Proc. 23rd Int. Conf. on Very Large Databases (VLDB), Athens, Greece, 1997

Efficient User-Adaptable Similarity Search in Large Multimedia Databases

Thomas Seidl
Institute for Computer Science
University of Munich, Germany
seidl@dbs.informatik.uni-muenchen.de

Hans-Peter Kriegel
Institute for Computer Science
University of Munich, Germany
kriegel@dbs.informatik.uni-muenchen.de

Abstract

Efficient user-adaptable similarity search more and more increases in its importance for multimedia and spatial database systems. As a general similarity model for multi-dimensional vectors that is adaptable to application requirements and user preferences, we use quadratic form distance functions $d_A^2(x, y) = (x - y) \cdot A \cdot (x - y)^{\mathrm{T}}$ which have been successfully applied to color histograms in image databases [Fal+ 94]. The components a_{ij} of the matrix A denote similarity of the components i and j of the vectors. Beyond the Euclidean distance which produces spherical query ranges, the similarity distance defines a new query type, the ellipsoid query. We present new algorithms to efficiently support ellipsoid query processing for various user-defined similarity matrices on existing precomputed indexes. By adapting techniques for reducing the dimensionality and employing a multi-step query processing architecture, the method is extended to high-dimensional data spaces. In particular, from our algorithm to reduce the similarity matrix, we obtain the greatest lower-bounding similarity function thus guaranteeing no false drops. We implemented our algorithms in C++ and tested them on an image database containing 12,000 color histograms. The experiments demonstrate the flexibility of our method in conjunction with a high selectivity and efficiency.

This research was funded by the German Ministry for Education, Science, Research and Technology (BMBF) under grant no. 01 IB 307 B. The authors are responsible for the content of this paper.

Proceedings of the 23rd VLDB Conference
Athens, Greece, 1997

1. Introduction

In recent years, an increasing number of database applications has emerged for which efficient support for similarity search is substantial. The requirements of modern information retrieval, spatial database and image database systems cannot be satisfied by the classic exact and partial match queries which specify all or some of the object features that have to fit exactly to the query features. The importance of similarity search grows in application areas such as multimedia, medical imaging, molecular biology, computer aided engineering, marketing and purchasing assistance, etc. [Jag 91] [AFS 93] [GM 93] [Fal+ 94] [FRM 94] [ALSS 95] [BKK 97] [BK 97]. Due to the immense and even increasing size of current databases, a high efficiency of query processing is crucial.

In this paper, we present a general technique for efficient similarity search in large databases that supports user-specified distance functions. The objects may be represented as high-dimensional vectors such as histograms over arbitrary distributions. Typical examples are color histograms which are obtained from images, shape histograms of spatial objects for geometric shape retrieval (e.g. section coding [BK 97]), multidimensional feature vectors for CAD objects and many others. In general, one- or multidimensional distributions can be characterized by histograms which are vectors or matrices that represent the distributions by discrete values.

For many applications concerning similarity search, the Euclidean distance of feature vectors is not adequate. The square of the Euclidean distance of two N-vectors x and y is defined as following:

$$d^2_{\mathrm{euclid}}(x, y) = (x - y) \cdot (x - y)^{\mathrm{T}} = \sum_{i=1}^{N} (x_i - y_i)^2$$

The basic assumption of the Euclidean distance is the independence of the dimensions, i.e. there is no influence of one component to the other. This fact does not reflect correlations of features such as substitutability or compensability. Therefore, it is recommended to provide similarity search techniques that use generalized distance functions. A distance model that has been successfully applied to image databases [Fal+ 94] and that has the power to model dependencies between different components of feature or histogram vectors, is provided by the class of quadratic form distance functions. Thereby, the distance measurement of two N-vectors is based on an $N \times N$-matrix $A = [a_{ij}]$ where the weights a_{ij} denote similarity between the components i and j of the vectors:

$$d_A^2(x, y) = (x - y) \cdot A \cdot (x - y)^{\mathrm{T}} =$$
$$= \sum_{i=1}^{N} \sum_{j=1}^{N} a_{ij} \cdot (x_i - y_i) \cdot (x_j - y_j)$$

This definition also includes the (squared) Euclidean distance when A is equal to the identity matrix, as well as (squared) weighted Euclidean distances when the matrix A is diagonal, $A = \mathrm{diag}(w_1, \ldots, w_N)$ with w_i denoting the weight of dimension i.

Section 2 contains some application examples which illustrate the relevance of adaptable distance functions, and a discussion of related work. In section 3, we present an efficient technique for similarity search in low-dimensional data spaces for a new query type, the ellipsoid query. Section 4 extends the method for efficient similarity query processing to high-dimensional spaces by employing techniques to reduce the dimensionality which leads to a multi-step query processing architecture. Section 5 presents the results from experimental evaluation, and section 6 concludes the paper.

2. Problem Characterization

There are two types of queries that are relevant for similarity search: First, range queries are specified by a query object q and a range value ε, defining the answer set to contain all the objects s from the database that have a distance less than ε to the query object q. Second, k-nearest neighbor queries for a query object q and a cardinal number k specify the retrieval of those k objects from the database that have the smallest distances to q.

We are faced with the general problem of similarity search in large databases whose objects are represented by vectors of any arbitrary dimension N, e.g. histograms or feature vectors. The similarity between two objects x and y is measured by quadratic form distance functions, $d_A^2(x, y) = (x - y) \cdot A \cdot (x - y)^{\mathrm{T}}$, where the similarity matrix A may be modified by the user at query time, according to the user-specific or even query-specific preferences. The $N \times N$-matrix A is only required to be positive definite, i.e. $z \cdot A \cdot z^{\mathrm{T}} > 0$ for all $z \in \Re^N, z \neq \mathbf{0}$, in order to obtain non-negative distance values. Since current and future databases are assumed to be very large, similarity query processing should be supported by spatial access methods (SAMs). If a SAM has already been precomputed, and if reduction of dimensionality has been applied to the vectors, it should be employed.

From the examples below, we observe that the dimensionality of the histograms may range from a few bins to tens and hundreds of bins (e.g. 256 colors in image databases). Therefore, a method for similarity search also has to provide efficient support for searching in high-dimensional data spaces.

