Analysis of Nearest Neighbor Query Performance in Multidimensional Index Structures

Guang-Ho Cha
Dept. of Multimedia Engineering, Tongmyong University of Information Technology, Pusan, South Korea

Ho-Hyun Park, Chin-Wan Chung
Dept. of Computer Science, Korea Advanced Institute of Science and Technology, Taejon, South Korea

Abstract

A frequently encountered type of query in geographic information systems and multimedia database systems is to find k nearest neighbors to a given point in a multidimensional space. Examples would be to find the nearest bus stop to a given location or to find some most similar images when an image is given. In this paper, we develop an analytic formula that estimates the performance for nearest neighbor queries and characterize the efficiency of multidimensional index structures for nearest neighbor queries. The developed formula can be used directly in the query optimizers and the characteristics of efficiency will become the basis for the design of the index structure. Experimental results show that our analytic formula is accurate within some acceptable error range. It is exhibited that the efficiency of the index structure depends on the storage utilization and the directory coverage of it.

1. Introduction

A great variety of applications require to find near neighbors in a multidimensional space. For example, typical applications are: finding the nearest MRI brain scans to an given image in medical image databases, retrieving video shots containing the frame similar to a given image in video databases, finding near hotels from a given location in geographic information systems, finding similar DNA's from a large genetics databases, etc.

Efficient processing of nearest neighbor queries requires an efficient index structure which capitalizes on the similarity of the objects to focus the search on the potential neighbors. In recent years, some index structures and algorithms have been developed for fast retrieval of near neighbors in a multidimensional space [1, 3, 7]. However, no attempt has been made to analyze the performance of the nearest neighbor queries on index structures as far as we know. The focus of most analyses [4, 6, 9] has been on the performance of range queries. In this paper, we develop an analytic formula to predict the performance of the nearest neighbor query in multidimensional index structures and investigate the factors that influence the performance of the nearest neighbor query. The developed formula can be used directly in the query optimizers and the investigated performance factors will be used for the basis of the design of multidimensional index structures.

2. Background

2.1 Basic Concepts of Multidimensional Index Structures

Most of multidimensional index structures have a tree structure which consists of internal and leaf nodes. A *leaf node* contains at most C_l entries of the form

$$(oid, R)$$

where C_l is the capacity of a leaf node, *oid* is a pointer to the object in the database, and R is a representative of the real object, e.g., it is a minimum bounding rectangle for the R-tree [5] or an n-dimensional feature vector for image databases [2, 3]. An *internal node* contains at most C_n entries of the form

$$(ptr, R)$$

where C_n is the capacity of an internal node, *ptr* is a pointer to the child node, and R is a bounding device that encloses all objects at lower levels, e.g., it is a minimum bounding rectangle in the R-tree or a minimum bounding interval in the HG-tree [2].

2.2 Basic Definitions

Let n be the dimensionality of the data space, $W_i = [0,1)$, $1 \le i \le n$, and $W = W_1 \times W_2 \times \ldots \times W_n$ be the n-dimensional unit data space in which all data objects are defined. Let us assume that for storing N data objects the index structure consumes m nodes s_1, $s_2, \ldots, s_m$, each corresponds to one disk page. The *directory region*, $DR(s_j)$ of node s_j is a minimal n-dimensional bounding rectangle enclosing all objects in s_j. This corresponds to the node entry component R described in the above subsection. We assume that $DR(s_j)$ is represented by $[l_1, r_1] \times \ldots \times [l_n, r_n]$, $l_i, r_i \in W_i$, $l_i \le r_i$ for $1 \le i \le n$. The *directory coverage*, $C_d(T)$, of the index structure T is defined by the union of the directory regions of all leaf nodes in the index structure [2]:

$$C_d(T) = \bigcup_{i=1}^{k} DR(s_i), \text{ where } k \text{ is the number of leaf nodes in the index structure.}$$

The *storage utilization* (U) of an index structure can be defined as follows [2]:

$$U = \frac{1}{m} \sum_{i=1}^{m} \frac{F_i}{P_i}$$

where F_i is the number of entries in node i, the P_i is the maximum number of entries that a node i can have, and m is the total number of nodes in the index structure.

