On (not) indexing quadratic form distance by metric access methods

Tomáš Skopal, Tomáš Bartoš, Jakub Lokoč

SIRET Research Group, Faculty of Mathematics and Physics,
Charles University in Prague, Malostranské nám. 25, 118 00, Prague, Czech Republic
{skopal, bartos, lokoc}@ksi.mff.cuni.cz

ABSTRACT

The quadratic form distance (QFD) has been utilized as an effective similarity function in multimedia retrieval, in particular, when a histogram representation of objects is used. Unlike the widely used Euclidean distance, the QFD allows to arbitrarily correlate the histogram bins (dimensions), allowing thus to better model the similarity between histograms. However, unlike Euclidean distance, which is of linear time complexity, the QFD requires quadratic time to evaluate the similarity of two objects. In consequence, indexing and querying a database under QFD are expensive operations. In this paper we show that, given static correlations between dimensions, the QFD space can be transformed into an equivalent Euclidean space. Thus, the overall complexity of indexing and searching in the QFD similarity model can be reduced qualitatively. Besides the theoretical time complexity analysis of our approach applied to several metric access methods, in experimental evaluation we show the real-time speedup on a real-world image database.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content analysis and indexing—*Indexing methods*

General Terms

Algorithms, theory, performance

Keywords

quadratic form distance, metric indexing, similarity search

1. INTRODUCTION

In the area of multimedia databases, we distinguish between two means of multimedia retrieval [8]. The first one is based on the keyword search, where a multimedia object (e.g., an image) is annotated and searched by keywords describing its content. The second mean is natively based on searching by the content itself, so the multimedia objects are searched according to the semantics of their (low-level) features. The most common content-based model considers the similarity search, where a pairwise similarity (or distance) function is defined on the descriptors of multimedia objects. The descriptor is determined from the respective multimedia object by a suitable feature extraction procedure, while it is often represented as a high-dimensional vector. The query is modeled by the usage of a sample multimedia object (its descriptor, respectively), so that the database objects can be thought as ordered by their similarity/distance to the query example. The most popular query types, the k nearest neighbors (kNN) query and the range query, are then defined as a prefix of this ordering (i.e., either first k objects in the ordering, or a part bounded by an object beyond the user-defined similarity threshold, respectively).

From the effectiveness point of view, the research is focused on proposing models complying with the human perception of the similarity, hence, seeking for the appropriate feature extraction technique and the similarity function. From the efficiency point of view, the research efforts are devoted to indexing techniques that allow fast query processing, given a particular similarity model. In this paper, we are concerned with the efficiency of indexing and searching, when using similarity models employing *quadratic form distance* (QFD) as the similarity function.

1.1 Measuring Vector Similarity

The widely used functions for measuring similarity of n-dimensional vectors is the class of Minkowski (L_p) distances:

$$\mathrm{L}_p(u,v) = \left(\sum_{i=1}^{n} |u_i - v_i|^p\right)^{1/p}, p \geq 1$$

Among all Minkowski metrics, the *Manhattan distance* (L_1), the *Euclidean distance* (L_2), and the *Chessboard distance* (L_∞) are used in multimedia retrieval.

When analyzing the suitability of a distance function for the similarity search, we observe that L_p distances suppose all the dimensions of the vector space to be independent (not correlated). In other words, the difference $|u_i - v_i|$ in the i-th dimension of the two vectors contributes to the final distance value independently, regardless of differences in other dimensions. Hence, an L_p distance is suitable for vector space models with no "crosstalk" between dimensions. For example, consider a vector describing a person as $u =$ [age: 54, height: 180cm, siblings: 2]. Here the dimensions are obviously independent, so measuring similarity in this simple model by a (weighted) L_p distance makes sense.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
EDBT 2011, March 22–24, 2011, Uppsala, Sweden.
Copyright 2011 ACM 978-1-4503-0528-0/11/0003 ...$10.00

However, in multimedia retrieval we often encounter vector models where the individual dimensions are (expected to be) correlated. The perfect case is a *histogram* on a homogeneous domain, that is, a vector collecting occurrences of particular properties in individual bins (dimensions). For instance, a simple representation of an image could be a 256-dimensional histogram describing the proportions of individual colors. The i-th histogram bin represents the number of pixels in the image having the i-th color. Measuring the similarity of two histograms by an L_p distance would be inappropriate, since it ignores the dimension ordering. In turn, an L_p distance would incorrectly measure an image of a sunset (red tones) as more similar to a tennis ball (yellow tones) than to an orange fruit (orange tones).

1.2 Quadratic Form Distance

To overcome the insensitivity of L_p to other dimensions, the *Quadratic Form Distance* (QFD) was proposed as a suitable function for similarity measuring, defined as

$$QFD_A(u,v) = \sqrt{(u-v)A(u-v)^T},$$

where A is required to be an $n \times n$ positive-definite matrix (called the *similarity matrix* or the *QFD matrix*). If the matrix A is diagonal, we get a reduction to weighted Euclidean distance, while an identity matrix A reduces QFD to the ordinary Euclidean distance. Importantly, the QFD is a metric distance and from now on we will call the respective metric space the *QFD space* (we will discuss the properties of QFD in detail in Section 3.2).

One of the first widely-known works where the QFD was defined and successfully used were [19], [14] and later [18]. In particular, the QFD was proved as an effective way of searching for similarities in a set of color images. The reason of better applicability was the support of correlations between individual dimensions (using the QFD matrix A), contributing to more robust similarity measuring.

For example, consider a 3-dimensional space, where the dimensions represent the numbers of red, green, and blue pixels in an image. Because the human perception views green and blue colors as more similar than reds and blues or reds and greens, the matrix A could be set as follows:

$$A = \begin{array}{c} \\ R \\ G \\ B \end{array} \begin{array}{c} \begin{array}{ccc} R & G & B \end{array} \\ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0.5 \\ 0 & 0.5 & 1 \end{pmatrix} \end{array}$$

A number of algorithms and image retrieval methods using QFD were developed within the QBIC project [24, 18, 14]. Other applications of QFD similarity search include 2D & 3D shapes [2, 3], protein structures [4, 15, 16], or flow cytometry [6]. In almost all the cited applications, the QFD matrix A is supposed static (not changing from query to query), while the correlations between dimensions are defined based on a scoring function related to the particular domain. Hence, the matrix A is not a query parameter and also it is not data-dependent. For RGB image histograms, the matrix A might be defined according to [18] as:

$$A_{ij} = 1 - \frac{d_{ij}}{d_{max}}, \text{where } d_{max} = max_{i,j}(d_{ij})$$

where d_{ij} is the Euclidean distance between representatives of colors i and j in the RGB color space.

