity Measure" (ASM). The characteristic of this measure is that similarities between two feature vectors are asymmetric. For example, if a key vector $\vec{a}$ is given, $d(\vec{a},\vec{0})$ is larger than $d(\vec{a},2\vec{a})$, provided that $d(\vec{x},\vec{y})$ refers to $\vec{y}$'s dissimilarity to $\vec{x}$. For Euclidean or Manhattan distance measures, $d(\vec{a},\vec{0})$ and $d(\vec{a},2\vec{a})$ are equal. However, with human perception, we assume that the sensed difference between $(\vec{a},\vec{0})$ is larger than $(\vec{a},2\vec{a})$. Thus, we expect that the ASM better matches human perception than distance measures that are symmetric.

Mathematically, ASM is defined as follows:

$$d_a(\vec{x},\vec{y}) = \sum_{i=1}^{D} d_{a,i} \quad (1)$$

where

$$d_{a,i} = \begin{cases} c\cdot(x_i - y_i) & \text{if } x_i > y_i \\ y_i - x_i & \text{if } x_i \le y_i \end{cases}$$

$d_a(\vec{x},\vec{y})$ is thus the asymmetric dissimilarity between $\vec{x}$ and $\vec{y}$, where $\vec{x}$ and $\vec{y}$ are D-dimensional vectors, x_i and y_i are the coordinates of $\vec{x}$ and $\vec{y}$, and c is a constant.

3.3 Evaluation of ASM

To confirm our assumption, we performed an experiment. First, we chose 5 keys from a test data set of 11,691 sub-images (Fig. 4). The test data set was created from 1075 Japanese paintings. Next, we determined correct answers for each key sub-image in terms of shape. We did this as follows:

1. Ask three people, who don't know much about the technicalities of image retrieval, to select correct answers for every key sub-image in terms of shape from the test data set. The number of answers was not restricted.
2. Merge the results of step 1, and choose the sub-images that were indicated as the correct answer by more than one person. Define the result of this step as the final correct answer set.

We then calculated outershape similarity for every key against all the records (i.e. sub-images) in the data set, using ASM and the Manhattan distance measure (MDM), and obtained top-k (differs from 20 to 100) images as an answer set. Finally, we examined how many correct answers were contained in each answer set.

Table 1 shows the results of the experiment. For almost every case, retrievals using ASM yielded the same or better results than MDM. In particular, ASM retrievals using keys 3 and 5 were distinctly superior.

From Tbl. 1, ASM and MDM retrieval performance with keys 2, 4 and 5 seem fairly low. We think it is because the number of correct answers for these keys (31 for Key 2, 21 for Key 4, and 28 for Key 5) were too few to get satisfactory retrieval precision. However, we think that this does not negate the advantage of ASM for these queries.

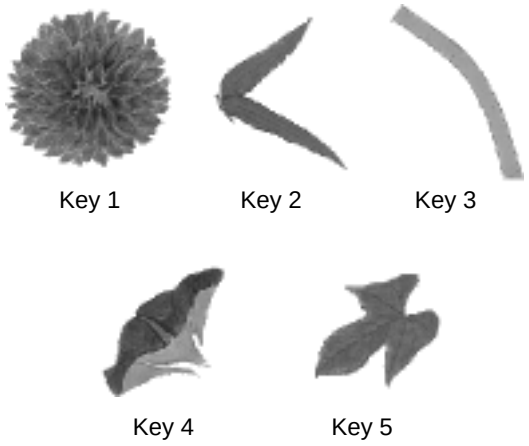

Figure 4: Key images

Key sub-image	1		2		3		4		5	
Total no. of correct	74		31		146		21		28	
Similarity measure	M	A	M	A	M	A	M	A	M	A
No. of returned	No. of correct in returned									
20	12	10	1	1	4	3	0	1	0	4
40	19	17	3	3	4	7	0	1	0	4
60	24	23	3	4	5	13	0	1	0	4
80	26	25	3	4	6	17	1	1	0	5
100	28	28	3	4	6	23	1	1	0	5

Table 1: Comparison of similarity measures

4 Multiple Inverted Array index structure

Though ASM is good for evaluating shape similarity, there was a problem with its access method. To eliminate this problem, we developed a novel access method using an index structure called ”Multiple Inverted Arrays” (MIA).