2.1 Adaptable Distance Functions

The following examples illustrate the relevance of generalized distance functions. The first one is taken from [Haf+ 95] who developed techniques for efficient color histogram indexing in image databases within the QBIC (Query By Image Content) project [Fal+ 94]. Consider a simplified color histogram space with three bins (red, orange, blue), and let x, y, and z be three normalized histograms of a pure red image, $x = (1, 0, 0)$, a pure orange image, $y = (0, 1, 0)$, and a pure blue image, $z = (0, 0, 1)$. The Euclidean distance d_{euclid} of x, y, and z in pairs is $\sqrt{2}$, whereas the histogram distance d_A for the application-specific matrix $A_{\mathrm{red,orange,blue}} = \begin{bmatrix} 1.0 & 0.9 & 0.0 \\ 0.9 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$ yields a similarity of $\sqrt{0.2}$ for x and y, and a distance of $\sqrt{2}$ for z and x as well as for z and y. Thus, the histogram distance d_A provides a more adequate model for the characteristics of the given color histogram space.

In our second example, let us consider three histograms a, b, and c over an ordered space (cf. figure 1). Although c is closer to b than to a, which may reflect a higher similarity of the object c to b than to a, the Euclidean distance neglects such relationships of vector components. In such a case, a distance matrix $A = [a_{ij}]$ seems to be adequate which is populated in a more or less broad band along the diagonal, i.e. whose weights a_{ij} depend on the distance $|i - j|$ between the histogram bins i and j.

For our last example, we assume an image similarity search system that supports a variety of different user preferences. For instance, user 1 requires a strong distinction of the hues, whereas user 2 only looks for images with a simi-

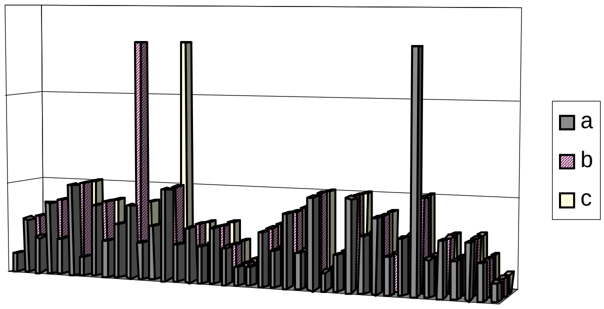

Figure 1: Sample histograms of three similar distributions over an ordered space

lar lightness but does not insist on the same hues. User 3 may be interested in pictures having the same mossy green while all the red and orange hues count for being similar. In order to obtain adequate distance functions, for each of the various preferences, a different matrix has to be composed that has suitable weighting factors at the appropriate positions. To provide an optimal support for all possible user profiles, the similarity retrieval system should support efficient query processing for various matrices.

2.2 Related work

Recently, various methods have been proposed concerning feature-based similarity retrieval [AFS 93] [GM 93] [FRM 94] [ALSS 95] [Kor+ 96] [BK 97] [BKK 97]. Typically, the architecture follows the multi-step query processing paradigm [OM 88] [BHKS 93] which in general provides efficient query processing, in particular when combined with a PAM or a SAM. However, these methods are restricted to manage only the Euclidean distance in the filter step, i.e. in the feature space.

In the QBIC (Query By Image Content) project [Fal+ 94], algorithms were developed for image retrieval based on shape and color. Two of the proposed algorithms use color histogram similarity [Haf+ 95]. The histogram distance function is defined as a quadratic form function $d^2_{\text{hist}}(x, y) = (x - y) \cdot A \cdot (x - y)^{\mathrm{T}}$ (see above). Since the dimensionality of the color histograms is 256, filter steps are used to efficiently support similarity query processing.

As a primary approach for an index-based filter step, [Fal+ 94] uses a concept of average colors: for each color histogram x, the average color x_{avg} is computed which is a three-dimensional vector since also the color space has three dimensions. The authors show that the average color distance, $d^2_{\text{avg}}(x, y) = (x_{\text{avg}} - y_{\text{avg}}) \cdot (x_{avg} - y_{avg})^{\mathrm{T}}$ scaled with a factor λ_A that depends on the matrix A, represents a lower bound for the histogram distance, i.e. $\lambda_A d^2_{\text{avg}}(x, y) \le d^2_{\text{hist}}(x, y)$. This lower-bounding property guarantees no false drops when using the average color distance in the filter step. The three-dimensional average color vectors are managed by using an R^*-tree, and a considerable performance gain has been obtained. The computation of the index only depends on the color map that is assigned to the histogram bins, but not on the similarity matrix A. Therefore, the index maybe precomputed without knowledge of any query matrix A. In other words, at query processing time, the method may fall back on an available index for arbitrary (previously unknown) user-specified similarity matrices. However, the dimensionality of the index is fixed to three, i.e. the dimensionality of the underlying color space. Thus, the average color method may not profit from advances in high-dimensional indexing methods.

As a generalization of d_{avg}, [Haf+ 95] introduce a scalable k-dimensional distance function d_k in order to operate with a k-dimensional index in the filter step. The k-index entries are obtained by a dimension reduction such that d_k is equal to the Euclidean distance. As shown by the authors, again the fundamental lower-bounding property $d^2_k(x, y) \le d^2_{\text{hist}}(x, y)$ holds, thus preventing the filter step from producing false drops. Contrary to the average color approach, this method provides more flexibility. The parameter k may be tuned in order to obtain an optimal filter selectivity and query processing performance with respect to the technical and application-specific environment. However, the main disadvantage of the method is its dependency on the similarity matrix A. In particular, the reduction of the high-dimensional histograms to k-dimensional index entries is done by using a symmetric decomposition of the similarity matrix A. Thus, when the query matrix A changes, the complete index would have to be recomputed in general. In other words, the method only supports the predefined similarity matrix for a given index.

In our approach, we efficiently support similarity processing and provide both, flexibility for the user to modify the similarity matrix, and scalability for the access method to use an optimal dimensionality of the underlying index structure according to the technical and application-specific environment.

3. Similarity Query Processing in Low Dimensions

A key technique to efficiently support query processing in spatial database systems is the use of point access methods (PAMs) or spatial access methods (SAMs). Although our method works with a variety of PAMs and SAMs, in this paper, we focus on access methods that manage the secondary storage pages by rectilinear hyperrectangles, e.g. minimum bounding rectangles (MBRs), for forming higher level directory pages. For instance, this paradigm is realized in the R-tree [Gut 84] and its derivatives, R^+-tree [SRF 87], R*-tree [BKSS 90], as well as in the X-tree [BKK 96], which has already been used successfully to support query processing for dimensionalities up to 16.