1.2.1 Dynamic distances based on QFD

Although most of the approaches employ static QFD matrix, there appear also applications that use dynamic QFD matrix or even a kind of generalized QFD distance.

A characteristic usage of a dynamic similarity matrix in QFD is shown in several papers such as [20], [26], or [2]. In [20] the authors propose a general method of iteratively guessing the distance function based on user preferences (i.e., *MindReader*). This concept uses QFD and tries to determine which attributes are important and find correlations between them to satisfy the user query. The principle of finding the ideal distance function (changing the similarity matrix in QFD) is very similar to relevance feedback techniques. According to multiple data examples (with scores), the method guesses and refines the implied distance function within several iterations.

A recent promising variant of QFD, the *signature quadratic form distance* (SQFD) [5], enables to use feature signatures (vectors of variable dimensionality) instead of just feature histograms (vectors of fixed dimensionality). In fact, the SQFD concatenates the compared signatures u, v into a vector $(u| - v)$, followed by the usual QFD computation, i.e., $SQFD(u,v) = \sqrt{(u| - v)A(u| - v)^T}$. This also requires a dynamic QFD matrix A that fits the particular features included in the signatures u, v. In other words, the SQFD constitutes a dynamic extension over the original QFD function, and it has proved a superior effectiveness in image classification applications based on feature signatures.

2. INDEXING SIMILARITY

In addition to the modeling of domain-specific similarity search problems in vector spaces, there were substantial efforts spent on developing indexing techniques that speed up the similarity queries in a large database. We distinguish two classes of database indexing methods that are used for the similarity search in *vector spaces*, both with some pros and cons.

2.1 Spatial Access Methods

The *spatial access methods* (SAM) [9] mostly treat the vector space independently of the distance function used for the similarity search. Hence, a SAM index is constructed using the vectorial structure of the descriptor (the values in individual dimensions are used). In particular, we name the R-tree family, X-tree, or VA-file, as representative SAMs.

As the SAM index is not dependent on a particular distance function, a distance function could be provided right at the query time as a parameter, allowing thus flexible similarity searches. This is especially important for QFD in applications when the matrix A needs to be adjusted from query to query, for example, when user preferences have to be incorporated into the similarity function.

On the other hand, the independence from the distance function is also the drawback of SAMs. Since a SAM indexes the database objects within (rectangular) regions minimizing the volumes, surfaces and overlaps (e.g., the volume of MBRs in case of R-tree), the objects in regions do not form tight clusters with respect to the distance function that will be used for querying. In consequence, the regions could be unnecessarily large, which leads to poor filtering ability and thus slower query processing. This negative effect is even magnified with the increasing dimensionality of the space

(the so-called *curse of dimensionality*), while it was many times proved that the data of the dimensionality beyond 10-20 cannot be efficiently searched by SAMs[1], regardless of the distance function used for queries [7, 1].

2.2 Metric Access Methods

The *metric access methods* (or metric indexes) [28, 12] represent a different indexing concept, treating the vector space together with the distance function as a black-box metric space. That is, only the distances between vectors can be utilized to build the index, not the particular vector coordinates. Here the pros and cons are exactly opposite as for SAMs. Since MAMs build the index using a particular *static* distance function, they are not suitable when the distance function has to be modified after indexing (e.g., at the query time). For example, changing the QFD matrix A would result in a different distance function than the one used for indexing. Such a change would require a reorganization of the metric index, making thus the actual index (and the distance values stored within) invalid. Nevertheless, as the matrix A is mostly regarded as not changing at query time (see Section 1.2), the requirement on static A should not be a big limitation.

To name the advantages, the MAM index regions are more compact than those of SAMs, since the database objects are organized in clusters gathering objects close in terms of the distance function. In turn, the MAMs are more successful in the fight with the curse of dimensionality, because the embedding (vector) dimensionality is irrelevant for them. Instead, the complexity of MAM indexing is determined by the distance distribution, namely, the *intrinsic dimensionality* [12, 27], which is usually smaller than the embedding dimensionality. In particular, we name the M-tree family, M-index, vp-tree, Pivot tables, GNAT, or SAT as representative MAMs. Since we focus on indexing QFD by MAMs, we further discuss the details of several MAMs in Section 4.

2.3 Indexing Quadratic Form Distance

When looking for an indexing technique suitable for efficient similarity search, we recognize two performance components contributing to the overall real-time cost – the cost of a single distance evaluation and the number of evaluations needed to answer a similarity query. To analyze the first component, we observe the QFD computation is expensive due to the matrix-to-vector multiplications (i.e., $O(n^2)$). Hence, the quadratic complexity of the QFD computation is a serious argument when deciding whether to employ an L_p distance (linear in time) or QFD. When thinking in absolute numbers, the QFD computation is extremely slow when used on high-dimensional histograms (say, $n > 100$). The indexing of QFD should ideally minimize both performance components: the complexity of QFD computation and the number of distances spent by query processing.

2.3.1 Lower-bounding Approaches

In order to increase the efficiency of querying in QFD spaces, there have been several transformational approaches proposed (e.g., in IBM's QBIC system [14, 18]), addressing both of the above discussed performance components.

In the classic work [14], authors suggest an indexing scheme for QFD that considers RGB color images. The scheme includes a lower-bounding of the QFD by an Euclidean-like function (specific to RGB image histograms) together with contractive dimensionality reduction techniques using average RGB colors. The database is then searched in a filter-and-refine way using a simple sequential scan not considering any indexing technique. Because of the applicability only to RGB color images and reduction to 3 dimensions, we consider this method as specific image retrieval model.

A transformational approach that uses a lower-bounding and a dimensionality reduction is proposed in [18] (also for color histograms) with further generalization of the transformation to a k-dimensional space. As in [14], the whole idea is based on a simple filter-and-refine strategy (i.e., no database indexing, just sequential search) where the QFD is replaced by L_p distance during the filtering step. In the refinement step, the QFD is being computed for a possibly smaller set of non-filtered candidates. The authors mention the generalized method only briefly, supposing rank-k SVD decomposition of the QFD matrix, which is used to obtain a matrix providing transformation to a k-dimensional L_p space. They do not explicitly propose $k = n$, that is, a (possibly) homeomorphic transformation of the QFD space into the Euclidean space (exactly preserving the distances). Instead, they present the transformation to k-dimensional space as an integral part of the whole retrieval method. The main contribution of our paper also uses the decomposition of the QFD matrix, however, we only consider the case $k = n$ which allows us to separate the transformation of the QFD space itself from the subsequent indexing method (any metric access method in our case).