4.1 Related work

Multi-dimensional tree indices such as R-tree [3], k-d tree [4], and their successors are usually used for multi-dimensional nearest neighbor search to shorten the similarity retrieval time. However, it is difficult to use such multi-dimensional trees if ASM is used to evaluate similarity because the node configuration of those multi-dimensional tree indices is based on symmetry of distance calculations.

Several file structure indices have been developed for multi-dimensional spaces. The best known one is Grid File [5]. In Grid File, a given multi-dimensional space is split into a number of buckets, and a directory is built for providing direct access to each bucket. Multi-dimensional trees would better suit ASM if a specialized directory were created. However, as the number of dimensions increases (to more than ten [5]), the number of buckets increases rapidly. This makes it difficult to construct a directory because the directory structure becomes very complex. Another file structure index is called Multi-Dimensional Directory (MDD) [6], but it does not well support high-dimensional spaces.

Therefore, we have developed a novel access method using an MIA index structure.

4.2 Data structure

MIA is a structure composed of inverted arrays. An inverted array is an array assigned to each dimension. Each array is composed of a certain number of buckets, and each bucket is assigned to a same sized region of the dimension. The buckets hold information about the feature vectors that fall within their assigned region.

Imagine a feature space that has two or more dimensions. It can be thought of as finite for any given data set. All the records in the given data set are distributed in the space as feature vectors. Assume that each vector is a point in the space.

To simplify the explanation, we consider only one dimension. Geometrically, it can be assumed to be a finite straight line. The points (i.e., records in the data set) are distributed along the line. Divide the line equally into a fixed number of regions. Now, every record in the data set belongs to a region. Assume each region is a bucket and the set of buckets is an array. We call this an ”inverted array”. An inverted array is similar to an inverted list. To emphasize that it consists of buckets dividing the length of the data set distribution, we use the term ”inverted array”.

An MIA structure is a collection of these inverted arrays, where each dimension has one inverted array.

Because this data structure is completely independent of the similarity measures, it can be used with various similarity measures or search methods.

4.3 Construction (insertion/deletion) method

In generally, it is very simple to construct a multiple inverted array structure.

1. Create an inverted array for each dimension of the feature space.
2. Observe the value of the dimension of the feature vector corresponding to the target inverted array.
3. Calculate the bucket position of the vector from the observed value.
4. Store the vector’s reference (e.g. ID) into the appropriate bucket of the array.

Step 1 is executed by splitting the dimension axis into a certain number of same length regions and construct an array, each of whose buckets correspond to one unique region on the axis.

Steps 2-4 represent insertion. By repeating these steps for each dimension of each feature vector, we can build a multiple inverted array structure. Deletion is done by changing step 4 to

4’. Remove the vector’s reference from the bucket.

An example of the construction of inverted arrays for a data set with two records (i.e., feature vectors) in a 2-dimensional space is shown in Fig. 5. First, an empty array with 10 buckets is prepared for each dimension. In this example, the length of each bucket's region is set to 0.5 because the data is known to be in the range $[0, 5]$ on each dimension. The arrays are named X and Y, corresponding to the first and second dimension of the feature vectors, respectively. Next, a reference to each feature vector is placed in each array according to its value in the dimension corresponding to that array. For example, a reference to feature vector $A(2.2, 3.8)$ is placed in the sixth bucket of array X ($X[5]$) because the value of the item in the first dimension falls between 2 and 2.5. It is similarly placed in $Y[7]$. References to vector $B(4.8, 3.1)$ are inserted into the arrays in a similar manner. It is possible to insert more than one reference into an array bucket.

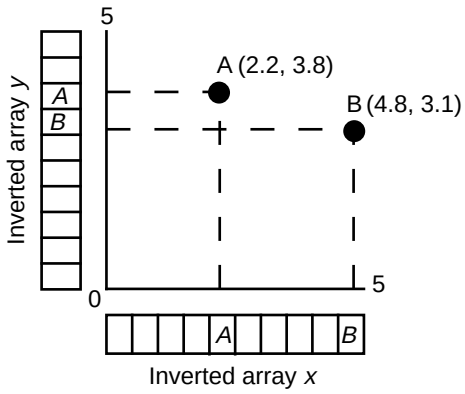

Figure 5: Construction of MIA structure

This example shows that inserting, deleting, or updating a record (i.e., a feature vector) only affects that a part of the data structure corresponding to the specified vector. Therefore, it is easy to change the data structure dynamically.