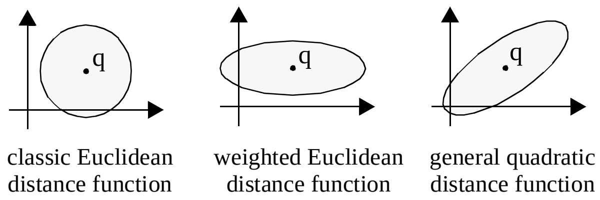

Figure 2: Typical shapes of query ranges $\{p \mid d(p, q) \le \varepsilon\}$ for various distance functions d and a fixed query range ε.

Up to now, similarity query processing using PAMs and SAMs supports only the Euclidean distance where query ranges are spherical, and weighted Euclidean distances which correspond to iso-oriented ellipsoids. However, query ranges for quadratic form distance functions $d_A^2(p, q) = (p - q) \cdot A \cdot (p - q)^T$ for positive definite query matrices A and query points q lead to arbitrary oriented ellipsoids (cf. figure 2). We call the new query type an *ellipsoid query* and present efficient algorithms for ellipsoid query processing in the following.

3.1 Ellipsoid Queries on Spatial Access Methods

Both, similarity range queries and k-nn queries, are based on the distance function for query objects $ellip_q(A)$ and database objects p. When employing SAMs that organize their directory by rectilinear hyperrectangles, an additional distance function *mindist* is required (cf. [HS 95] [RKV 95] [BBKK 97] [Ber+ 97]) which returns the minimum distance of the query object $ellip_q(A)$ to any iso-oriented hyperrectangle *box*.

As a generalization of *mindist*, we introduce the operation $distance(A, q, box, \varepsilon)$ which returns the minimum distance $d_{min} = \min\{d_A^2(p, q) \mid p \in box\}$ from $ellip_{A,q}$ to *box* if $d_{min} \ge \varepsilon$, and an arbitrary value below ε if $d_{min} < \varepsilon$. The relationship of *distance* to the operations *intersect* and *mindist* is shown by the following lemma.

Lemma 1. The function $distance(A, q, box, \varepsilon)$ fulfills the following correspondences:

(i) $ellip.intersects\,(box, \varepsilon) \equiv distance(A, q, box, \varepsilon) \le \varepsilon$

(ii) $ellip.mindist\,(box) \equiv distance(A, q, box, 0)$

Proof. (i) The estimation $distance(A, q, box, \varepsilon) \le \varepsilon$ holds by definition if and only if the minimum distance d_{min} of *ellip* to *box* is lower or equal to ε. On the other hand, $d_{min} \le \varepsilon$ is true if and only if the hyperrectangle *box* intersects the ellipsoid *ellip* of level ε. (ii) Since $d_{min} \ge 0$, $distance(A, q, box, 0)$ always returns the actual minimum $d_{min} = \min\{d_A^2(p, q) \mid p \in box\}$ which is never less than $\varepsilon = 0$. ◊

For range query processing, only intersection has to be tested. Lemma 1 helps to improve the runtime efficiency, since the exact value of *mindist* is not required as long as it is smaller than ε (cf. figure 3).

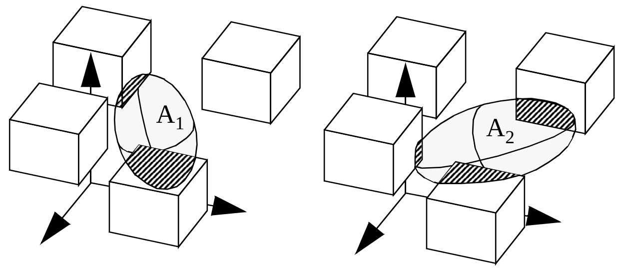

Figure 3: Problem *ellipsoid intersects box* for two different similarity matrices A_1 and A_2

3.2 Basic Distance Algorithm for Ellipsoids and Boxes

For the evaluation of $distance(A, q, box, \varepsilon)$, we combined two paradigms, the steepest descent method, and iteration over feasible points (cf. figure 4).

For the steepest descent, the gradient $\nabla_{\mathrm{ellip}}(p_i)$ of the ellipsoid function $d^2_{A,\,\mathrm{origo}}(\mathrm{p}) = p \cdot A \cdot p^T$ at p_i is determined (step 4). In step 7, the linear minimization returns the scaling factor s for which $p + sg$ is minimum with respect to the ellipsoid; this holds if $\nabla_{\mathrm{ellip}}(p + sg) \cdot g = 0$. The steepest descent works correctly and stops after a finite number of iterations (step 9) [PTVF 92].

The feasible points paradigm is adapted from the linear programming algorithm of [BR 85]. The basic idea is that every point that is visited on the way down to the minimum should belong to the feasible region which is the box in our case. The algorithm ensures the feasibility of the visited

```
method distance(A, q, box, ε) → float;
0   box := box.move (– q);                        // consider difference vectors p = x – q, x ∈ box
1   p_0 := box.closest (origo);                   // ‘closest’ with respect to the Euclidean distance
2   loop
3       if (d^2_{A, origo}(p_i) ≤ ε ) break;      // ellipsoid is reached
4       g := – ∇_ellip (p_i) ;                    // descending gradient of the ellipsoid at p
5       g := box.truncate (p_i, g);               // gradient truncation with respect to the box
6       if (|g| = 0) break;                       // no feasible progress in the truncated gradient direction
7       s := – ∇_ellip (p_i)*g/∇_ellip (g)*g ;    // linear minimization from p along the direction g
8       p_{i+1} := box.closest (p_i + s*g);       // projection of the new location p onto the box
9       if (d^2_{A, origo}(p_i) ≈ d^2_{A, origo}(p_{i+1}) ) break;   // no more progress has been achieved
10  endloop
11  return d^2_{A, origo}(p_i) ;                  // return final ellipsoid distance
end distance;
```

Figure 4: The basic algorithm $distance(A, q, box, \varepsilon)$ iterates over feasible points p_i within the *box* until ε or the constrained minimum of the ellipsoid is reached

points by the closest point operation provided for the box type. As well as the starting point, all the points that are reached by the iteration are projected onto the box (steps 1 and 8). For any point *p*, this projection yields the closest point of the box according to the Euclidean distance. Since the boxes are rectilinear, the projection is simple: For each dimension *d*, set *p*[*d*] to *box*.lower[*d*] if $p[d] < box.\text{lower}[d]$, and set it to *box*.upper[*d*] if $p[d] > box.\text{upper}[d]$. Nothing has to be done if already $box.\text{lower}[d] \le p[d] \le box.\text{upper}[d]$.