A general approach of lower-bounding the QFD distance is used in [26, 2, 4, 21], where SVD-based dimensionality reduction into the lower-dimensional QFD space is considered. Moreover, the transformed database is indexed by the X-tree for more efficient similarity search (with the dimensionality reduced up to 20 in experiments).

The mentioned lower-bounding techniques suffer from several drawbacks. First, some approaches [26, 4] use a dimensionality reduction that treats the entire database as a large matrix that needs to be decomposed (e.g., by SVD or KLT). Hence, the dimensionality reduction represents a pre-processing step that is extremely expensive, for example, $O(m^2 + n^3)$ in case of SVD, where m is the size of database and n is the dimensionality. Second, as the transformation to a smaller dimensionality is only contractive (i.e., the new distances are smaller than the original ones), the lower bounds become less tight as the target dimensionality decreases, leading to more false positives. The false positives then need to be expensively checked by the QFD in the original space. For larger target dimensionality the number of false positives decreases, however, at the cost of inefficient indexing by SAMs (due to the curse of dimensionality). Third, as the dimensionality reduction of data requires the entire database beforehand, some subsequent updates of the database (insertions and deletions) without complete re-indexing could result in distorted embedding, leading to false dismissals and/or less effective filtering.

2.3.2 Indexing QFD using MAMs

Simply said, the research community dealing with general metric access methods views the quadratic form distance as any other expensive distance, like the edit distance. There have been published many papers on MAMs (includ-

[1]Observed and proved for uniformly distributed data and exact search (i.e., a search not allowing false dismissals).

ing the classic monograph [28]) and their applications in image retrieval [11], where the QFD is cited or even used in the experiments, together with other metric distances. From the conceptual point of view, the MAMs are able to improve just one of the discussed performance components – the number of QFD computations. Hence, even though MAMs usually outperform SAMs (as discussed in Section 2.2), they are still too expensive for native indexing of the QFD in real-world applications. Nevertheless, in the following subsection we sketch the idea of our contribution, that allows a MAM-based efficient and exact similarity search in high-dimensional spaces under the QFD.

2.4 Paper Contribution

Comparing to the previous research mentioned in Section 2.3.1, we also consider a space transformation (we call it the QMap model). However, the transformation is not only approximate, but it is a homeomorphic mapping of the source QFD space onto a target Euclidean space, so that the distances are *exactly preserved*. A database described in the transformed space can be then indexed by any MAM or SAM. Obviously, the cost of indexing and querying in Euclidean space is qualitatively cheaper than in QFD space.

To the best of our knowledge, none of the papers mentioned in Section 2.3 explicitly formulates the main outcome of our paper – that there exists such homeomorphic mapping of a QFD space into the Euclidean space. Moreover, unlike the previous approaches, we separate the QMap model from its usage in an access method, providing the freedom of choosing an arbitrary MAM or SAM.

In the following, we show the mentioned transformation always exists, and that it can be cheaply obtained just from the QFD matrix. Later in the paper, we discuss the utilization of the QMap model together with MAMs, including the analysis of the indexing/querying complexity, as well as an experimental evaluation on a real-world image database.

3. THE QMAP MODEL

The classic QFD model manipulates vectors in the QFD space and computes the distances using directly the QFD form. At this moment, we define the *QMap model* that encapsulates the whole concept of transforming the QFD space and using the Euclidean space instead. Like the approaches presented in Section 2.3.1, also the QMap model assumes a static QFD matrix A for the transformation purposes.

The QMap model ensures that all vectors (database or query vectors) from the QFD space will be transformed, prior to any indexing or querying operation, into the Euclidean space, yet preserving exactly the same distances. Moreover, the QMap model constitutes an indexing framework that allows to employ the existing MAMs without the need of modifying them.

3.1 The Idea

The idea of the proposed transformation is sketched in Figure 1, where we show a simple linear transformation of a QFD space into the Euclidean space. When visualizing a QFD ball of an arbitrary radius, we obtain an ellipsoid that is freely oriented in the space, while the orientation is a consequence of the dimension correlations specified in the QFD matrix A. Note that the points on the ellipsoid boundary are equally distant to the ellipsoid center (with respect to the QFD). Also note that all possible ellipsoids

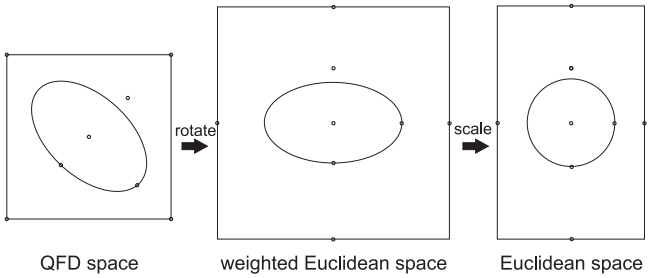

Figure 1: The idea of QMap – homeomorphic transformation of a QFD space into the Euclidean space.

are oriented in the same way. Hence, to obtain an equivalent Euclidean space, it should be sufficient:

- first, to *rotate* the space to obtain the QFD balls having axes parallel to the axes of coordinate system, i.e., transformation to a weighted Euclidean space,
- and, second, to *scale* the space independently in all dimensions to obtain the same diameters of the ellipsoids in all dimensions, that is, obtaining Euclidean balls.

Since the rotation and the scaling are linear transformations, they can be composed to form a single transformation matrix B. The matrix B can be used to transform the database vectors (or a query vector) from the source QFD space into vectors of the target Euclidean space. Most importantly, we require the distances of mapped vectors in the target Euclidean space to be exactly the same as they were in the QFD space.

Although the presented transformation might be viewed as well-known from the mathematic point of view, its application in database indexing, surprisingly, was not explicitly formulated so far (to the best of our knowledge).

As an additional minor contribution of the paper, in the following technical section we prove that it is sufficient to assume the QFD matrix to be symmetric and positive-definite.

3.2 QMap Preliminaries

Before we introduce the QMap transformation in Section 3.3, we summarize some matrix operations and also clarify the assumptions on the QFD matrix (Section 3.2.3).

