pISSN 1976-913X
eISSN 2092-805X

J Inf Process Syst, Vol.9, No.2, June 2013 http://dx.doi.org/10.3745/JIPS.2013.9.2.315

An Improvement Video Search Method for VP-Tree by using a Trigonometric Inequality

Samuel Sangkon Lee*, Masami Shishibori** and Chia Y. Han***

Abstract—This paper presents an approach for improving the use of VP-tree in video indexing and searching. A vantage-point tree or VP-tree is one of the metric space-based indexing methods used in multimedia database searches and data retrieval. Instead of relying on the Euclidean distance as a measure of search space, the proposed approach focuses on the trigonometric inequality for compressing the search range, which thus, improves the search performance. A test result of using 10,000 video files shows that this method reduced the search time by 5-12%, as compared to the existing method that uses the AESA algorithm.

Keywords—Vantage Point; VP-Tree; Trigonometric Inequality; Search Algorithm; Range Search; Nearest Neighbor Search; AESA algorithm; Multimedia Database

1. INTRODUCTION

Thanks to both the low pricing and large capacity of primary or secondary storage units nowadays, it has become possible to store a huge amount of multimedia data such as text, video, image, music, etc. even on a personal computer. As a result, there has been a demand for a technology that would search for data quickly and accurately. To enhance search efficiency, features from the stored data are extracted and organized into indexes [1], which are accessed during data search and retrieval. So, search efficiency is greatly influenced by the index-organizing method that is used.

In general, the features of multimedia data are expressed in vectors. A search is performed based on the similarity of features or on the distance between features in vectors [2, 3]. There are many schemes for the multidimensional indexing of feature vectors, which include the following methods: R-tree [4], R*-tree [5], SS-tree [6], SR-tree [7], X-tree [8], and VA-FILE [9]. For all of these methods, only the Euclidean distance is normally used as a metric for searching. While others, such as in [10], a quadratic form distance measure, which considers correlations among multidimensional data, is used in addition to the Euclidean distance. The edited distance, which calculates similarities among strings, and the Earth Mover's Distance, which calculates the structural similarities among videos, are used [11]. In essence, it is important to consider metric dis-

Manuscript received May 7, 2012; first revision August 29, 2012; accepted Fubruary 22, 2013.
Corresponding Author: Samuel Sangkon Lee
* Dept. of Computer Science and Engineering, Jeonju University, Jeonju, 560-759, South Korea (samuel@jj.ac.kr)
** Institute of Technology and Science, The University of Tokushima, Tokushima, 770-8506, Japan (bori@is.tokushima-u.ac.jp)
*** Dept. of Computer Science, College of Engineering and Applied Science, University of Cincinnati, Cincinnati, 45221, OH, USA (han@ucmail.uc.edu)

tance information when an index is prepared based on the coordinates in a multidimensional index, and a metric-based indexing would only be possible if a metric axiom were formed. By having metric axioms, indexing is made simpler, and metrics besides the Euclidean distance become applicable.

Generally speaking, the metric space index is represented as a hierarchical tree structure. By recursively dividing the space (data set) based on metric information, the search space is reduced. Based on the space division scheme that was used, different tree structures are formed. The M-tree [12], VP-tree [13, 14], MVP-tree [15] and MI-tree [16] have been proposed. The M-tree consists of a bottom-up index tree where at the division of space the search efficiency decreases in the amount of common areas in the divided spaces. But, the VP-tree divides the search space in a top-down approach. After choosing a good reference point, which is called the 'vantage point' (vp), the distance is calculated from this reference point to others in the search. It becomes efficient if the search range under consideration can approach a leaf object, which is linked to the leaf node, along the appropriate node from the root node. However, it slows down the search speed by increasing the distance calculation frequency in the lead node.

To reduce the distance calculation frequency an improvement in the search algorithm in the leaf node of VP-tree is sought. By applying the trigonometric inequality to several spaces (i.e., given the vectors of two sides, x and y, in a norm vector space, the inequality shall be as follows: $\|\mathrm{x}+\mathrm{y}\| \leq \|\mathrm{x}\| + \|\mathrm{y}\|$). In addition, provided that x, y and z are in the metric space, M, and the distance between them is set to d, the inequality shall be as follows: $d(\mathrm{x},\mathrm{z}) \leq d(\mathrm{x},\mathrm{y}) + d(\mathrm{y},\mathrm{z})$. This work focuses on the fact that as the distance between the base point of the trigonometric inequality and a query object gets closer, the analysis range for the search is narrowed. Therefore, the distance calculation frequency can be reduced by significantly narrowing down the search range after using the nearest node on the query object in terms of the base point of trigonometric inequality. Also, instead of using a vp as the base of inequality, the method uses a base in the leaf node of the VP-tree.

However, the nearest node cannot be determined in advance. Therefore, the object that is nearest to the query object in the search list is assumed to be the nearest virtual node. Then, the compression technology on the range based on the nearest node can be realized. In addition, in order to use the trigonometric inequality using the base point as the nearest virtual node, it is necessary to be aware of the distance between the nearest node and all object distances in the leaf node. Here, because the nearest virtual node cannot be limited in advance, it is necessary to obtain all of the inter-object distances. Therefore, a distance list file that calculates the distance between objects at indexing is constructed. However, constructing a huge distance list file in each object can reduce the size of file that can be read through the memory. In the search, the Approximating and Eliminating Search Algorithm (AESA) [17] compresses a range using the distance list file against all objects. Unlike AESA, which targets all of the objects, and where file reading frequency substantially increases, the proposed method only targets the objects in a leaf node. By using a VP-tree, fewer objects in a leaf node are targeted, and files are read provided that the nearest virtual node (base point) is updated. Therefore, the file access frequency can be significantly reduced, thus, boosting the search efficiency.

The Locality-Sensitive Hashing (LSH) algorithm is relevant to our research method. The LSH is an algorithm for solving the (approximate/exact) Near Neighbor Search in high dimensional spaces. We were able to find pointers to the newest LSH algorithm in Euclidean spaces, as well as the description of the E2LSH package, which is an implementation of this new algorithm for

the Euclidean space. The earlier algorithm for Euclidean space is [23], which is a good introduction to the LSH, and the description of an affair is in [24]. For the LSH algorithm, [25] is a good survey of the LSH, and the most recent algorithm is [26]. However, this LSH algorithm is the approximate Near Neighbor Search in high dimensional spaces.

