Similarity Search for Adaptive Ellipsoid Queries Using Spatial Transformation

Yasushi Sakurai† Masatoshi Yoshikawa§ Ryoji Kataoka† Shunsuke Uemura§

† NTT Cyber Space Laboratories
{*ysakurai, kataoka*}*@dq.isl.ntt.co.jp*

§ Nara Institute of Science and Technology
{*yosikawa, uemura*}*@is.aist-nara.ac.jp*

Abstract

Similarity retrieval mechanisms should utilize generalized quadratic form distance functions as well as the Euclidean distance function since ellipsoid queries parameters may vary with the user and situation. In this paper, we present the spatial transformation technique that yields a new search method for adaptive ellipsoid queries with quadratic form distance functions. The basic idea is to transform the bounding rectangles in the original space, wherein distance from a query point is measured by quadratic form distance functions, into spatial objects in a new space wherein distance is measured by Euclidean distance functions. Our method significantly reduces CPU cost due to the distance approximation by the spatial transformation; exact distance evaluations are avoided for most of the accessed bounding rectangles in the index structures. We also present the multiple spatial transformation technique as an extension of the spatial transformation technique. The multiple spatial transformation technique adjusts the tree structures to suit typical ellipsoid queries; the search algorithm utilizes the adjusted structure. This technique reduces both page accesses and CPU time for ellipsoid queries. Experiments using various matrices and index structures demonstrate the superiority of the proposed methods.

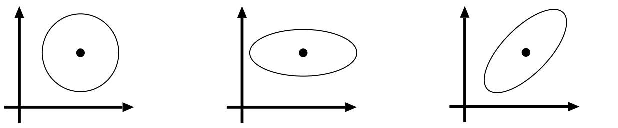
1 Introduction

Multimedia content-based retrieval systems use feature values extracted from multimedia data; they find data objects whose feature values are most similar to those of the query object. These systems include various pattern recognition mechanisms, and the databases on which they operate continue to grow in size. This means that, multimedia systems and spatial databases require (1) information retrieval methods with more general distance functions, and (2) improved search performance.

Since the Euclidean distance space makes all dimensions independent of each other, it fails to adequately represent the user's intention. Therefore, multimedia systems require the use of generalized quadratic form distance functions as well as the Euclidean distance function. Since quadratic form distance functions can represent correlations between dimensions, retrieval mechanisms using them have high search quality [HSE$^+$95]. The quadratic form distance function $d_M^2(p,q) = (p-q) \cdot M \cdot (p-q)^t$ is calculated from a query matrix M which is positive definite (i.e. $d_M^2(p,q) > 0$), where q is a query point and p is a data object in a data set. In d-dimensional spaces, the Euclidean distance function has circles for isosurfaces, and weighted Euclidean distance functions correspond to iso-oriented ellipsoids, whose major axis is aligned to the coordinate axis. Quadratic form distance functions have arbitrarily oriented ellipsoids that are not necessarily aligned to the coordinate axis (see Figure 1). Quadratic form distance functions are regarded as a generalization of the Euclidean distance function and weighted Euclidean distance functions. MindReader [ISF98] is an example of the application of quadratic form distance functions; based on relevance feedback, it guesses the correlations between dimensions, which reflect the user's preference. Unlike the Euclidean distance function, quadratic form distance functions more faithfully reflect the user's intention.

Given that the size of multimedia databases will continue to grow and that the dimensionality of feature data will continue to increase, high-performance data retrieval methods are essential. Many index methods have been proposed so far [GG98]. They

Proceedings of the 27th VLDB Conference,
Roma, Italy, 2001



(a) Euclidean (b) weighted Euclidean (c) quadratic form

Figure 1: Isosurfaces for various distance functions.

include data-partitioning index trees (e.g. the R*-tree [BKSS90], the X-tree [BKK96] and the A-tree [SYUK00]). Nearest neighbor search methods using such indices have also been proposed [RKV95] [HS95]. In particular, the A-tree is reported to offer good performance for high-dimensional data [SYUK00]. Unfortunately, most spatial access methods were designed for searches based on the Euclidean distance function, so new spatial access methods that are suited to ellipsoid queries based on quadratic form distance functions are needed. In addition, image retrieval mechanisms using various user-adaptable distance functions [FSA$^+$95] [HSE$^+$95] and relevance feedback mechanisms that guess the user's desires, such as MARS [RHM97] and MindReader [ISF98], deal with queries whose parameters can vary with the user and situation. These mechanisms require search methods that can support adaptive queries.

The goal of our work is to create a search method for adaptive ellipsoid queries that can find similar objects efficiently. Various metric indices (e.g. the M-tree [CPZ97] and the mvp-tree [BO97]) have been proposed as indexing methods for arbitrary distance functions. However, these indices cannot be applied to systems that handle changeable distance functions and, thus, they are not functionally adequate to support adaptive ellipsoid queries. In [SK97], Seidl et al. presented a search algorithm for adaptive ellipsoid queries on index structures that calculates exact distances between query points and MBRs (Minimum Bounding Rectangles). In [ABKS98], Ankerst et al. presented a search method that reduces the number of exact quadratic form distance calculations needed and so reduces the CPU time by using MBB (Minimum Bounding Box) distance functions and MBS (Minimum Bounding Sphere) distance functions, which we call the MBB-MBS approximation technique in this paper. However, this technique's search cost increases as dimensionality grows or as the query ellipsoid becomes flatter.

To overcome the disadvantages of the MBB-MBS approximation technique for ellipsoid queries, we have developed an approximation technique, the Spatial Transformation Technique (STT). The basic idea of STT is to transform the MBRs, whose distance from a query point is measured by the quadratic form distance functions, into rectangles, whose distance is measured by the Euclidean distance functions. Its approximation quality is high even when the flatness of query ellipsoids and dimensionality are high, and yet STT has a low CPU cost. We also developed the Multiple Spatial Transformation Technique (MSTT). MSTT adjusts the tree structures to suit typical ellipsoid queries; this technique reduces the number of page accesses as well as the CPU cost because the search algorithm utilizes the adjusted structure.

The remainder of this paper is organized as follows. Section 2 is a summary of the analysis of the MBB-MBS approximation technique. Section 3 describes the motivation, definitions and algorithms of STT. Section 4 describes MSTT. Section 5 gives the results of a performance evaluation of STT and MSTT. Finally, Section 6 concludes the paper.