Let D be a distance function defined in n-dimensional data space. Given a point $(x_1, x_2, \ldots, x_n)$ and a positive integer k, the k-nearest neighbor query finds the k nearest neighbors on $(x_1, x_2, \ldots, x_n)$ with respect to the distance function D. The typical way to compute the distance between two points is using the Euclidean distance. Let $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, y_2, \ldots, y_n)$ be two points in an n-dimensional space. Then the Euclidean distance, $D_2(X,Y)$ between X and Y is as follows:

$$D_2(X, Y) = \left[\sum_{i=1}^{n} |x_i - y_i|^2 \right]^{1/2}$$

3. Analysis of Index Structures for Nearest Neighbor Queries

In this section we develop an analytic formula to evaluate the average response time

for k-nearest neighbor queries. For the performance measure of the query processing, we will use the number of disk accesses necessary to perform k-nearest neighbor query because the average query cost is dominated by disk accesses. And we will use the Euclidean distance as the (dis)similarity measure.

Nearest neighbor search algorithms should focus on eliminating unlikely candidates rather than pin-pointing the targets directly to find the k nearest neighbors, because a data object is not the nearest neighbor only when the index subsystem declares that it is not [3]. Thus it is not clear how to estimate the performance of nearest neighbor queries as compared with range queries in which a specific query region is given and only the nodes overlapping the regions are necessary to be inspected. Contrary to the range queries where the query region is fixed, the query region for the k-nearest neighbor query is widely variable depending on the query location and the data distribution. These problems make the analysis of the nearest neighbor query performance difficult, and may cause the relative errors of the performance analysis to be fluctuated.

At first, we provide an example to get an intuition for the analysis of the nearest neighbor query performance.

Example 1. Fig. 1 shows a collection of directory regions numbered 1 to 12 on a normalized 2-dimensional unit domain space, organized in a multidimensional index structure with fanout = 3. Let us assume that we want to find 5 nearest neighbors from the query location denoted by + in Fig. 1. Then the dotted circle will be the minimal search area of the 5-nearest neighbor query, i.e., all 5 nearest points are within the dotted circle. The directory regions numbered by 1, 3, 5, 10, 11 should be inspected.

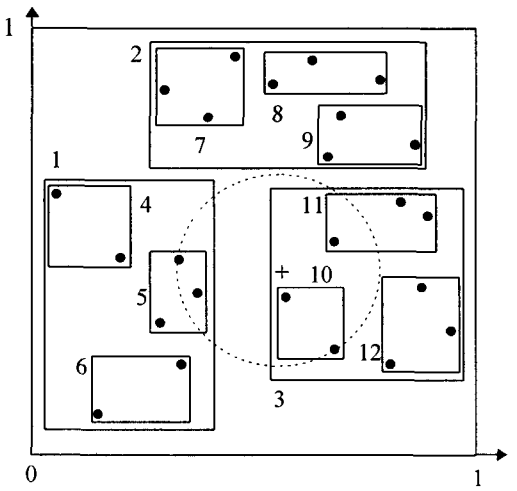


Fig. 1. Data (bullets) organized in the multidimensional index structure with fanout = 3.

Considering this observation, we can catch an insight that the smallest area needed to process the k-nearest neighbor query is the n-dimensional sphere containing the k nearest objects from the query location. Therefore, in the index structure, we can estimate the cost necessary to process the k-nearest neighbor query by computing the number of nodes that intersect the n-dimensional sphere. Then the expected number

DA of disk accesses needed to perform k-nearest neighbor query Q is given by

$$\sum_{i=1}^{m} Probability(R(k) \cap DR(s_i)) \quad (1)$$

where $R(k)$ is the expected query region covered by Q and m is the total number of nodes in the index structure. Formula (1) says that every node s_i whose directory region $DR(s_i)$ is overlapping the query region $R(k)$ must be inspected. The $R(k)$ becomes the volume $S_n(r)$ of the n-dimensional sphere containing k nearest objects. We will call it *hyper-volume*. We can get the size of each directory region $DR(s_i)$ from the index structure or can compute from the characteristics of the data set [9]. Table 1 gives the summary of symbols and their definitions that will be used in the paper.

