Hierarchical linear subspace indexing method

Andreas Wichert
Mário J. Silva

DI–FCUL TR–06-3

March 2006

Departamento de Informática
Faculdade de Ciências da Universidade de Lisboa
Campo Grande, 1749–016 Lisboa
Portugal

Technical reports are available at `http://www.di.fc.ul.pt/tech-reports`. The files are stored in PDF, with the report number as filename. Alternatively, reports are available by post from the above address.

Hierarchical linear subspace indexing method

Andreas Wichert
Department of Computer Science
University of Lisbon, Portugal
wichert@xldb.di.fc.ul.pt

Mário J. Silva
Department of Computer Science
University of Lisbon, Portugal
mjs@di.fc.ul.pt

March 2006

Abstract

Traditional multimedia indexing methods are based on the principle of hierarchical clustering of the data space where metric properties are used to build a tree that can then be used to prune branches while processing the queries. However, the performance of these methods will deteriorate rapidly when the dimensionality of the data space is increased.

We describe a new hierarchical linear subspace indexing method will based on the generic multimedia indexing (GEMINI) approach, which does not suffer from the dimensionality problem. The hierarchical subspace approach offers a fast searching method for large content-based multimedia databases.

The approach will be demonstrated on image indexing, in which the subspaces correspond to different resolutions of the images. During content-based image retrieval the search starts in the subspace with the lowest resolution of the images. In this subspace the set off all possible similar images is determined. In the next subspace additional metric information corresponding to a higher resolution is used to reduce this set. This procedure is repeated until the similar images can be determined eliminating the false candidates.

1 Introduction

In this report we describe fast content-based methods for searching large image database. In content-based image retrieval technique, a query is posed in the form of an example image that is matched against the stored images. Most current Web search engines support image searches, but these searches only consider the text surrounding images [3]. In traditional images databases matching is based on textual descriptions and tags that annotate the images.

Content-based image retrieval methods use features, which describe important properties of the images. The most used features are: color, texture, shape, position and spatial coordinates and layout [6],[9],[8]. These features are mapped into points in a high-dimensional feature space, and the search is based on points that are close to a given query point in this space.

In our approach, we combine the color information and its spatial distribution by simple image matching. We scale the digital images to a fixed size and map them into a 3-band RGB (Red, Green, Blue) representation where each color is represented by 8 bits. With this transformation we are able to represent the images as vectors and to compute the Euclidian distance between them. Two images $\vec{x}$ and $\vec{y}$ are similar if their distance is smaller or equal to ϵ, $d(\vec{x}, \vec{y}) \leq \epsilon$. The result of a query computed by this method is a set of images that have similar spatial color characteristics as the query image.

However, the size of the result set is previously unknown; it depends on ϵ and may reach the size of the entire database or no image at all. Worse yet, most multimedia indexing methods become worse with huge dimensionality eventually reducing to sequential scanning, which is not acceptable for large databases.

Traditional indexing methods are based on the principle of hierarchical clustering of the data space, in which metric properties are used to build a tree that then can be used to prune branches while processing the queries. They operate efficiently when the number of the dimensions is small. Tree-based methods become worse with huge dimensionality, eventually reducing the computing costs to sequential scanning [1]. These negative effects are also named as the "curse of dimensionality." Most problems arise from the fact that the volume of a sphere with constant radius size grows exponentially with increasing dimension. A nearest-neighbor query in a high dimensional space corresponds to a hyper-sphere with a huge radius which is mostly larger than the extension of the data space in most dimensions [1].

Based on the generic multimedia indexing approach and lower bounding methods [5], [4], we developed a hierarchical subspace indexing method which does not suffer from the dimensionality problem. In our image indexing method, the subspaces correspond to different resolutions of the images. The search starts in the subspace with the lowest resolution of the images in which the set off all possible similar images is determined. In this subspace the set off all possible similar images is determined. In the next subspace additional information corresponding to a higher resolution is used to reduce this set. This procedure is repeated until the similar images can be determined eliminating the false candidates.

2 Spatial access methods

2.1 Tree-based spatial access methods

The basic idea of metric indexes is to derive metrics from item properties to build a tree that then can be used to prune branches in processing queries. Those trees resemble in their design decision trees and B-trees [2]. B-trees may have a variable number of keys and children. In B-trees the nodes hold ordered ranges of numbers representing sets of keys. A key in a non leaf node has a left and a right child pointer (branch). The left pointer is the root of a sub tree which contains nodes with keys less than or equal to it. It is the right child pointer of the proceeding key. The right pointer is the a sub tree containing all keys greater than any keys in that node, and it is the left child pointer of the following key (see Figure 1).

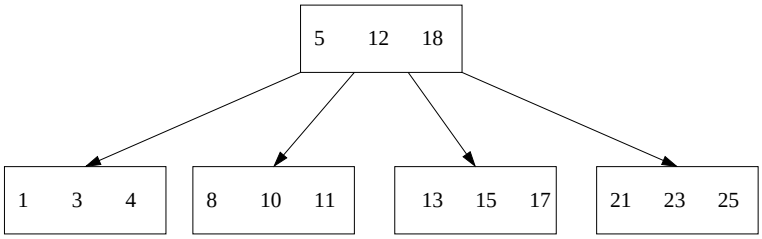

Figure 1: In B-trees, the keys are ranges of numbers, they are stored in non-decreasing order in a node. The left pointer of a key is the root of a sub tree which contains nodes with keys less than or equal to it, the right pointer is the a sub tree containing all keys greater than any keys in that node.

The root node of the tree serves as an entry point for query processing. The information is stored in the leaves and in the keys. The leaves are called data pages

and the nodes of the tree are called directory pages. During the search operation to a given point the correct child is chosen by a linear search of the key values in the node. A key value greater than or equal to the desired value is searched. If present the child pointer to the immediate left of that value is followed, otherwise the rightmost child pointer is followed.

In metric indexes the keys in the trees represent regions which are subsets in the data space [1]. The d dimensional data space is recursively split by $d-1$ dimensional hyper-planes until the number of data items in a partition is below a certain threshold. Data is represented by vectors that are stored in data pages such that spatially adjacent vectors are stored in the same page. Each data vector is stored in exactly one data page and there is no object duplication within data pages. The directory nodes are organized hierarchically, each directory node points to a set of sub trees. Assigned to each directory node is a key represented by the page region, which is a subset of the data space. Each key has an associated child, which is the root of a sub tree containing all nodes with regions that are contained in the key region of the proceeding key. However, unlike in the B-trees, where the keys are numbers which define intervals which do not overlap, key regions may overlap because the space has dimension greater one. In these spaces the points are not ordered. Ordering of points requires a one to one mapping into a line (injective mapping). This means that regions of pages in different branches of the tree may overlap, which leads to high computing costs. Because of that, special heuristics are often used to avoid or to minimize the overlapping. Trees mainly differ by the definition of the region and the resulting search and building algorithms. The metric indexes tree can be used to prune branches in processing the queries because of the lower bounding property. The distance of a query point to each key region is greater than the distance to the key regions of its children.

In R-trees, the key regions are minimum bounding rectangles (MBR) [1]. MBR is a multidimensional interval of the data space which is a minimal axis-parallel rectangle enclosing the complete point set with at least one data point. In the tree hierarchy the key MBR of a parent node contain the MBR keys of its children, that are allowed to overlap (see Figure 2). In a range query, search covers all points in the space whose Euclidian distance to the query point is smaller or equal to ϵ. A MBR which includes the sphere with the radius ϵ around the query point is determined. Then the R-tree is descended recursively excluding all the branches whose MBR do not intersect with the query MBR. Because the regions may overlap at each level, the descend may include several branches.