In the following text, we will often use vectors, matrices, and operations between them. Because we can see an n-dimensional vector as an $1 \times n$ matrix, all matrix operations apply also to vectors and their combinations with matrices.

3.2.1 Basic Matrix Operations

For any two conformable matrices[2] C, D, the matrix *transposition rule* $(CD)^T = D^T C^T$ holds.

Suppose we have three conformable matrices C, D, and E. Then, the matrix *right-hand* distributive law

$$(C + D)E = CE + DE$$

holds (similarly with the left-hand distributive law [22]).

The *matrix associativity* assures that we get the same result no matter in which order we multiply three conformable matrices C, D, E:

$$(CD)E = C(DE)$$

[2]Such matrices, the dimensions of which are suitable for defining some operations (e.g., addition, multiplication, etc).

Suppose we have an $a \times b$ matrix A and a $b \times c$ matrix B. The product of the matrix multiplication AB is an $a \times c$ matrix C for which any item C_{ij} is defined as

$$C_{ij} = \sum_{k=1}^{b} A_{ik} * B_{kj}$$

3.2.2 Cholesky Decomposition

If A is a symmetric positive-definite $n \times n$ matrix, then there exists a unique lower triangular $n \times n$ matrix B (called *Cholesky triangle*) with positive diagonal entries such that

$$A = BB^T.$$

The process of decomposing the matrix A into matrices B and B^T is called *Cholesky decomposition* (or Cholesky factorization [17]).

In order to be clear, we provide the Cholesky decomposition algorithm that gives us the unique lower triangular matrix B from the given QFD matrix A (see Algorithm 1). The time complexity of the presented algorithm is $O(n^3)$, while in common settings the running time is negligible (e.g., 30 milliseconds for $n = 512$ when running on an office PC).

Algorithm 1 Cholesky decomposition (matrix A)

Require: $n \times n$ symmetric positive-definite matrix A

```
 1: B ← A.clone()
 2: for i = 0 to n − 1 do
 3:   for j = i to n − 1 do
 4:     sum = B[i][j]
 5:     for k = i − 1 downto 0 do
 6:       sum = sum − (B[i][k] ∗ B[j][k])
 7:     end for
 8:     if (i = j) then
 9:       if (sum ≤ 0) then
10:         error("Matrix is not positive definite!")
11:         return
12:       end if
13:       B[i][i] = sqrt(sum)
14:     else
15:       B[j][i] = sum / B[i][i]
16:     end if
17:   end for
18: end for
19: B.clearUpperTriangle()
20: return B
```

3.2.3 Properties of the QFD Matrix

Since there appear different assumptions on the QFD matrix A over the available literature, in the following subsection we show that without loss of generality we can assume the matrix A as *positive-definite* and *symmetric*.

Positive Definiteness

In some sources, the QFD matrix A is assumed as positive-semidefinite (i.e., $zAz^T \geq 0$), while in others as positive-definite (i.e., $zAz^T > 0, \forall z \neq \mathbf{0}$).

Let $z = (u - v)$, then from the identity (metric space postulate) it follows $zAz^T = 0 \Leftrightarrow u = v$ (i.e., $z = \mathbf{0}$). Otherwise (for $u \neq v$), it must hold $zAz^T > 0$. Hence, to preserve metric postulates of QFD, the QFD matrix A must be strictly positive-definite.

Symmetry

According to the previous requirements, the QFD matrix must be positive-definite. Furthermore, we show that for any general matrix, we are able to define a symmetric one that will give us the same results when we use it in the QFD form. The following sequence of steps shows how to build the corresponding symmetric matrix.

1. Let A be a general QFD matrix. When computing the quadratic form distance d for any two vectors u, v, we define $z = (u - v)$ to be an n-dimensional vector and we get:

$$d = QFD(u, v) = \sqrt{(u - v)A(u - v)^T} = \sqrt{zAz^T}$$

2. We decompose the inner matrix multiplications into summations according to the matrix multiplication rules. Let us define d' as follows:

$$d' = zAz^T = \sum_{j=1}^{n} (\sum_{i=1}^{n} z_i * A_{ij}) z_j = \sum_{j=1}^{n} \sum_{i=1}^{n} A_{ij} * z_i * z_j$$

3. Furthermore, we would like to replace the matrix A with a symmetric matrix B which will give us the same result. To build this new matrix from the given matrix A, we analyze the result of the previous step according to all $z_i * z_j$ multiplications.

   We can see that for any two dimensions i, j of the vector z, the product of $z_i * z_j$ occurs in the final summation once if $i = j$ or twice if $i \neq j$ (with multiplicators A_{ij} and A_{ji}) because $z_i * z_j = z_j * z_i$.

   Therefore, we can define the elements in a new symmetric matrix B as follows

   - $B_{ii} = A_{ii}$ for $i = 1, 2, .., n$
   - $B_{ij} = B_{ji} = \frac{A_{ij} + A_{ji}}{2}$ for $i, j = 1, 2, .., n$ when $i \neq j$

4. Replacing the matrix A with the matrix B we get

$$\tilde{d} = \sum_{j=1}^{n} (\sum_{i=1}^{n} z_i * B_{ij}) z_j = \sum_{j=1}^{n} \sum_{i=1}^{n} B_{ij} * z_i * z_j$$

   When we analyze the multiplication of $z_i * z_j$ in $\tilde{d}$ (with matrix B) and d' (with matrix A), we get:

   - $i = j$ $(z_i * z_i)$
     The multiplicator in $\tilde{d}$ is B_{ii} which is the same as the multiplicator in d' (A_{ii}) according to the definition of B.
   - $i \neq j$ $(z_i * z_j,\ z_j * z_i)$
     The product $z_i * z_j$ occurs in $\tilde{d}$ twice with multiplicators B_{ij} and B_{ji}. This leads to

$$B_{ij} + B_{ji} = \frac{A_{ij} + A_{ji}}{2} + \frac{A_{ij} + A_{ji}}{2} = A_{ij} + A_{ji}$$

     which is the same as the multiplicator in d'.

   Thus, we proved that replacing the matrix A with the symmetric matrix B does not change the QFD distance computed for any vector z.

The previous steps constitute a guide how to build a symmetric matrix B for a general matrix A that will return the same results when we replace A with B in the QFD form. Hence, we can assume that the QFD matrix is symmetric. Note also that because A is positive-definite and $\tilde{d} = d'$, then also B must be positive-definite.