In order to proceed fast from the current point p_i to the desired minimum, we decompose the gradient *g* into two components $g = g_{feasible} + g_{leaving}$, and reduce *g* to the direction $g_{feasible}$ that does not leave the box when it is affixed to *p* (step 5). For rectilinear boxes, the operation *box*.truncate (*p*, *g*) is easily performed by nullifying the leaving components of the gradient *g*: For each dimension *d*, set *g*[*d*] to 0 if $g[d] < 0$ and $p[d] = box.\text{lower}[d]$, or if $g[d] > 0$ and $p[d] = box.\text{upper}[d]$.

Since the evaluation of both, the ellipsoid value $d^2_{A,\,\text{origo}}(\text{p}) = p \cdot A \cdot p^{\mathrm{T}}$, and the gradient vector $\nabla_{\text{ellip}}(\text{p}) = 2 \cdot A \cdot p^{\mathrm{T}}$, requires $O(N^2)$ time for dimensionality *N*, the overall runtime of $distance(A, q, box, \varepsilon)$ is $O(\#iter \cdot N^2)$ where *#iter* denotes the number of iterations. Note that our starting point $p_0 \in box$ is closest to the query point in the Euclidean sense. Thus, if *A* is a diagonal matrix, the algorithm immediately stops within the first iteration, which guarantees a runtime complexity of $O(N^2)$. For the non-Euclidean case, we typically obtained *#iter* close to 1 and never greater than 8 from our experiments over various dimensions and query matrices.

4. Efficient Similarity Search in High Dimensions

In principle, the algorithms presented above also apply to high-dimensional feature spaces. In practice, however, efficiency problems will occur due to the following two obstacles (cf. [Fal+ 94]): First, the *quadratic nature of the distance function* causes an evaluation time per object that is quadratic in the number of dimensions. Second, the *curse of dimensionality* strongly restricts the usefulness of index structures for very high dimensions. Although access methods are available that efficiently support query processing in high dimensions, such as the X-tree [BKK 96], the lower dimensionality promises the better performance.

A key to efficiently support query processing in high-dimensional spaces is the paradigm of multi-step query processing [OM 88] [BHKS 93] [BKSS 94] in combination with techniques for reducing the dimensionality (cf. [Fal+ 94]). In [Kor+ 96], index-based algorithms for similarity query processing are presented that guarantee no false drops if the feature distance function is a lower bound of the actual object distance function. Adapting these techniques, we use a reduced similarity function as feature distance for which we prove a greatest lower bound property thus even ensuring optimal filtering in the reduced data space.

4.1 Reduction of Dimensionality

A common technique for indexing high-dimensional spaces is to reduce the dimensionality of the objects in order to obtain lower-dimensional index entries. A variety of reduction techniques is available: The data-dependent Karhunen-Loève transform (KLT) as well as data-independent methods such as feature sub-selection, histogram coarsening, Discrete Cosine (DCT), Fourier (DFT), or wavelet transforms (cf. [Fal+ 94]).

All these techniques conceptually perform the reduction in two steps: First, they map the N-vectors into a space of the same dimensionality N using an information-preserving transformation. Second, they select r components (e.g. the first ones, or the most significant ones) from the transformed N-vectors to compose the reduced r-vectors that will be managed by using an r-dimensional index.

Every linear reduction of dimensionality can be represented by a $N \times r$-matrix R when including the truncation of $N - r$ components. Thus, the reduction can be performed in $O(N \cdot r)$ time. As an example, consider the KLT that is based on a principal component analysis of the vectors in the database. By sorting the components according to their decreasing significance, the first positions of the transformed N-vectors carry the highest amount of information and are selected to form the reduced r-vectors. The linear reduction to k dimensions from [Haf+ 95] depends on the similarity matrix A and is determined by a decomposition of A. Note that also feature sub-selection is a linear reduction technique which can be represented by an $N \times r$-matrix R containing $N - 1$ zeros and a single 1.0 in every of its r columns.

As a final and illustrative example, let us consider coarsening of histograms, i.e. reducing the resolution of a histogram by joining bins. For instance, an N-vector $(x_1, \ldots, x_N)$ is mapped to the corresponding r-vector $(x_1 + \ldots + x_{i_1}, x_{i_1+1} + \ldots + x_{i_2}, \ldots, x_{i_{r-1}+1} + \ldots + x_N)$ simply by summing up the values of neighboring histogram bins. This method is a linear reduction technique which is represented by an $N \times r$-matrix
$$R = \begin{bmatrix} 1\ldots1\ 0\ldots & & \ldots0 \\ 0\ldots0\ 1\ldots1\ 0\ldots & & \ldots0 \\ \ldots & & \ldots \\ 0\ldots & & \ldots0\ 1\ldots1 \end{bmatrix}^{\mathrm{T}}$$
whose entries almost are zero. If a component i of the N-bins contributes to the component j of the r-bins, the entry r_{ij} of the matrix R is set to one.

Although for the KLT, the original $N \times N$ transformation matrix may be given, in general, only the truncated $N \times r$ reduction matrix R will be available. Obviously, such a reduction matrix cannot be inverted. However, our algorithm requires a certain kind of 'inversion', which we will introduce now. For a given $N \times r$ reduction matrix R, let us define the B-complemented $N \times N$-matrix R^B by appending an arbitrary $N \times (N - r)$-matrix B to the right of R. For instance, B may be the $N \times (N - r)$ null matrix, leading to the 0-complementation R^0.

Lemma 2. For any $N \times r$ reduction matrix R for which R^0 has a rank of r, a B-complementation R^B can be computed that is non-singular, i.e. whose inverse $(R^B)^{-1}$ exists.

Proof. Let B be an orthonormal set of basis vectors that span the $(N - r)$-dimensional nullspace of the matrix R^0. These vectors can be obtained from the Singular Value Decomposition (SVD) of R^0 (cf. [PTVF 92]). Since the 0-complementation R^0 is assumed to have a rank of r, the B-complementation R^B has the full rank of N and, therefore, its inverse $(R^B)^{-1}$ exists. ◊

Note, if R^0 would have a rank lower than r, the reduction of dimensionality using the matrix R would produce redundancy which we neglect for the subsequent.