2 Problems of Search Methods for Adaptive Ellipsoid Queries

This section summarizes the properties and problems of the MBB-MBS approximation technique.

2.1 Search Methods for Adaptive Ellipsoid Queries

The search algorithm of [SK97] for ellipsoid queries on index structures calculates exact distances between query points and MBRs (Minimum Bounding Rectangles). This search method supports adaptive ellipsoid queries with variable distance functions by using index structures. However, the calculations of distance between query points and MBRs incurs CPU costs as high as $O(\omega \cdot d^2)$ time, where d is dimensionality and ω denotes the number of iterations. The CPU time represents a high percentage of the overall search time. Ankerst et al. [ABKS98] developed the MBB-MBS approximation technique that reduces the number of exact quadratic form distance calculations needed (and, in so doing, reduces the CPU time for ellipsoid queries) by using MBB (Minimum Bounding Box) distance functions and MBS (Minimum Bounding Sphere) distance functions. The following definitions formalize the MBB distance function and the MBS distance function for d-dimensional spaces:

$$\begin{aligned} d^2_{MBB(M)}(p,q) &= \max_{i=1}^{d}\left(\frac{(p_i-q_i)^2}{(M^{-1})_{ii}}\right), && (1)\\ d^2_{MBS(M)}(p,q) &= \lambda_{M_{min}} \cdot (p-q)^2, && (2)\end{aligned}$$

where λ_{M_i} $(i = 1, \cdots, d)$ are the eigenvalues of M, and $\lambda_{M_{min}}$ is the lowest eigenvalue of M. The MBB distance functions approximate an ellipsoid query area by a bounding box that totally encloses the query area. The MBS distance functions use a bounding sphere for the approximation. Both approximation techniques require $O(d)$ time for their calculations.

2.2 Summary of Experimental Evaluation and Analysis

We have performed extensive experiments to analyze the MBB-MBS approximation technique. In the evaluation, we varied matrix flatness and dimensionality.

The flatness of a query matrix M ($det(M) = 1$) is evaluated as follows:

$$\sigma_M^2 = \sum_{i=1}^{d} (\lambda_{M_i} - \overline{\lambda}_M)^2, \quad \overline{\lambda}_M = \sum_{j=0}^{d} \frac{\lambda_{M_j}}{d},$$

where λ_{M_i} is the i-th dimensional eigenvalue and $\overline{\lambda}_M$ is the average of the eigenvalues of M. In this paper, the variance σ_M^2 is called the flatness of M. Before calculating σ_M^2, all matrices were normalized [1] with $det(M) = 1$. Here, the flatness of the unit matrix that represents searching in the Euclidean space, is 0.

The details of the experiments are described in [SYKU01]. Our experiments revealed that the MBB-MBS approximation technique has the following problems:

(P1) **CPU time**
For both MBB and MBS approximation functions, approximation quality decreases as either dimensionality or matrix flatness grows. As a result, the number of exact quadratic form distance calculations increases, thus leading to a high CPU time.

(P2) **Node accesses**
Index structures are constructed to find target objects in the Euclidean space efficiently, and the search algorithms utilize the resulting index structures. Therefore, as matrix flatness grows, the number of node accesses increases. This leads to increases in both CPU time and number of page accesses.

The problems revealed serve as the basis for developing our proposed methods. To cope with (P1), we developed the spatial transform technique; this technique achieves high quality approximations and superior performance. To overcome (P2), we developed an extension of the spatial transformation technique, called the multiple spatial transformation technique. This technique reduces both CPU time and the number of page accesses.

3 Spatial Transformation Technique

In this section, we describe the Spatial Transformation Technique (STT), which approximates the quadratic form distance between a query point and a bounding rectangle. Like the MBB-MBS approximation technique, STT guarantees no false drops and so returns exact answers to any query.

3.1 Basic Ideas

In computing the exact distance for quadratic form distance functions, calculating the distance between a query point and MBRs in the index structures incurs a high CPU cost, and moreover, the calculations need to be iterated. That is, the complexity is $O(\omega \cdot d^2)$, where ω denotes the number of iterations. The basic idea of STT is to transform the MBRs, whose distance from a query point is measured by the quadratic form distance functions, into rectangles whose distance is measured by the Euclidean distance functions. STT requires no iteration. The transformation contributes to reducing CPU time. As shown in the problem (P1), the MBB-MBS approximation technique is not effective when either the dimensionality or flatness of query matrices is high. STT requires less CPU time and offers great efficiency even when dimensionality and matrix flatness are high, because of its high approximation quality. In this section, we first define the spatial transformation and then describe a spatial transformation technique for bounding rectangles in index structures.

3.2 Definition of Spatial Transformation

Given a query matrix M and a query point q, the quadratic form distance between q and a point p in a d-dimensional space $\mathcal{S}$ is defined as follows:

$$d_M^2(p, q) = (p - q) \cdot M \cdot (p - q)^t. \tag{3}$$

Since M is positive definite, the spectral decomposition of M can be calculated as:

$$M = E_M \cdot \Lambda_M \cdot E_M^t, \tag{4}$$

where E_M is the set of the eigenvectors of M, and the diagonal matrix
$\Lambda_M = diag(\lambda_{M_1}, \lambda_{M_2}, \ldots, \lambda_{M_d})$ consists of the eigenvalues $\lambda_{M_1}, \lambda_{M_2}, \ldots, \lambda_{M_d}$ of M. From Equations (3) (4), we obtain:

$$d_M^2(p, q) = (p - q) \cdot E_M \cdot \Lambda_M \cdot E_M^t \cdot (p - q)^t. \tag{5}$$

When considering point $p' = (p - q) \cdot E_M \cdot \Lambda_M^{\frac{1}{2}}$ in the Euclidean space $\mathcal{S}'$, Equation (5) denotes that the Euclidean distance between the origin O and p' in $\mathcal{S}'$ is equal to the quadratic form distance $d_M^2(p, q)$ (i.e. $d_M^2(p, q) = p' \cdot p'^t$). Here, the transformation matrix of M is defined as:

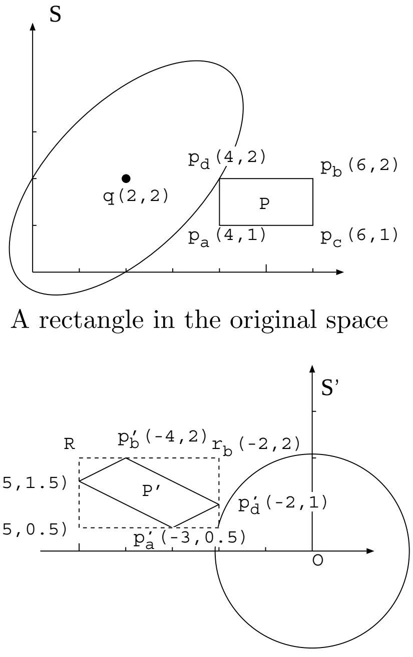
$$A_M = E_M \cdot \Lambda_M^{\frac{1}{2}}. \tag{6}$$

A_M transforms the quadratic form distances within $\mathcal{S}$ into the Euclidean distances within $\mathcal{S}'$. It yields the so called spatial transformation of p into p'.

3.3 Spatial Transformation of Rectangles for Distance Calculation

STT gives the spatial transformation of rectangles in index structures. Figure 2 illustrates the spatial transformation of a rectangle. In this figure, the bounding rectangle P in $\mathcal{S}$ is transformed into the d-dimensional parallelogram P' in $\mathcal{S}'$. Since the calculation of distance between the origin O and polygons in high-dimensional spaces incurs a high CPU cost, STT approximates P' by rectangle R as shown in Figure 2(b).

[1] Normalization of matrices is described in Section 4.4.



(a) A rectangle in the original space

(b) A rectangle calculated by STT

Figure 2: An example of spatial transformation.

This approximation reduces the calculation cost of ellipsoid queries.

We assume a rectangle P within $\mathcal{S}$ and a query point q. Let p_a and p_b be endpoints of the major diagonal of P and l_i be the i-th dimensional edge length of P. It follows that point p'_a in $\mathcal{S}'$ can be calculated by the spatial transformation of p_a:

$$p'_a = (p_a - q) \cdot A_M. \tag{7}$$

We extract the following components from the components a_{ij} of A_M:

$$\begin{aligned} \phi_{ij} &= \begin{cases} a_{ij} & (a_{ij} < 0) \\ 0 & (\text{otherwise}), \end{cases} \\ \psi_{ij} &= \begin{cases} a_{ij} & (a_{ij} > 0) \\ 0 & (\text{otherwise}). \end{cases} \end{aligned} \tag{8}$$

From Equations (7) (8), the rectangle R that totally encloses the d-dimensional parallelogram with respect to the spatial transformation of P can be calculated as [2] :

$$\begin{aligned} R &= (r_a, r_b), \qquad\qquad\qquad\qquad\qquad\qquad (9) \\ r_{a_j} &= p'_{a_j} + \sum_{i=1}^{d} l_i \cdot \phi_{ij}, \quad r_{b_j} = p'_{a_j} + \sum_{i=1}^{d} l_i \cdot \psi_{ij} \\ & \quad (1 \le j \le d), \end{aligned}$$

where r_a and r_b are endpoints of the major diagonal of R. Since R totally encloses P' in $\mathcal{S}'$, the search algorithm can use the Euclidean distance $d^2(R, O)$ instead of the quadratic form distance $d^2_M(P, q)$ (i.e. $d^2(R, O) \le d^2_M(P, q)$).

For example, as shown in Figure 2, the query point $q = (2, 2)$ and matrix:

$$M = \begin{pmatrix} 1.25 & -0.75 \\ -0.75 & 1.25 \end{pmatrix}$$

are given. When using M, the vertices p_a, p_b, p_c and p_d of the bounding rectangle P in $\mathcal{S}$ are transformed into the vertices p'_a, p'_b, p'_c and p'_d of the parallelogram P' in $\mathcal{S}'$, respectively. Also, $R = (r_a, r_b)$ encloses P'. $d^2_M(q, P)$ is approximated by $d^2(R, O)$, and we can utilize $d^2(R, O)$ instead of $d^2_M(q, P)$.

3.4 Search Algorithm

Range queries and k-nearest neighbor queries are useful for multi-dimensional databases. An algorithm based on spatial transformation can efficiently support both types of queries. Since k-nearest neighbor queries are more complex and require higher cost than range queries, we will focus on the k-nearest neighbor search in this paper and describe one such algorithm for STT. Note that the idea of STT can be applied to any range query.

The search algorithm implements Equations (6) (7) (8) (9) for spatial transformation. However, its CPU cost would become excessive if it required these formulas to be used in the spatial transformation of all accessed rectangles. Therefore, we use the following two ideas to reduce the CPU cost.

First, the result of Equations (6) (8) does not depend on the position of the bounding rectangles accessed. Thus, the search algorithm solves these formulas before accessing the rectangles. The result can be applied to the spatial transformation of all rectangles visited.

Second, we reduce the calculation time relative to Equation (9). Note that, on average, half of the components ϕ_{ij} and ψ_{ij} are 0. Therefore, in the implementation, the algorithm searches for all pairs of row number i and column number j whose components are $\phi_{ij} \ne 0, \psi_{ij} \ne 0$ before accessing nodes in index structures. This preprocessing halves the CPU cost when calculating R (i.e. r_{a_j} and r_{b_j}) with Equation (9).