This paper is structured as follows: in Section 2, the construction and search algorithms of the VP-tree are described, and a method to compress the range of the leaf node is introduced. In Section 3, the method of an improved search algorithm for the leaf node is stated. In Section 4, the tests and evaluations that were based on the improvement method are explained. In Section 5, the conclusion and future prospects are described.

2. AN IMPROVED METHOD FOR THE VP-TREE

2.1 Construction Algorithm

In this section, the construction algorithm of the VP-tree is described. Provided that an indexing is performed against the data set S, which consists of N data, each tree node selects the vantage point (hereinafter, vp) according to the random algorithm as shown below.

• VP-tree Construction Algorithm
Step 1. Randomly select the virtual vp from a data set;
Step 2. Calculate the distance from the virtual vp to the (n-1)th object;
Step 3. Estimate the median and variance of the distance;
Step 4. Repeat the said processes (Steps 1-3) and decide on the point in which the variance reaches the maximum level as vp.

The median of the distance against all data from the vp to S is μ. Provided that $d(p,q)$ is the distance between p and q, the data set (S) shall be divided into S_1 and S_2 as follows:

$$S_1 = \{s \in S \mid d(s, vp) < \mu\}$$
$$S_2 = \{s \in S \mid d(s, vp) \geq \mu\}$$

Then, an index is created by recursively applying the said division mechanism to S_1 and S_2. All subsets, such as S_1 and S_2, are equivalent to a single node of the VP-tree. In addition, several objects can be stored in a leaf node.

2.2 Search Algorithm

In this section, the search algorithms of the range search of the VP-tree and the k-nearest neighbor search are described. The range search is a search method that is used to get a set of objects that exist within a radius from the center after designating the query objects and search range (radius). The *k*-nearest neighbor search is a search method that is used for getting a set of objects in order of closeness in terms of distance by designating query objects and k (search frequency). In this paper, the *k*-nearest neighboring search has been used. Because this method is based on the range search algorithm, both search methods will be described.

In the range search, first, the distance between the leaf object, which is linked to a leaf along

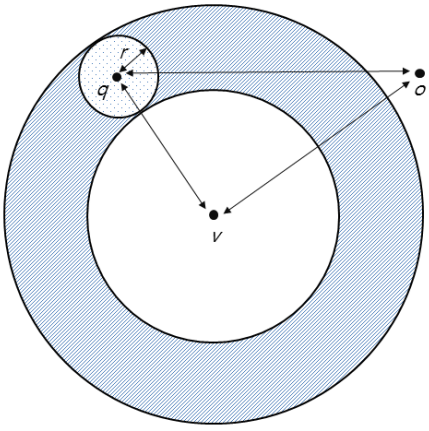


Fig. 1. Compression of the search range with vp as the base point

the appropriate node from the root node, and the query object is calculated, and objects that exist in the search range are obtained. On the other hand, in the k-nearest neighbor search, objects are added to the search list along the root after setting the search radius to an infinite level. If the number of searches on the search list exceeds the designated search number, the search objects with a maximum distance are deleted from the list. In addition, the search radius is narrowed by repeating the search with the maximum distance of search list as search radius. Ultimately, the designated number can be obtained as the search result.

2.3 Search Range Compression in the Leaf Node

As described in Section 2.2, in a conventional VP-tree the distance with the query object was calculated by approaching all of the objects in a leaf node during the search. As an improvement plan a search range reduction method [18], which uses a trigonometric inequality in the search of each object in a leaf node has been proposed, as shown below.

The distance between the vp object and each leaf object is preserved in a distance list in a leaf node. In terms of the distance between the vp object and each leaf object, the trigonometric inequality-based distance calculation frequency can be reduced. Let the query object be q; the radius, r; the vp object in a leaf node, v; and a leaf object linked to the leaf node, o; as such the search algorithm can be then summarized as follows:

• Leaf Node-based Search Algorithm w/the K-nearest Neighbor Search

Input: q, r, L
Output: L,
Search Leaf (q, r, L)
{
for each (all leaf objects) {
if ($|d(v,o) - d(v,q)| \leq r$) {
if ($|d\ (o,q)| \leq r$)

{as the maximum distance value by adding o to L}
}
}
}

Here, q: query object, r: search radius, o: leaf object, v: vp object and L: search result.

Both $d(v,o)$ and r in theorem ① are already known at the start of searching for the leaf node, and $d(v,q)$ can get each leaf node once. Therefore, it is possible to find out if a lead object exists in the search range without calculating the distance between each leaf object. Hence, the frequency of distance access to the distance calculation and leaf object can be reduced. A method to reduce the candidate of the leaf node (analysis candidate) is shown in Fig. 1, and the *k*-nearest neighbor search algorithm is proposed. In the figure, the parts other than the diagonal lines represent that theorem ① is proven. The distance calculation on the objects here can be omitted. Meanwhile, the diagonal line sections are the area in which theorem ① is NOT proven. Therefore, the distance of the objects that exist here should be calculated.

- Theorem ①
  If $d(v,o) - d(v,q) > r$, the leaf object (o) does NOT exist in the search range.

[Proof] Based on the trigonometric inequality $(d(v,q) + d(q,o) \geq d(v,o))$, $d(v,o) + d(v,q) > r$ becomes $d(v,o) - d(v,q) > r$. In other words, 'o' does NOT exist in the search range. The inequality $(-d(v,o) + d(v,q) > r)$ also becomes $d(q,o) > r$. Therefore, the said theorem ① is proved.

In addition, the search range can be compressed using all of the vp objects that exist on the path from the root node to the leaf node, as well as compressing all of the vp objects in the leaf node. In this case, the distance of all the leaf objects linked to the leaf node and all vp objects, which exist on the path from the root node to the leaf node, should be saved before the leaf node.