4.2 Lower Bounding of Similarity Distances

Let A_N be an $N \times N$ positive definite similarity matrix. For readability, we denote the difference of two N-vectors s and q by $s - q = \Delta_N = (\delta_1, \ldots, \delta_N) \in \Re^N$. Then, the A_N-distance appears as $d_{A_N}(s, q) = \Delta_N \cdot A_N \cdot \Delta_N^{\mathrm{T}}$. Note that the index only manages reduced entries sR and qR. In our notation, the difference vector is $sR - qR = (s - q)R = \Delta_N R \in \Re^r$. Recall that R may be complemented by any matrix B and the reduced vector $\boldsymbol{x}R$ is obtained from $\boldsymbol{x}R^B$ by truncating (nullifying) the last $N - r$ components.

Lemma 3. (*Distance-Preserving Transformation* A_N^R).
For any non-redundant $N \times r$ reduction matrix R and any similarity matrix A_N, there exists a B-complemented $N \times N$-matrix R^B for which the $N \times N$-matrix $A_N^R = (R^B)^{-1} \cdot A_N \cdot (R^{B\mathrm{T}})^{-1}$ preserves the A_N-distance:
$$d_{A_N}(s, q) = d_{A_N^R}(sR^B, qR^B)$$

Proof. According to Lemma 2, for every reduction matrix R, a B-complementation R^B exists that is invertible. We use this particular R^B and get:

$$d_{A_N^R}(sR^B, qR^B) = (\Delta_N R^B) \cdot A_N^R \cdot (\Delta_N R^B)^{\mathrm{T}} =$$

$$= \Delta_N \cdot R^B \cdot (R^B)^{-1} \cdot A_N \cdot (R^{B^{\mathrm{T}}})^{-1} \cdot R^{B^{\mathrm{T}}} \cdot \Delta_N^{\mathrm{T}} =$$

$$= \Delta_N \cdot A_N \cdot \Delta_N^{\mathrm{T}} = d_{A_N}(s, q) \text{ . } \Diamond$$

The transformation of A_N to A_N^R is the first step on our way to a filter distance function $d_{A_r^R}(sR, qR)$ that lower-bounds the object distance function $d_{A_N}(s, q)$, as it is required to guarantee no false drops. In the following, we present the reduction of the query matrix from dimension $N \times N$ to $r \times r$ which is performed in a recursive way from A_k to A_{k-1}, $k = N, \ldots, r+1$, such that the lower-bounding property holds in each step:

$$\forall \Delta_k \in \Re^k\text{: } \Delta_{k-1} \cdot A_{k-1} \cdot \Delta_{k-1}^{\mathrm{T}} \le \Delta_k \cdot A_k \cdot \Delta_k^{\mathrm{T}}$$

However, beyond this weak lower-bounding property, we use a matrix A_{k-1} that represents the greatest lower bound, i.e. the optimum of all reduced distance functions. For this purpose, we partition the matrix A_k by splitting off the last column and row, resulting in $A_k = \begin{bmatrix} \tilde{A}_{k-1} & \boldsymbol{col}_k \\ \boldsymbol{row}_k & \tilde{a}_{kk} \end{bmatrix}$ where $\tilde{A}_{k-1}$ is a $(k-1) \times (k-1)$-matrix, $\tilde{a}_{kk} \in \Re$ is a scalar, and $\boldsymbol{row}_k, \boldsymbol{col}_k^{\mathrm{T}} \in \Re^{k-1}$ are two row vectors which we combine to the row vector $\boldsymbol{last}_k = \frac{1}{2}(\boldsymbol{row}_k + \boldsymbol{col}_k^{\mathrm{T}})$.

Remember that reducing the dimensionality of a vector includes truncation of the component *k*. Therefore, we assume that $\delta_k \in \Re$ is not available.

Lemma 4. (*Greatest Lower Bound*)
For any positive definite $k \times k$ similarity matrix A_k, the $(k-1) \times (k-1)$-matrix $A_{k-1} = \tilde{A}_{k-1} - \dfrac{(\boldsymbol{last}_k^{\mathrm{T}} \cdot \boldsymbol{last}_k)}{\tilde{a}_{kk}}$, which consists of $a_{ij} = \tilde{a}_{ij} - \dfrac{(\tilde{a}_{ik} + \tilde{a}_{ki})(\tilde{a}_{kj} + \tilde{a}_{jk})}{4\tilde{a}_{kk}}$ for $1 \le i, j \le k-1$, defines a distance function $d_{A_{k-1}}$ which is the minimum and, therefore, the greatest lower bound of d_{A_k} for the case that the value of $\delta_k \in \Re$ is unknown:

$$\forall \Delta_k \in \Re^k\text{: } \Delta_{k-1} \cdot A_{k-1} \cdot \Delta_{k-1}^{\mathrm{T}} = \min\{\Delta_k \cdot A_k \cdot \Delta_k^{\mathrm{T}} \mid \delta_k \in \Re\}$$

Proof. Note that the matrix A_{k-1} is well defined since $\tilde{a}_{kk} = (0, \ldots, 0, 1) \cdot A_k \cdot (0, \ldots, 0, 1)^{\mathrm{T}} > 0$ for any positive definite matrix A_k. By expansion of A_k, followed by a quadratic complementation, we obtain:

$$\Delta_k \cdot A_k \cdot \Delta_k^{\mathrm{T}} =$$

$$= \Delta_{k-1} \cdot \tilde{A}_{k-1} \cdot \Delta_{k-1}^{\mathrm{T}} + 2\delta_k \boldsymbol{last}_k \Delta_{k-1}^{\mathrm{T}} + \tilde{a}_{kk}\delta_k^2 =$$

$$= \Delta_{k-1} \cdot \tilde{A}_{k-1} \cdot \Delta_{k-1}^{\mathrm{T}} - \frac{1}{\tilde{a}_{kk}}(\boldsymbol{last}_k \cdot \Delta_{k-1}^{\mathrm{T}})^2 + $$

$$+ \frac{1}{\tilde{a}_{kk}}(\boldsymbol{last}_k \cdot \Delta_{k-1}^{\mathrm{T}} + \tilde{a}_{kk}\delta_k)^2 =$$

$$= \Delta_{k-1} \cdot \left(\tilde{A}_{k-1} - \frac{1}{\tilde{a}_{kk}}(\boldsymbol{last}_k^{\mathrm{T}} \cdot \boldsymbol{last}_k)\right) \cdot \Delta_{k-1}^{\mathrm{T}} +$$

$$+ \frac{1}{\tilde{a}_{kk}}(\boldsymbol{last}_k \cdot \Delta_{k-1}^{\mathrm{T}} + \tilde{a}_{kk}\delta_k)^2 =$$

$$= \Delta_{k-1} \cdot A_{k-1} \cdot \Delta_{k-1}^{\mathrm{T}} + \frac{1}{\tilde{a}_{kk}}(\boldsymbol{last}_k \cdot \Delta_{k-1}^{\mathrm{T}} + \tilde{a}_{kk}\delta_k)^2 .$$