Let c_{a_j} be the number of components in the j-th column where $\phi_{ij} \ne 0$, and u_{jk} be c_{a_j} row numbers of the components in the j-th column ($k = 1, \ldots, c_{a_j}$). Similarly, for ψ_{ij}, let c_{b_j} be the number of components in the j-th column where $\psi_{ij} \ne 0$, and v_{jk} be c_{b_j} row numbers of the components in the j-that column ($k = 1, \ldots, c_{b_j}$). Functions for calculating the position of R with less computation time can be obtained by using u_{jk} and v_{jk}:

$$\begin{aligned} R &= (r_a, r_b), \qquad\qquad\qquad (10) \\ r_{a_j} &= p'_{a_j} + \sum_{k=1}^{c_{a_j}} l_k \cdot \phi_{(u_{jk})j}, \end{aligned}$$

[2] The proof of Equation (9) is described in [SYKU01].

```
Procedure search(point $query$, matrix $M$,
                                          integer $k$)
 1. $\Phi_M$ := analyzeMatrix($M$);
 2. enqueue($a\_pointer\_to\_the\_root$, 0);
 3. while emptyQueue() = false do
 4.   $N$ := dequeue();
 5.   if $N$ is a data node then
 6.     for each $entry \in N$ do
 7.       if $d_{MBB\text{-}MBS(M)}$($query$, $entry$.vector)
                              $\leq nnlist[k].dist$ then
 8.         if $d_M$($query$, $entry$.vector)
                              $\leq nnlist[k].dist$ then
 9.           $nnlist[k].id$ := $entry$.id;
10.           $nnlist[k].dist$ := $d_M$($query$,
                                   $entry$.vector);
11.           sort $nnlist$ by distance;
12.           pruneQueue($nnlist[k].dist$);
13.         endif
14.   else
15.     for each $entry \in N$ do
16.       if $d_{MBB\text{-}MBS(M)}$($query$, $entry$.rectangle)
                              $\leq nnlist[k].dist$ then
17.         $R$ := spatialTransformation($query$,
                         $entry$.rectangle, $\Phi_M$);
18.         if d($R$, $O$ ) $\leq nnlist[k].dist$ then
19.           if $d_M$($query$, $entry$.rectangle)
                              $\leq nnlist[k].dist$ then
20.             enqueue($entry$.ptr, $d_M$ ($query$,
                              $entry$.rectangle));
21.       endif
22.   endif
23. enddo
24. output($nnlist$);
```

Figure 3: k-nearest neighbor search algorithm for ellipsoid queries.

$$r_{b_j} \quad = \quad p'_{a_j} + \sum_{k=1}^{c_{b_j}} l_k \cdot \psi_{(v_{jk})j},$$

where each c_{a_j} and c_{b_j} averages $d/2$.

Figure 3 shows the search algorithm for ellipsoid queries using tree structures of the R-tree family. The search algorithm utilizes the spatial transformation of rectangles to evaluate the distance of a query point to the rectangles. STT and MBB-MBS approximation techniques incur lower CPU costs compared with the exact quadratic form distance function for distance calculations. Therefore, the search algorithm first calculates the approximation distance between a query point and a bounding rectangle when evaluating the distance to the bounding rectangle. If the calculated approximation distance is less than or equal to the distance of the query point to the actual k-th nearest neighbor, the exact distance to the rectangle is evaluated using the exact quadratic form distance function.

In the search procedure (see Figure 3), for initialization, the transformation matrix is calculated and its components are checked (step 1), and then the pair of a pointer to the root and 0 is stored in the priority queue (step 2). In step 4, the function `dequeue()` dequeues the pair from the top of the priority queue, and extracts a node N. If N is a data node, the MBB-MBS approximation distance of every data object in the node is evaluated. If the approximation distance is less than or equal to the actual k-th nearest neighbor distance, the exact distance is evaluated (steps 5 to 8), and the data object together with its distance is stored in the nearest neighbor list (steps 9 to 12). If N is not a data node, the MBB-MBS approximation distance of every bounding rectangle is evaluated (step 16). If the MBB-MBS approximation distance of a rectangle is less than or equal to the actual k-th nearest neighbor distance, the spatial transformation of the rectangle is calculated from Φ_M (step 17). In step 18, the Euclidean distance between O and R obtained by the spatial transformation is evaluated. If the distance calculated by the spatial transformation is less than or equal to the actual k-that nearest neighbor distance, the exact distance is evaluated (step 19).

Our experiments used not only the A-tree but also the R*-tree. The A-tree is useful for ellipsoid queries as well as queries based on the Euclidean distance function. The A-tree search algorithm differs somewhat from the other methods in the R-tree family. The details of the A-tree search algorithm are described in [SYUK00].

3.5 Dimensionality Reduction

When the flatness of a query matrix is high, there are eigenvectors whose eigenvalue is small. In the space created by the spatial transformation, the dimensions corresponding to the eigenvalues contribute less to approximation quality although the dimensions require the same CPU cost as the others. The STT with dimensionality reduction eliminates dimensions whose eigenvalues are small in order to save on CPU costs.

Let $r = (r_1, r_2, \ldots, r_d)$ be the point in R, which is the closest to O in $\mathcal{S}'$ created by the spatial transformation. When using dimensionality reduction, the distance of R to O can be determined as:

$$\begin{aligned} \tilde{d}^2(R,O) &= \sum_{i=1}^{n} (r_i)^2, \qquad (11) \\ n &= COUNT\left(\lambda_j \geq \frac{\eta}{d} \cdot \sum_{i=1}^{d} \lambda_i\right) \\ & \quad (j = 1, \ldots, d), \end{aligned}$$

where η is a threshold for dimensionality reduction, and λ_i is arranged in descending order (i.e. $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d > 0$). The function $COUNT(\Gamma)$ gives the number of elements that satisfy requirement Γ. This formula shows that the dimensionality for distance calculation in $\mathcal{S}'$ is limited to n ($n \leq d$). Thus, the dimensionality reduction reduces the calculation time relative to Equations (7) (8) (10) as well as Equation (11) to n/d. As query matrix flatness increases, n decreases and higher efficiency is achieved for the distance calculations.

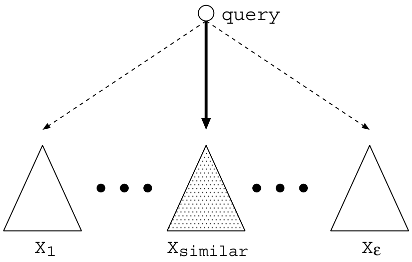


Figure 4: The multiple spatial transformation technique.

4 Multiple Spatial Transformation Technique

In this section, we present the Multiple Spatial Transformation Technique (MSTT), which is an extension of STT. STT's approximation quality is high, however, the number of node accesses increases as query matrix flatness grows. This is because STT, as well as conventional search methods, utilizes a structure that is constructed by the Euclidean distance function. To overcome this problem, MSTT constructs tree structures based on various quadratic form distance functions and then chooses a structure that gives sufficient search performance; the search algorithm described in Section 3.4 utilizes such a chosen structure.