4.4 Retrieval methods

4.4.1 Basic concept

We can use many types of retrieval methods with a single MIA structure. The basic concept of retrieval with an MIA is as follows.

In a multi-dimensional feature space, consider a hyper-rectangle whose edges are defined by a hyper-plane orthogonal to a dimension axis. We can assume that there are hyper-rectangles that contain all the vectors requested for a given query. For example, if a user requested the 10 most similar images to a key, there must be hyper-rectangles containing the 10 most similar feature vectors to the key vector. If we can find the smallest such hyper-rectangle, we can get an answer set without costly similarity calculations. We call this smallest hyper-rectangle the "answer space". Retrieval using MIA is based on this concept.

Each edge of the hyper-rectangle defines a start or a end point of a region to be searched on a dimension. We call this region a "search scope". A search scope can be translated into a sequence of buckets of an inverted array for a dimension. We call this sequence of buckets "search scope buckets". A set of feature vectors contained in those buckets are then retrieved from each inverted array. Finally, these retrieved sets are combined to produce a final set of feature vectors.

With an MIA structure, it is easy to obtain records in a search scope by checking records in the search scope buckets. However, in general, the set of records thus obtained is not exactly the same as the records in the answer space. We thus call this set of records a "candidate set", and each record in the candidate set is called a "candidate". A candidate set can be shrunk by checking the search scope of another dimension (for example, obtaining a candidate

set from another dimension and taking the intersection of the two sets). Repeating this shrink operation yields a "final candidate set", whose elements are the records in the answer space.

In summary,

1. Given an answer space, get a candidate set from a dimension.
2. Shrink the set by checking the search scope of another dimension.
   Repeating this step an appropriate number of times yields a final candidate set.
3. Compute similarity of every record in the final candidate set for the given key and similarity measure.
4. Return the specified number of records in order of their similarity

This procedure can be roughly divided into the "Filtering" process composed of steps 1 and 2 and the "Calculation / Selection" process composed of steps 3 and 4. The latter is identical to brute-force operation, so the former is the key point of the retrieval method.

Fig. 6 shows graphically an example of the filtering process. First, candidate set 1, containing eight records, is obtained from dimension X. Next, candidate set 2 is obtained from dimension Y. The final candidate set is given as the intersection of these two sets. The similarity of the records in the final candidate set is computed with the specified similarity, and a specified number of records are returned as the result for the query.

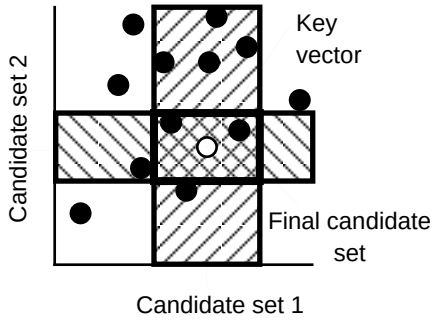

Figure 6: Filtering operation with MIA

This method can be easily extended to searching multiple feature spaces. Because each inverted array can be treated independently, the method can be easily applied to a combined space created from the spaces of the user-specified features.

4.4.2 Retrieval method using statistics-based approximation

A search based on the concept presented in the previous section has two big issues: 1) how to determine the answer space, and 2) how fast/accurate. These issues are highly correlated.

With perfect knowledge of the record distribution in a given data set and of the characteristics of the given similarity measure, the answer space for a given query can be perfectly estimated. However, this is not feasible in real situations because real data sets tend to have arbitrary distributions, making it difficult to get the perfect knowledge. The same can be said about the characteristic of the similarity measure.

Therefore we can only know an 'assumed' answer space, and because it is assumed, it may be inaccurate. For a certain types of applications, inaccuracy to some extent may be acceptable. We suppose that similar image retrieval is one such an application. Regarding the second issue, the greater the accuracy that is needed, the larger the assumed answer space would be, which would make the search slower. That is, there is a trade-off between speed and accuracy.

To optimize the speed/accuracy trade-off, we extended the method presented in Sec. 4.4.1 We did this by

- Using statistics of the record distribution

- Reducing the number of shrink operation iterations
- Reducing the number of candidates in the final candidate set

Using statistics of the given data set aims to make the answer space assumption more accurate. Reducing the number of repetitions and the number of candidates speeds up the search. In general, shrinking is not a cost-free operation, so it is desirable to minimize the number of iterations. However, if shrinkage is not enough, there will be too many candidates increasing the time taken to compute similarities. Optimizing the balance between the number of shrink iterations and the number of candidates can be achieved by setting an appropriate "stop condition" that determines when to stop shrink iterations while watching the status of the candidate set dynamically.

