
Multimedia Tools and Applications, 13, 235–254, 2001

Efficient Near Neighbor Searching Using Multi-Indexes for Content-Based Multimedia Data Retrieval

CHIH-CHIN LIU
JIA-LIEN HSU
ARBEE L.P. CHEN* alpchen@cs.nthu.edu.tw
Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan 300, R.O.C

Abstract. Many content-based multimedia data retrieval problems can be transformed into the near neighbor searching problem in multidimensional feature space. An efficient near neighbor searching algorithm is needed when developing a multimedia database system. In this paper, we propose an approach to efficiently solve the near neighbor searching problem. In this approach, along each dimension an index is constructed according to the values of feature points of multimedia objects. A user can pose a content-based query by specifying a multimedia query example and a similarity measure. The specified query example will be transformed into a query point in the multi-dimensional feature space. The possible result points in each dimension are then retrieved by searching the value of the query point in the corresponding dimension. The sets of the possible result points are merged one by one by removing the points which are not within the query radius. The resultant points and their distances from the query point form the answer of the query. To show the efficiency of our approach, a series of experiments are performed to compare with the related approaches.

Keywords: near neighbor searching, spatial index, content-based multimedia data retrieval, multimedia databases

1. Introduction

As multimedia applications such as World-Wide-Web, video-on-demand, and digital library become more and more popular, the need for content-based multimedia data retrieval is getting more important [5–9, 12, 15, 17, 18, 26–28, 30]. Since each kind of multimedia data such as image, video, music, and graphics has its own characteristics, it is very difficult to develop a uniform method for content-based retrieval. To support the content-based retrieval capability for the multimedia applications, we have started a multimedia database system project called *Vega* [17–19]. Currently, the *Vega* project focuses on how to represent, extract, and query the content of image, video, and music data. By analyzing the characteristics of each kind of multimedia data, we conclude that many content-based retrieval problems can be transformed into the near neighbor searching problem (in which the content of a multimedia object is mapped into a point, called *feature point*, in a multidimensional feature space [15]) or the string matching problem [19]. The following examples are typical features by which the near neighbor searching techniques can be applied.

*http://db.nthu.edu.tw.

- Color histograms of image objects
- Average *RGB* values of image objects
- Texture feature vectors of image objects
- Tone histograms of music objects

There are two major groups of approaches to solve the near neighbor searching problems. The first group of approaches in based on the *R*-tree and its variants, such as the R^*-tree and the R^+-tree. The *R*-tree family is the fundamental data structure for indexing the geometric data [1, 14, 25]. Roussopoulos et al. present an efficient branch-and-bound traversal algorithm to find the nearest neighbor object to a given point based on the *R*-tree [23]. In this approach, the containment property, the information of minimum bounding region, and the triangular inequality rule are utilized to effectively prune branches during the nearest neighbor searching process. However, this approach is not suitable for higher dimensional data since the containment property suffers in higher dimensional feature space (explode exponentially with the dimensionality [21]). And the feature points extracted from the multimedia objects are often in high dimensional feature spaces.

The other group of approaches partitions the feature space in a relative way [4, 6, 29]. Yianilos organizes the feature points into a tree structure called *vp-tree* [29]. In a vp-tree, each node, termed *vantage point*, partitions the space by a sphere, and divides the remaining nodes into two groups. These two groups are corresponding to the left and right subtree of the vantage point. By carefully choosing the vantage points, the vp-tree will be balancedly constructed. Since the distances among feature points are pre-computed at tree-construction phase, the vp-tree is a feasible solution to index higher dimensional points. The GNAT (Geometric Near-neighbor Access Tree), also recursively partitions the feature space into subspaces [4]. The difference between the vp-tree from the GNAT is on how the feature space is partitioned. These approaches have two major drawbacks: first, the numbers of the node visiting operations in both the vp-tree approach and the GNAT approach depend on the choices of the branching points. However, it is difficult to find the branching points that result least number of node visiting operations for a given query. Second, the cost for distance calculation spends in each node visiting operation is very high as the number of dimensions increases.

Unlike the partition-based approaches, we propose an elimination-based approach to efficiently solve the near neighbor searching problem. For each type of media objects to be indexed in the multimedia database, we first automatically or manually extract a set of features for each multimedia object to represent their content. For the features whose values can be represented as points (or vectors) in a multidimensional space, we build a *multi-index* for each feature. The construction of a multi-index is quite simple: the values of the feature points in each dimension are sorted and stored in an index structure. When users pose a query, the feature that the query is based and the similarity measure should be specified. To process the query, it will first be transformed into a *query point* in the feature space. Then in each dimension, the corresponding value of the query point is used as the searching key to find the closest point in that dimension. The feature points retrieved in each dimension are called the *candidate points* in the dimension. By merging the candidate points in each dimension, we can get the feature points satisfying the query conditions.

The rest of this paper is organized as follows. In Section 2, we discuss the role of the near neighbor searching techniques in content-based multimedia data retrieval. A new near neighbor searching technique is proposed in Section 3. To show the efficiency of our approach we explain the experiment results in Section 4. Section 5 concludes this paper and describes our future directions.

2. The near neighbor searching problem

A content-based multimedia query usually consists of a multimedia object as the query example and a *similarity measure* which specifies how close the result multimedia objects should be similar to the query example. If the features of the multimedia objects and the query examples can be represented as feature points in Euclidean space, the query processing of the content-based query will be transformed into a near neighbor searching problem.

The near neighbor searching problem can be formulated as follows. Let $P_1, P_2, \ldots, P_k$ be a set of k points in the n-dimensional feature space, which represent the feature points of the k multimedia objects in the multimedia database. Given a multimedia query example whose corresponding feature point is Q and the similarity measure r, the answer of this query is $\{P_i | d(P_i, Q) \leq r, P_i \in \{P_1, P_2, \ldots, P_k\}\}$, where $d(P_i, Q)$ is the distance between P_i and Q in the n-dimensional feature space. Figure 1 shows a near neighbor searching example in which P_1 and P_4 are the results of the query Q.