In nearest neighbor search algorithms the upper bound on every surface is determined and among those the minimum is taken, because only the nearer surface can contain the minimum. The region description of an MBR comprises a lower and an upper bound. Among each pair of opposite edges of an MBR, only the edges closer to the query point are considered for each dimension. Given the MBR definition, the lower bound is defined by the nearest corner that contain those edges and the upper bound by the farthest corner which contain one of this edges.

In SS-Tree, the page regions are spheres [1]. The average value of all points (centroid) is used as the center for the sphere and the minimum radius is chosen such that all objects are included in the sphere. Spheres do not allow an easy overlap-free split. In the tree hierarchy the key spheres of a parent node contain the keys spheres of its children, which are allowed to overlap. In a range query the search is performed on all points in the space whose Euclidian distance to the query point is smaller or equal to ϵ.

The description of regions by spheres allows a very fast determination of a lower and an upper bound, which is the distance of the query point to the centroid minus the radius for the lower bound, and plus the radius for the upper bound value. According to Böhm SS-trees outperform the R-trees [1].

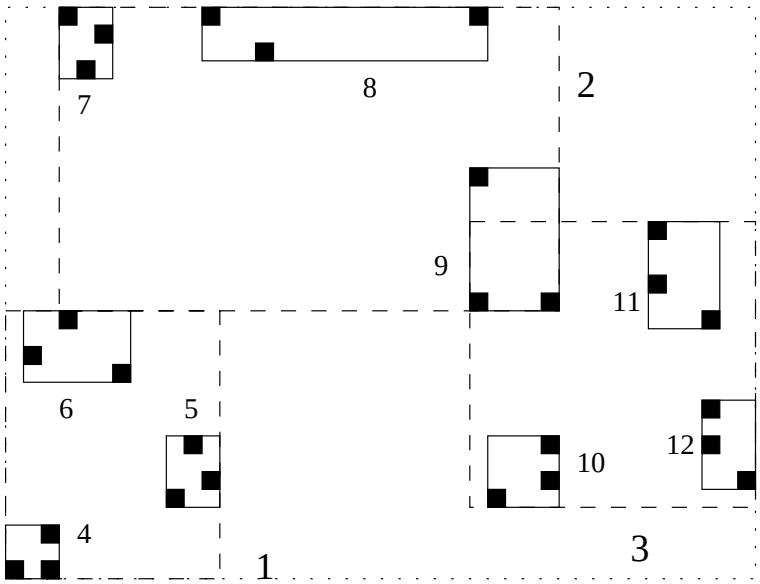

Figure 2: In R-trees, key regions are minimum bounding rectangles. The children of the root (doted big rectangle) are represented by the region keys indicated by the numbers $1, 2$, and 3 (dashed rectangles). The key region 1 has the children indicated by the numbers $4, 5, 6$, which themselves have key regions representing the data (indicated by the black dots) and so on. The fan-out of the tree is three.

There are many more tree structures, like the SR-trees, which can be regarded as the combination of the R-tree and the SS-tree, or R*-trees, X-trees, TV-trees or kd-tress, which use different heuristics to minimize or to avoid the overlap of the key regions.

No extensive and objective comparison between the different tree index structures has been published. For example Faloutsos claims: *" ..that R-trees based methods seem to be most robust for higher dimensions"* [4]. Hoverer empirical comparison depends strongly on the data and all these tree index structures suffer from the problems which result from the "curse of dimensionality" [1] as explained in the next section.

2.1.1 Curse of dimensionality

The metric indexes trees operate efficiently when the number of dimensions is small. The growth of the number of dimensions has negative implications for the performance of multidimensional index trees; these negative effects are also named as the "curse of dimensionality." Most problems arise from the fact that the volume of a sphere or a minimum bounding rectangles with the constant radius or edge size grows exponentially with increasing dimension. For example, the volume of high dimensional cube approaches its surface with the growing dimension [1]. In high-dimensional spaces a partition is performed only in few dimensions touching the boundary of the data space in most dimensions. The probability that the key regions may overlap grows with the grow of the dimensions which means that most regions of pages in different branches of the tree overlap. A nearest-neighbor query in a high dimensional space corresponds to a hyper-sphere with a huge radius which is mostly larger than the extension of the data space in most dimensions [1]. Because of these problems tree indexing methods explode exponentially with the dimensionality eventually reducing the search time to sequential scanning.

A solution to this problem is to map the objects into points in low dimensional space so that tree spatial access methods can be used [4].

3 Subspace sequence method

3.1 Generic multimedia indexing approaches

The idea behind the generic multimedia indexing (GEMINI) [5],[4] approaches is to find a feature extraction function that maps the high dimensional objects into a low dimensional space. In this low dimensional space, a so-called 'quick-and-dirty' test can discard the non-qualifying objects. It is supposed that objects that are very dissimilar in the feature space are also very dissimilar in the original space (see Figure 3).

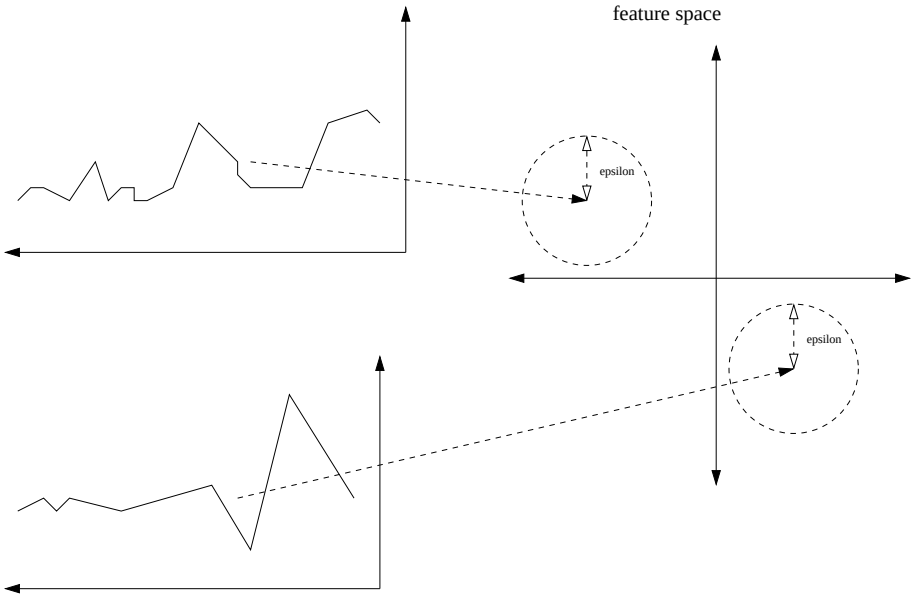

Figure 3: Feature extraction function which maps the high dimensional objects into a low dimensional space. The distance of similar objects should be smaller or equal to ϵ. This tolerance is represented by a sphere with the radius ϵ in the feature space.