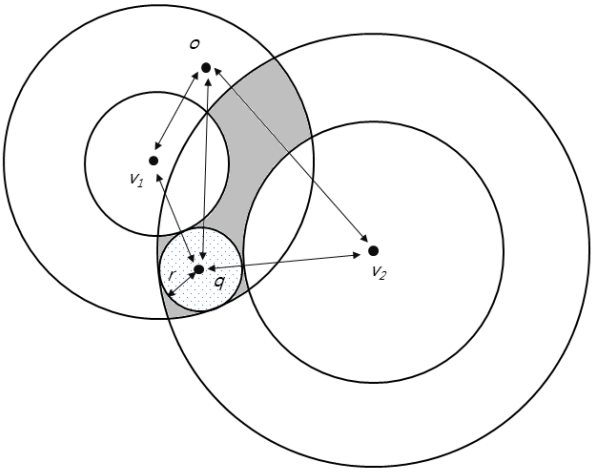


Fig. 2. Compression of the search range when vp is taken as the base point

In terms of the distance between multiple vp objects and each leaf object, using a trigonometric inequality can reduce the distance calculation frequency. If the query object = q, the search radius = r, the ' k ' vp objects on the path from the root node to the leaf node = v_i $(i = 1, 2, 3, \ldots, k)$, and the leaf object linked to the leaf node = o, the analysis candidates can be reduced as shown in Fig. 2. This algorithm makes it possible to further narrow down analysis candidates using several vp objects.

3. SEARCH RANGE COMPRESSION USING THE NEAREST NODE

When the vp is used as the base point of a trigonometric inequality, The less diagonal line section which theorem ① is not satisfied, the better compression effect of the search range. The radius of the circumference can be obtained by adding the radius (r) to the distance from vp to the query object (q). Here, because r is a fixed value against the search request, the compression of the search range becomes advantageous as the distance between vp and q decreases. In fact, the object that is nearest to q is the nearest node. In other words, the search range can be reduced by eliminating the parts in which theorem ① is not proven from the search target by using the object (the nearest node to q instead to vp) as the base point of a trigonometric inequality. Therefore, this paper proposes a search range compression method using a trigonometric inequality that takes the nearest node as the base point.

If the query object = q, the search radius = r, the nearest node that is the nearest to the query object in the search list = o_1, and the leaf object linked to the leaf node = o, the following theorem ② is proven.

● Theorem ② If $d(o_1, o) - d(o_1, q) > r$, the leaf object ($o$) does NOT exist in the search range.
[Proof] $d(o_1, o) - d(o_1, q) > r$ becomes $d(q, o) > r$ because of the trigonometric inequality ($d(o_1, q) + d(q, o) \geq d(o_1, o)$). It is confirmed that o does NOT exist in the search range.

Here, if $d(o_1, o)$ and $d(o_1, q)$ are already known, it can be confirmed that an object does not exist in the search range without calculating the distance with each leaf object. This process is illustrated in Fig. 3. Unless a leaf object exists in the diagonal line sections as shown in this figure, a calculation of the distance with the query object can be omitted. In fact, the search algorithm of the leaf node is proposed as described below.

• The Leaf Node-based Search Algorithm by the Proposed Method

```
Input: q, r, L
Output: L,
Search_Leaf(q, r, L)
{
  foreach o (all leaf objects) {
        if ( ((d(o1, q) + r) < d(o1, o))) {
            if (d(o, q) ≤ r)
```

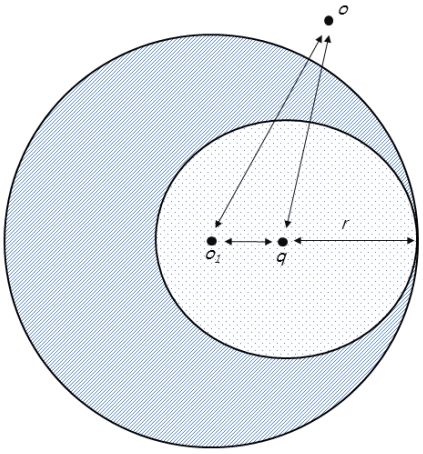


Fig. 3. Compression of the search range that takes the nearest node as the base

{r as the maximum distance value by adding o to L}
}
}
}

Here, q: query object, r: search radius, o: leaf object, o_1: object that is nearest to q in the search list, and L: search list.

In terms of $d(o_1, o)$ in theorem ②, value is given if a list of the distance between the nearest node and an object in the leaf node exists. However, because which object is the nearest node to q cannot be confirmed in advance, o_1 does not consider all objects in reality. Here, it is necessary to make a file that calculates the distance between a leaf object that is linked to the leaf node and other objects. However, it is not a good idea to construct this huge file in the memory. In this paper, a file set has been constructed with each object ID as a file name just like the Akama method [19]. However, to avoid the restriction on the number of files in a directory, the lower three digits of the ID were set to a file name, while the upper three digits were set to a directory name. For example, the maximum number of files in a single directory is set to 1,000.

In addition, the nearest node cannot be decided in advance in $d(o_1, q)$, or for the distance between the query object (q) and the nearest node (o_1), as well. As proposed in the leaf node-based search algorithm as shown above, the object that is nearest to the query object (q) in the search list (o_1) is presumed to be the nearest node (o_1). In addition, the nearest node is updated whenever an object in the search range is newly discovered.

4. RESULTS

4.1 Test Method

Experiments have been conducted using our method to create a VP-tree for a video indexing

search. The improvement method mentioned in this paper as applied to the VP-tree. Then, a test on the similar video search was performed. For the test, a 2 GB computer (Pentium D 3.2 GHz) with Linux was used. A total of 10,000 videos were prepared, and their features were collected using the HSI histogram. The HSI histogram is a color histogram that consists of hue, saturation, and intensity. In terms of dimensions, four types were used; 12 (4×3), 24 (8×3), 48 (16×3), and 96 (32×3). In this case, 4, 8, 16, and 32 mean the corresponding vector, and 3 means HIS. In addition, 10,000 video objects whose features were already collected were indexed with the VP-tree. The vp at indexing has been calculated based on random data (up to 100 data sets every time). From the 1,000 input videos that were not used for indexing, the distance calculation frequency and the average per video (CPU time) were calculated using the *k*-nearest neighbor search.