3. The multi-index approach

Figure 2 shows the overall approach for content-based multimedia data retrieval using multi-index. It consists two phases: index construction and query processing. In the index construction phase, for each multimedia object in the database, its feature is extracted and represented as a feature point in the feature space. Then a multi-index is constructed to organize these feature points. In the query processing phase, the candidate point set in each

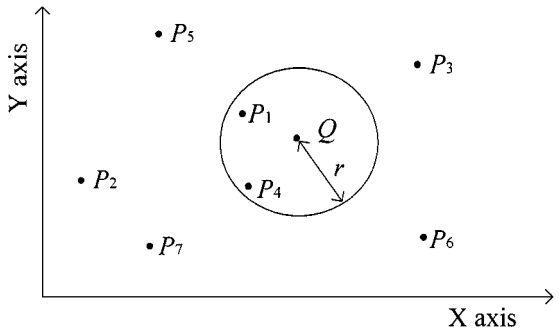


Figure 1. A near neighbor searching example in a two-dimensional feature space.

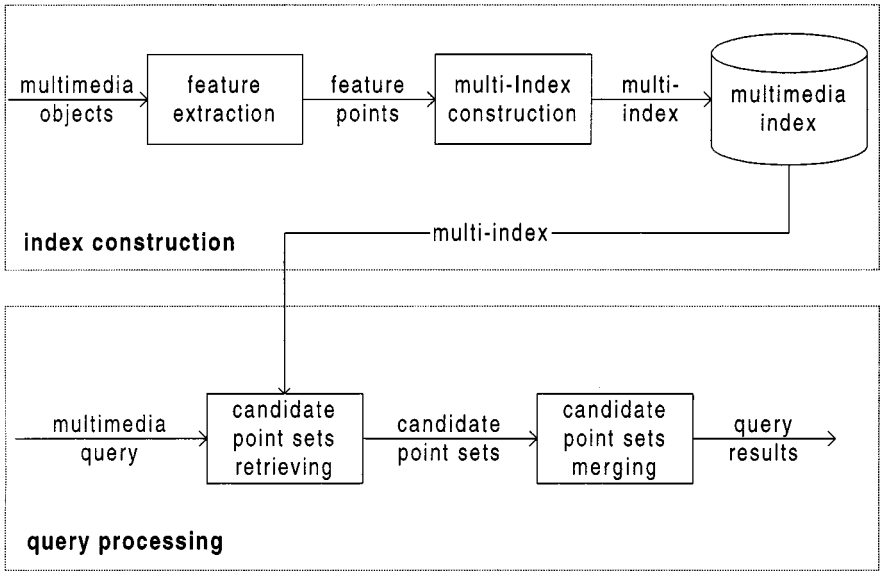


Figure 2. Overview of the multi-index approach.

dimension is retrieved according to the information of the multi-index in that dimension. These candidate point sets are merged by removing the candidate points that are not within the query range. The merged candidate points form results of the query.

We will use average **RGB** values of image objects as examples to illustrate the notion of the proposed near neighbor searching algorithms proposed in this section.

3.1. The construction of a multi-index

As we have described in the previous section, before a content-based multimedia query can be processed, the values of the feature points of the multimedia objects in the multimedia database should be indexed. Assume there are k multimedia objects in the multimedia database, k feature points are extracted and stored as k points $P_1, P_2, \ldots, P_k$ in an n-dimensional feature space. We build a multi-index MI for this feature as follows.

1. **Algorithm 1: Muild_Multi_Index**
2. /* input: a set of k points P_i, in n-dimensional feature spaces */
3. /* assume $P_i = (x_{1,i}, x_{2,i}, \ldots, x_{n,i})$ */
4. /* output: a multi-index $MI = \{I_1, I_2, \ldots, I_n\}$ */
5. begin
6. for $i = 1$ to n
7. begin

```
 8.                sort {x_{i,1}, x_{i,2}, ..., x_{i,k}} and store the result in I_i
 9.            end
10.        return {I_1, I_2, ..., I_n}
11.    end
```

The time complexity of Algorithm 1 is $O(nk \log k)$ since there are n sets of k values to be sorted. The space required to store the multi-index is $O(nk)$. We use the following example to illustrate the process of Algorithm 1.

Example 1: Assume there is a picture database which contains 10 pictures $P_1, P_2, \ldots, P_{10}$ as shown in figure 3(a). The average R, G, B values of these pictures are shown in figure 3(c). To provide the ability to efficiently retrieve the pictures whose average R, G, B values are similar to those of a user specified picture example, these pictures should be indexed. To achieve this, we apply algorithm 1 on the ten pictures. A multi-index set $MI = \{I_R, I_G, I_B\}$ is then constructed by sorting the values in the corresponding dimension. The result is shown in Table 1.

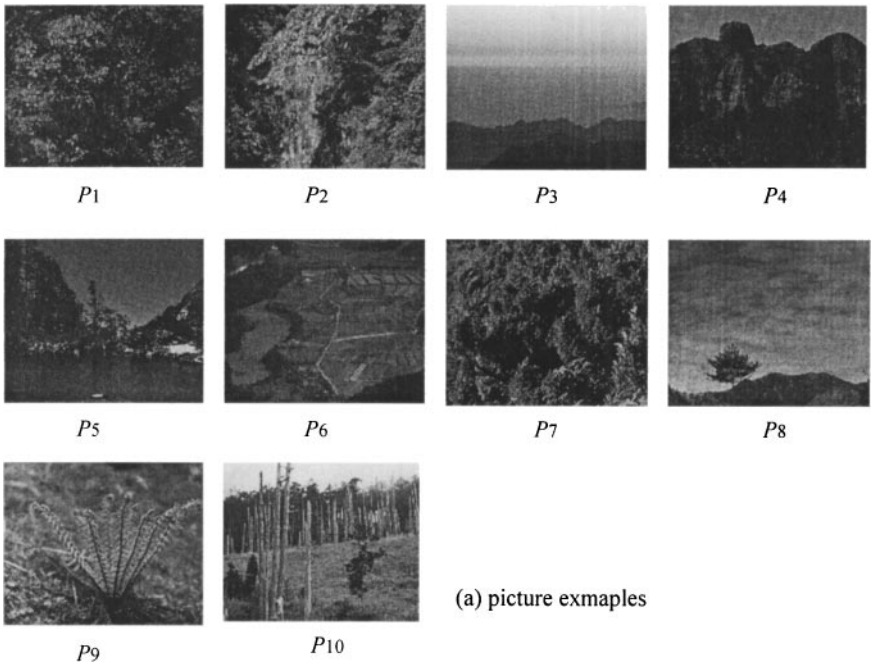
3.2. Retrieving the candidate points

To process a content-based multimedia query whose query point is $Q = (q_1, q_2, \ldots, q_n)$ and the similarity measure (also called *query radius*) is r, we first compute the difference δ_i of q_i and its closest value in the i-th index. We call the result the *difference vector*, denoted as $\Delta = (\delta_1, \delta_2, \ldots, \delta_n)$.