Ideally the feature map should preserve the distances exactly, but this is only possible if the dimension of both spaces are equal. However, if the distances in the feature space are always smaller or equal then the distances in the original space, a bound can be determined which is valid in both spaces. The distance of similar objects is smaller or equal to ϵ in original space and consequently it is as well smaller or equal ϵ in the feature space. No object in the feature space will be missed (false dismissals) in the feature space. However, there will be some objects that are not similar in the original space (false hints/alarms). That means that we are guaranteed to have selected all the objects we wanted plus some additional false hits in the feature space. In a second pass on this selected set has to be separate out the false hits by comparison in the original space. The size of the collection in the feature space depends on ϵ and the proportion between both spaces and may reach the size of the entire database, if the feature space is not carefully chosen. The lemma which guarantees that no objects will be missed in the feature space is called the lower bounding lemma and is expressed mathematically as follows; let O_1 and O_2 be two objects; $F()$, the mapping of objects into f dimensional space

should satisfy the following formula for all objects, where d is a distance function in the original space and $d_{feature}$ in the feature subspace:

$$d_{feature}(F(O_1), F(O_2)) \leq d(O_1, O_2). \quad (1)$$

In the first step in the GEMINI approach the distance function has to be defined. The second step consists in finding the feature extraction function $F()$ that satisfies the bounding lemma has to be determined. Such a function has to capture most of the characteristics of the objects into a low dimensional feature space. In most cases the used distance functions in the original space and in the feature space are equal. Given the Parseval's theorem which states that the Discrete Fourier Transform (DFT) preserve Euclidian distances between signals, the DTF which keeps the first coefficients of the transform is an example for a feature function $F()$ [5], [4]. Accordingly one can use any orthonormal transform because they all preserve the distance between the original and the transformed space. One can also use data dependent transforms as feature functions $F()$, such as Karhunen Loeve transform. However, they have to be recalculated as soon as new data arrives [5]. Once the data are mapped into the low dimensional feature space, tree spatial access methods can be used.

During the search a range query is performed. All points whose distance to the query point is smaller or equal to ϵ in the feature space are searched. In the second step false hits are filtered from the set of selected objects by comparison in the original space.

3.2 Lower bounding approach

Based on the analysis of GEMINI approach the subspace method will be developed. Let DB be a database of s multimedia objects $\vec{x}^{(i)}$ represented by vectors of dimension m in which the index i is an explicit key identifying each object,

$$\{\vec{x}^{(i)} \in DB | i \in \{1..s\}\}.$$

The set DB can be ordered relatively to a given multimedia object $\vec{y}$ using a distance function d. This is done by a monotone increasing sequence corresponding to the increasing distance of $\vec{y}$ to $\vec{x}^{(i)}$ with an explicit key which identifies each object indicated by the index i,

$$d[y]_n := \{d(x^{(i)}, y) \;\; | \;\; \forall n \in \{1..s\} : d[y]_n \leq d[y]_{n+1}\}$$

if $\vec{y} \in DB$, then $d[y]_1 := 0$. The set of similar multimedia objects in correspondence to $\vec{y}$, $DB[y]_\epsilon$ is the subset of DB, $DB[y]_\epsilon \subseteq DB$ with the size $\sigma = |DB[y]_\epsilon|$, $\sigma \leq s$:

$$DB[y]_\epsilon := \{x^{(i)} \in DB \;\; | \;\; d[y]_n = d(x^{(i)}, y) \leq \epsilon\}.$$

Now lets map all the multimedia objects of the DB with $F()$, the mapping which satisfies the lower bounding lema into f dimensional space,

$$\{F(\vec{x})^{(i)} \in F(DB) | i \in \{1..s\}\}.$$

The set $F(DB)$ can be ordered in a relation to a given multimedia object $F(\vec{y})$ and a distance function $d_{feature}$.

This is done by a monotone increasing sequence corresponding to the increasing distance of $F(\vec{y})$ to $F(\vec{x}^{(i)})$ with an explicit key which identifies each object indicated by the index i,

$$d[F(y)])_n := \{d_{feature}(F(x^{(i)}), F(y)) \;\; | \;\; \forall n \in \{1..s\} : d[F(y)]_n \leq d[F(y)]_{n+1}\}$$

The set of similar multimedia objects in correspondence to $F(\vec{y})$, $F(DB[y])_\epsilon$ is the subset of $F(DB)$, $F(DB[y])_\epsilon \subseteq F(DB)$ with the size $F(\sigma) = |F(DB[y])_\epsilon|$, $\sigma \leq F(\sigma) \leq s$:

$$F(DB[y])_\epsilon := \{F(x)_n^{(i)} \in F(DB) \mid d[F(y)]_n = d_{feature}(F(x)^{(i)}, F(y)) \leq \epsilon\}.$$

An ϵ exists only if $min^*(d[y]_n) < d[F(y)]_s$ and is chosen from the interval $[min^*(d[y]_n), d[F(y)]_s]$ where

$$min^*(d[y]_n) = \begin{cases} d[y]_1 & \text{if } d[y]_1 \neq 0 \\ d[y]_2 & \text{if } d[y]_1 = 0. \end{cases}$$

To determine $DB[y]_\epsilon$ by linear search, we need $s \cdot m$ computing steps, if we suppose that computation of distance between two m-dimensional vectors requires m computing steps. To determine $DB[y]_\epsilon$ when $F(DB[y])_\epsilon$ is present, we need $F(\sigma) \cdot m$ steps; the false hints are separated from the selected objects by comparison in the original space. If no metric tree is used to index the feature space the savings using result from the size of $F(DB[y])_\epsilon$ in proportion to the dimensions of both spaces $\frac{f}{m}$. The computing time of $DB[y]_\epsilon$ is saved compared to linear matching in the original space if:

$$s \cdot m \geq F(\sigma) \cdot m + s \cdot f \tag{2}$$

$$\left\lfloor s \cdot (1 - \frac{f}{m}) \right\rfloor \geq F(\sigma). \tag{3}$$

In the next section we expand the analysis into the subspace method.

3.3 Linear subspace sequence method

Let V be a m-dimensional vector space and $F()$ a linear mapping that obeys the lower bound lemma from the vector space into a f-dimensional subspace U. Contrary to GEMINI approach where the feature space needs not to be a subspace, we can map the computed metric distance between objects in the f-dimensional subspace U into the m-dimensional space V which contains the subspace U. In this case the lower bounding lemma is extended; let O_1 and O_2 be two objects; $F()$, the mapping of objects into f dimensional subspace U should satisfy the following formula for all objects, where d is a distance function in the space V and d_U in the subspace U:

$$d_U(F(O_1), F(O_2)) \leq d(F(O_1), F(O_2)) \leq d(O_1, O_2). \tag{4}$$

We can define a sequence of subspaces $V = U_0, U_1, U_2, \ldots, U_n$ in which each subspace is a subspace of another space

$$U_0 \supset U_1 \supset U_2 \supset \ldots \supset U_n$$

and

$$dim(U_0) > dim(U_1) > dim(U_2) \ldots > dim(U_n).$$

Let $F_{a,b}()$ be the mapping from subspace U_a to subspace U_b that obeys the lower bounding lemma. An example of a sequence of subspaces is the sequence of real vector subspaces

$$\mathbf{R^m} \supset \mathbf{R^{m-1}} \supset \mathbf{R^{m-2}} \supset \ldots \supset \mathbf{R^1}$$

formed by the mapping from one subspace to another which always set the last coordinate ($\neq 0$) to 0, in this case the used distance functions in the original space and in the subspace are equal.

All s multimedia objects of DB are in space $V = U_0$, which is represented by $V(DB) = U_0(DB)$. The DB mapped by $F_{0,1}()$ from space U_0 to its subspace U_1 is indicated by $U_1(DB)$.