The second term of the sum is a square, and therefore, its minimum is zero which will be reached for a certain $\delta_k \in \Re$. However, since δ_k is assumed to be not available, we may only rely on the minimum, zero. Therefore, the first term of the sum, $\Delta_{k-1} \cdot A_{k-1} \cdot \Delta_{k-1}^{\mathrm{T}}$, represents the minimum of the left hand side, $\Delta_k \cdot A_k \cdot \Delta_k^{\mathrm{T}}$, for all $\Delta_k \in \Re^k$. $\Diamond$

4.3 Multi-Step Similarity Query Processing

For multi-step similarity query processing, we adapt the algorithms of [Kor+ 96] for range queries and *k*-nn queries. In our case, we reduce the *N*-vectors *s* to *r*-vectors *sR* using an $N \times r$ reduction matrix *R*. In order to obtain an appropriate filter distance function, we also reduce the original $N \times N$ similarity matrix A_N to the $r \times r$ query matrix A_r^R. Since the lower-bounding property holds, $d_{A_r^R}(sR, qR) \le d_{A_N}(s, q)$, the method prevents false drops. In addition, the greatest-lower-bound property ensures optimal filtering for a given reduction *R*.

In figure 5, we present our algorithm for matrix reduction. We assume the inverse complemented reduction matrix $(R^B)^{-1}$ to be precomputed (cf. Lemma 2). The Lemmata 3 and 4 ensure the correctness of the algorithm. From a geometric point of view, step 1 performs a rotation, and step 2 a projection (not intersection!) of the *N*-dimensional query ellipsoid to *r* dimensions according to the reduction matrix *R*. The overall runtime complexity is $O(N^3)$.

Finally, figure 6 shows the adapted versions of the algorithm from [Kor+ 96] for multi-step similarity query processing which we call $\mathrm{SIM}_{\mathrm{range}}(A, q, \varepsilon)$ for range queries

REDUCE_MATRIX $(A_N, (R^B)^{-1}) \longrightarrow A_r^R$

(1) *Distance-preserving rotation (cf. Lemma 3):*

Transform A_N to $A_N^R = (R^B)^{-1} \cdot A_N \cdot (R^{B^{\mathrm{T}}})^{-1}$

(2) *Projection (cf. Lemma 4):*

For k from N down to $r+1$, reduce A_k^R to A_{k-1}^R

Figure 5: Algorithm to transform the original query matrix A_N into the reduced query matrix A_r^R

and $\mathrm{SIM}_{k\text{-nn}}(A, q, k)$ for k-nn queries, where q denotes the query object.

Algorithm SIM$_{\mathbf{range}}$ $(A_N, A_r^R, \boldsymbol{q}, \varepsilon)$

- *Preprocessing*. Reduce the query point q to qR
- *Filter Step*. Perform an ellipsoid range query on the SAM to obtain $\{s \mid d_{A_r^R}(sR, qR) \le \varepsilon\}$
- *Refinement Step*. From the candidates set, report the objects s that fulfill $d_{A_N}(s, q) \le \varepsilon$

Algorithm SIM$_{\boldsymbol{k}\text{-}\mathbf{nn}}$ $(A_N, A_r^R, \boldsymbol{q}, \boldsymbol{k})$

- *Preprocessing*. Reduce the query point q to qR
- *Primary Candidates*. Perform an ellipsoid k-nn query around qR with respect to $d_{A_r^R}$ on the SAM
- *Range Determination*. For the primary candidates s, determine $d_{\max} = \max\{d_{A_N}(s, q)\}$
- *Final Candidates*. Perform an ellipsoid range query $\{s \mid d_{A_r^R}(sR, qR) \le d_{\max}\}$ on the SAM
- *Final Result*. Rank the final candidates s by $d_{A_N}(s, q)$, and report the top k objects

Figure 6: Algorithms for range queries and k-nn queries based on a SAM (adapted from [Kor+ 96])

5. Experimental Evaluation

We implemented and evaluated our algorithms on image databases containing some 12,000 color pictures from commercially available CD-ROMs. We compare our method to the QBIC techniques which had been tested on a database of some 950 images [Haf+ 95] [Fal+ 94]. According to the QBIC evaluation, we computed 64D and 256D color histograms for the images and used the formula $A_\sigma[i, j] = \exp(-\sigma(d_{ij}/d_{\max}))$ to generate similarity matrices. Since our method supports query processing for a variety of similarity matrices on the same index, we instantiated query matrices A_σ for various values of σ. According to [Haf+ 95], we performed the symmetric decomposition $A_\sigma = L_\sigma \cdot L_\sigma^{\mathrm{T}}$ and selected the first r columns of L_σ to obtain the reduction matrices R for various dimensionalities r. We managed the reduced data spaces by using X-trees [BKK 96]. All algorithms were implemented in C++ and evaluated on an HP-735 running under HP-UX 9.01.

Figure 7 demonstrates the selectivity of the filter step. For reducing the dimensionality, we used various r-indexes (k-indexes in [Haf+ 95]) for the similarity matrix A_{10} and the reduced dimensions $r \in \{3, 6, 9, 12, 15\}$. We measured the average selectivity of some 100 sample queries retrieving fractions up to 1% (120 images) of the database. Hardly, a user may visually handle more than this number of results. We simulated the method of [Haf+ 95] while decomposing the query matrix A_{10} to the reduction matrix. Additionally, we changed the query matrix to A_8 and A_{12} thus demonstrating the flexibility of our method without loss of efficiency.