When the dimension number n of the feature space is large, it is possible that in certain dimension i, the difference of q_i and the closest value in the i-th index is larger than the query radius. It means that the multimedia database does not contain any answer of this query. When this occurs, the near neighbor searching process terminates and it only takes $O(n \log k)$ processing time. This is one of the main advantages of our approach. However, if it is not the case, the difference vector will be used to decide the order of the dimensions

Table 1. (a) The index for R values. (b) The index for G values. (c) The index for B values.

(a)										
R Value	0.102	0.141	0.145	0.180	0.212	0.275	0.318	0.361	0.451	0.627
Picture ID	P_1	P_5	P_4	P_7	P_6	P_2	P_8	P_9	P_{10}	P_3
(b)										
G Value	0.101	0.137	0.153	0.180	0.200	0.251	0.302	0.365	0.396	0.447
Picture ID	P_1	P_5	P_4	P_7	P_6	P_2	P_9	P_8	P_{10}	P_3
(c)										
B Value	0.086	0.102	0.161	0.184	0.184	0.227	0.231	0.302	0.400	0.561
Picture ID	P_1	P_7	P_2	P_5	P_9	P_4	P_6	P_3	P_{10}	P_8



Picture ID	(R, G, B)	Picture ID	(R, G, B)
P_1	(0.102, 0.101, 0.086)	P_8	(0.318, 0.365, 0.561)
P_2	(0.275, 0.251, 0.161)	P_9	(0.361, 0.302, 0.184)
P_3	(0.627, 0.447, 0.302)	P_{10}	(0.451, 0.396, 0.400)
P_4	(0.145, 0.153, 0.227)	Q_1	(0.478, 0.541, 0.753)
P_5	(0.141, 0.137, 0.184)	Q_2	(0.302, 0.310, 0.416)
P_6	(0.212, 0.200, 0.231)	Q_3	(0.302, 0.223, 0.161)
P_7	(0.180, 0.180, 0.102)		

(c)

Figure 3. (a) Ten picture examples. (b) Three picture query examples. (c) Their average (R, G, B) values.

to search. The computing of the difference vector is described in Algorithm 2.

```
1.  Algorithm 2 Find_Difference_Vector
2.  /* input: a query point Q = (q1, q2, ..., qn) in n-dimensional feature space */
3.  /*        a query radius r                                                    */
4.  /*        a multi-index MI = {I1, I2, ..., In}                               */
5.  /* output: a difference vector Δ = (δ1, δ2, ..., δn)                         */
6.  begin
7.      for i = 1 to n
8.          begin
9.              search the nearest value to qi in Ii
10.             let xi be the nearest value
11.             δi = |qi − xi|
12.             if δi ≥ r
13.                 return null
14.             endif
15.         end
16.     return Δ = (δ1, δ2, ..., δn)
17. end
```

The feature points of a multimedia database usually cluster together rather than randomly distribute in the feature space. If a user specifies a multimedia query example which is not similar to any multimedia object in the database, our query processing will complete quickly. Let us see the following example.

Example 2: Assume there is a picture query Q_1 whose (R, G, B) feature point is (0.478, 0.541, 0.753), as shown in figure 3. Let the query radius r be 0.15. The picture database to be searched is the one described in Example 1. To find the answers of this query, first we apply Algorithm 2 on this query.

$$\begin{aligned}\Delta &= (\delta_R, \delta_G, \delta_B)\\ &= (|0.478 - 0.451|, |0.541 - 0.447|, |0.753 - 0.561|)\\ &= (0.027, 0.094, 0.192)\end{aligned}$$

Since $\delta_B = 0.192 > r = 0.15$, the picture database does not contain any picture that satisfies Q_1.

If the value of the difference vector in each dimension is smaller than the query radius, the next step of the query processing is to find the *candidate point set* in each dimension. In each dimension, the interval used to determine whether a feature point is a candidate point is called the *search range* in the dimension. The search range for each dimension can be set to equal to the query radius. If the number of points in a candidate point set can be reduced, less processing time will be needed to merge the candidate point sets. Thus, the cost for distance computing will be reduced. To achieve this goal, we find that by arranging the order of the searching in each dimension according to the corresponding value of the

difference vector, the search range may be decreased in some dimensions. We called this the *search range decreasing rule*. This rule can be formulated as follows.

Theorem 1. *Given a query whose query point $Q = (q_1, q_2, \ldots, q_n)$ in an n-dimensional feature space and the query radius r, there exist n search ranges $r_1, r_2, \ldots, r_n$ such that $r_1 \geq r_2 \geq \cdots \geq r_n$.*

Proof: Assume the difference vector of this query is $\Delta = (\delta_1, \delta_2, \ldots, \delta_n)$, where $\delta_i \geq 0$ for $1 \leq i \leq n$. Let $r_1 = r$, i.e., the search range in the first dimension equals the query radius. Since the distance from every feature point to q_1 is larger than or equal to δ_1 in the first dimension, the feature points whose distances to q_2 in the second dimension are larger than $({r_1}^2 - {\delta_1}^2)^{1/2}$ can not be candidate points. Thus, the search range in the second dimension can be set to $r_2 = ({r_1}^2 - {\delta_1}^2)^{1/2}$. Similarly we can set $r_3 = ({r_2}^2 - {\delta_2}^2)^{1/2}$, $r_4 = (r_3^2 - \delta_3^2)^{1/2}, \ldots, r_n = (r_{n-1}^2 - \delta_{n-1}^2)^{1/2}$. Since $\delta_i \geq 0$, for $1 \leq i \leq n$, $r_i \leq r_{i-1}$ holds, i.e., $r_1 \geq r_2 \geq \cdots \geq r_n$. □