4.1 Basic Ideas

We have revealed the node access problem (P2) in Section 2.2. The search methods for adaptive ellipsoid queries presented in [SK97] and [ABKS98] use index structures based on the Euclidean distance function. Accordingly, the number of node accesses increases as query matrix flatness grows, which leads to an increase in CPU cost and number of page accesses. MSTT overcomes this problem by selecting an arbitrary quadratic form distance function before constructing the index structures; the search algorithm utilizes the resulting structure. MSTT reduces both page accesses and CPU cost for ellipsoid queries.

MSTT can handle more than one index structure. For multimedia systems that attach importance to retrieval performance and can well afford the disk space, the use of more than one structure is effective in improving search performance. Figure 4 illustrates a retrieval mechanism based on MSTT. The mechanism first determines a typical ellipsoid query matrix X_i ($i = 1, \ldots, \varepsilon$) from the user's query logs, and then constructs index structures based on X_i. In query processing, the matrix $X_{similar}$ closest to the query matrix M is chosen, and target objects are found using the structure constructed by $X_{similar}$. In particular, the query shown in Figure 4 requires search processing based on the Euclidean distance function if $M = X_{similar}$. This retrieval mechanism that adopts multiple indexing can accelerate search performance.

Disk prices continue to fall and disk unit capacity is increasing rapidly. [GG97] shows that disk unit capacity and storage cost have increased /decreased a hundred times and ten thousand times, respectively; whereas disk access speeds have increased only ten-fold in the last twenty years. The resulting trend is to emphasize disk access speed counts over storage cost. In addition, reducing the search cost has a higher priority than reducing the insertion cost in many multimedia databases. It follows that there is a strong rationale for using more than one index to improve search performance.

4.2 Indexing and Retrieval Mechanisms

Structure Construction:

Let C be a matrix for constructing an index structure. The transformation matrix A_C of C is:

$$A_C = E_C \cdot \Lambda_C^{\frac{1}{2}}.$$

All data points included in the data set for constructing an index are transformed by A_C. For instance, A_C transforms a data point p in the data set into $p' = p \cdot A_C$. MSTT constructs an index structure $\mathcal{I}_C$ based on the transformed data points. $\mathcal{I}_C$ can efficiently support queries whose matrix is C.

Query Processing by MSTT:

For a query point q and a query matrix M, we first transform q into $q' = q \cdot A_C$ to perform this query using $\mathcal{I}_C$. Given a new matrix M' for the query processing of M using $\mathcal{I}_C$:

$$M' = A_C^{-1} \cdot M \cdot (A_C^{-1})^t, \qquad (12)$$

the quadratic form distance of M between p and q can be expanded as follows:

$$\begin{aligned} d_M^2(p,q) &= (p-q) \cdot M \cdot (p-q)^t \\ &= (p'-q') \cdot A_C^{-1} \cdot M \cdot (A_C^{-1})^t \cdot (p'-q')^t \\ &= (p'-q') \cdot M' \cdot (p'-q')^t. \end{aligned}$$

Thus, the query whose matrix is M' and point is q' using $\mathcal{I}_C$ leads to the search result of the ellipsoid query of M. In particular, if $M = C$, $\mathcal{I}_C$ can efficiently support the query whose matrix is M, since M' is a unit matrix, which means the search is based on the Euclidean distance function.

4.3 Similarity of Matrices

When more than one index structure is constructed, the search process must choose one of them for access. To do so, we define the dissimilarity between a query matrix M and an index $\mathcal{I}_C$ by using the matrix flatness.

Queries of M using $\mathcal{I}_C$ utilize M', calculated by Equation (12), as the query matrix. Let $\lambda_{M'_i}$ be the

Table 1: Variance of eigenvalues.

	w_r	1	10	100	1000
σ^2_M	$d = 8$	0.0307	76.489	7998.6	800214
	$d = 27$	64.777	93372	9.29e8	9.29e12

Table 2: Dimensions used for ellipsoid queries.

	w_r	1	10	100	1000
n	$d = 8$	8	8	4	4
	$d = 27$	27	18	9	9

i-th dimensional eigenvalue of M' and $\overline{\lambda}_{M'}$ be the average of the eigenvalues of M'. The variance $\sigma^2_{M'}$ of the eigenvalues of M' is determined as follows:

$$\sigma^2_{M'} = \sum_{i=1}^{d}(\lambda_{M'_i} - \overline{\lambda}_{M'})^2, \quad \overline{\lambda}_{M'} = \sum_{j=0}^{d}\frac{\lambda_{M'_j}}{d} \qquad (13)$$

We employ $\sigma^2_{M'}$ as the measure of dissimilarity between M and $\mathcal{I}_C$. For a similarity search using MSTT, the effectiveness of $\mathcal{I}_C$ relative to M improves as $\sigma^2_{M'}$ decreases.

4.4 Normalization of Matrices

To calculate the dissimilarity of queries and indices, all matrices must be normalized, i.e., $det(C) = det(M) = 1$ for matrices C and M. The normalized matrix N of M is obtained by:

$$N = E_M \cdot \Lambda_N \cdot E^t_M, \quad \lambda_{N_i} = \lambda_{M_i} \cdot \left(\prod_{i=1}^{d}\lambda_{M_i}\right)^{-\frac{1}{d}}$$

where the diagonal matrix Λ_N consists of the eigenvalues λ_{N_i} $(i = 1, \ldots, d)$ of N. C can also normalized in the same way.

5 Performance Evaluation

To verify the effectiveness of STT, we implemented the algorithm and compared it with the MBB-MBS approximation technique. We then measured the performance of MSTT.