A subspace U_k can be mapped from different spaces by different functions

$$\{U_k : F_{l,k}()|U_l \rightarrow U_k, l < k\}$$

in contrast to the universal GEMINI approach, in which the mapped DB is only depended on the function $F()$. We use a notation which depends on the subspace U_k and on the mapped function:

$$\{U_k(\vec{x})^{(i)} \in U_k(DB)|i \in \{1..s\}\}$$

$$d[U_k(y)]_n := \{d(U_k(x^{(i)}), U_k(y)) \;\mid\; \forall n \in \{1..s\} : d[U_k(y)]_n \leq d[U_k(y)]_{n+1}\} \quad (5)$$

$$U_k(DB[y])_\epsilon := \{U_k(x)_n^{(i)} \in U_k(DB) \;\mid\; d[U_k(y)]_n = d(U_k(x)^{(i)}, U_k(y)) \leq \epsilon\}, \quad (6)$$

with the size $U_k(\sigma) = |U_k(DB[y]_\epsilon)|$, $U_0(\sigma) < U_k(\sigma) < s$ and,

$$U_0(\sigma) < U_1(\sigma) < U_2(\sigma) < \ldots < U_{(n)}(\sigma) < s$$

An ϵ exists only if $min^*(d[U_0(y)]_n) < d[U_n(y)]_s$ and is chosen from the interval $[min^*(d[U_0(y)]_n), d[U_n(y)]_s]$ where

$$min^*(d[U_0(y)]_n) = \begin{cases} d[U_0(y)]_1 & \text{if } d[U_0(y)]_1 \neq 0 \\ d[U_0(y)]_2 & \text{if } d[U_0(y)]_1 = 0. \end{cases}$$

The computing time of $U_0(DB[y])_\epsilon$ is saved compared to linear matching in the original space if:

$$s \cdot m \geq U_1(\sigma) \cdot m + s \cdot dim(U_1)$$

$$\left\lfloor s \cdot (1 - \frac{dim(U_1)}{m}) \right\rfloor \geq U_1(\sigma).$$

When we apply the same procedure recursively,

$$s \cdot dim(U_1) \geq U_2(\sigma) \cdot dim(U_1) + s \cdot dim(U_2)$$

$$\left\lfloor s \cdot (1 - \frac{dim(U_2)}{dim(U_1)}) \right\rfloor \geq U_2(\sigma)$$

it follows:

$$s \cdot m \geq U_1(\sigma) \cdot m + s \cdot dim(U_1) \geq U_1(\sigma) \cdot m + U_2(\sigma) \cdot dim(U_1) + s \cdot dim(U_2).$$

Because the computing time of a subspace sequence is saved compared to computation using only the subspace U_2 and the resulting $U_2(\sigma)$, it should also be:

$$U_2(\sigma) \cdot m + s \cdot dim(U_2) \geq U_1(\sigma) \cdot m + U_2(\sigma) \cdot dim(U_1) + s \cdot dim(U_2)$$

$$U_2(\sigma) \cdot m \geq U_1(\sigma) \cdot m + U_2(\sigma) \cdot dim(U_1),$$

which is true when

$$\left\lfloor U_2(\sigma) \cdot (1 - \frac{dim(U_1)}{m}) \right\rfloor \geq U_1(\sigma).$$

Generic for $k \in [1, \ldots, (n-1)]$

$$\left\lfloor U_{(k+1)}(\sigma) \cdot (1 - \frac{dim(U_k)}{dim(U_{(k-1)})}) \right\rfloor \geq U_k(\sigma), \tag{7}$$

and for $k = n$

$$\left\lfloor s \cdot (1 - \frac{dim(U_n)}{dim(U_{(n-1)})}) \right\rfloor \geq U_n(\sigma) \tag{8}$$

and the computing costs are

$$U_1(\sigma) \cdot m + U_2(\sigma) \cdot dim(U_1) + \ldots + s \cdot dim(U_n) =$$

$$U_1(\sigma) \cdot dim(U_0) + U_2(\sigma) \cdot dim(U_1) + \ldots + s \cdot dim(U_n) =$$

$$= \sum_{i=1}^{n} U_i(\sigma) \cdot dim(U_{(i-1)}) + s \cdot dim(U_n). \tag{9}$$

In the next section we introduce a linear mapping $F()$ that meets all required properties.

3.4 Orthogonal projection

Let be $V = \mathbf{R^m}$ a vector space and let be U a f-dimensional subspace obtained by a projection, and an Euclidian distance function $d = l_2$. An orthogonal projection P into U is a mapping $P : \mathbf{R^m} \rightarrow U$, it orders every vector $\vec{x} \in \mathbf{R^m}$ a vector $P(\vec{x})$ with a shortest distance to $\vec{x} \in \mathbf{R^m}$. Let be $(w^{(1)}, w^{(2)}, \ldots, w^{(m)})$ be the orthonormalbasis of $\mathbf{R^m}$, and $(w^{(1)}, w^{(2)}, \ldots, w^{(f)})$ the orthonormalbsis of U. Then, $\vec{x}$ can be represented by an unique decomposition

$$\vec{x} = \sum_{i=1}^{f} < \vec{x}, w^{(i)} > \cdot w^{(i)} + \sum_{i=f+1}^{m} < \vec{x}, w^{(i)} > \cdot w^{(i)}$$

and the orthogonal projection of $\vec{x}$ onto U can be written by

$$P(\vec{x}) = \sum_{i=1}^{f} < \vec{x}, w^{(i)} > \cdot w^{(i)}$$

$$O(\vec{x})^{\perp} = \sum_{i=f+1}^{m} < \vec{x}, w^{(i)} > \cdot w^{(i)}$$

An orthogonal basis can be decomposed for example by the classical method named 'Gram-Schmidt orthogonalization' process. According to the Pythagorean theorem, $||\vec{x}||^2 = ||P(\vec{x})||^2 + ||O(\vec{x})^{\perp}||^2$; consequently, $||\vec{x}|| \geq ||P(\vec{x})||$, from which

the lower bound lemma follows. A projection is always a linear transformation and can be represented by a projection matrix $\wp$ with the dimension $m \times m$ with $\wp = \wp^2$. Any vector $\vec{u} \in U$ is fixed by the projection matrix $\vec{u} = \wp \cdot \vec{u}$, it is the eigenvector of $\wp$ with eigenvalue 1.

Furthermore, we can map the computed metric distance d_U between objects in the f-dimensional orthogonal subspace U into the m-dimensional space V which contains the orthogonal subspace U by just multiplying the distance d_u by a constant $c = \sqrt{\frac{m}{f}}$.

For example the orthogonal projection of points $\vec{x} = (x_1, x_2) \in \mathbf{R^2}$ on the bisecting line $U = \{(x_1, x_2) \in \mathbf{R^2} | x_1 = x_2\} = \{(x_1, x_1) = \mathbf{R^1}\}$ corresponds to the mean value of the projected points. It can be represented by the projection matrix with $rank(\wp) = 1$

$$\wp = \begin{pmatrix} \frac{1}{m/f} & \frac{1}{m/f} \\ \\ \frac{1}{m/f} & \frac{1}{m/f} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix},$$

and the orthonormalbsis of U is $x^{(1)} = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$. The points $\vec{a} = (2, 4)$ is mapped into $P(\vec{a}) = 3$, and $\vec{b} = (7, 5)$ into $P(\vec{b}) = 6$. The distance in U is $d_u(P(\vec{a}), P(\vec{b})) = \sqrt{|6-3|^2}$, $c = \sqrt{2}$, so the distance in $\mathbf{R^2}$ is $d(P(\vec{a}), P(\vec{b})) = 3 \cdot \sqrt{2} \leq d(\vec{a}, \vec{b}) = \sqrt{26}$ (see Figure 4).