For example, assume the query radius is 0.5 and the largest value in the difference vector is 0.3. Then the difference of the result points, if any, in any other dimension and the corresponding value of the query point should be smaller than $(0.5^2 - 0.3^2)^{1/2} = 0.4$. Thus, we can set the search range of the next dimension to be 0.4. Repeating the decreasing of the search range until the search range is less than zero, which means the query has no answer, or all candidate point sets are obtained. Of course if the number of feature points in any candidate point set is zero, the query has no answer. The candidate point retrieval algorithm goes as follows.

```
1.  Algorithm 3 Candidate_Point_Retrieval
2.  /* input: a query point Q = (q1, q2, ..., qn) in n-dimensional feature space   */
3.  /*        a query radius r                                                     */
4.  /*        a multi-index MI = {I1, I2, ..., In}                                 */
5.  /*        a difference vector Δ = (δ1, δ2, ..., δn)                            */
6.  /* output: the candidate point set in each dimension                          */
7.  begin
8.      sort Δ = (δ1, δ2, ..., δn) into Δ' = (δ1', δ2', ..., δn')
9.          where δ' = δ_oi for 1 ≤ i ≤ n, oi denotes the subscript of δi' in Δ
10.     let R = (r1, r2, ..., rn)
11.     let ri = r
12.     for i = 2 to n
13.         begin
14.             ri = (r_{i-1}^2 - δ_{i-1}^2)^{1/2}
15.             if ri^2 ≤ 0
16.                 return null
17.             endif
18.         end
19.     for i = 1 to n
```

```
20.             search δ_{o_i} ± r_i, in I_{O_i} and store the candidate points to S_i
21.             if NumberOf(S_i) = 0
22.                 return null
23.             endif
24.         return S_1, S_2, ..., S_n
25.     end
```

Example 3: Assume there is a picture query Q_2 whose (R, G, B) feature vector is (0.302, 0.310, 0.416), as shown in figure 3. Let the query radius r be 0.02. The picture database to be searched is the one described in Example 1. To find the answers of this query, first we apply Algorithm 2 on this query.

$$\begin{aligned}\Delta &= (\delta_R, \delta_G, \delta_B)\\ &= (|0.302 - 0.318|, |0.310 - 0.302|, |0.416 - 0.400|)\\ &= (0.016, 0.008, 0.016)\end{aligned}$$

Since all δ_R, δ_G, and δ_B are smaller than r, we then apply Algorithm 3.

$$R = (r_1, r_2, r_3) = (0.02, (0.02^2 - 0.016^2)^{1/2}, (0.02^2 - 0.016^2 - 0.016^2)^{1/2})$$

Since ${r_3}^2 < 0$, we find that the picture database does not contain any picture that satisfies Q_2.

3.3. Merging the candidate points

The results generated by Algorithm 3 are the candidate point sets of all the dimensions. To generate the answer of the query, the final step is to merge these candidate point sets. The answer consists of a set of result feature points and their similarity measures to the query example Q. Two tables are used to record the intermediate results of candidate point set merging: the *Counter* table and the *Distance* table. For each candidate point P_i, the Counter table keeps the number of dimensions in which P_i satisfies the associated query conditions. When the merging process is executed in the j-th dimension, *Distance*$[i]$ will keep the distance between P_i and Q in the j-th dimensional subspace. If P_i passes the query condition in the j-th dimension, i.e., *Distance*$[i]$ is less than the query radius in the j-th dimensional subspace, increase the value of *Counter*$[i]$ by 1. After all candidate point sets are merged, the points whose associated Counter values are the result feature points of the query. Their similarity measures are stored in the *Distance* table. Algorithm 4 formulates the candidate point set merging process.

```
1. Algorithm 4 Candidate_Point_Set_Merging
2. /* input: the candidate point sets S_1, S_2, ..., S_n                    */
3. /*        the search order list O_1, O_2, ..., O_n, generated by Algorithm 3  */
```

```
/*      a query point Q = (q1, q2, ..., qn) in n-dimensional feature space      */
/*      a query radius r                                                        */
/* output: the result feature points and their similarity measures              */
begin
    for i = 1 to k          /* initial two tables: Counter and Distance        */
        begin
            Counter[i] = 0
            /* store the number of dimensional conditions satisfied by Pi      */
            Distance[i] = ∞
                /* store the partial computed distance between Pi and Q        */
        end
    for each candidate point Pi in Si
        begin
            Counter[i] = 1
                        /* qOi is the value of Q in Qi-th dimension            */
            Distance[i] = |qOi − xOi,i|
                    /* xOi,i is the value of Pi in the Qi-th dimension         */
        end
    for j = 2 to n
        begin
            for each candidate point Pi in Sj
                begin
                    if Counter[i] = j − 1
                        begin
                            Distance[i] = (Distance[i]^2 + (qOi − xOi,i)^2)^1/2
                            if Distance[i] ≤ qOi
                                Counter[i] = Counter[i] + 1
                    /* Pi satisfies the query condition in the j-th dimension  */
                            endif
                        end
                    endif
                end
        end
    for each Pi with Counter[i] = n
        begin
            output(Pi, Distance[i])
        end
end
```

The time complexity of Algorithm 4 is $O(\sum_{i=1}^{n} N_i)$ where n is the number of dimensions in the feature space and N_i is the number of candidate points in the i-th dimension.

Example 4: Assume there is a picture query Q_3 whose (R, G, B) feature vector is (0.302, 0.223, 0.161), as shown in figure 3. Let the query radius r be 0.05. The picture database

to be searched is the one described in Example 1. To find the answers of this query, first we apply Algorithm 2 on this query.

$$\begin{aligned}\Delta &= (\delta_R, \delta_G, \delta_B)\\ &= (|0.302 - 0.318|, |0.223 - 0.200|, |0.161 - 0.161|)\\ &= (0.016, 0.023, 0)\end{aligned}$$

Since all δ_R, δ_G, and δ_B are smaller than r, we then apply Algorithm 3. The search order is "$G \rightarrow R \rightarrow B$".

Search the index for G values with $0.223 \pm 0.05 = [0.173, 0.273]$, wet get $S_G = \{P_7, P_6, P_2\}$.

Search the index for R values with $0.302 \pm 0.044 = [0.258, 0.346]$, wet get $S_R = \{P_2, P_8\}$.

Search the index for B values with $0.161 \pm 0.041 = [0.120, 0.202]$, wet get $S_B = \{P_2, P_5, P_9\}$.

The candidate point retrieving procedure is shown in figure 4(a).

Apply Algorithm 4 on the three candidate point sets, the values in the Counter table and the Distance table are initialized to be 0 and ∞, respectively as shown in figure 4(b). The candidate point set S_G is merged by the following two procedures: increasing the values of *Counter*[2], *Counter*[6], and *Counter*[7] by one and calculating *Distance*[2], *Distance*[6], and *Distance*[7] in the one-dimensional subspace. The result is shown in figure 4(c). Similarly, the candidate point set S_R is merged and the result is shown in figure 4(d). Finally, the candidate point set S_B is merged and we find the answer of this query is P_2 and its similar measure is 0.039. The result is shown in figure 4(e).