The method thus developed uses statistics-based approximation. It is based on the assumption that there are important and non-important dimensions in calculating similarity. For example, if a dimension has all the records in one bucket of its inverted array, the difference derived from this dimension would be too small to affect the order of similarity. The dimension is thus assumed as non-important. Non-important dimension can be neglected, so the feature space can be reconfigured to lower dimensionality.

There are two benefits in classifying the dimensions as either important or non-important.

First, the number of shrink iterations can be reduced. This is because we need consider only those dimensions estimated to be important.

Second, the number of candidates in a candidate set will be reduced. The number of important dimensions is correlated to the number of shrink iterations needed. In general, the more often shrinking is done, the fewer candidates there will be in the final candidate set. Thus, to obtain a specified number of records in the result, decreasing the number of shrink iterations decreases the number of candidates needed in each candidate set.

Presented below is a procedural description of our method.

1. Predict the number of important dimensions based on the statistics for each dimension. These statistics are calculated when the database is created.
2. Determine the minimum number of candidates in a candidate set, by using the result of step 1.
3. Determine the search scope of each dimension for the given key, that satisfy the result of step 2.
4. Evaluate each dimension, by using the statistics and the results of a dynamic observation of the search scope.
5. Determine the order of dimensions to search using the result of step 4.
6. Following the search order, get a candidate set from a dimension's inverted array.
7. Following the search order, get a candidate set from the next dimension's inverted array. Take the intersection of these two candidate sets, and make the intersection the new candidate set. Repeat this step until the given stop condition is satisfied.
8. Compute the similarities with all records in the final candidate set.
9. Return a specified number of records in order of their similarity.

We can illustrate this with an example. Imagine a feature space with 8 dimensions that contains 10,000 records. Besides creating an inverted array for each dimension, the standard deviation of each dimension is calculated and stored as a statistic indicator of the distribution. Assume that the user requested the top 10 similar images and indicated that the dimension whose standard deviation was more than $0.5 \times \sqrt{\frac{1}{12}}$ was 'important'. $\sqrt{\frac{1}{12}}$ is the standard deviation of the uniform distribution along one dimension, where the distribution region is $(0, 1)$. The reason for choosing this threshold is that the uniform distribution is thought to be good for showing distinct differences in Euclidean distance calculations, because the records are well distributed, i.e., not concentrated. We assume that there are 4 dimensions passing the threshold (step 1).

From the number of important dimensions, the minimum number of candidates (k') is determined as 1779 in this example, which is calculated from the formula,

$$k' = N \cdot \sqrt[d']{\frac{k}{N}} \tag{2}$$

where N : Total number of records in the data set (= 10000 in this example)
k : Number of similar sub-images requested (= 10)
d' : Number of important dimensions (= 4)

The reason for using this formula is that with uniform distributions, the number of candidates becomes k'/N times the number after each shrink operation (step 2). We then determine the search scopes of each dimension, so that each search scope contains 1779 records. We do this by checking the inverted array of the dimension, starting from the bucket of the key and extending the scope in both (plus/minus) directions. For use with ASM, it should be done asymmetrically in the plus and minus directions (step 3). Additionally, the standard deviation of each search scope is calculated (step 4). The dimensions are sorted based on the calculated values (step 5).

Assume that the order of the dimension is $(D_4, D_7, D_8, D_1, D_3, D_2, D_6, D_5)$, where "$D_x$" is the name of a dimension. The first candidate set (call it C_4) was obtained from dimension D_4 (step 6). The second candidate set (C_7) is then obtained from dimension D_7. The intersection of C_4 and C_7 is made the current candidate set (C_c). Next the third candidate set (C_8) is obtained from dimension D_8. The intersection of C_c and C_8 is made the new current candidate set (again called C_c). At this point, assume that the given stop condition, i.e., the number of candidates in C_c is less than $5 \times k$ (= 50) or $N/10$ (= 100), is satisfied (step 7). Then, with C_c as the final candidate set, calculate the similarity between the key and each record in C_c (step 8), and return k (= 10) records in order of their similarity (step 9).

The behavior of this method is determined by the major factors listed below.