Symbols	Definitions
n	number of dimensions
N	number of data points in the index structure
m	number of nodes in the index structure
s_i	i-th node in the index structure
$DR(s_i)$	directory region of node s_i
k	number of neighbors to find
$R(k)$	area covered by k objects
α	value to correct the area covered by k objects
$B(n)$	n-dimensional sphere
r	radius of the n-dimensional sphere
$S_n(r)$	hyper-volume of the n-dimensional sphere with radius r
DA	number of disk accesses for a k-nearest neighbor query

Table 1. Summary of symbols and Definitions

Contrary to the range query where the query region is given directly, in the nearest neighbor query, determining the area $R(k)$ covered by k objects is an important problem because the area $R(k)$ is widely variable depending on the query location and the data distribution. Since the shape of the area covered by the k-nearest neighbor query is determined by n-dimensional sphere, i.e., $R(k) = S_n(r)$, we must compute $R(k)$ and $S_n(r)$.

Assumed that N uniformly distributed points are stored in a multidimensional index structure and the domain space is n-dimensional unit data space $[0,1)^n$, the area $R(k)$ covered by k points out of a total N points will be k/N. However, this value of $R(k)$ would sometimes lead to inaccurate estimations because $R(k)$ is widely variable depending on the query location and data distributions. Let us consider Examples 2 and 3.

Example 2. Fig. 2 shows two 3-nearest neighbor queries Q_1 and Q_2 in which their query locations are at + and ×, respectively. All data points are clustered within the area denoted by A. The query locations of Q_1 and Q_2 lie outside and inside A, respectively. Two areas covered by Q_1 and Q_2 to include 3 nearest neighbors show significant difference.

The $R(k)$ is also dependent on the data distribution. For example, in uniform distribution, the size of area $R(k)$ may be independent of the query location. On the other

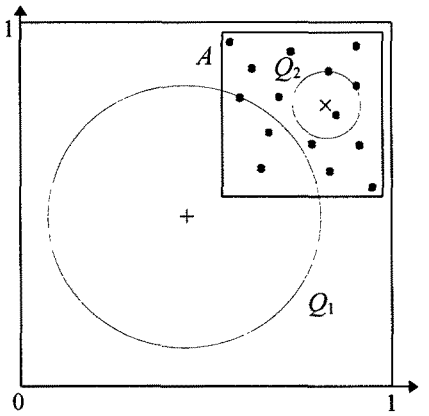
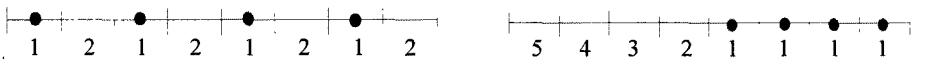


Fig. 2 Two 3-nearest neighbor queries with query points located in + and ×

hand, the influence of the query location on the size of $R(k)$ may be severe in skewed distribution. In particular, when the area covered by the whole data set is smaller, the effect of the query location to the size of $R(k)$ is severer. We give the next example to illustrate the effect how the area covered by the data set influences the size of $R(k)$.

Example 3. Fig. 3 illustrates two sample data distributions on a 1-dimensional space. The bullets on line segments represent data points. We assume that the whole data space is 8 units. Assuming that the search direction is restricted to only one direction, the number beneath each line segment represents the area (= the number of units) that must be searched to find 1-nearest neighbor when the query location lies in that segment. Summing up the results of 1-nearest neighbor query at each line segment, the sums of areas that must to be searched are 12 (= 1+2+1+2+1+2+1+2) units and 18 (= 5+4+3+2+1+1+1+1) units in (a) and (b), respectively.