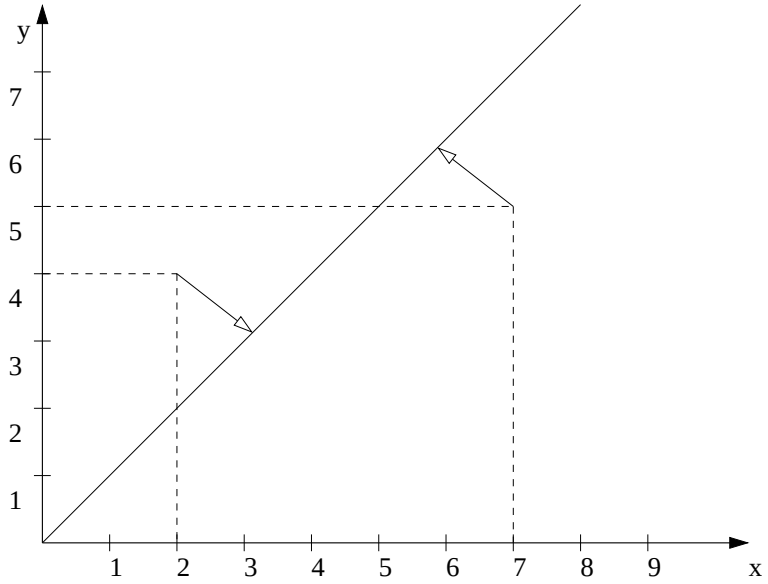

Figure 4: For example the orthogonal projection of points $\vec{x} = (x_1, x_2) \in \mathbf{R^2}$ on the bisecting line $U = \{(x_1, x_2) \in \mathbf{R^2} | x_1 = x_2\} = \{(x_1, x_1) = \mathbf{R^1}\}$ corresponds to the mean value of the projected points. $\vec{a} = (2, 4)$ is mapped into $P(\vec{a}) = 3$, and $\vec{b} = (7, 5)$ into $P(\vec{b}) = 6$.

Orthogonal projection by a linear mapping $F_{a,b}()$ is used to map subspace U_a into to subspace U_b. Because it obeys the lower bounding lemma, a sequence of subspaces can be applied for content-based image retrieval.

3.5 Metric information

Ideally the mapping function should reduce the computing costs as much as possible and preserve the distances as much as possible. For an orthogonal projection

we suppose equality in the Equation 4 that is dependent on multiplication with constants c_k and ι_k:

$$d_{U_k}(U_k(O_1), U_k(O_2)) \cdot c_k^0 = d_{U_0}(U_k(O_1), U_k(O_2)) = d(U_k(O_1), U_k(O_2)) \quad (10)$$

$$d_{U_0}(U_k(O_1), U_k(O_2)) \cdot \iota_k^0 \approx d_{U_0}(U_0(O_1), U_0(O_2)) = d(O_1, O_2). \quad (11)$$

A stated before, for an orthogonal projection the constant c_k^0 can be easily determined, it is $c_k^0 = \sqrt{\frac{dim(U_0)}{dim(U_k)}}$. The bigger c_k^0 is, the more computation time is saved. On the other side, ι_k^0 corresponds to the metric information gain of the transition from $U_k(DB)$ to $U_0(DB)$. This transition depends on the multimedia objects in the DB and on the distance between them. It can be described by the ratio of the mean distance between the objects in $U_0(DB)$ that is $mu(U_0(DB))$, and the mean distance between the objects in $U_k(DB)$ that is $mu(U_k(DB))$:

$$\iota_k^0(DB) := \frac{mu(U_0(DB))}{mu(U_k(DB)) \cdot c_k^0}. \quad (12)$$

Because a fully connected graph with s nodes has $\frac{(s-1)\cdot s}{2}$ edges, the mean distance $mu(U_k(DB))$ between all objects of $mu(U_k(DB))$ is

$$mu(U_k(DB)) := \{\sum \frac{2 \cdot d(U_k(x^{(i)}), U_k(x^{(j)}))}{(s-1) \cdot s} | \forall i \neq j \in \{1..s\}\} \quad (13)$$

The distances are preserved better by a mapping, the closer ι_k^0 is to 1. Better preserved distances lead to reduced computing costs. Exactly preserved distances would correspond to $\iota_k^0 = 1$. The quality of the mapping function is expressed by the combination of c_k^0 with ι_k^0,

$$q_k^0(DB) = \frac{1}{\iota_k^0(DB) - \frac{1}{c_k^0}} \quad (14)$$

$q_k^0(DB)$ takes the values from the interval between 0 and 1, the closer q_k^0 to 1, the better the quality of the mapping in the relation to the computing costs.

By the metric information loss of the transition from $U_0(DB)$ to $U_k(DB)$ errors emerge as indicated by the resulting sequence in $U_k(DB)$ in relation to $U_0(DB)$

$$d[U_k^0(y)]_n := \{d(U_k(x^{(i)}), U_k(y)) \mid \forall n \in \{1..s\} : d[U_0(y)]_n \leq d[U_0(y)]_{n+1}\}. \quad (15)$$

The resulting error in U_k is indicted by the distance of the ordered sub-sequences in U_k from the sequence in U_0. The distance is measured from the beginning of the ordered sub-sequence g, it is indicated by α_g with $g \in \{1..s\}$. The ordered subsequence g in U_k is defined as

$$s[U_k^0(y)]_{N_g} := \{d(U_k(x^{(i_{s(0)})}), U_k(y)) \mid \forall N \in \{\alpha_g..\gamma_g\} \subseteq \{1..s\}$$
$$: d[U_k(y)]_N \leq d[U_k(y)]_{N+1} \quad \wedge \quad d[U_0(y)]_N \leq d[U_0(y)]_{N+1}\} \quad (16)$$

and the error as,

$$Error_k^0(m) := \{\sum_{n=1}^{m} |i - \alpha_g| \mid m \leq s\}. \quad (17)$$

Figure 5: For two color images representing land and sea the images with the most similar color characteristic which corresponds to the most atmospheric similar images are determined.

4 Content based image retrieval experiments

4.1 Image Database

The linear subspace sequence method and orthogonal projection will be demonstrated on a practical example of fast searching methods for content-based image retrieval. The given task is to determine $DB[y]_\epsilon$ with either $y \in DB$ or $y \notin DB$. For example for two color images $land_a, sea_a \notin DB$; $d[land_a]_1, d[sea_a]_1$ are determined (see Figure 5). The determined images have the most similar color characteristic as the query images and correspond to the most atmospheric similar images. On the other hand binary image representation is more suitable for simple form matching. Boundaries of the objects that are matched are defined by a threshold. The operation takes a black and white image, and maps all the pixels of the image whose values are over the threshold to one and all the others to zero. The test database DB consists of 1000 color images of size 384×256 with photos of landscapes and people, with several outliers consisting of color drawings of dinosaurs or photos of flowers with only a view colors. The image database was used in the SIMPLIcity project [10]. The images are mapped into a 3-band RGB (Red, Green, Blue) representation in which each color is represented by 8 bits and scaled to the size of 240×180 by a bilinear method [7]. This transformation represents the color images by vectors of the dimension $240 \cdot 180 \cdot 3 = 129600$, where each pixel is represented by three bands R, G, B and in which each component has a value between 0 and 255. The images can be also converted into a black and white representation by computing the mean value of the three RGB-bands $(R+G+B)/3$ representing the color of the pixel. In this case, the resulting vectors have the dimension $240 \cdot 180 = 43200$.