In addition, the quadratic form distance has been used as a metric scale among video objects. The quadratic form distance between histogram H and histogram K can be expressed as follows:

$$\mathrm{D_q(H,K)} = \sqrt{(\mathrm{h}-\mathrm{k})^{\mathrm{T}}\mathrm{A}(\mathrm{h}-\mathrm{k})}$$

$$= \sqrt{\sum_{\mathrm{i}=1}^{\mathrm{N}}\sum_{\mathrm{j}=1}^{\mathrm{N}}\mathrm{a_{ij}}(\mathrm{h_i}-\mathrm{k_i})\left(\mathrm{h_j}-\mathrm{k_j}\right)} \quad (1)$$

Here, $\mathrm{A}=[\mathrm{a_{ij}}]$ is a matrix which represents similarity between ith side and jth side. In this paper, a determinant [18] as shown in the Equation (2) below has been used:

$$\mathrm{a_{ij}} = 1 - \frac{\mathrm{d(i,j)}}{\mathrm{d_{max}}} \quad (2)$$

In the Equation (2) above, $d(i,j)$ is the distance of the color space between the ith side and the jth side, while d_{max} refers to the maximum value of $d(i,j)$. In addition, this paper has proposed a search range compression method that uses the nearest node of the VP-tree. However, AESA [19] is also available as a search range compression algorithm that uses the nearest node. In AESA, the search range is decided after preparing and using a file that has calculated the distance between objects without constructing an index tree such as a VP-tree. In terms of search range compression, the nearest node is compressed using a trigonometric inequality just like the method proposed in this paper. AESA has an algorithm that is very similar to the method mentioned in this paper. The difference between the two methods is described in the following section. This paper has used AESA for making a comparison with the search range compression improvement method, which uses the nearest node in the VP-tree to evaluate the effect of the range compression. The specific algorithm of AESA is stated in the next section.

4.2 AESA

AESA can predict the nearest node (s) by using Equation (3) as shown below:

$$\mathrm{s} = \mathrm{argmin}_{\forall \mathrm{o}\epsilon \mathrm{P}-\mathrm{E}} \sum_{\forall \mathrm{o}\epsilon \mathrm{U}} |\mathrm{d(o,u)} - \mathrm{d(q,u)}| \quad (3)$$

Here, $P =$ a set of all objects, E = a set of objects that have been removed after not having

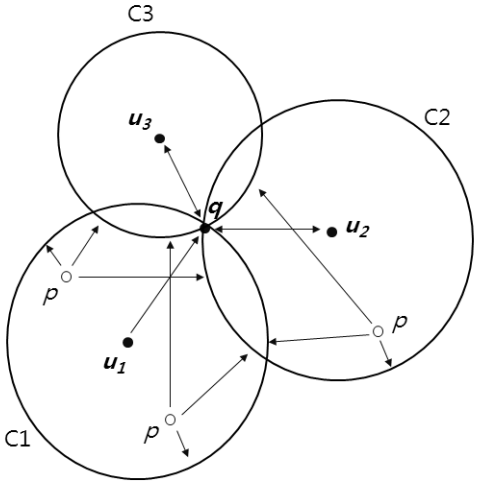


Fig. 4. The base point (s) selection method

been chosen as search targets, U = a set of objects that have been chosen as the nearest node (s) in the past, and q = query object. This equation is described in Fig. 4.

If the circles C1, C2, and C3, which have $d(q,u)$ as their radius are assumed, $\sum|d(o,u) - d(q,u)|$ is the sum of the distance between each circle and o. The lowest sum of the distance (o) is chosen as s. In other words, o, which is expected to be the nearest to q when viewed from each u, is calculated. Therefore, the point that is the nearest to q without calculating the distance between the query object and all objects in person can be estimated. The distance between s, which has been chosen based on the said method, and the query object (q) is calculated. If it is likely that the distance would be included in the search list, it should be added to the search list. In addition, if it is nearer distance between the nearest node in the past (the No.1 candidate in the search list) and q, s is updated into the nearest node. Lastly, the compression of search range is continued using a trigonometric inequality. Just like theorem ① and theorem ②, the following theorem ③ is proven.

- Theorem ③
  If $d(s,o) - d(s,q) > r$, the leaf object (p) does NOT exist in the search range.

Here, the radius r refers to the distance between the query object (q) and the object positioned at the bottom of the search list. In addition, $d(s,o)$ can be read and updated from the distance list file among the objects that have been prepared in advance. $d(s,q)$ is calculated when the nearest node is updated. Therefore, all of the metric information used in theorem ③ is already known when being applied to a trigonometric inequality. It can be said that an object is not included in the search range without calculating the distance between each object and q in person. In other words, the unnecessary calculation processes can be omitted, and the distance calculation frequency can be reduced. By repeating these processes until all objects are removed, the search results per case can be finally obtained.

4.3 Test Results

A test on the *k*-nearest neighbor search has been performed using an improvement method for the compression of each search range. The explanatory notes in each graph are test results that have been obtained after performing the four methods mentioned below. According to a conventional research method (VP-tree), it is more effective to use several vp objects than to use a single vp object. Therefore, the VP-tree has been chosen in this paper.

- vp_all: search range compression method using several vp objects
- vp_nn: search range compression method using the nearest node
- vp_all_nn: search range compression method that combines both vp_all and vp_nn
- AESA: AESA-based search range compression method

First of all, the test results on the metric calculation frequency in each dimension are shown in the figures below (Fig. 5 to Fig. 8). These figures are the results of 12, 24, 48, and 96-dimension data. The horizontal axis (k) and vertical axis (calc_num) refer to search frequency and metric calculation frequency, respectively. As shown in the VP-tree in the figure, metric calculation frequency decreased in order of vp_all, vp_nn and vp_all_num. Meanwhile, AESA is lower than the VP-tree in terms of metric calculation frequency (See the curves labeled 1, 2, 3, 4, respectively).

Then, the test results on the search time in each dimension are shown in the Fig. 9 to Fig. 12. These figures are the test results of 12, 24, 48, and 96-dimension data. Fig. 13 shows the data search execution time in all dimensions (12 to 96) using AESA. In these figures, the horizontal (k) and vertical (cpu-time) axes are expressed in the second of time. The cpu-time of the VP-tree of the figures (Fig. 9 to Fig. 12) decreases in the order of vp_all, vp_nn and vp_all_nn. In particular,