3.4. k-nearest neighbor searching

The algorithms proposed above solve the near neighbor searching problem with a search range which can be called *range query algorithms*. However, in many multimedia applications, users would like to retrieve k multimedia objects which are most similar to the given query example. This problem is known as the *k-nearest neighbor searching* problem. To find the k nearest neighbors of a query point Q, we repeatedly apply the range query algorithm with various query radiuses. The initial query radius r_0 can be set as follows to find the k result feature points in the first round. Assume there are N feature points uniformly distributed in the n-dimensional feature space. r_0 can be set as

$$r_0 = \sqrt[n]{\frac{k}{N}}$$

N : total number of objects in the databases

n : number of dimensions

Assume there are m_0 result feature points retrieved by the range query algorithm in the first round. If $m_0 \geq k$, the first k result feature points with larger similarity measures will

						$\longleftarrow 0.302 \pm 0.044 \longrightarrow$			
R Value	0.102	0.141	0.145	0.180	0.212	0.275	0.318	0.361	0.451	0.627
Picture ID	P_1	P_5	P_4	P_7	P_6	$\boldsymbol{P_2}$	$\boldsymbol{P_8}$	P_9	P_{10}	P_3

					$\longleftarrow 0.223 \pm 0.050 \longrightarrow$					
G Value	0.101	0.137	0.153	0.180	0.200	0.251	0.302	0.365	0.396	0.447
Picture ID	P_1	P_5	P_4	$\boldsymbol{P_7}$	$\boldsymbol{P_6}$	$\boldsymbol{P_2}$	P_9	P_8	P_{10}	P_3

				$\longleftarrow 0.161 \pm 0.041 \longrightarrow$						
B Vlaue	0.086	0.102	0.161	0.184	0.184	0.227	0.231	0.302	0.400	0.561
Pciture ID	P_1	P_7	$\boldsymbol{P_2}$	$\boldsymbol{P_5}$	$\boldsymbol{P_9}$	P_4	P_6	P_3	P_{10}	P_8

(a)

Pciture ID	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}
Counter	0	0	0	0	0	0	0	0	0	0
Distance	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

(b)

Pciture ID	P_1	$\boldsymbol{P_2}$	P_3	P_4	P_5	$\boldsymbol{P_6}$	$\boldsymbol{P_7}$	P_8	P_9	P_{10}
Counter	0	1	0	0	0	1	1	0	0	0
Distance	∞	0.028	∞	∞	∞	0.023	0.043	∞	∞	∞

(c)

Pciture ID	P_1	$\boldsymbol{P_2}$	P_3	P_4	P_5	P_6	P_7	$\boldsymbol{P_8}$	P_9	P_{10}
Counter	0	2	0	0	0	1	1	0	0	0
Distance	∞	0.039	∞	∞	∞	0.023	0.043	∞	∞	∞

(d)

Pciture ID	P_1	$\boldsymbol{P_2}$	P_3	P_4	$\boldsymbol{P_5}$	P_6	P_7	P_8	$\boldsymbol{P_9}$	P_{10}
Counter	0	3	0	0	0	1	1	0	0	0
Distance	∞	0.039	∞	∞	∞	0.023	0.043	∞	∞	∞

(e)

Figure 4. (a) Candidate retrieving procedure of example 4. (b)(c)(d)(e) Candidate merging procedure of Example 4.

be the answer. Otherwise, the query radius in the second round will be set as

$$r_1 = \sqrt[n]{\frac{k}{m_0}} \times r_0 \times S$$

S : speedup factor

S is a constant (e.g., $S = 1.5$) which enlarges the query range such that less rounds will be needed. In general, the range query algorithm will be applied for t times such that $m_t \geq k$

and $m_{t-1} < k$. The query radius of the i-th ($i \geq 1$) round will be set as

$$r_i = \sqrt[n]{\frac{k}{m_{i-1}}} \times r_{i-1} \times S$$

4. Performance analysis

To show the efficiency of our approach, we perform a series of experiments and compare our approach with the vp-tree approach. The environment and assumptions of the experiments are described first. Based on these assumptions, we implement the algorithms described in the previous section. There are four factors which dominate the performance of the near neighbor searching: the dimension of feature space, the size of data set, the query radius, and the distance function. For each factor, we show the experiment results and explain its behaviors.

4.1. Experiment set-up

A near neighbor searching method should perform well when dealing with real data. Since the distribution of the feature points in a multimedia database are clustered, not uniform nor random, the near neighbor searching method should take this characteristics into consideration. Since the dimension of a feature point based on the color histogram of image objects depends on the number of colors the image object possesses, we do a series of experiments on the color histograms of image objects to show the performance of our approach. For example, to form a feature point in a 64-dimension feature space, we cut a sub-picture of size 8×8 (also called a block) from a picture of 64 colors. For a picture of size 128×128, we can generate 256 feature points in a 64-dimension feature space. Given an example image block and a similarity measure, the blocks which are similar to the example block will be retrieved. Adjusting the number of colors of the image objects to be indexed, we can generate many sets of feature points in different dimensional feature spaces.

The experiments are performed based on the following assumptions. First, we use the operation count as the unit to measure the performance of the near neighbor searching algorithms to avoid the effect due to the different computing powers of various machines. We also assume that the memory size is large enough to keep the whole indexing structure such that the I/O considerations can be omitted. Finally, the number of node visitings is the most popular measurement for tree structures. However, this measurement is not suitable for non-tree data structures as in our approach. Thus, we transform the number of node visitings when traversing the vp-trees into the corresponding operation counts such that the comparisons can be made on the same measure.

For comparison, we first implement a brute-force method. However, the performance is much poorer than both our approach and the vp-tree approach. For example, processing a query whose query point is in a 64-dimension feature space, and the size of the data set is 2048, the brute-force method needs 526366 operations while our method only needs 1093 operations. Therefore, we omits the results of the brute-force method. To implement the

vp-tree method, we follow the algorithms in [29], which is briefly described as follows. For the vp-tree, the root node corresponds to the entire feature space. Among the remaining points, we choose one point as the vantage point. It splits the space into left and right subspaces corresponding to the root's left and right subtree. Repeating the process, a binary tree is formed. Reviewing the construction, one major concern is the algorithm of selecting vantage points. The selection algorithm we used in the one that stated in [29]. It first constructs a set of candidate vantage points by random sampling, and then chooses the best one among them.