3.3 QFD-to-L_2 Space Transformation

Without loss of generality, let us have the squared QFD form on n-dimensional vectors u, v:

$$QFD(u,v)^2 = (u-v)A(u-v)^T \quad (1)$$

We proved that without any consequences, we can suppose the QFD matrix A to be positive-definite and symmetric. Therefore, we can apply the Cholesky decomposition (see Section 3.2.2) to the matrix A and we get a unique lower triangular matrix B such that

$$BB^T = A$$

After replacing the QFD matrix A with the product of BB^T, in the squared QFD form we get

$$QFD(u,v)^2 = (u-v)BB^T(u-v)^T$$

Then, we subsequently apply the matrix associativity rule $(CD)E = C(DE)$, the matrix transposition rule $D^TC^T = (CD)^T$ and, lastly, the matrix distributivity rule $(C-D)E = CE - DE$, leading to

$$\begin{aligned}(u-v)BB^T(u-v)^T &= [(u-v)B][B^T(u-v)^T] \\ &= [(u-v)B][(u-v)B]^T \\ &= (uB-vB)(uB-vB)^T\end{aligned}$$

Precisely analyzing the product of previous steps, we find out that the result is the squared Euclidean distance on modified n-dimensional vectors $u' = uB$ and $v' = vB$:

$$L_2(u',v')^2 = (u'-v')(u'-v')^T \quad (2)$$

We obtained an $n \times n$ matrix B which transforms the QFD space into an equivalent Euclidean space – any vectors u, v into vectors u', v' in the Euclidean space.

Although we start with squared QFD form (1) and obtain the squared Euclidean form (2), the results hold also for the typical (non-squared) forms as defined in Section 1.1. The reason is that both spaces are metric spaces in which any distance is non-negative.

4. APPLICATION OF QMAP IN MAMS

In this section we show the comparison of the QFD model and the QMap model in the process of indexing and querying using several MAMs. In particular, we consider three representatives of different MAM concepts, the simple sequential file (naïve referential method), the Pivot tables (flat index – distance matrix), and the M-tree (hierarchical index). We focus mostly on the time complexity, considering the database size as m and the vector dimensionality as n.

When transforming to the L_2 space using the QMap model, we need the result of the Cholesky decomposition of the QFD matrix A (see Section 3.2.2). The resulting $n \times n$ matrix B enables us to transform vectors from the QFD space into the desired Euclidean space. As the decomposition is computed for static A only once (it could be computed at the time of designing the similarity), we suppose that the matrix B is available when we begin with indexing or querying.

The transformation of a vector using the matrix B takes $O(n^2)$ time (matrix-to-vector multiplication). We also remark that a single distance computation takes $O(n^2)$ time in a case of the QFD and $O(n)$ time in a case of the L_2 distance. Importantly, as the QMap model preserves the QFD distances exactly, the number of distance computations spent on indexing/querying in both models is the same, whatever MAM is used. Also note that we analyze just the time spent on the distance computations and on the transformations to the Euclidean space. To be completely correct, we should also include the overhead time, e.g., traversing the index, I/O cost, etc. However, the overhead time is the same for both models, because the MAMs index the data based on exactly the same distances.

4.1 Sequential file

The sequential file is a flat binary file that is built from a series of dynamic insertions by just appending the inserted objects at the end of the file. Any query involves a sequential scan over all the objects in the binary file. For a query object q and every data object o_i, a distance $\delta(q, o_i)$ must be computed (regardless of the query selectivity). Although this kind of "MAM" is not very smart, it is a baseline structure that also can take advantage of the QMap model.

4.1.1 Indexing

The time analysis of QFD indexing is straightforward because we need to simply insert all n-dimensional vectors. Thus, indexing in the QFD model (just storing m n-dimensional vectors) gives us the time complexity of $O(mn)$. On the other hand, in the QMap model, we have to additionally transform each source vector into the Euclidean space, which gives us the result time $O(mn^2)$.

4.1.2 Querying

Having a similarity query in the QFD model, we can directly compute the distances for all m vectors with the QFD distance. Because the QFD is computed in $O(n^2)$ time, we get the total time complexity of $O(mn^2)$. Compare this result to the QMap model with $O(mn)$ searching time, as we only need $O(n)$ time to compute the L_2 distance instead of the QFD.

4.2 Pivot tables

A simple but efficient solution to the similarity search in metric spaces represent methods called *Pivot tables*, like LAESA [23]. In general, a set of p objects (so-called pivots) is selected from the database, while for every database object a p-dimensional vector of distances to the pivots is created. The distance vectors belonging to the database objects then form a distance matrix – the pivot table. When performing a range query (q, rad), a distance vector for the query object q is determined the same way as for a database object. From the distance vector of the query and the query radius rad a p-dimensional hyper-cube is created, centered in the query and with edges of length $2rad$. Then, the range query is processed on the pivot table, such that the p-dimensional vectors of the database objects that do not fall into the query cube are filtered out from the further processing. The database objects that cannot be filtered have to be subsequently checked by the usual sequential search.

4.2.1 Indexing

First, based on a pivot selection technique [10], we need to select the p pivots from a database sample of a size s, which takes c distance computations (usually $c >> m > s$). Hence, in the QFD model the pivot selection requires $O(cn^2)$ time, while in the QMap model it takes $O(sn^2 + cn)$ time.

Second, we need to insert the m data vectors into the index, including the computation of the distances in the pivot table. In the QFD model, we take the m data vectors and compute the QFD to all p pivots which takes $O(mpn^2)$ time. In the QMap model, each data vector is transformed into the L_2 space ($O(mn^2)$ time) and then the p distances to the pivots are computed in the L_2 space and stored in the pivot table ($O(mpn)$ time), thus leading to $O(mn^2 + mpn)$ time[3].

Hence, the total indexing time is $O(cn^2+mpn^2) = O(cn^2+mn(pn))$ in the QFD model and $O(sn^2 + cn + mn^2 + mpn)$ in the QMap model. Because m is an upper bound to s, we can simplify the complexity of indexing in QMap model to $O(cn + mn(p + n))$. Thus, we proved that indexing in the QMap model is cheaper than indexing in the QFD model.

4.2.2 Querying

Given a query in the QFD model, we need to compute QFD from query to the p pivots in $O(pn^2)$ time as the first step. Then, we filter distance vectors using pivots and for the remaining x non-filtered vectors, we evaluate the QFD. This leads to $O(pn^2 + mp + xn^2) = O(n(pn) + mp + xn^2)$.