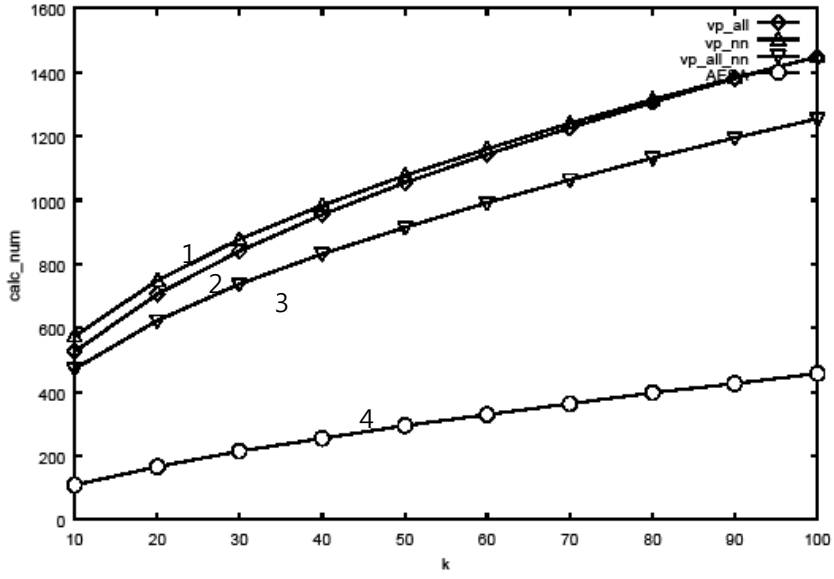


Fig. 5. Metric calculation frequency on 12-dimension data

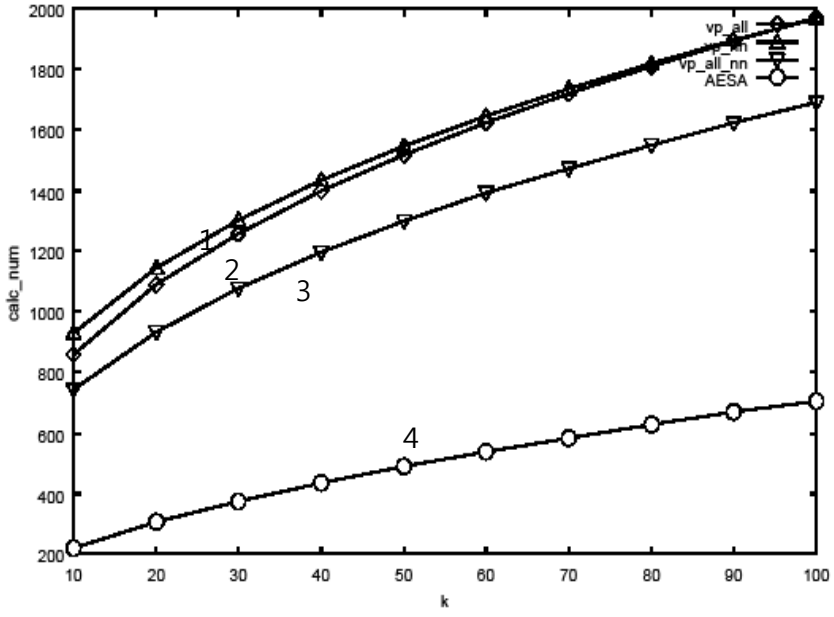


Fig. 6. Metric calculation frequency on 24-dimension data

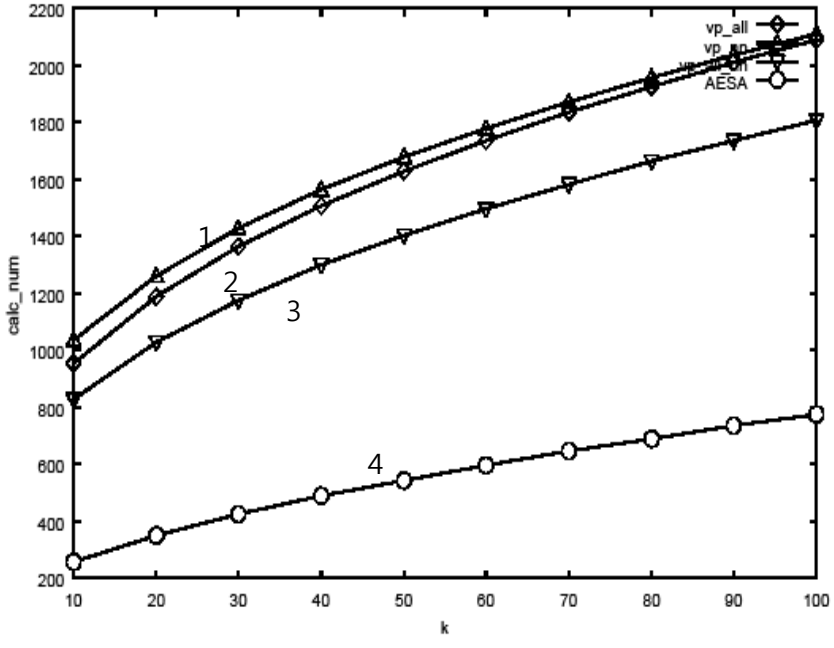


Fig. 7. Metric calculation frequency on 48-dimension data

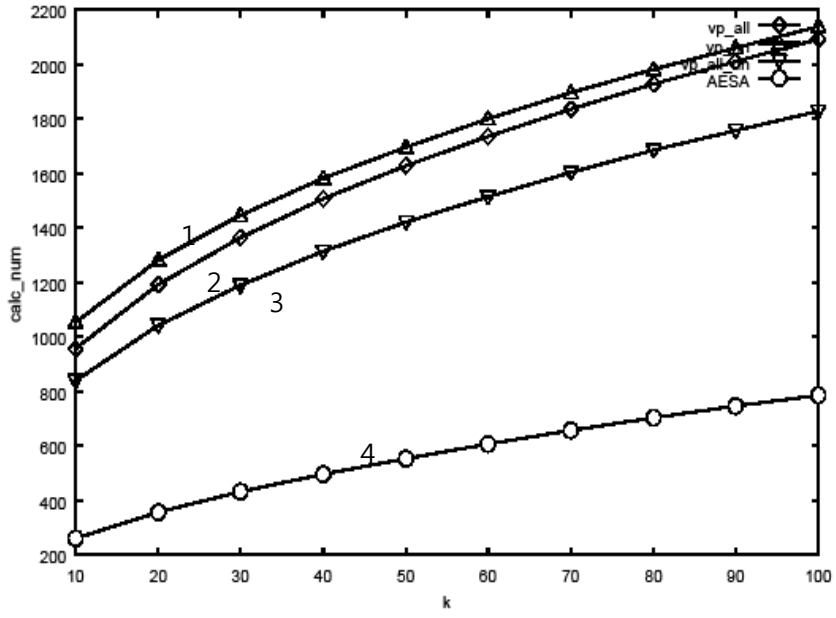


Fig. 8. Metric calculation frequency on 96-dimension data

the difference is minor compared to vp_all and vp_nn. However, about a 10% improvement is observed in vp_all_nn. The range compression based on the nearest node is processed in the ranges rather than in the conventional vp. It appears that all of them have been possible because the search range compression has significantly narrowed.