Based on these experiment settings, we perform the following experiments

4.2. Result and analysis

As we mentioned before, there are four major factors that affect the performance of the near neighbor searching algorithms. Thus, we perform four groups of experiments. In each group of experiments, we use both the randomly generated data set and the real image data set.

4.2.1. The dimensions of feature space. The first group of experiments shows the effect due to the number of dimensions of the feature space. The results obtained based on the random data set are shown in figure 5. The operation counts of our approach are linear to the number of dimensions of the feature space. Both methods conquer *dimensional curse* (i.e., the number of operation counts quadratically explodes as the number of dimensions increases) [21]. When the number of feature points is less than 8192, our method is better than the vp-tree in all the feature spaces we have experimented. The difference of the operation counts in the 1024-vp and the 1024-MI (denote the vp-tree approach and our approach with 1024 feature points, respectively) cases is larger than that in the 4196-vp and in the 4196-MI cases, which means that our method benefits when the feature space is sparse (i.e., higher dimensions and less feature points). This behavior is more obvious in

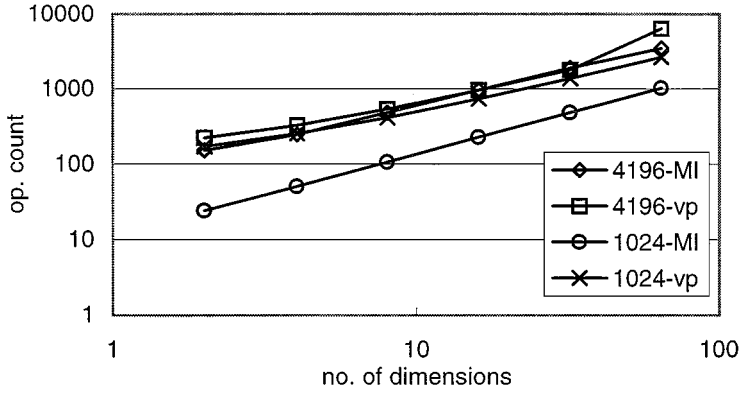


Figure 5. Operation counts vs. number of dimensions of the feature space for random data set.

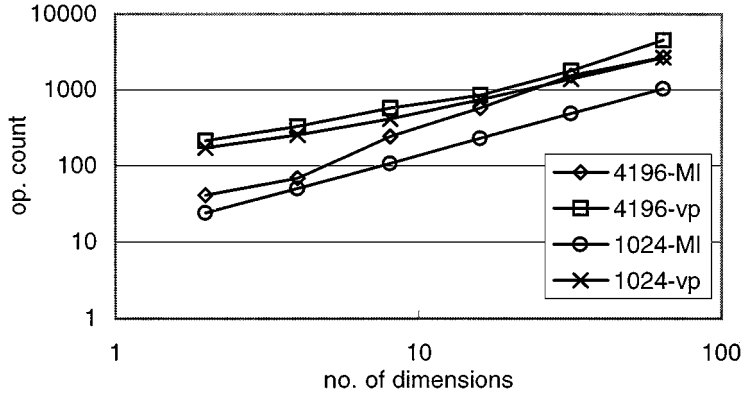


Figure 6. Operation counts vs. number of dimensions of the feature space for real data set.

the real data set cases since the feature points in the real data set are clustered rather than randomly distributed, as shown in figure 6.

4.2.2. The size of the data set. The second group of experiments show the operation counts versus the size of the data set. If the vantage points are well chosen, the operation counts of the experiments using vp-trees grow logarithmically when the number of feature points increases. However, since our method is elimination-based, when the density of data points is high, more candidate points will be retrieved and needed to be merged. Thus, our method is better than vp-tree when the size of data set is small (i.e., the number of feature points is less than 8192 in our experiments) as shown in figure 7. On the contrary, the vp-tree is better than our method when the size is large. In the case of the real data set, since our approach benefits from the skewed characteristics of the real data set, the crossing points of the MI and the vp curves move to 16384, as shown in figure 8.

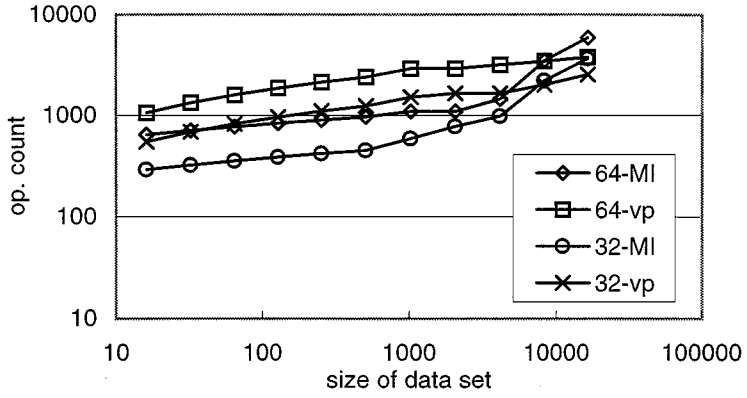


Figure 7. Operation counts vs. size of data set for random data set.

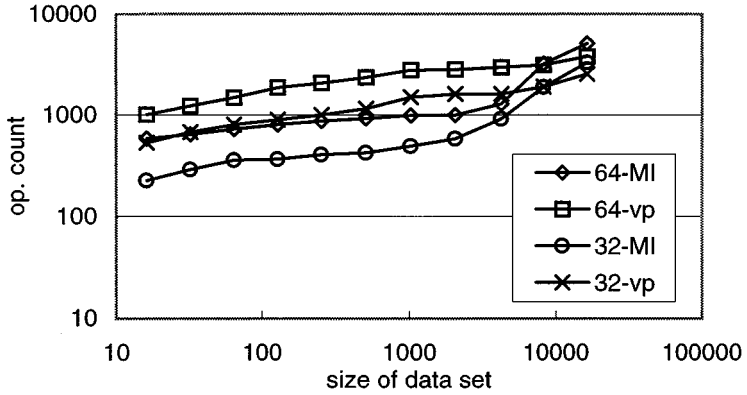


Figure 8. Operation counts vs. size of data set for real data set.