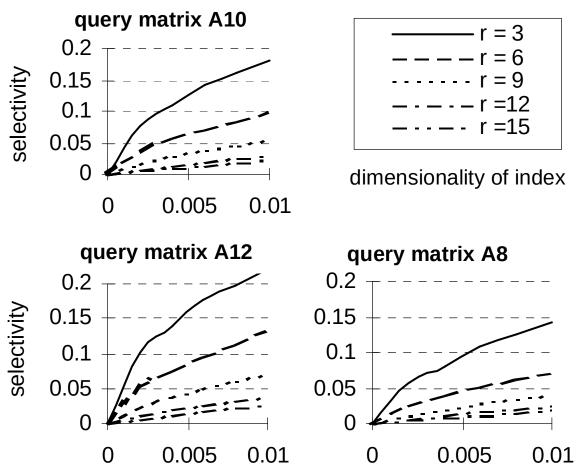

Figure 7: Selectivity of the filter step for various query matrices. The diagrams depict the fractions retrieved from indexes (y-axis) of dimensionality 3 to 15 depending on the fraction requested from all 12,000 images (x-axis).

In all examples, the selectivity increases with the dimensionality of the index and is approximately 20% for $r = 3$, 10% for $r = 6$, 5% for $r = 9$, and below 3% for $r > 12$. For the modified query matrices, the selectivity values change only slightly.

Figure 8 indicates that the selectivity is affected by the technique for reducing the dimensionality which may be adapted to the application characteristics as an advantage of our approach.

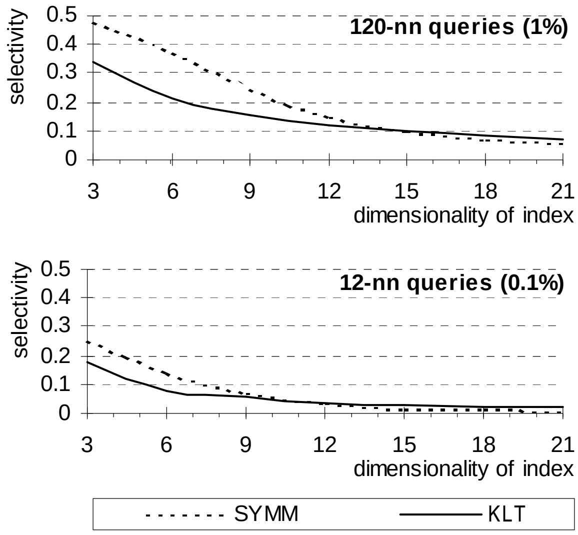

Figure 8: Selectivity of the filter step for various techniques to reduce the dimensionality. The symmetric decomposition of the query matrix (SYMM) is compared to the Karhunen-Loeve Transform (KLT).

Finally, we show the efficiency of the multi-step query processing technique, averaged over 120 k-nn queries with $k = 12$ on 256D histograms. Figure 9 depicts the number of candidates and the number of index page accesses depending on the dimensionality of the index. A good selectivity of the filter step is important since each candidate will cause a page access in the refinement step, and the exact evaluation is expensive in 256D. Figure 10 depicts the overall runtime and its components depending on the index dimensionality. As expected, the refinement time decreases with the dimensionality due to the decreasing number of candidates. On the other hand, the time for the filter step increases with the index dimensionality. Acceptable runtimes (below 20 sec) are achieved for dimensions $r \geq 15$, and the overall minimum (below 10 sec) is reached for $r \approx 30$. We observe that the overall runtime does not significantly vary for a wide range of index dimensionalities.

6. Conclusion

In this paper, we address the problem of similarity search in large databases. Many applications require that the similarity function reflects mutual dependencies of components in feature vectors, e.g. of neighboring histogram bins. Whereas the Euclidean distance ignores correlations of vector components even in the weighted case, quadratic form distance functions fulfill this requirement leading to ellipsoid queries as a new query type. In addition,

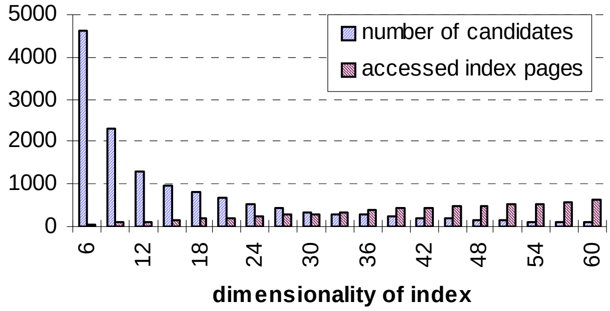

Figure 9: Efficiency of multi-step query processing for 120 k-nn queries with $k = 12$ which represents 0.1% of the data. The diagram indicates the number of candidates obtained from the filter step and the number of pages read from the index depending on the index dimensionality.

the similarity function should be adaptable to user preferences at query time. While current index-based query processing does not adequately support this task, we present efficient algorithms to process ellipsoid queries using spatial access methods. The method directly applies to low-dimensional spaces, and the multi-step query processing paradigm efficiently supports similarity search in high-dimensional spaces. Available techniques for reducing the dimensionality apply to data vectors but have to be adapted to reduce the query matrix, too. We present an algorithm to reduce similarity matrices leading to reduced ellipsoid queries that are efficiently supported by the index. We prove that the resulting reduced similarity function represents the greatest lower bound of the original similarity function thus guaranteeing no false drops as well as optimal selectivity for any given reduction.

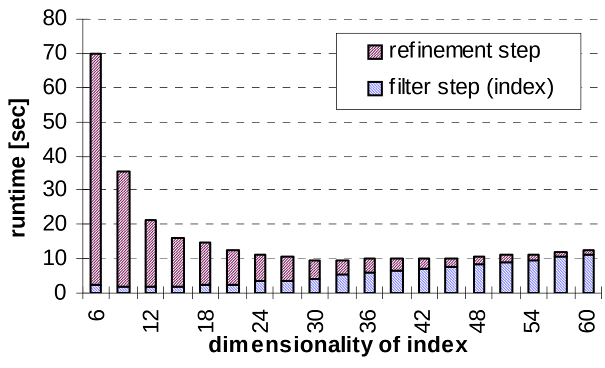

Figure 10: Overall runtime of multi-step query processing, divided into the times of filter and refinement step and averaged over 120 k-nn queries with $k = 12$, depending on the dimensionality of the index.

We implemented our algorithms and compared them to techniques which were developed in the QBIC project [Fal+ 94] [Haf+ 95]. Our approach provides two advantages: It is not committed to a fixed similarity matrix after the index has been created, and the dimensionality of the index may be adapted to the characteristics of the application. In other words, query processing is supported for a variety of similarity matrices on any existing precomputed index. The experiments were performed on image databases containing color histograms of some 12,000 images. The good efficiency of our method is demonstrated by both, the high selectivity of the filter step as well as the good performance of ellipsoid query processing on the index.