In the QMap model, we need to transform the query vector into L_2 space and compute the L_2 distances to the pivots in $O(n^2 + pn)$ time. Then, the pivot table is searched in $O(mp)$ time, while the remaining x non-filtered vectors are checked in $O(xn)$ time. In total, we obtain the time complexity $O(n^2+pn+mp+xn) = O(n(p+n)+mp+xn)$. Hence, we proved that querying in the QMap model is cheaper.

4.3 M-tree

The *M-tree* [13] is a dynamic index structure that provides a good performance in the secondary memory (i.e., in database environments). The M-tree is a hierarchical index, where some of the data objects are selected as centers (local pivots) of ball-shaped regions, while the remaining objects are partitioned among the regions in order to build up a balanced and compact hierarchy of data regions.

In its original version, the M-tree is built by dynamic insertions in the same way as B-tree. First, a suitable leaf for the newly inserted object must be found, which takes $log(m)$ time. Next, the insertion into the leaf could cause an overfill, which results in splitting along the path to the root. Simply said, the time complexity of the dynamic insertion in the M-tree is analogous to the insertion complexity in B-trees, hence leading to $O(mlog(m))$ distance computations.

The similarity queries are implemented by traversing the tree, starting at the root. In general, those M-tree nodes are accessed, whose regions are overlapped by the query.

4.3.1 Indexing

As mentioned earlier, the construction of M-tree takes $O(mlog(m))$ considering the number of distance computations. In the QFD model, this means the time complexity $O(mn^2log(m))$. In the QMap model, we need to transform each data vector to the Euclidean space, leading to total time complexity $O(mn^2 + mnlog(m))$. Again, indexing using the QMap model is cheaper.

4.3.2 Querying

Let x be the number of distance computations needed while evaluating a similarity query. In the QFD model, the query processing takes $O(xn^2)$ time. In the QMap model, the query vector needs to be additionally transformed into the Euclidean space, leading to time complexity $O(n^2+xn)$.

4.4 Summary

To emphasize and show the effect of replacing the QFD model with the QMap model when indexing, we present the Table 1. As might be seen, the only MAM where the QMap model is worse than the QFD model is the sequential file, because all vectors must be transformed to the L_2 space. We notice this drawback but we do not see it as a big disadvantage. This is also the only situation where the QMap model is less efficient. For all other cases, the QMap model outperforms the QFD model by far.

Table 1: Indexing Time Complexity Comparison

Method (model)	Indexing	Better
seq. file (QFD)	$O(mn)$	QFD
seq. file (QMap)	$O(mn^2)$	
Pivot tables (QFD)	$O(cn^2 + mn(pn))$	QMap
Pivot tables (QMap)	$O(cn + mn(p + n))$	
M-tree (QFD)	$O(mn^2log(m))$	QMap
M-tree (QMap)	$O(mn^2 + mnlog(m)))$	

The Table 2 accents the improved time complexity of the proposed QMap model for the query evaluations. The QMap is more efficient in all cases, mainly because it leverages cheaper distance computations compared to QFD model.

Table 2: Querying Time Complexity Comparison

Method (model)	Querying	Better
seq. file (QFD)	$O(mn^2)$	QMap
seq. file (QMap)	$O(mn)$	
Pivot tables (QFD)	$O(n(pn) + mp + xn^2)$	QMap
Pivot tables (QMap)	$O(n(p + n) + mp + xn)$	
M-tree (QFD)	$O(xn^2)$	QMap
M-tree (QMap)	$O(n^2 + xn)$	

5. EXPERIMENTAL EVALUATION

In addition to the theoretical analysis of the computational complexity of indexing/querying in both models, in this section we present also an experimental evaluation on a real-world database. In the experiments, we have used the same MAMs as those analyzed in Section 4, that is, the sequential file, the Pivot tables, and the M-tree.

5.1 The Testbed

We have used a database of 1,000,000 images downloaded from Flickr.com. As the image representation we used the standard RGB histogram of the dimensionality 512, where

[3]We do not need to transform the p pivots into the L_2 space, as they were already transformed during the pivot selection.

the R,G,B components were divided in 8 bins each, thus 8*8*8 = 512 bins. Each histogram was normalized to have the sum equal to 1, while the value of each bin was stored in a float. Then, for each bin we computed as "color prototype" the color in the center of the bin $((R_{min}+R_{max})/2, (G_{min}+G_{max})/2, (B_{min}+B_{max})/2)$, and this color was transformed to CIE Lab color space [25]. The QFD matrix A was computed as described in Section 1.2, where the d_{ij} was the Euclidean distance between the "color prototypes" (after transforming to CIE Lab) of bins i and j.

We have indexed and queried the database using all the MAMs in both models (QFD model and QMap model), measuring the real-time in seconds (note that some graphs use log-scales). The query times were averaged for 500 different queries, while the query histograms were not indexed.

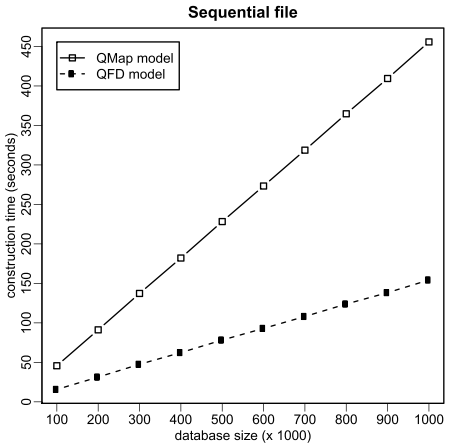

Figure 2: Indexing: sequential file.

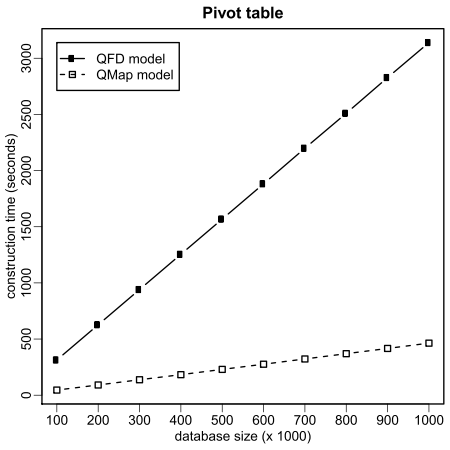

Figure 3: Indexing: pivot table.