(a) uniform distribution (b) skewed distribution

Fig. 3 Two sample data distributions on a 1-dimensional line

From the above example, we can get the knowledge that the area that must be searched to find k nearest neighbors tends to increase as the area covered by the data set becomes smaller when the query location is uniformly distributed. Based on the above observation, we correct the area $R(k)$ as follows:

$$R(k) = \frac{k}{N\alpha} \tag{2}$$

where α is a heuristic value in $(0,1]$ to correct $R(k)$. When the data set has a uniform-like random distribution, α becomes 1. As the area covered by the data set is smaller, we make the α smaller. Smaller area covered by the data set makes it lower that the probability of laying the query location inside the area covered by the directory regions, and thus it tends to increase the size of $R(k)$.

Now, let us find the hyper-volume $S_n(r)$ and radius r of the n-dimensional sphere $B(n)$. Starting from 2-dimensional data space, we will generalize them to n-dimension. In the case of 2-dimension, we have the area $S_2(r) = \pi r^2$ of the circle $B(2)$ and can get the radius r from $\pi r^2 = R(k)$:

$$r = \sqrt{\frac{R(k)}{\pi}}$$

Generalizing the problem to n-dimension, we can compute recursively the hyper-volume S_n of the n-dimensional sphere $B(n)$ as follows:

$$S_n(r) = \int_{-r}^{r} S_{n-1}(f(x))\,dx, \quad n \geq 2 \tag{3}$$
$$S_1(r) = 2r,$$
$$f(x) = \sqrt{r^2 - x^2}$$

For n=2 and n=3, $S_2 = \pi r^2$ and $S_3 = (4/3)\pi r^3$, respectively. We can compute the radius r of an n-dimensional sphere from Eq. 2 = Eq. 3. The hyper-volume of this n-dimensional sphere $B(n)$ with radius r is the estimator of the area covered by the k nearest neighbors from the query location.

Our goal is to estimate the expected number of disk accesses by finding the probability that the n-dimensional sphere $B(n)$ with radius r intersects a directory region $DR(s_i)$. To simplify the situation, we start from 2-dimensional space and then generalize it to n-dimension. As an example, see Fig. 4. $W = [0,1)^2$ is a 2-dimensional unit data space. A square DR is a directory region. Circle C with radius r represents the search area $R(k)$ including the k nearest neighbors, whose center is a query location. The probability that the directory region DR intersects the query circle C becomes the probability that the directory node may be accessed in the nearest neighbor query. We get

$$Probability(C \cap DR) = area(P) = area(DR) + perimeter(DR) \cdot r + area(C)$$

Summing it up to the index structure with m nodes, we obtain the expected number $DA(2)$ of disk accesses on a 2-dimensional space using Eq. 1:

$$DA(2) = \sum_{i=1}^{m} Probability(C \cap DR(s_i))$$
$$= \sum_{i=1}^{m} area(DR(s_i)) + r \sum_{i=1}^{m} perimeter(DR(s_i)) + area(C) \cdot m$$

Generalizing it to n-dimension we derive the following formula for DA:

$$DA(n) = \sum_{i=1}^{m} hypervolume_n(DR(s_i)) + r \sum_{i=1}^{m} hypervolume_{n-1}(DR(s_i))$$
$$+ \sum_{i=1}^{m} \left(\sum_{j=2}^{n} \frac{1}{2^j} hypervolume_{n-j}(DR(s_i)) \cdot S_j(r) \right) \tag{4}$$

This the estimator to evaluate the number of disk accesses necessary to perform the k-nearest neighbor query on n-dimensional data space. By the function $hypervolume_n(x)$ we refer to the hyper-volume of an n-dimensional solid x. For example, n=3, n=2, n=1, and n=0, $hypervolume_3(x)$ = volume of x, $hypervolume_2(x)$ = area of x, *hy-*

$pervolume_1(x)$ = perimeter of x, and $hypervolume_0(x)$ = number of vertices of x, respectively. $S_j(r)$ is the hyper-volume of the j-dimensional sphere with radius r.