4.2 Orthogonal projection

The sequence of subspaces correspond to different resolutions of the images (see Figure 6). To compute a different resolution of an image, it has to be tiled with rectangle windows W which define sub-images of the window. The arithmetic mean

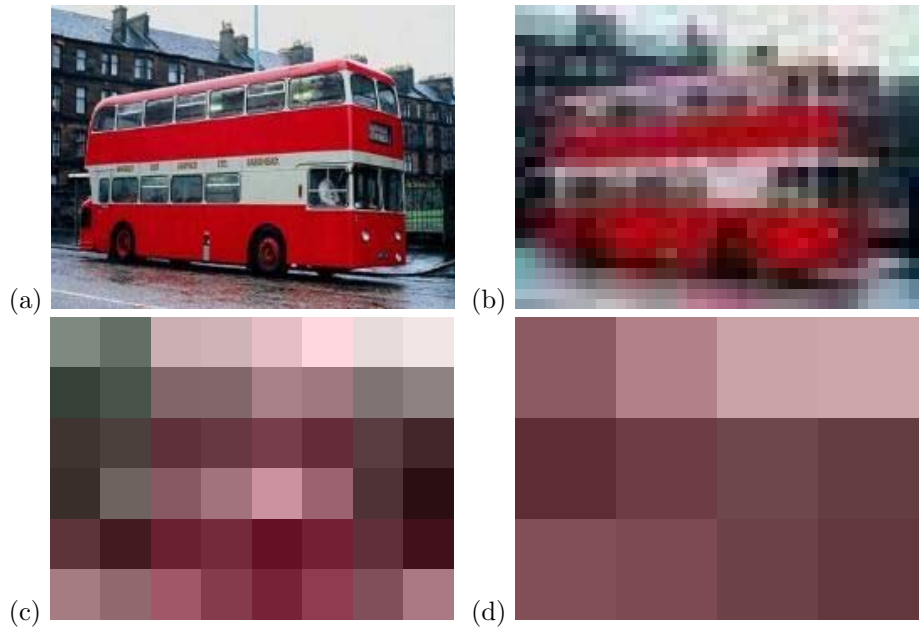

Figure 6: (a) Image of a bus, the size 240×180. (b) Image of the bus, resolution 40×30. (c) The image of the bus resolution 8×6. (d) The image of the bus, resolution 4×3.

is computed using the pixels in each rectangle window, and each sub-image in a window is replaced by this computed mean value. For black and white images, all pixels in the window are summed and divided by their total number. For color images, where each pixel is represented by three bands R, G, B, all components of a band in the window are summed and divided by their total number. The arithmetic mean value computation in a window corresponds to an orthogonal projection of these values onto a bisecting line. Because of this, the different resolutions of an image correspond to a sequence of subspaces that satisfy the lower bounding lemma. Formally the linear subspace sequence is defined by the mapping function $F_{a,b}()$, which corresponds to the rectangle windows W of size $j \times k$ in which the mean value is computed. We define a sequence of three subspaces of the space $V = U_0$

$$U_0 \supset U_1 \supset U_2 \supset U_3$$

which correspond to the functions $F_{0,1}() := \{mean\ over\ W | W\ with\ size\ 6 \times 6\}$, $F_{1,2}() := \{mean\ over\ W | W\ with\ size\ 5{\times}5\}$, $F_{2,3}() := \{mean\ over\ W | W\ with\ size\ 2{\times}2\}$. The dimensions of the subspaces are

$$dim(U_0) = 43200 > dim(U_1) = 1200 > dim(U_2) = 48 > dim(U_3) = 12$$

multiplied with factor 3 for color images (see Figure 6).

The distance between objects d_{U_0} in the space U_0 can be obtained from the distance d_{U_k} between objects in the orthogonal subspace U_k by multiplying the distance d_{U_k} by a constant $c_k = \sqrt{\frac{dim(U_0)}{dim(U_k)}}$, with $c_1 = 6$, $c_2 = 30$ and $c_3 = 60$.

The computing time of $U_0(DB[y])_\epsilon$ is saved compared to linear matching in the original space, if according to Equation 8 following constraints are valid,

$$\left\lfloor 1000 \cdot (1 - \frac{12}{48}) \right\rfloor = 750 \geq U_3(\sigma),$$

and according to Equation 7,

$$\left\lfloor 745 \cdot (1 - \frac{48}{1200}) \right\rfloor = 742 \geq U_2(\sigma),$$

$$\left\lfloor 735 \cdot (1 - \frac{1200}{43200}) \right\rfloor = 530 \geq U_1(\sigma).$$

4.3 Mean computational costs

The compting costs of $U_0(\sigma)$ are depending on y and the characteristics of $d[U_k(y)]_n$ in corresponding sequence of subspaces. To predict the mean computing costs of $U_0(\sigma)$ and to indicate the validity of the constraints for a given image database the mean sequence $d[U_k(DB)]_n$ is defined. For all s images $x^{(i)} \in DB$ the monotone increasing sequence $d[U_k(y)]_n$ in space U_k with $y = x^{(i)}$ is computed and the mean sequence is formed,

$$d[U_k(DB)]_n := \sum_{i=1}^{s} \frac{d[U_k(x^{(i)})]_n}{s}. \tag{18}$$

Note that $d[U_k(DB)]_n$ need not to be a monotone increasing sequence,

$$d[U_k(DB)]_n \leq d[U_k(DB)]_{n+1}$$

is not always valid. A subsequence of $d[U_k(DB)]_n$ whose all elements are $\leq \epsilon$ has to be determined for $U_k(DB)_\epsilon$,

$$U_k(DB)_\epsilon := \{\forall n : [d[U_k(DB)]_n \leq \epsilon\} \tag{19}$$

The size of the subsequence is $U_k(\overline{\sigma}) = |U_k(DB)_\epsilon)|$. However, for a sufficiently large s $d[U_k(DB)]_n$ is mostly a monotone increasing sequence, so that $U_k(\overline{\sigma})$ can be easily estimated. In Figure 7 we see the characteristics $d[U_0(DB)]_n$, $d[U_1(DB)]_n$, $d[U_2(DB)]_n$ and $d[U_3(DB)]_n$.

The maximal number of similar images $U_0(\overline{\sigma})$ to a given query image can be estimated under the assumption that the mean computing costs are only saved if $750 > U_3(\overline{\sigma})$ and $U_3(\overline{\sigma}) > \cdots > U_0(\overline{\sigma})$. It follows that the maximal number of similar images is around 134. Supposed $U_3(\overline{\sigma}) = 745$ then the corresponding ϵ is $d[U_3(DB)]_{745} = 466.02 \cdot 60 = 27961$. According to Equation 19 all elements of $d[U_k(DB)]_n$ have to be < 27961, which is true for $d[U_2(DB)]_{582}$, $d[U_1(DB)]_{261}$ and $d[U_0(DB)]_{134}$ (see Figure). It is also true that $U_2(\sigma) = 582$, $U_1(\sigma) = 261$ satisfy the constraints which were determined according to Equation 7.