Regarding the improvement rate of the test time by change in dimensions, the improvement rate stayed at about 5% when 100 data sets were searched at 12 dimensions. However, the improvement rate increased up to 12% when 100 data sets were searched at 96 dimensions. As shown above, if the method mentioned in this paper is used, a sufficient search range compression effect can be obtained despite there being high dimensions.

The detailed information on the indexed files, which was constructed for the test, has been stated in Table 1. Here, 'dim,' 'node,' 'leaf_object,' and 'index_size' refer to dimensions, the number of nodes, the number of leaf objects, and the size of index data, respectively. In a leaf node, the maximum node was set to 10. Regardless of the dimensions, the size of files in which a distance list is recorded using the latest node was 313 Mbytes.

The test results from the comparison with AESA are shown in the Fig. 13 below. Dimensions 12, 24, 48, and 96 are curves 1, 2, 3, and 4, respectively. Even though AESA is superior to the

Table 1. Specification of the index file

Dim	Node	Leaf Object	Index Size (bytes)
12	2357	7643	6,000,640
24	2255	7745	5,742,592
48	2265	7735	5,767,168
96	2295	7705	5,844,992

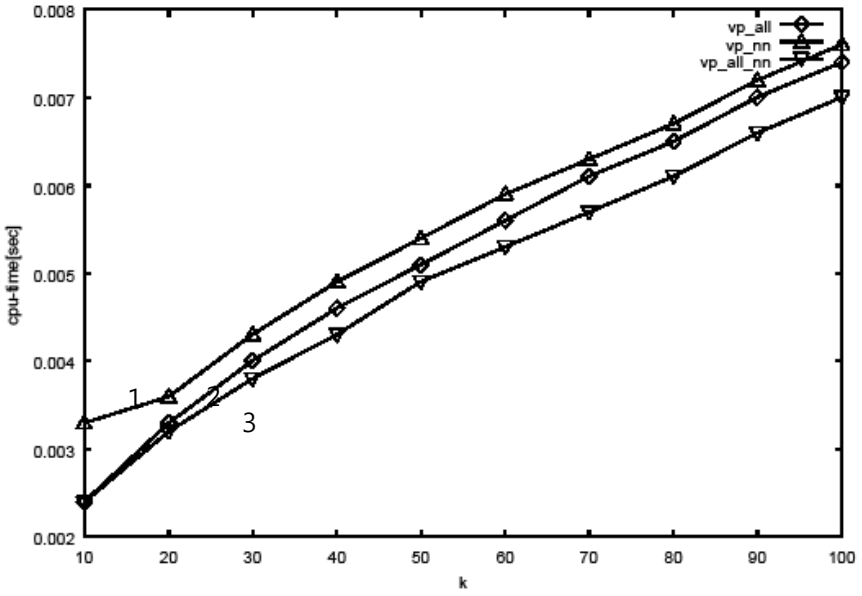


Fig. 9. Execution time on 12-dimension data

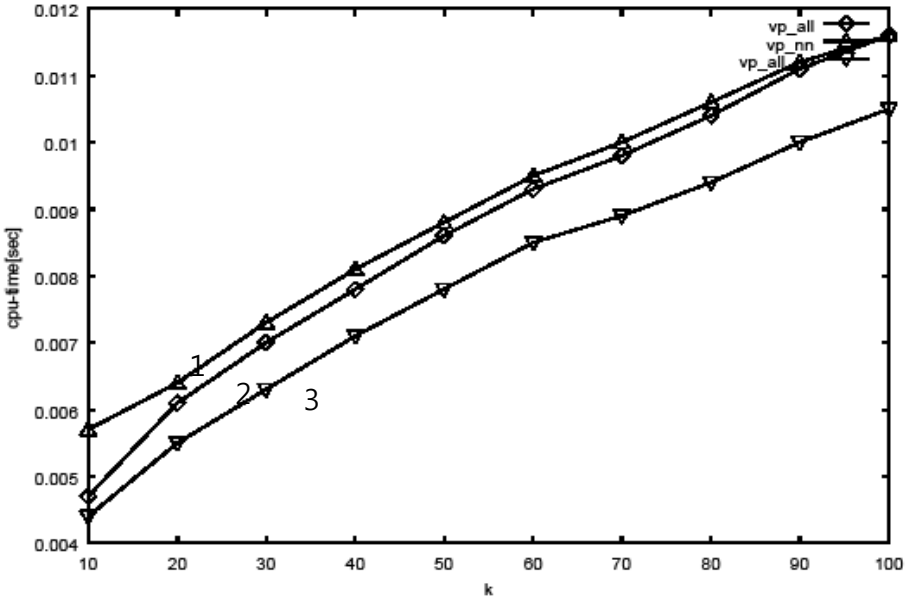


Fig. 10. Execution time on 24-dimension data

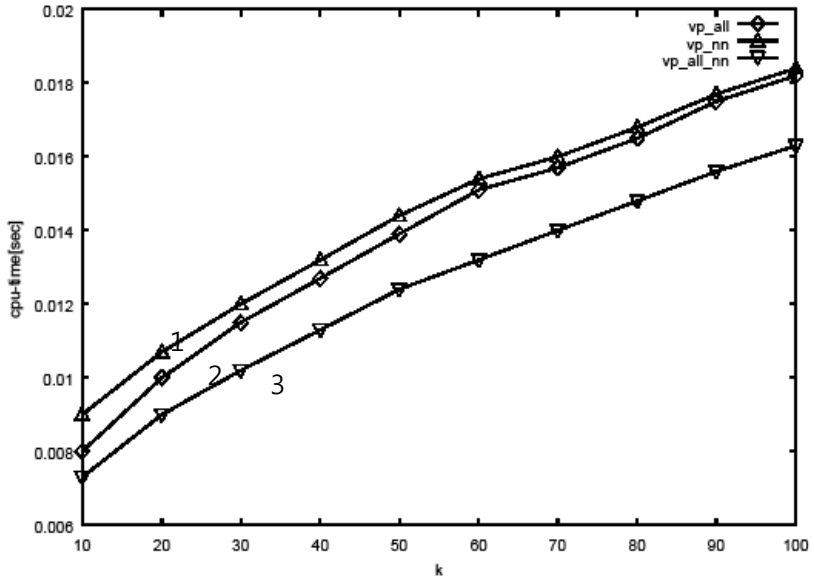


Fig. 11. Execution time on 48-dimension data

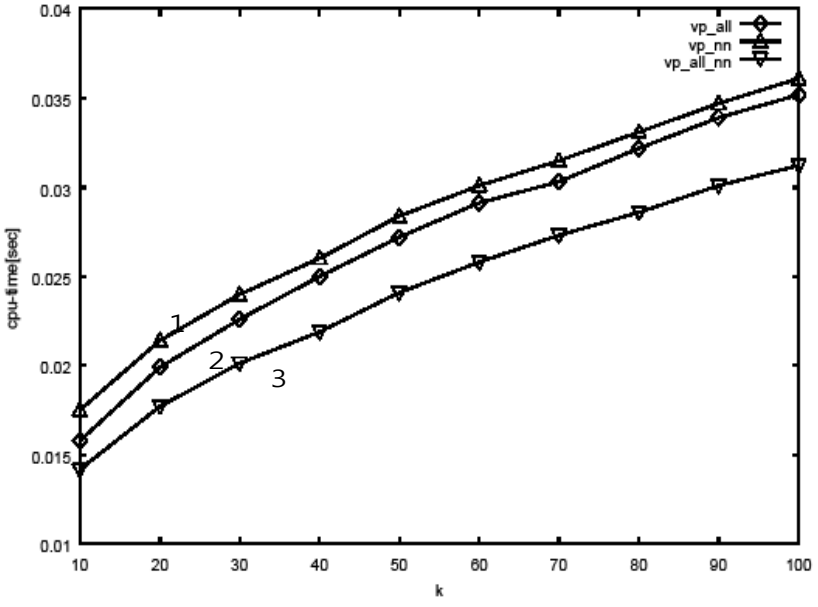


Fig. 12. Execution time on 96-dimension data

Table 2. Comparison of the frequency of reading files from the distance list

Dimension	AESA	VP-tree
12	110	6
24	219	7
48	257	7
96	262	7

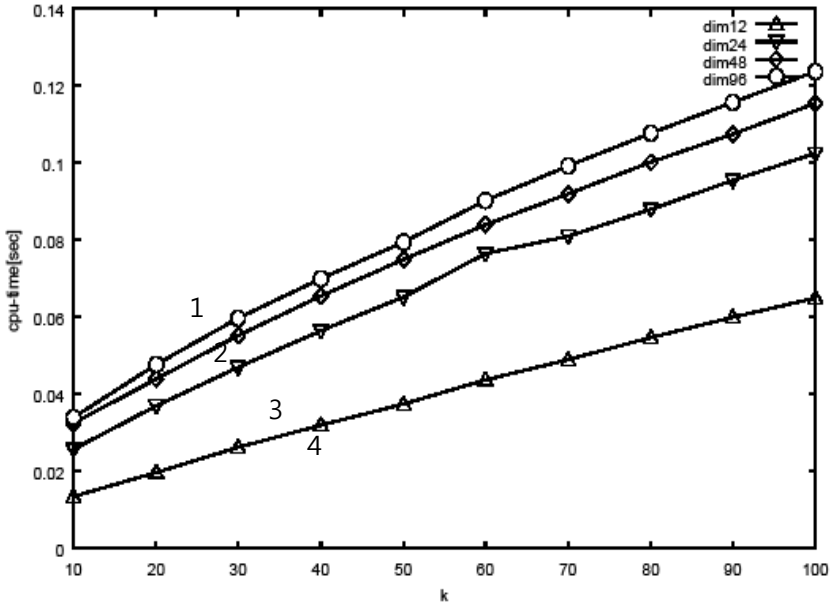


Fig. 13. The execution time of AESA

VP-tree in terms of metric calculation frequency, the former was slower than the latter in terms of search speed because of a difference in the frequency of reading files from a distance list. The frequency for the VP-tree and AESA to read files from the distance list is stated in Table 2 below. The distance list file of the VP-tree is a file that calculated the distance between the leaf object and all of the other objects. On the contrary, the distance list file of AESA is a file that estimated the distance between all of the objects.