5.2 Indexing

In Figures 2, 3, 4 see the real times of indexing for different MAMs, where we observe different sizes of the indexed database. The results reproduce the theoretical costs analyzed in the previous section. In particular, the sequential file performs faster in the QFD model where just storing of the vectors into the binary file takes place. The Pivot tables and M-tree in the QMap model beat their QFD variants by an order of magnitude (the M-tree is even 36x faster).

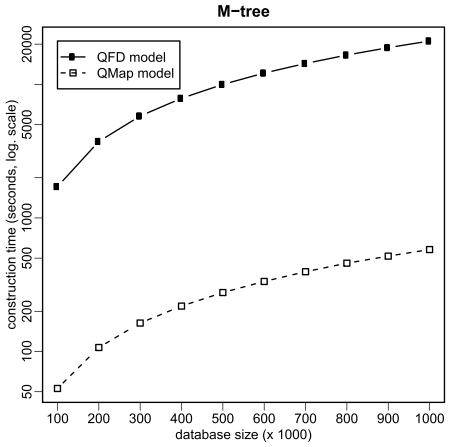

Figure 4: Indexing: M-tree.

5.3 Querying

After indexing, we searched the databases by k nearest neighbors queries (kNN). In Figures 5, 6, 7, see the results for 1NN queries and growing volumes of the indexed database. The sequential file in the QMap model was up to 227x faster than in the QFD model. Also the M-tree in the QMap model exhibits a 200x speedup.

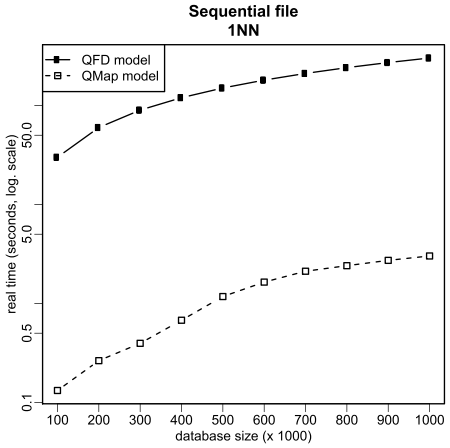

Figure 5: 1NN on growing databases – seq. file.

However, the Pivot tables in QMap model were just 24x faster than in the QFD model. The reason of relatively smaller speedup in Pivot tables is in the smaller proportion of the real-time spent within the distance computations, when compared to other operations needed to answer the query. In particular, in the Pivot tables the query algorithm traverses a large distance matrix, while the number of non-filtered objects x that require the QFD computation is relatively small (see Section 4.2.2). Note also that the speedup was decreasing with the growing database (e.g., from 227x to 100x speedup in a case of the sequential file); this relative

slowdown was caused by a fixed-size disk cache used in the experiments, hence the query processing on larger databases needed more frequent access to the disk.

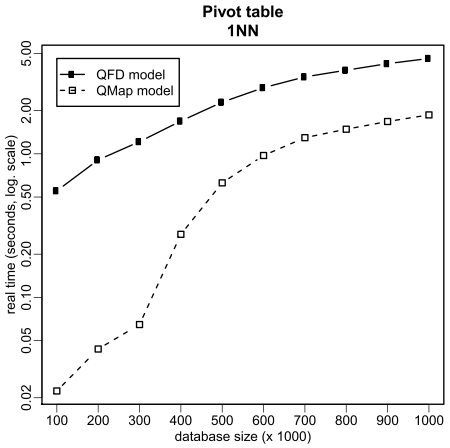

Figure 6: 1NN on growing databases – Pivot tables.

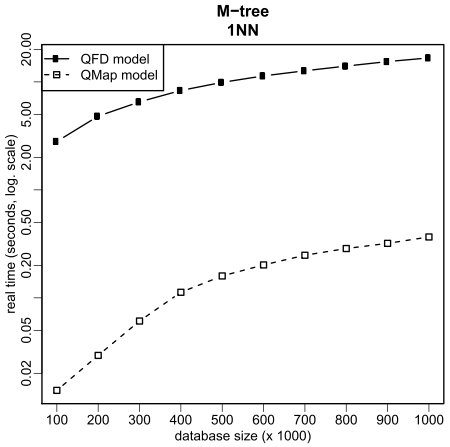

Figure 7: 1NN on growing databases – M-tree.

In Figures 8 and 9, see the real-time for kNN queries on the largest database (one million histograms) for Pivot tables and the M-tree. The M-tree was up to 47x faster in the QMap model when compared to the QFD model. The speedup of Pivot tables was 15x in the QMap model. Concerning Pivot tables, the speedup was, again, relatively smaller than for the M-tree, because of relatively small number of non-filtered objects x (see Section 4.2.2). Also, the M-tree is less demanding in the disk accesses than Pivot tables, so for M-tree the overhead cost was relatively smaller.

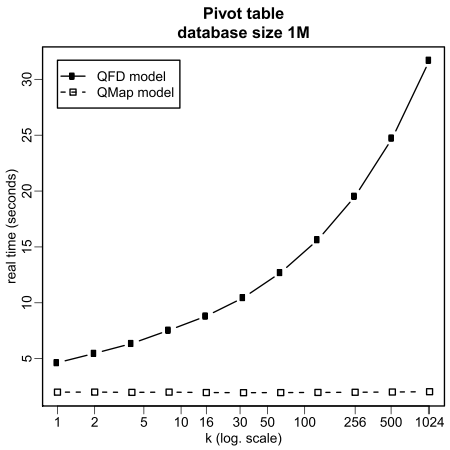

Figure 8: kNN on 1M database – Pivot tables.

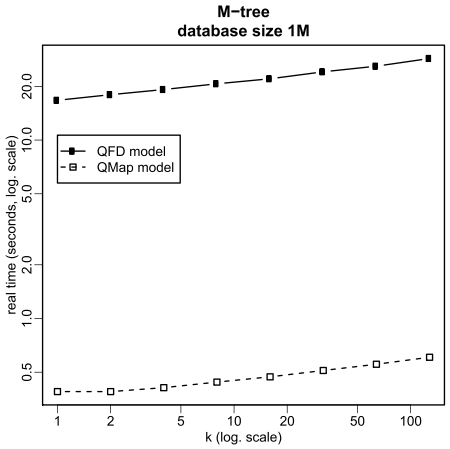

Figure 9: kNN on 1M database – M-tree.