In our future work, we plan to investigate how various techniques for the reduction of dimensionality affect the performance of query processing. Additionally, we will apply our method to other application domains such as geometric shape retrieval in CAD and 3D protein databases.

References

[AFS 93] Agrawal R., Faloutsos C., Swami A.: *'Efficient Similarity Search in Sequence Databases'*, Proc. 4th. Int. Conf. on Foundations of Data Organization and Algorithms (FODO'93), Evanston, IL, in: Lecture Notes in Computer Science, Vol. 730, Springer, 1993, pp. 69-84.

[ALSS 95] Agrawal R., Lin K.-I., Sawhney H. S., Shim K.: *'Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases'*, Proc. 21st Int. Conf. on Very Large Databases (VLDB'95), Morgan Kaufmann, 1995, pp. 490-501.

[BBKK 97] Berchtold S., Böhm C., Keim D. A., Kriegel H.-P.: *'A Cost Model for Nearest Neighbor Search in High-Dimensional Data Spaces'*, Proc. 16th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems (PODS), Tucson, AZ, 1997, pp. 78-86.

[Ber+ 97] Berchtold S., Böhm C., Braunmüller B., Keim D. A., Kriegel H.-P.: *'Fast Parallel Similarity Search in Multimedia Databases'*, SIGMOD'97 best paper award, Proc. ACM SIGMOD Int. Conf. on Management of Data, Tucson, AZ, 1997, pp. 1-12.

[BHKS 93] Brinkhoff T., Horn H., Kriegel H.-P., Schneider R.: *'A Storage and Access Architecture for Efficient Query Processing in Spatial Database Systems'*, Proc. 3rd Int. Symp. on Large Spatial Databases (SSD'93), Singapore, 1993, in: Lecture Notes in Computer Science, Vol. 692, Springer, pp. 357-376.

[BKK 96] Berchtold S., Keim D. A., Kriegel H.-P.: *'The X-tree: An Index Structure for High-Dimensional Data'*, Proc. 22nd Int. Conf. on Very Large Data Bases (VLDB'96), Mumbai, India, 1996, pp. 28-39.

[BKK 97] Berchtold S., Keim D. A., Kriegel H.-P.: *'Using Extended Feature Objects for Partial Similarity Retrieval'*, accepted for publication in the VLDB Journal.

[BK 97] Berchtold S., Kriegel H.-P.: *'S3: Similarity Search in CAD Database Systems'*, Proc. ACM SIGMOD Int. Conf. on Management of Data, Tucson, AZ, 1997, pp. 564-567.

[BKSS 90] Beckmann N., Kriegel H.-P., Schneider R., Seeger B.: *'The R*-tree: An Efficient and Robust Access Method for Points and Rectangles'*, Proc. ACM SIGMOD Int. Conf. on Management of Data, Atlantic City, NJ, 1990, pp. 322-331.

[BKSS 94] Brinkhoff T., Kriegel H.-P., Schneider R., Seeger B.: *'Efficient Multi-Step Processing of Spatial Joins'*, Proc. ACM SIGMOD Int. Conf. on Management of Data, Minneapolis, MN, 1994, pp. 197-208.

[BR 85] Best M. J., Ritter K.: *'Linear Programming. Active Set Analysis and Computer Programs'*, Englewood Cliffs, NJ, Prentice Hall, 1985.

[Fal+ 94] Faloutsos C., Barber R., Flickner M., Hafner J., Niblack W., Petkovic D., Equitz W.: *'Efficient and Effective Querying by Image Content'*, Journal of Intelligent Information Systems, Vol. 3, 1994, pp. 231-262.

[FRM 94] Faloutsos C., Ranganathan M., Manolopoulos Y.: *'Fast Subsequence Matching in Time-Series Databases'*, Proc. ACM SIGMOD Int. Conf. on Management of Data, Minneapolis, MN, 1994, pp. 419-429.

[GM 93] Gary J. E., Mehrotra R.: *'Similar Shape Retrieval using a Structural Feature Index'*, Information Systems, Vol. 18, No. 7, 1993, pp. 525-537.

[Gut 84] Guttman A.: *'R-trees: A Dynamic Index Structure for Spatial Searching'*, Proc. ACM SIGMOD Int. Conf. on Management of Data, Boston, MA, 1984, pp. 47-57.

[Haf+ 95] Hafner J., Sawhney H. S., Equitz W., Flickner M., Niblack W.: *'Efficient Color Histogram indexing for Quadratic Form Distance Functions'*, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 17, No. 7, 1995, pp. 729-736.

[HS 95] Hjaltason G. R., Samet H.: *'Ranking in Spatial Databases'*, Proc. 4th Int. Symposium on Large Spatial Databases (SSD'95), in: Lecture Notes in Computer Science, Vol. 951, Springer, 1995, pp. 83-95.

[Jag 91] Jagadish H. V.: *'A Retrieval Technique for Similar Shapes'*, Proc. ACM SIGMOD Int. Conf. on Management of Data, Denver, CO, 1991, pp. 208-217.

[Kor+ 96] Korn F., Sidiropoulos N., Faloutsos C., Siegel E., Protopapas Z.: *'Fast Nearest Neighbor Search in Medical Image Databases'*, Proc. 22nd VLDB Conference, Mumbai, India, 1996, pp. 215-226.

[OM 88] Orenstein J. A., Manola F. A.: *'PROBE Spatial Data Modeling and Query Processing in an Image Database Application'*, IEEE Trans. on Software Engineering, Vol. 14, No. 5, 1988, pp. 611-629.

[PTVF 92] Press W. H., Teukolsky S. A., Vetterling W. T., Flannery B. P.: *'Numerical Recipes in C'*, 2nd ed., Cambridge University Press, 1992.

[RKV 95] Roussopoulos N., Kelley S., Vincent F.: *'Nearest Neighbor Queries'*, Proc. ACM SIGMOD Int. Conf. on Management of Data, San Jose, CA, 1995, pp. 71-79.

[SRF 87] Sellis T., Roussopoulos N., Faloutsos C.: *'The R+-Tree: A Dynamic Index for Multi-Dimensional Objects'*, Proc. 13th Int. Conf. on Very Large Databases, Brighton, England, 1987, pp. 507-518.