- The statistics used in step 1.
- How the number of important dimensions is determined in step 1
- How the minimum number of candidates is determined in step 2
- How the search scope of each dimension is determined in step 3
- How the dimension search order is determined in steps 4 and 5
- The stop condition is used in step 7

We implemented this method and evaluated its performance in some experiments.

5 Evaluation of retrieval using Multiple Inverted Array structure

Some experiments were performed to examine the effectiveness of the access method described in the preceding section.

5.1 Experimental conditions

◇Environment

Machine: Sun SparcStation IPX, 64MB Memory
OS: SunOS 4.1.4
Compiler: GNU C compiler with -O2 optimization

◇Data set We prepared 50,000 sub-images extracted from 218 digital photographs. The size of the sub-images ranged from 100 to 126,564 pixels. From each sub-image we extracted an outershape feature vector of 24 dimensions. These vectors were the target of our retrieval experiments. From this 50,000-vector set, 1,000-, 3,000-, 5,000-, and 10,000-vector sets were derived.

◇Queries

- Keys ... For each query, one sub-image from the given data set was selected as the key. 200 queries were made using different key sub-images selected from the smallest (1000-sub-image) data set. All queries requested the 11-nearest neighbour sub-images (They were actually 10-nearest neighbour queries because the key sub-image must be contained in the results).
- Similarity measure ... ASM was used to calculate dissimilarity between a key and a record. In the experiments, c in Eq. 1 was set to 2.
- Performance measure ... Each query was executed 50 times and the average CPU time measured by the *clock()* function was recorded to evaluate retrieval time becase our system is designed to be work entirely in memory.

◇**Factors in MIA** The factors listed in Sec. 4.4.2 etc. were set as follows:

- Number of buckets ... the number of buckets of every inverted array was set to 4096.
- Statistic indicator of data distribution ... standard deviation
- Number of important dimensions ... set to 8 (constant) to eliminate the effect of difference in distributions between data sets of different sizes.
- Minimum number of candidates ... calculated using Eq. 2 (given in the example in Sec. 4.4.2). The calculated values for each data set were:

Data set size	Min. no. of candidates
1000	471
3000	1178
5000	1803
10000	3212
50000	12284

- Search scope ... because we used Eq. 1 as the similarity measure with c = 2, we decided to make the search scope in the plus direction twice that in the minus direction. Fig. 7 shows simply how the search scope is configured.

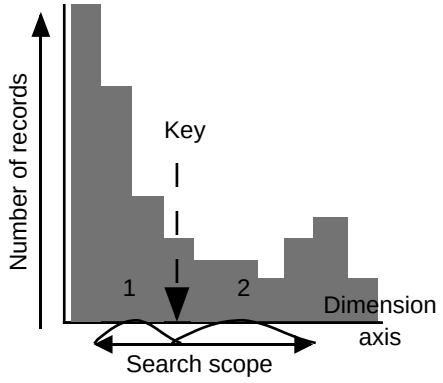

Figure 7: Search scope for asymmetric similarity measure

- Order of dimensions to search ... determined by the variance of the record distribution in the search scope of each dimension, in decreasing order of the value.
- Stop condition ... the dynamic stop condition was not tested in order to observe how the method would behave.

5.2 Experimental results and discussion

Figure 8 shows the total retrieval time (average of 200 queries) versus the number of shrink iterations for the 50,000-record data set. The calculation/selection time decreased rapidly with the number of shrink iterations. Because this effect exceeded the increase in the filtering time, the total time decreased monotonously. A decrease like this was seen for the data sets of other sizes.

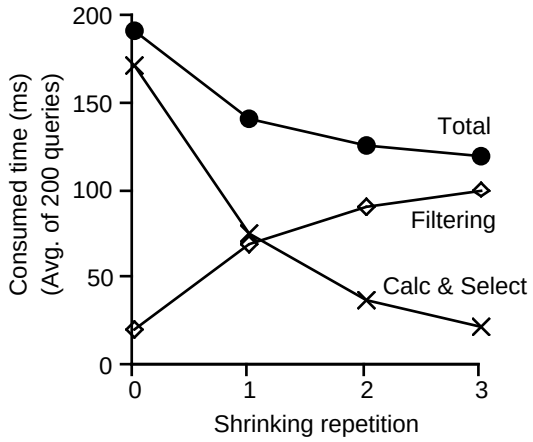

Figure 8: Time consumed by each process

The number of correct answers remaining in the final candidate set (10 is the best; average of 200 queries) versus the number of shrink repetitions is shown in Fig. 9. We use the result at no shrinkage (i.e. the first candidate set is the final candidate set), which yields the highest accuracy of the proposed method. Even for the largest (50,000-record) data set, the number of correct answers was 7.45.