In AESA, the distance list file must be read whenever the process is repeated. In other words, distance list files should be read as many times as the distance calculation frequency, which has had a significant impact on search time. Meanwhile, the distance list file is read in the compression of the search range of the leaf objects that are in the VP-tree. Therefore, the frequency for reading files can be reduced to the minimum level. In conclusion, it can be said that the VP-tree is better than AESA in improving search efficiency.

5. CONCLUSION

As the search descends down the tree, some distance calculations can be omitted, and instead the distance is inferred from the distance of the query object from the parent node, and the distance of the parent node from the child node is pre-calculated in indexing time. The search through multimedia data has been discussed. The features of multimedia data are expressed in one vector, and a search is performed based on the similarity between the features in vectors representing indexed data and the vector representing the query. The vectors representing the indexed data are hierarchically arranged as a tree structure. In terms of search time, the search space is reduced by recursively dividing the data set based on metric information, while descending down the tree [21]. We suggest narrowing the search space and consequently reducing the number of distance calculations, by applying trigonometric inequalities over several dimensions that are simultaneously searched.

This paper has attempted to reduce distance calculation frequency and to improve search speed by enhancing the search algorithm of the leaf node in the VP-tree. Experimental results are reported from a comparison against the AESA algorithm. We illustrated that AESA made much less distance calculations than the proposed VP-tree schemes. Our proposed schemes run faster than AESA. The reason why is their smaller number of readings of the distance list file, which calculates the distance between objects during the indexing time. This, seemingly, too, is a result of applying the trigonometric inequalities, or the larger number of items stored in the AESA files as compared with the VP-tree files, which again is a result of applying the trigonometric inequalities.

Using the improvement method, a test has been performed on a similar video search. As a result, the search time was reduced by 5-12%. In addition, it has been confirmed that the VP-tree was more effective than AESA in reducing the search time. It is necessary to perform an additional study on the search algorithm, which can further reduce the distance calculation by using a smaller index size.