6. CONCLUSIONS

We have introduced the QMap model, a transformational approach that maps the quadratic form distance (QFD) space into the Euclidean space, while exactly preserving the distances. Unlike different approaches based on expensive dimensionality reduction and/or lower-bounding of the QFD, the transformation of the QMap model is based just on a relatively cheap decomposition of the static QFD matrix, hence, being not dependent on the database objects. In consequence, the QMap model allows similarity searching in dynamically changing databases without any distortion of the transformation. We have also shown that the QMap model can be easily combined with the existing metric access methods, achieving superior performance (even in terms of the time complexity) when compared with the straightforward indexing of the QFD. Besides a theoretical analysis of the time complexity of operations on MAMs used within the QMap model, in the experimental evaluation we have shown that the speedup in both indexing and querying could reach several orders of magnitude.

6.1 Future work

In the future work we would like to experimentally compare the QMap model with the QBIC approach [18]. However, as the QBIC implementation details were not properly discussed, the comparison would be based on our own reconstruction of QBIC. Furthermore, we plan to perform more experiments with additional datasets.

Acknowledgments

This research has been supported in part by Czech Science Foundation projects GAČR 201/09/0683, 202/11/0968, and by the grant SVV-2011-263312.

7. REFERENCES

[1] C. C. Aggarwal, A. Hinneburg, and D. A. Keim. On the surprising behavior of distance metrics in high dimensional spaces. In *ICDT*. LNCS, Springer, 2001.

[2] M. Ankerst, B. Braunmüller, H.-P. Kriegel, and T. Seidl. Improving adaptable similarity query processing by using approximations. In *Proc. 24th int. conf. on Very large data bases (VLDB)*, pages 206–217. Morgan Kaufmann, 1998.

[3] M. Ankerst, G. Kastenmüller, H.-P. Kriegel, and T. Seidl. 3d shape histograms for similarity search and classification in spatial databases. In *Advances in Spatial Databases*, volume 1651 of *Lecture Notes in Computer Science*, pages 207–226. Springer Berlin / Heidelberg, 1999.

[4] M. Ankerst, G. Kastenmüller, H.-P. Kriegel, and T. Seidl. Nearest neighbor classification in 3d protein databases. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 34–43. AAAI Press, 1999.

[5] C. Beecks, M. S. Uysal, and T. Seidl. Signature quadratic form distances for content-based similarity. In *Proceedings of the seventeen ACM international conference on Multimedia*, MM '09, pages 697–700, New York, NY, USA, 2009. ACM.

[6] T. Bernas, E. Asem, J. Robinson, and B. Rajwa. Quadratic form: a robust metric for quantitative comparison of flow cytometric histograms. *Cytometry, Part A.*, 73(8):715–726, 2008.

[7] K. S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In *ICDT '99: Proceedings of the 7th International Conference on Database Theory*, pages 217–235, London, UK, 1999. Springer-Verlag.

[8] H. M. Blanken, A. P. de Vries, H. E. Blok, and L. Feng. *Multimedia Retrieval*. Springer, 2007.

[9] C. Böhm, S. Berchtold, and D. Keim. Searching in High-Dimensional Spaces – Index Structures for Improving the Performance of Multimedia Databases. *ACM Computing Surveys*, 33(3):322–373, 2001.

[10] B. Bustos, G. Navarro, and E. Chávez. Pivot selection techniques for proximity searching in metric spaces. *Pattern Recognition Letters*, 24(14):2357–2366, 2003.

[11] V. Castelli and L. D. Bergman, editors. *Image Databases : Search and Retrieval of Digital Imagery*. Wiley-Inter., 2002.

[12] E. Chávez, G. Navarro, R. Baeza-Yates, and J. L. Marroquín. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, 2001.

[13] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In *VLDB'97*, pages 426–435, 1997.

[14] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz. Efficient and Effective Querying by Image Content. *J. Intell. Inf. Syst.*, 3:231–262, July 1994.

[15] T. Fober and E. Hüllermeier. Similarity measures for protein structures based on fuzzy histogram comparison. In *IEEE World Congress on Computational Intelligence*. IEEE, 2010.

[16] T. Fober, M. Mernberger, G. Klebe, and E. Hüllermeier. Efficient similarity retrieval for protein binding sites based on histogram comparison. In *German Conference on Bioinformatics*, 2010.

[17] G. H. Golub and C. F. Van Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[18] J. Hafner, H. S. Sawhney, W. Equitz, M. Flickner, and W. Niblack. Efficient color histogram indexing for quadratic form distance functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:729–736, 1995.

[19] M. Ioka. A method of defining the similarity of images on the basis of color information. Technical Report Tech. Report RT-0030, IBM Tokyo Research Lab, 1989.

[20] Y. Ishikawa, R. Subramanya, and C. Faloutsos. Mindreader: Querying databases through multiple examples. In *Proceedings of the 24th International Conference on Very Large Data Bases*, VLDB '98, pages 218–227, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[21] M. Luo, X. Bai, and G. Xu. Svd-based hierarchical algorithm for similarity indexing in quadratic form distance space. In *Fifth Asian Conference on Computer Vision*, 2002.

[22] C. D. Meyer, editor. *Matrix analysis and applied linear algebra*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[23] M. L. Mico, J. Oncina, and E. Vidal. A new version of the nearest-neighbour approximating and eliminating search algorithm (aesa) with linear preprocessing time and memory requirements. *Pattern Recogn. Lett.*, 15(1):9–17, 1994.

[24] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. H. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin. The QBIC Project: Querying Images by Content, Using Color, Texture, and Shape. In *Storage and Retrieval for Image and Video Databases (SPIE)'93*, pages 173–187, 1993.

[25] Y. Rubner, J. Puzicha, C. Tomasi, and J. M. Buhmann. Empirical evaluation of dissimilarity measures for color and texture. *Comput. Vis. Image Underst.*, 84(1):25–43, 2001.

[26] T. Seidl and H.-P. Kriegel. Efficient user-adaptable similarity search in large multimedia databases. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, VLDB '97, pages 506–515, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[27] T. Skopal. Unified framework for fast exact and approximate search in dissimilarity spaces. *ACM Trans. Database Syst.*, 32, November 2007.

[28] P. Zezula, G. Amato, V. Dohnal, and M. Batko. *Similarity Search - The Metric Space Approach*, volume 32. Springer, 2006.