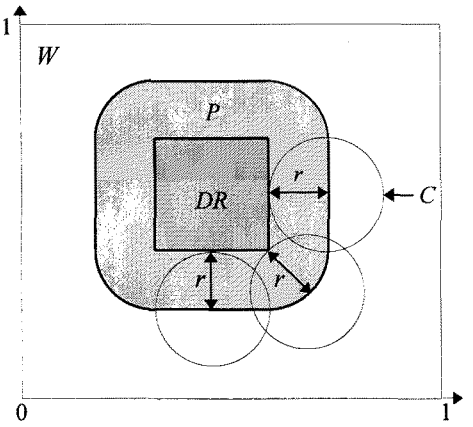


Fig. 4 Probability P that the directory region DR intersects the query circle C

We can make several comments related to the above formula. The performance of the nearest neighbor query relies on three factors: the total number of nodes in the index structure, the hyper-volumes of the directory regions, and the area covered by k nearest objects, which are represented by m, $hypervolume_j(DR(s_i))$, and $S_j(r)$, respectively, in Eq. 4. The size of m is mainly affected by the *storage utilization* of the index structure. Higher storage utilization reduces the total number of nodes, m, in the index structure. The $hypervolume_n(DR(s_i))$ is defined by the *directory coverage* of the index structure. Smaller directory coverage decreases the size of the $hypervolume_n(DR(s_i))$. In addition, all of the j-th hyper-volume $hypervolume_j(DR(s_i))$ are also important factors. The search area $S_j(r)$ is a j-dimensional sphere defined by the required number of objects out of a total number of objects.

4. Experimental Results and Analysis

In order to evaluate our analysis we carried out several experiments on various data distributions. We employed the HG-tree [2] and the buddy-tree [8] as underlying multidimensional index structures for our experiments, but other structures such as R-tree can be used. The nearest neighbor search is based on the algorithm given by Roussopoulos et al. [7]. During each experiment, the 100,000 4-dimensional points from the data space $[0,1)^4$ has been inserted into the initially empty HG-tree and buddy-tree. In order to achieve statistically significant results the node size was set to 512 bytes which is at the lower end of realistic node sizes [8]. For our tests we used four groups of data sets as in [8]: uniform distribution, clustered distribution, bit distribution, and diagonal distribution.

We based our tests on the number of nodes (= pages) retrieved by k-nearest neighbor query. In Figs. 5 to 8 we see the average of 1,000 queries for each of several different number of nearest neighbors. In each experiment, we used the following heuristic value of α:

$$\alpha = (\text{the area covered by the data set})^{1/n}$$

where n is the dimensionality of the data space. In practice, we approximated the area covered by the data set by the directory coverage of the index structure.

In the experiments, we used two groups of nearest neighbor queries. In one group of queries, query locations are distributed uniformly. Query locations of the other group follow the distributions similar to the data set. The experimental result of the former group is represented by Experimental1 and that of the latter group is represented by Experimental2 in Figs. 6 to 8. In the data set with uniform distribution, only the uniformly distributed queries were tested because the query locations of these queries follow the distributions of the data set.

The observation of the results in uniform distribution is that the analytical estimate is close to the actual result: the relative error is below 9%. In clustered distribution, on the other hand, the errors in the queries with small number of neighbors are relatively large compared with those of the uniform distribution. However, the average errors are within an acceptable range (less than 14%). In the bit distribution, all relative errors are below 16%. The errors in the diagonal distribution are large. However, when the distribution of the query locations is similar to the distribution of the data set, the average error is acceptable (less than 19%).

In the case that the ratio of the area covered by the data set to the whole data space is very small, which is the case of our diagonal distribution, the relative errors tend to be large. In this case, assuming that query locations are uniformly distributed, the probability that the query locations lie close to the desired data points or directory regions is very low. Therefore, the area $R(k)$ is very large, and thus it may cause the difference between the estimated area $R(k)$ and the actual area covered by k points to be high. Compared to the analysis of the range query performance [4, 9], the reason that the error rates in the analysis of the nearest neighbor query performance are somewhat high is because the fluctuation of the area $R(k)$ is relatively high depending on the query location and the data distribution. However, since we may make an optimistic assumption that the query locations given by users are not so far away from desired data objects, we may expect that the actual error rates are lower than those in the experimental results. For example, in content-based image databases, users may provide an query image similar to what they want to retrieve.