REFERENCES

[1] K. Kita, K. Tsuda, and M. Shishibori, *Information Retrieval Algorithm*, Kyoritsu Shuppan, 2002, pp.5-25.

[2] M. Yoshikawa, and S. Uemura, "Indexing Techniques for Multimedia Data," *Journal of Information Processing Society of Japan*, Vol.42, No.10, 2001, pp.953-957.

[3] N. Katayama, and S. Satoh, "Indexing Techniques for Similarity Retrieval," *Journal of Information Processing Society of Japan*, Vol.42, No.10, 2001. pp.958-963.

[4] A. Guttman, *"A Dynamic Index Structure for Spatial Searching," Proceedings of the ACM SIGMOD*, Boston, MA, 1984, pp.47-57.

[5] N. Beckmann, H. -P. Kriegel, R. Schneider, and B. Seeger, *"The R*-tree: An Efficient and Robust Access Method for Points and Rectangles," Proceedings of the ACM SIGMOD*, Atlantic City, NJ, 1990, pp.322-331.

[6] D. A. White, and R. Jain, *"Similarity Indexing with SS-tree," Proceedings of the 12th International Conference on Data Engineering*, 1996, pp.516-523.

[7] N. Katayama, and S. Satoh, "SR-Tree : An Index Structure for Nearest Neighbor Searching of High-Dimensional Point Data," *IEICE Trans. on Information and Systems*, Vol.J80-D-I, No.8, 1997, pp.703-717.

[8] S. Berchtold, D. A. Keim, and H. -P. Kriegel, *"The X-tree An Index Structure for High Dimensional Data," Proceedings of the 22nd VLDB*, 1996, pp.28-39.

[9] R. Weber, F. J. Schek, and S. Blott, *"A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces," Proceedings of the 24th VLDB*, 1998, pp.194-205.

[10] M. Ioka, *A Method of Defining the Similarity of Images on the Basis of Color Information*, Technical Report RT0030, IBM Tokyo Research Lab., 1989.

[11] Y. Rubner, C. Tomasi, and L. J. Guibas, *"The Earch Mover's Distance, Multi-Dimensional Scaling, and Color-Based Image Retrieval," Proceedings of the ARPA Image Understanding Workshop*, 1999, pp.661-668.

[12] P. Ciaccia, M. Patella, and P. Zezula, *"M-tree: An Efficient Access Method for Similarity Search in Metric Spaces," Proceedings of the ACM SIGMOD*, San Jose, CA, 1995, pp.71-79.

[13] P. N. Yianilos, *"Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces," Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1993, pp.311-321.

[14] A. W. -C. Fu, P. M. S. Chan, Y. L. Cheung, and Y. S. Moon, "Dynamic VP-Tree Indexing for N-Nearest Neighbor Search Given Pair-Wise Distances," *The VLDB Journal*, Vol.9, No.2, 2000, pp.154-173.

[15] T. Bozkaya, and M. Ozsoyoglu, *"Distance-based Indexing for High Dimensional Metric Spaces," Proceedings of the 1997 ACM SIGMOD*, Tucson, AZ, 1997, pp.357-368.

[16] M. Ishikawa, J. Notoya, H. Chen, and N. Ohbo, "A Metric Index MItree," *Trans. of Information Processing Society of Japan*, Vol.40, No.SIG 6 (TOD 3), 1999, pp.104-114.

[17] V. Ruiz, "An Algorithm for Finding Nearest Neighbors in (approximately) Constant Average Time," *Pattern Recognition Letters*, Vol.4, iss.3, 1986, pp.145-157.

[18] M. Iwasaki, "Implementation and Evaluation of Metric Space Indices for Similarity Search," *Trans. of Information Processing Society of Japan*, Vol.40, No.SIG 3(TOD 1), 1999, pp.24-33.

[19] H. Akama, F. Konishi, T. Yoshida, M. Yamamuro, and K. Kushima, "External Key Search and Dynamic Data Insertion in Inverted File Indexing Method Applied for Nearest Neighbor Search," *Trans. of Information Processing Society of Japan*, Vol.40, No.SIG 8(TOD 4), 1999, pp.51-62.

[20] Corel Image, http://www.corel.com, 2011.

[21] M. Shishibori, S. S. Lee, and K. Kita, *"A Method to Improve Metric Index VP-tree for Multimedia Databases," The 17th Korea-Japan Joint Workshop on Frontiers of Computer Vision*, Woolsan, Korea, 2011, pp.21-26.

[22] P. Zezula, G. Amato, V. Dohnal, M. Batko, *Similarity Search: The Metric Space Approach*, 1st ed., Springer, New York, 2005, pp.9-16.

[23] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni, "Locality-Sensitive Hashing Scheme Based on p-Stable Distributions," Proceedings of the Twentieth Annual Symposium on Computational Geometry (SCG '04), 2004, pp.253-262.

[24] T. Darrell and P. Indyk and G. Shakhnarovich (eds.), "Nearest Neighbor Methods in Learning and Vision: Theory and Practice," MIT Press, 2006.

[25] A. Andoni and P. Indyk, "Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions," Communications of the ACM, Vol.51, No.1, 2008, pp.117-122.

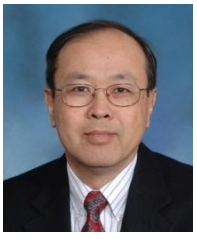
[26] A. Andoni and P. Indyk "Near-Optimal Hashing Algorithms for Near Neighbor Problem in High Dimensions," In Proceedings of the Symposium on Foundations of Computer Science (FOCS '06), 2006, pp.459-468.

Samuel Sangkon Lee

He received his PhD degree in 2001, from Tokushima University, Japan. He is currently an Associate Professor in the Department of Computer Science and Engineering at Jeonju University, Republic of Korea. His research interests include information retrieval, document classification, natural language processing.

Masami Shishibori

He received his BS Degree in 1991, his MS Degree in 1993 and PhD Degree in 1997, from Tokushima University, Japan. He is currently a Professor in the Department of Information Science and Intelligent Systems at Tokushima University, Japan. His research interests include multimedia processing, information retrieval, natural language processing.



Chia Y. Han

He received his PhD degree from University of Cincinnati in 1985. He is currently an Associate Professor of Computer Science in the School of Computing Sciences and Informatics at University of Cincinnati, Cincinnati, Ohio, USA. His research interests include computer graphics, multimedia and augmented reality, human computer interaction, and intelligent systems.