We evaluated its performance using real data sets with size of 100,000. For the data sets, 8-D and 27-D feature vectors of color histograms were extracted from images. In assessing search performance, the page access number and CPU time were measured by the average of 100 queries. In our evaluation, we used 20-nearest neighbor queries; query data were different from the point data included in the indices, that is, query points were generated randomly and independently of data points. Page size was 8 KB. CPU time was measured on a SUN UltraSPARC-II 450 MHz. We used the A-tree [SYUK00], which provides superior performance for high-dimensional data, and chose the code with size of 6 bits per dimension for approximating the bounding rectangles and data objects in the A-tree structure. To obtain the similarity matrices M, we calculated the components m_{ij} of M using the following formula [HSE+95] [ABKS98]:

$$m_{ij} = exp(-\alpha(d_w(c_i, c_j)/d_{max})^2),$$

where α is a positive constant, and $d_w(c_i, c_j)$ denotes the weighted Euclidean distance between the color c_i and c_j. The factors $w = (w_r, w_g, w_b)$ represent the weighting of the red, green and blue components in RGB color space. In our evaluation, α was 10, and both w_g and w_b were fixed to 1. w_r was varied from 1 to 1,000. We calculated the eigenvalues of every matrix for 8-D and 27-D data. Table 1 shows the result of this calculation for various values of w_r. As shown in the table, matrix flatness increases as w_r grows when α, w_g and w_b are fixed.

For the dimensionality reduction technique, the best threshold ($\eta = 0.01$) from among three alternatives, $\eta = 0.1$, $\eta = 0.01$, $\eta = 0.001$, was chosen. Table 2 shows the dimensions n used for STT with respect to the matrices created in our experiments.

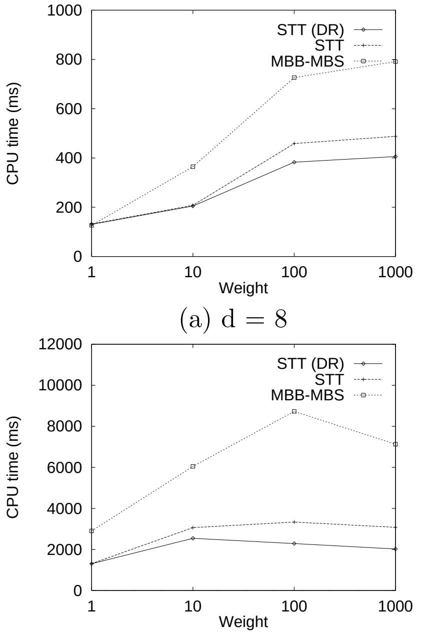
5.1 Search Performance

Figure 5 compares STT and the MBB-MBS approximation technique in terms of CPU cost. The A-tree was used as the index structure. The symbol `STT(DR)` means the CPU cost for the STT with dimensionality reduction. The number of page accesses is shown in Figure 6. Since STT and the MBB-MBS approximation technique utilize exact quadratic form distance functions, both require the same number of page accesses to perform ellipsoid queries. Thus, the difference in search time between STT and the MBB-MBS approximation technique depends on calculation complexity. As described in Section 1, ellipsoid queries incur high costs in calculating the distance between bounding rectangles and query points. Figure 5 shows that STT reduces CPU cost for all data sets. The effectiveness of STT increases as either dimensionality or matrix flatness grows. In particular, STT achieves a 74 % reduction in CPU cost over that of the MBB-MBS approximation technique for high dimensionality and matrix flatness.

We evaluated the STT performance using the R*-tree as well as the A-tree. STT using either the R*-tree or the A-tree was superior to the MBB-MBS approximation technique. The details of the experiments using the R*-tree are described in [SYKU01].

5.2 Analysis of Approximation Techniques for Elliptical Queries

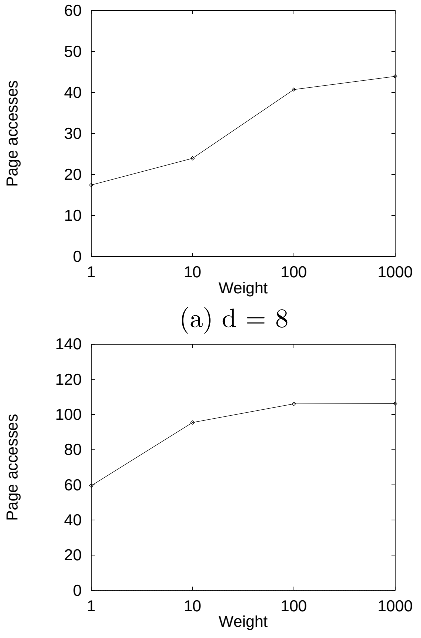
STT does not utilize the exact quadratic form distance functions to access bounding rectangles whose approximation distance from the query point exceeds the actual k-nearest neighbor distance, similar to the MBB-MBS approximation technique. Figure 7 shows the percentage of filtered exact quadratic form distance



(a) d = 8

(b) d = 27

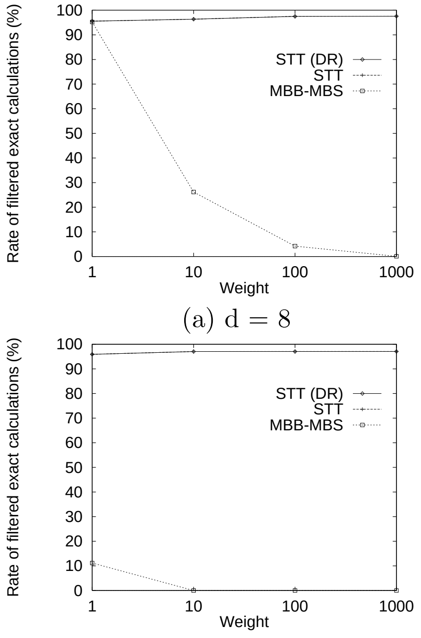
Figure 5: Comparison of STT with the MBB-MBS approximation technique in terms of CPU cost.



(a) d = 8

(b) d = 27

Figure 6: The number of page accesses for the MBB-MBS approximation technique.



(a) d = 8

(b) d = 27

Figure 7: Rate of filtered exact distance calculations for STT.

calculations versus the number of bounding rectangles accessed in search processing, that is, this figure illustrates the effectiveness of the approximation techniques. Although the efficiency of the MBB-MBS approximation technique decreases as the flatness of query matrix grows, the STT approximations efficiently filter exact quadratic form distance calculations for all queries. STT proves to be highly effective with high-dimensional data and queries whose matrix flatness is high as well as those with lower dimensionality and flatness. The effectiveness of STT yields a low CPU cost as shown in Figure 5.