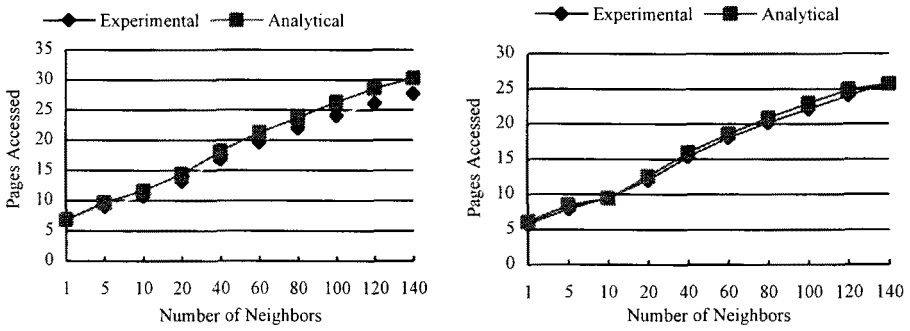


Fig. 5 Experimental vs. analytical results in uniform data distributions

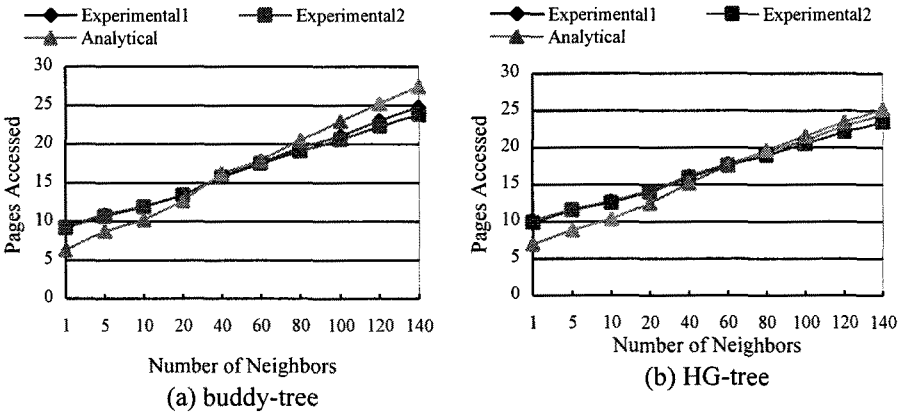


Fig. 6 Experimental vs. analytical results in clustered distributions

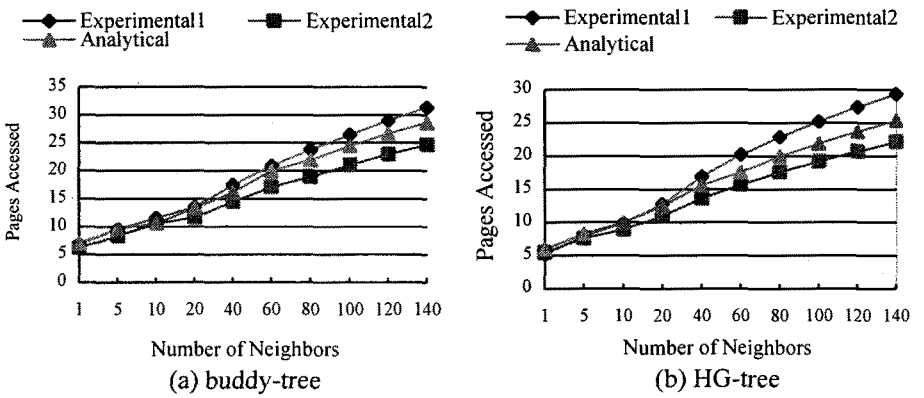


Fig. 7 Experimental vs. analytical results in bit distributions

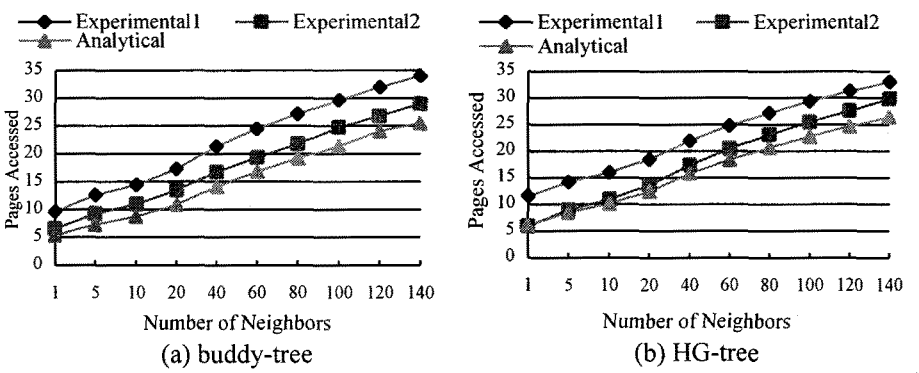


Fig. 8 Experimental vs. analytical results in diagonal distributions

5. Conclusions

There are two contributions in this work. First, we derived an analytic formula that can be used as a good estimate for the cost of nearest neighbor query performance. Using it we estimated the performance of the nearest neighbor query on various data distributions and showed that the analysis is applicable for prediction of the nearest neighbor query performance. Experimental results show that the estimation errors are within an acceptable range in most case, the average error is usually around 5%-20%. Even in the case that the area covered by the data set is extremely small, the average error is below 19% if the query distribution is similar to the data distribution.

Second, we showed that the storage utilization, the directory coverage, the hyper-volume of the directory region, and the area covered by the required k data objects are the major factors that influence the performance of the nearest neighbor query. These factors will become the basis for the design of multidimensional index structures. Up to now, it has not yet been well known what are the critical factors that influence the nearest neighbor query performance.