4.2.3. The query radius. In the third group of experiments, we discuss the effect of query radius on the performance of the near neighbor searching algorithms. As figures 9 and 10 show, the values of the query radium can be partitioned into three ranges. When the query radius is small (less than 0.02 in the 1024-MI and the 1024-vp cases), since less candidate points are retrieved, the possibility that the candidate point set merging algorithm needs to be applied is reduced. In this case, the operation counts of our method will be less than that of the vp-tree. When the query radius is in the middle range (between 0.02 and 0.18 in the 1024-MI and the 1024-vp cases), by carefully choosing the vantage points, the vp-tree method only needs to traverse a few branches. On the other hand, the possibility that the candidate point set merging algorithm needs to be applied increases. Thus the vp-tree approach is better than ours in this case. When the query radius is large (larger than 0.18 in the 1024-MI and the 1024-vp cases), since most feature points in a vp-tree should be

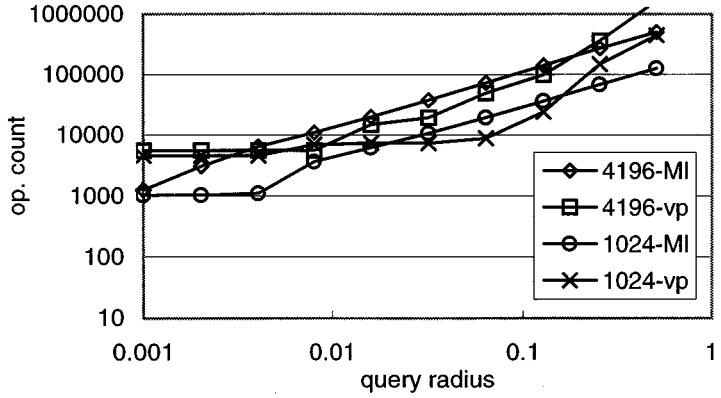


Figure 9. Operation counts vs. size of data set for random data set.

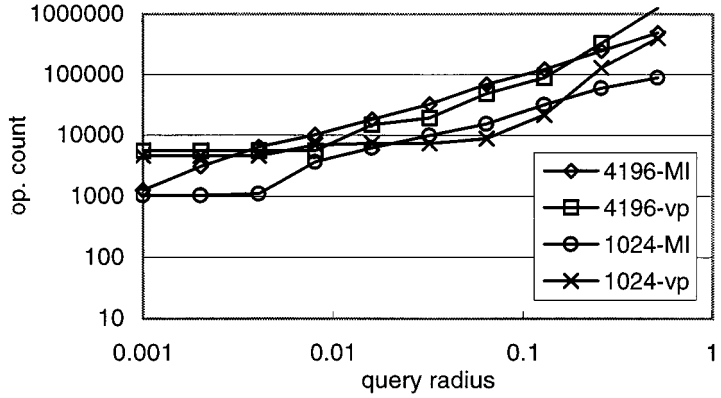


Figure 10. Operation counts vs. size of data set for real data set.

visited, few branches can be pruned by the vp-tree method. Thus, the operation counts of the vp-tree method will be larger than that of ours.

4.2.4. The distance function. In the fourth group of experiments, we discuss the effect of the distance function on the performance of the near neighbor searching algorithms. Basically, a distance function consists of three kinds of operations: the addition/subtraction operations, the multiplication/division operations, and the comparison operations. The execution costs for the three kinds of operations are different, which depend on the computing platform. Since we choose the *operation count* as the measure unit, a relative cost ratio among the three kinds of operations should be assigned. Based on the relative cost ratio, the total operation counts for calculating a distance function can be obtained. The relative cost ratio among the three kinds of operations is termed the *operation ratio.* According to the system statistical information in our experiment platform, the operation ratio among addition, multiplication and comparison is assigned $1:3:1$.

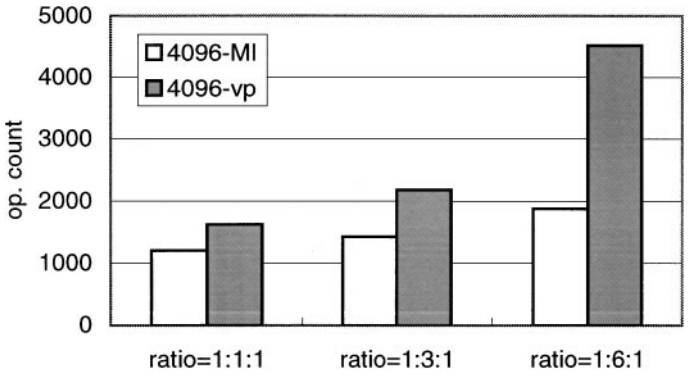


Figure 11. Operation counts vs. operation ratio for real data set.

Since our method eliminates the candidate points as early as possible, the number of the multiplication operations is reduced. Thus our method will benefit when the operation ratio of the multiplication operations increases. The results are shown in figure 11.

5. Conclusion

In this paper, we propose an approach to process near neighbor searching for content-based multimedia data retrieval. In this approach, an index is constructed for the values of feature points of multimedia objects in each dimension. A user can pose content-based query by specifying a multimedia query example and a similarity measure. The specified query example will be transformed into a query point in the multi-dimensional feature space. The candidate points in each dimension are then retrieved by searching the value of the query point in the corresponding dimension. These candidate point sets are merged one by one by removing the candidate points which are not within the query radius. The merged candidate points and their distances to the query point form the answer of the query.

There are three advantages that our method provides: First, if the feature points are clustered together or highly skewed, the query processing time will be very short since the possibility increases that in certain dimension there is no candidate point. Second, our approach benefits when the number of dimensions of the feature space is high. Finally, because the feature points outside the query radius will be eliminated as early as possible, less multiplication operations are needed compared with the vp-tree approach. It does mean that our approach is practical since in real environment and applications the multiplication operation is the most time-consuming operation in calculating the distance function. These properties are also observed throughout a series of experiments we performed.

In the near future, we will integrate the near neighbor searching engine into the *Vega* multimedia database system. The techniques for distributed multimedia query processing will also be investigated. Finally, we are currently developing an efficient approximate string matching mechanism to solve the multimedia feature matching problem.