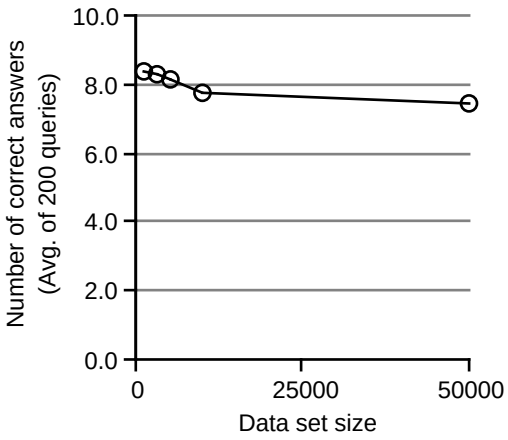

Figure 9: Accuracy of answer set

The relations between the size of the data set and the total retrieval time for the proposed and brute-force methods are compared in Fig. 10. Again we use the result at no shrinkage, which yields the slowest speed of the proposed

method. As shown, for data sets larger than 3,000, the proposed method is faster than the brute-force one. For the 50,000-record data set, the retrieval time of our proposed method is about 60% shorter than that of the brute-force method.

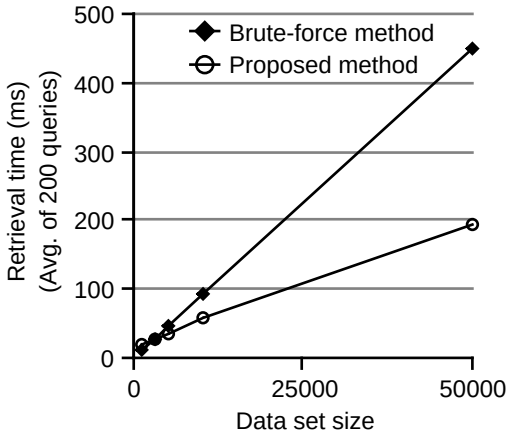

Figure 10: Retrieval time

6 Conclusion

We have proposed a similarity measure called ”Asymmetric Similarity Measure” for evaluating shape similarity. Because it is difficult to use any multi-dimensional tree index such as R-tree or k-d tree with this measure, we developed an access method that uses the Multiple Inverted Array structure for fast and flexible nearest-neighbor searching in multi-dimensional spaces. Experimental observations showed that it has good potential for use with ASM. The search speed is significantly faster than the brute-force method, while still giving acceptable accuracy for shape similarity retrieval.

We plan to improve the proposed method to obtain better performance. Additionally, for a wider variety of retrieval types, we plan to develop access methods using this data structure. Our goal is to provide the *ExSight* system with a flexible mechanism that chooses the optimal access method for a given query dynamically. We think that this will be important for a multimedia information retrieval system because in the future there will be more types of retrieval given the variety in multimedia information.

7 Acknowledgement

We would like to thank the relatives of the late Japanese painting artist Mr Gakuryo Nakamura and NHK Educational Corporation for granting their permission to use the artist’s drawings.

References

[1] Wyszecki G. and Stiles W.S. Colour science : concepts and methods, quantitative data and formulae, The Wiley series in pure and applied optics. John Wiley & Sons, Inc., New York, 2nd edition 1982

[2] Tanabe K. and Ohya J. A Study of Similarity Measure for Image Shape Retrieval. ICICE PRU88-68:65-72(In Japanese)

[3] Guttman A. R-trees: a dynamic index structure for spatial searching. Proceedings of the ACM SIGMOD International Conference on the Management of Data:47-57, 1984

[4] Bentley J. L. Multidimensional binary search trees used for associative searching. Communications of the ACM 18(9):509-517, 1975

[5] Nievergelt J. Hinterberger H. and Sevcik K. C. The grid file: an adaptable symmetric multikey file structure. ACM Transactions on Database Systems 9(1):38-71, 1984.

[6] Liou J. H. and Yao S. B. Multi-Dimensional Clustering for Data Base Organizations Inform. Systems Vol. 2:178-198, 1977.