To retrieve 134 most similar images to a given query image of the image test database the mean computation costs are according to Equation 9:

$$(43200 \cdot 261 + 1200 \cdot 582 + 48 \cdot 745 + 12 \cdot 1000) \cdot 3 = 12021360 \cdot 3$$

which is 3.5936 less complex than a list matching which requires $43200 \cdot 1000 \cdot 3$ operations.

In many query requests we search for around teen most similar images. In this case the computation cost are 15.654 less complex then a list matching. This is because the corresponding ϵ value for $U_0(\sigma) = 10$ is 22540, all elements of $d[U_k(DB)]_n$ have to be < 22540, which is true for $d[U_3(DB)]_{450}$, $d[U_2(DB)]_{236}$ and $d[U_1(DB)]_{57}$.

To retrieve 10 most similar images to a given query image of the image test database the mean computation costs are:

$$(43200 \cdot 57 + 1200 \cdot 236 + 48 \cdot 450 + 12 \cdot 1000) \cdot 3 = 2759760 \cdot 3.$$

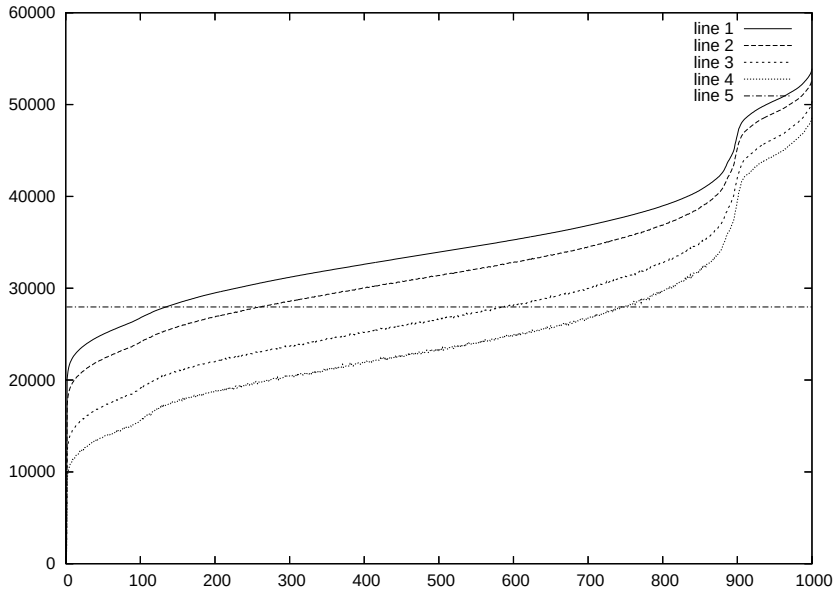

Figure 7: Characteristics of $d[U_0(DB)_n$ =line 1, $d[U_1(DB)]_n$ = line 2, $d[U_2(DB)]_n$ = line 3 and $d[U_3(DB)]_n$ = line 4 in the original space U_0, and 745 = line 5, below the value 745 computing time is saved.

Figure 8 indicates that the the mean computation cost are linearly dependent on the number of the most similar images which should be retrieved to a given query image.

It should be noted that this are estimated mean computational cost for DB, for example for $y = x^{(1)} \in DB$, $d[U_3(y)]_{745} = 386.26 \cdot 60 = 23175$ is and $U_0(\sigma) = |U_0(DB[y])_{23175}|$=12, $U_1(\sigma) = 104$ and $U_2(\sigma) = 540$, also satisfying Equation 7. It follows that to get the 12 most similar images, $U_0(DB[y])_{23175}$, the computation costs are according to Equation 9:

$$(43200 \cdot 104 + 1200 \cdot 540 + 48 \cdot 745 + 12 \cdot 1000) \cdot 3 = 5188560 \cdot 3$$

which is 8.326 times less then a list matching which requires 43200·1000·3 operations.

4.4 Estimation of the complexity by metric information

The determination of the mean retrieval computational costs for a large multimedia database is not practicable. However, the computation costs of the estimation of $U_0(\overline{\sigma})$, $U_1(\overline{\sigma})$ and $U_2(\overline{\sigma})$ can be reduced by $d^*[U_2(DB)]_n \approx \iota_3^2 \cdot d[U_3(DB)]_n$, and $d^*[U_1(DB)]_n \approx \iota_2^1 \cdot d^*[U_2(DB)]_n$ and $d^*[U_0(DB)]_n \approx \iota_1^0 \cdot d^*[U_1(DB)]_n$. However, it should be noted that the estimation is based on a raw approximation, as shown in Figure 9, where $d[U_0(DB)]_n$ and $d[U_k(DB)]_n \cdot \iota_b^a,\ k \in \{1,2,3\}$ are shown.

The metric information gain can be estimated for l objects $l \leq s$ in the database it is,

$$mu(U_k(DB)) \approx mu_{i=1}^{l}(U_k(DB)),$$

$$mu_{i=1}^{l}(U_k(DB)) := \{\sum \frac{2 \cdot d(U_k(x^{(i)}), U_k(x^{(j)}))}{(s-1) \cdot s} | \forall i \neq j \in \{1..l\}, l \leq s\} \qquad (20)$$

A graph representing $mu_{i=1}^{l}(U_k(DB))$ in dependency to l, for $l \in \{1..s\}$ and for $k \in \{0,1,2,3\}$ indicates strong correlated fluctuations in four spaces, corresponding

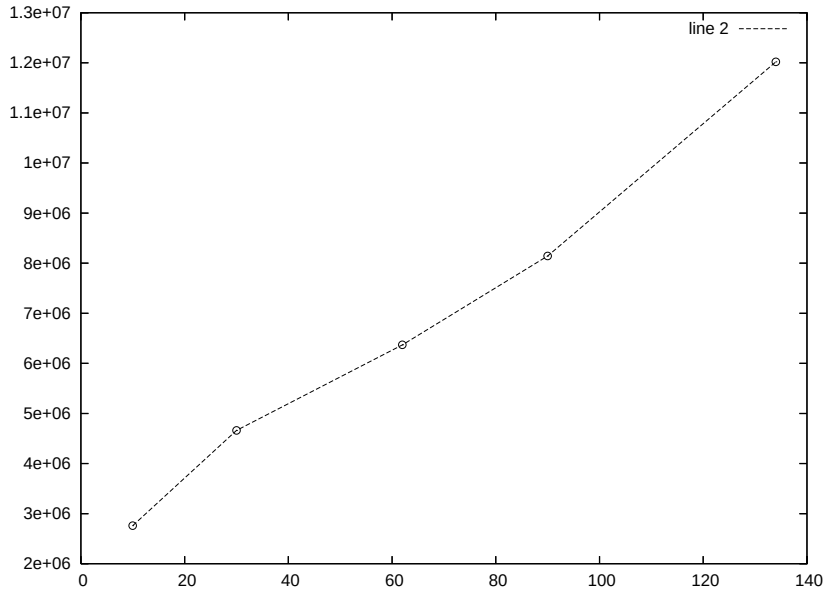

Figure 8: Computing costs, the x-axis indicates the number of the mot similar images which are retrieved, the y-axis the computer costs. The computing costs are dependent on the size $U_0(\sigma)$).

to the outliers consisting of the color drawings of dinosaurs or photos of flowers with only several colors (see Figure 10).