The dimensionality reduction provided by STT eliminates dimensions that make only a slight contribution to the approximation of distance between query points and bounding rectangles. This technique becomes more effective in ellipsoid searches as the flatness of the query matrix increases. Since the flatness of query $w_r = 1$ is relatively low, the query uses all dimensions in the search as shown in Table 2. On the other hand, for queries $w_r = 100$ and $w_r = 1000$, both of which have flat ellipsoids, distance calculations are based on lower dimensionality. As Figure 7 shows, STT has high approximation efficiency with and without dimensionality reduction; as a result, the STT with dimensionality reduction has superior performance and a lower CPU cost.

5.3 Effectiveness of MSTT

For a given query matrix, MSTT constructs an index structure based on the query matrix in order to support the query more efficiently. In these experiments, we measured the performance of MSTT with the following structures:

(1) **Unit:** the index structures constructed from the unit matrix.

(2) $\boldsymbol{W_r = 10}$**:** the index structures constructed from the matrix $w_r = 10$.

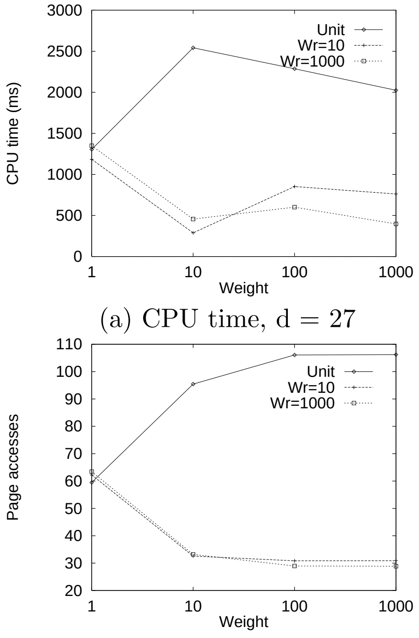
(3) $\boldsymbol{W_r = 1000}$**:** the index structures constructed from the matrix $w_r = 1000$.

Figure 8 depicts the search performance for ellipsoid queries using these index structures [3] . Table 3 shows the dissimilarities between the three index structures and the four ellipsoid queries for 8-D and 27-D dimensions. The dissimilarities of index structures to queries were calculated using Equation (13). For any query, choosing the index structure that is most similar to the query minimizes search cost of the query. MSTT significantly reduces CPU costs and the number of page accesses for any query.

In addition, search cost is not proportional to dissimilarity. For example, queries whose dissimilarity is 0 incur some search cost since similarity searches entail some cost even in the Euclidean distance space. Note that the function is not a cost model. Dissimilarity allows the search algorithm to choose the index structure well suited to query matrices.

In practical situations, the dissimilarity of a given query matrix must be calculated for each index when

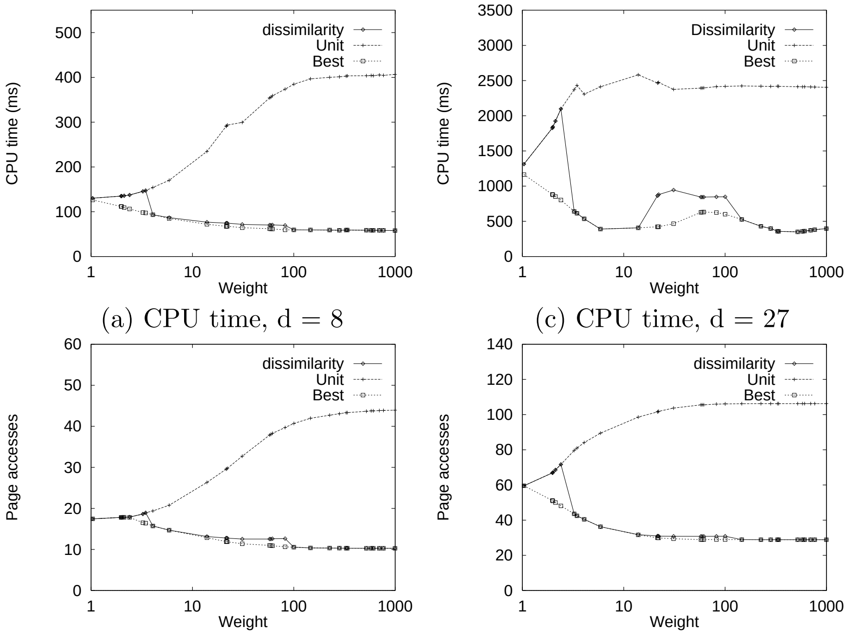
[3] Experiments for 8-D data are shown in [SYKU01].



(a) CPU time, d = 27

(b) Page accesses, d = 27

Figure 8: Search performance for MSTT.



(a) CPU time, d = 8

(c) CPU time, d = 27

(b) Page accesses, d = 8

(d) Page accesses, d = 27

Figure 9: Behavior of the dissimilarity function.

trying to find the best index structure. The number of dissimilarity calculations grows as the number of created indices increases since the number of dissimilarity calculations equals the number of indices. However, each calculation incurs only a small CPU cost: 2 ms for 27-D data in our experiments. Since this is negligible compared with the overall search time, constructing various structures substantially improves search performance.

5.4 Properties of the dissimilarity function

In this section, we analyze the properties of the dissimilarity function defined in Section 4.3. We created 30 query matrices for 8 and 27 dimensions using weight w_r as follows:

$$w_r = 10^{random},$$

where *random* is a randomly generated number between 0 and 3. Figure 9 shows how the dissimilarity function chooses an index structure. Each search cost for 30 queries was measured by the average of 100 queries. This experiment used three kinds of index, **Unit**, $\boldsymbol{W_r = 10}$ and $\boldsymbol{W_r = 1000}$, such as the experiment shown in Section 5.3. The figure shows the following search costs:

(1) **Dissimilarity:** the cost of search using index structures chosen by the dissimilarity function.

(2) **Unit:** the search cost on index structures constructed from the unit matrix.

(3) **Best:** the lowest search cost using the optimal index structure for each query matrix.

In the experiment using 27-dimensional data, the dissimilarity function chooses the index structure **Unit** for the query matrices whose w_r lies between 1 and 3, $\boldsymbol{W_r = 10}$ for w_r between 3 and 100, and $\boldsymbol{W_r = 1000}$ for w_r between 100 and 1,000. The choice of indices for 8-dimensional data is quite similar to that for 27-dimensional data. Although search cost determined by the function is not exactly equal to the search cost achieved by the optimal index structures, the function chooses a good structure for most of the query matrices. Moreover, compared with the index structure based on the Euclidean distance, index structures chosen by the dissimilarity function greatly reduce the search cost. Unlike the previous works [SK97] [ABKS98] that use one index structure based on the Euclidean distance, MSTT constructs various index structures, which allows the dissimilarity function to choose the structure well suited to the query matrices. This analysis demonstrates the effectiveness of MSTT.