Future work could focus on more accurate estimation of the area $R(k)$covered by k objects out of a total of N data objects. Here we need a technique for adjusting the heuristic value α depending on the data distribution or the directory coverage of the index structure.

References

[1] S. Brin, "Near Neighbor Search in Large Metric Spaces," *Proc. of the 21th Int'l. Conf. on VLDB*, pp. 574-584, 1995.

[2] G.-H. Cha and C.-W. Chung, "HG-tree: An Index Structure for Multimedia Databases," *Proc. of the IEEE Int'l. Conf. on Multimedia Computing and Systems*, pp. 449-452, June 1996.

[3] T. Chiueh, "Content-Based Image Indexing," *Proc. of the 20th Int'l. Conf. on VLDB*, pp. 582-593, 1994.

[4] C. Faloutsos and I. Kamel, "Beyond Uniformity and Independence: Analysis of R-trees Using the Concept of Fractal Dimension," *Proc. of the 13th ACM Symposium on Principles of Database Systems*, pp. 4-13, 1994.

[5] A. Guttman, "R-trees: a dynamic index structures for spatial searching," *Proc. of the ACM SIGMOD Int'l. Conf. on Management of Data*, pp. 47-57, 1984.

[6] B.-U. Pagel, H.-W. Six, H. Toben, and P. Widmayer, "Towards an Analysis of Range Query Performance in Spatial Data Structures," *Proc. of the 12th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pp. 214-221, 1993.

[7] N. Roussopoulos, S. Kelley, and F. Vincent, "Nearest Neighbor Queries," *Proc. of the ACM SIGMOD Int'l. Conf. on Management of Data*, pp. 71-79, 1995.

[8] B. Seeger and H.-P. Kriegel, "The Buddy-Tree: An Efficient and Robust Access Method for Spatial Data Base Systems," *Proc. of the 16th Int'l. Conf. on VLDB*, pp. 590-601, 1990.

[9] Y. Theodoridis and T. Sellis, "A Model for Prediction of R-tree Performance," *Proc. of the 16th ACM SIGACT-SIDMOD-SIGART Symposium on Principles of Database Systems*, pp. 161-169, 1996.