References

1. N. Bechmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: An efficient and robust access method for points and rectangles," in Proc. of ACM SIGMOD Conf., 1990, pp. 322–331.
2. J.L. Bentley and J.H. Friedman, "Data structure for range searching," ACM Computing Surveys, Vol. 11, No. 4, pp. 397–409, 1979.
3. J.L. Bentley, "Multidimensional binary search trees used for associative searching," Communication of the ACM, Vol. 18, No. 9, pp. 509–517, 1975.
4. S. Brin, "Near neighbor search in large metric spaces," in Proc. of the 21st International Conference on VLDB, 1995, pp. 574–584.
5. A.L.P. Chen, C.C. Liu, K.L. Lee, and C.C. Chen, "The design of a video database system," in Proc. Real-Time and Media Systems, 1995.
6. T.C. Chiueh, "Content-based image indexing," in Proc. of 20th VLDB Conf., 1994, pp. 582–593.
7. T.C. Chou, A.L.P. Chen, and C.C. Liu, "Music databases: Indexing techniques and implementation," in Proc. IEEE Intl. Workshop on Multimedia Data Base Management Systems, 1996.
8. N. Dimitrova and F. Golshani, "Motion recovery for video content classification," ACM Trans. on Info. Sys., Vol. 13, No. 4, pp. 408–439, 1995.

9. C. Faloutsos and K.-I. Lin, "FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets," in Proc. of ACM SIGMOD Conf., 1995.
10. R.A. Finkel and J.L. Bentley, "Quad trees: A data structure for retrieval on composite keys," Acta Informatica, Vol. 4, No. 1, pp. 1–9, 1974.
11. M. Freeston, "A general solution of the n-dimensional B-tree problem," in Proc. of ACM SIGMOD Conf., 1995, pp. 80–91.
12. V.N. Gudivada and V.V. Raghavan, "Design and evaluation of algorithms for image retrieval by spatial similarity," ACM Trans. on Info. Sys., Vol. 13, No. 2, pp. 115–144, 1995.
13. R.H. Guting, "An introduction to spatial database systems," VLDB Journal, Vol. 3, No. 4, pp. 357–399, 1994.
14. A. Guttman, "R-trees: A dynamic index structure for spatial searching," in Proc. of ACM SIGMOD Conf., 1984, pp. 47–75.
15. H.V. Jagadish, "A retrieval technique for similar shapes," in Proc. of ACM SIGMOD Conf., 1991, pp. 208–217.
16. T.C.T. Kuo, Y.B. Lin, A.L.P. Chen, S.C. Chen, and C.Y. Ni, "Efficient shot change detection on compressed video data," in Proc. IEEE Intl. Workshop on Multimedia Database Management Systems, 1996.
17. C.C. Liu and A.L.P. Chen, "Modeling and query processing of distributed multimedia databases," in Proc. Real-Time and Media Systems, 1996.
18. C.C. Liu and A.L.P. Chen, "3D-List: A data structure for efficient video query processing," IEEE Trans. on TKDE, to appear.
19. C.C. Liu and A.L.P. Chen, "Vega: A multimedia database system supporting content-based retrieval," Information Science and Engineering, 1997.
20. C.C. Liu and A.L.P. Chen, "1D-List: A data structure for efficient approximate string matching," NTHU Technical Report.
21. K.-I. Lin, H.V. Jagadish, and C. Faloutsos, "The TV-tree: An index structure for high-dimensional data," VLDB Journal, Vol. 3, No. 4, pp. 519–544, 1994.
22. J. Nievrgelt, H. Hinterberger, and K.C. Sevcik, "The grid file: An adaptable, symmetric multikey file structure," ACM Transactions on Database Systems, Vol. 9, No. 1, pp. 38–71, 1984.
23. N. Roussopoulos, S. Kelly, and F. Vincent, "Nearest neighbor query," in Proc. of ACM SIGMOD Conf., 1995, pp. 71–79.
24. H. Samet, The Design and Analysis of Spatial Data Structures, Addison-Wesley, 1990.
25. T. Sellis, N. Roussopoulos, and C. Faloutsoso, "The R+ tree: A dynamic index for multi-dimensional objects," in Proc. of the 13th International Conference on VLDB, 1987, pp. 507–518.
26. S.W. Smoliar and H.J. Zhang, "Content-based video indexing and retrieval," IEEE Multimedia, Vol. 1, No. 2, pp. 62–72, 1994.
27. T.T.Y. Wai and A.L.P. Chen, "Retrieving videodata via motion tracks of content symbols," in Proc. of ACM International Conference on Information and Knowledge Management (CIKM).
28. R. Weiss, A. Duda, and D.K. Gifford, "Composition and search with a video algebra," IEEE Multimedia, Vol. 2, No. 1, pp. 12–25, 1995.
29. N. Peter Yianilos, "Data structures and algorithms for nearest neighbor search in general metric spaces," in Proc. of the Forth Annual ACM-SIAM Symposium on Discrete Algorithm, 1993, pp. 311–321.
30. A. Yoshitaka, S. Kishida, M. Hirakawa and T. Ichikawa, "Knowledge-assisted content-based retrieval for multimedia databases," IEEE Multimedia, Vol. 1, No. 4, pp. 12–21, 1994.

Chih-Chin Liu was born in Hsinchu, Taiwan, R.O.C. on October 30, 1968. He received his B.S. degree in

electrical engineering in 1990 and the Ph.D. degree in computer science in 1998, both from National Tsing Hua University, Hsinchu, Taiwan. He is currently a researcher in Computer and Communications Research Laboratories at Industrial Technology and Research Institute. His research interests include multimedia databases, wireless multimedia communications, and mobile internet.

Jia-Lien Hsu is currently a Ph.D. candidate in Computer Science at National Tsing Hua University. He received the B.S. degree in Computer Science from National Tsing Hua University, Hsinchu, Taiwan, R.O.C. in 1994. His research interests include multimedia databases, music databases and content-based retrieval.

Arbee L.P. Chen received the B.S. degree in computer science from National Chiao Tung University, Taiwan, R.O.C. in 1977, and the Ph.D. degree in computer engineering from the University of Southern California in 1984. He joined National Tsing Hua University, Taiwan, in August 1990, and became a Professor in the Department of Computer Science in 1991. He was a Member of Technical Staff at Bell Communications Research, New Jersey, from 1987 to 1990, an Adjunct Associate Professor in the Department of Electrical Engineering and Computer Science, Polytechnic University, New York, and a Research Scientist at Unisys, California, from 1985 to 1986. His current research interests include multimedia databases, data mining and mobile computing. Dr. Chen organized the 1995 IEEE Data Engineering Conference and the 1999 International Conference on Database Systems for Advanced Applications (DASFAA). He received a Distinguished Research Award in 1996–1997 and 1998–1999.