6 Conclusions

This paper described the Spatial Transformation Technique (STT), which offers excellent performance when searching for adaptive ellipsoid queries. First, we analyzed the MBB-MBS approximation technique and discussed its problems. Then, based on this analysis, we showed how STT can overcome these problems.

STT's high level of performance is due to its use of spatial transformation. Since the spatial transformation provides highly accurate approximations of the distance between query points and bounding rectangles, STT eliminates exact distance evaluations for most of the bounding rectangles accessed in the index structures. The mechanism of STT is remarkably

Table 3: Dissimilarity of matrices.

(a) **Unit**

w_r		1	10	100	1000
$\sigma^2_{M'}$	$d = 8$	0.031	76.490	7999	800214
	$d = 27$	64.777	93372	9.29e8	9.29e12

(b) $\boldsymbol{W_r = 10}$

w_r		1	10	100	1000
$\sigma^2_{M'}$	$d = 8$	71.296	0	194.99	19892
	$d = 27$	42467	0	127814	1.30e9

(c) $\boldsymbol{W_r = 1000}$

w_r		1	10	100	1000
$\sigma^2_{M'}$	$d = 8$	748634	19892	196.01	0
	$d = 27$	4.37e12	1.33e9	132022	0

efficient, especially for queries whose dimensionality or matrix flatness is high. This technique guarantees no false drops. In experiments using various matrices and index structures, STT was found to be superior to the conventional MBB-MBS approximation technique.

This paper also described the Multiple Spatial Transformation Technique (MSTT). MSTT adjusts tree structures to suit ellipsoid queries; the search algorithm utilizes the adjusted structures. This technique reduces the number of page accesses as well as the CPU cost for ellipsoid queries.

MSTT can support ellipsoid queries efficiently because one or more index structures can be used. In the future, we plan to consider an algorithm that determines matrices from a log of user's queries to create various indices. We will also create a matrix decision algorithm whose parameters are a log of queries and the number of indices that can be stored on disk.

References

[ABKS98] Mihael Ankerst, Bernhard Braunmüller, Hans-Peter Kriegel, and Thomas Seidl: "Improving Adaptable Similarity Query Processing by Using Approximations", in *Proc. of the 24th International Conference on Very Large Data Bases (VLDB)*, pp. 206–217, New York City, NY, August 1998.

[BKK96] Stefan Berchtold, Daniel A. Keim, and Hans-Peter Kriegel: "The X-tree: An Index Structure for High-Dimensional Data", in *Proc. of the 22nd International Conference on Very Large Data Bases (VLDB)*, pp. 28–39, Bombay, September 1996.

[BKSS90] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger: "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles", in *Proc. ACM SIGMOD Conf.*, pp. 322–331, Atlantic City, NJ, May 1990.

[BO97] Tolga Bozkaya and Meral Ozoyoglu: "Distance-Based Indexing for High-Dimensional Metric Spaces", in *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 357–368, May 1997.

[CPZ97] Paolo Ciaccia, Marco Patella, and Pavel Zezula: "M-tree: An Efficient Access Method for Similarity Search in Metric Spaces", in *Proc. of the 23rd International Conference on Very Large Data Bases (VLDB)*, pp. 426—435, Athens, August 1997.

[FSA+95] M. Flickner, H. S. Sawhney, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker: "Query by image and video content: the QBIC system", *IEEE Computer*, Vol. 28, No. 9, pp. 23–32, September 1995.

[GG97] Jim Gray and Goetz Graefe: "The Five-Minute Rule Ten Years Later and Other Computer Storage Rules of Thumb", *SIGMOD Record*, Vol. 26, No. 4, pp. 63–68, December 1997.

[GG98] Volker Gaede and Oliver Günther: "Multidimensional Access Methods", *ACM Computing Surveys*, Vol. 30, No. 2, pp. 170–231, June 1998.

[HS95] G. R. Hjaltason and H. Samet: "Ranking in Spatial Databases", in *Proceedings of the 4th Symposium on Spatial Databases*, pp. 83–95, Portland, Maine, August 1995.

[HSE+95] James L. Hafner, Harpreet S. Sawhney, William Equitz, Myron Flickner, and Wayne Niblack: "Efficient Color Histogram Indexing for Quadratic Form Distance Functions", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 17, No. 7, pp. 729–736, July 1995.

[ISF98] Yoshiharu Ishikawa, Ravishankar Subramanya, and Christos Faloutsos: "MindReader: Querying databases through multiple examples", in *Proc. of the 24th International Conference on Very Large Data Bases (VLDB)*, pp. 218–227, New York City, NY, August 1998.

[RHM97] Y. Rui, T. S. Huang, and S. Mehrotra: "Content-based Image Retrieval with Relevance Feedback in MARS", in *Proc. of IEEE International Conference on Image Processing*, pp. II–815–818, October 1997.

[RKV95] Nick Roussopoulos, Stephen Kelley, and Frédéric Vincent: "Nearest Neighbor Queries", in *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 71–79, May 1995.

[SK97] Thomas Seidl and Hans-Peter Kriegel: "Efficient User-Adaptable Similarity Search in Large Multimedia Databases", in *Proc. of the 23rd International Conference on Very Large Data Bases (VLDB)*, pp. 506—515, Athens, August 1997.

[SYKU01] Yasushi Sakurai, Masatoshi Yoshikawa, Ryoji Kataoka, and Shunsuke Uemura: "Similarity Search for Adaptive Ellipsoid Queries Using Spatial Transformation", Technical report, Nara Institute of Science and Technology, 2001.

[SYUK00] Yasushi Sakurai, Masatoshi Yoshikawa, Shunsuke Uemura, and Haruhiko Kojima: "The A-tree: An Index Structure for High-Dimensional Spaces Using Relative Approximation", in *Proc. of the 26th International Conference on Very Large Data Bases (VLDB)*, pp. 516–526, Cairo, Egypt, September 2000.