These outliers can also be identified by fluctuations in the graph representing the image entropy of the $U_1(DB)$ (see Figure 11). Image entropy is a scalar; it is a statistical measure of randomness that can be used to characterize the texture of the input image. Entropy is defined as $-\sum_{i=0}^{3}\sum_{j=0}^{255}(p_{ji}\cdot\ln(p_{ij}))$ where p_{ij} contains the histogram counts returned from image histogram, 256 bins are used to compute the histogram for each color of an RGB image [7]. The mapping from U_1 to U_2 and U_2 to U_3 reduce the entropy, fluctuations become more and more washed-out, there is not enough entropy information. This is not the case with the mean distance $mu_{i=1}^{l}(U_k(DB))$, as shown in Figure 10.

Table 1 shows the quality of the mapping functions for the three subspaces of the space $V = U_0$ for the test image database of 1000 images.

a, b	c_b^a	ι_b^a	q_b^a
0, 1	6	1.084 (1,09)	0.756 (0.7489)
1 ,2	5	1.186 (1.197)	0.643 (0.636)
2, 3	2	1.159 (1.167)	0.363 (0.357)
0, 2	30	1.286 (1.307)	0.744 (0.732)
0, 3	60	1.491 (1.525)	0.654 (0.639)

Table 1: The quality of the mapping functions for the three subspaces of the space $V = U_0$ for the test image database of 1000 images.

$F_{0,1}() := \{mean\ over\ W|W\ with\ size\ 6 \times 6\}$ is the mapping with the best quality for the test database. $F_{2,3}() := \{mean\ over\ W|W\ with\ size\ 2 \times 2\}$ is the mapping the worst quality for the test database, because of that only minimal savings in computing costs occur. The quality of the mapping is not only dependent

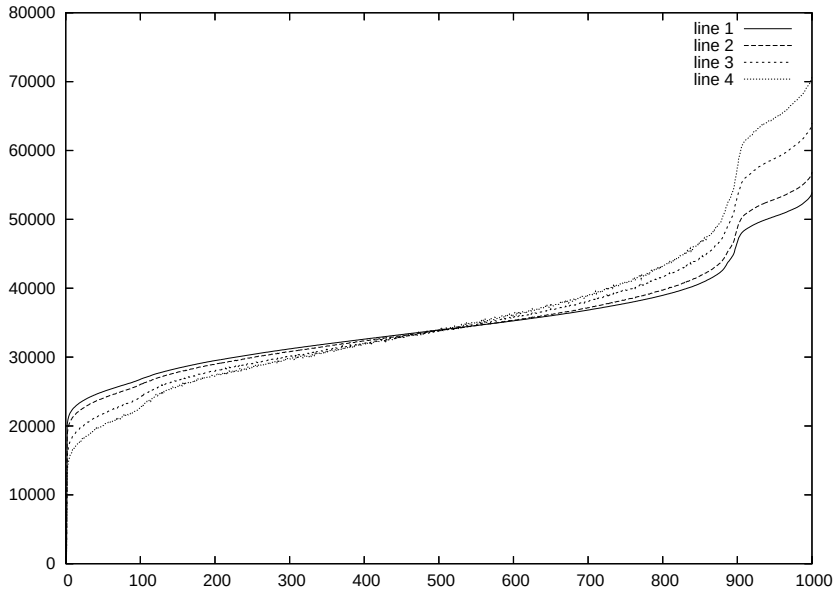

Figure 9: $d[U_0(DB)]_n$ =line 1 and $d[U_k(DB)]_n \cdot \iota_0^k$ =line k, $k \in \{1, 2, 3\}$.

on the size of the window as can be seen by the size of the following quality values; $q_1^0 > q_2^0 > q_3^0$.

The image entropy has less spatial information that is taken into account. Because of that, the proportion of the mean entropy values (image entropy loss) can not be used to estimate the metric information loss. Mean entropy values correspond to the entropy of $U_k(DB)$,

$$entopy(U_k(DB)) = \sum_{i=1}^{s} \frac{entropy(U_0(x^{(i)})}{s}$$

$$image\ entropy\ gain_0^k = \frac{entopy(U_0(DB))}{entopy(U_k(DB))}.$$

Table 2 shows the values for metric information loss compared with the corresponding proportion of the mean entropy values.

U_f/U_k	$metric\ information\ gain_k^f$	$image\ entropy\ gain_k^f$
U_0/U_1	1.08	1.0
U_1/U_2	1.19	1.36
U_2/U_3	1.16	1.5

Table 2: Metric information loss compared with the corresponding proportion of the mean entropy values (the image entropy loss). The image entropy has less spatial information that is taken into account, because of that more image entropy is lost in lower dimensional space compared to the metric information loss.

The metric information gain can be estimated for l objects $l \leq s$ in the database it is,

$$mu(U_k(DB)) \approx mu_{i=1}^{l}(U_k(DB)),$$

$$mu_{i=1}^{l}(U_k(DB)) := \{\sum \frac{2 \cdot d(U_k(x^{(i)}), U_k(x^{(j)}))}{(s-1) \cdot s} | \forall i \neq j \in \{1..l\}, l \leq s\} \qquad (21)$$

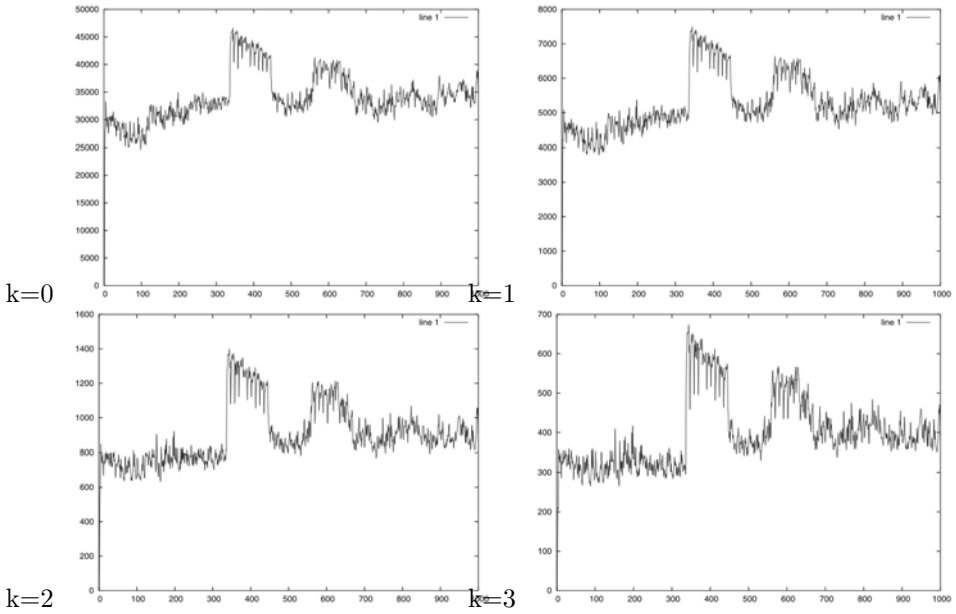

Figure 10: A graph representing $mu_{i=1}^{l}(U_k(DB))$ in dependency to l for $l \in \{1..s\}$ and for $k \in \{0,1,2,3\}$ indicates strong correlated fluctuations in four spaces corresponding to the outliers consisting of color drawings of dinosaurs or photos of flowers with only several colors. Shown are the values corresponding to the distance d_{U_k} inside the orthogonal space, the correctly scaled values are obtained by multiplication with a constant c_k.

and

$$\iota_k^0(DB) \approx \frac{mu_{i=1}^{l}(U_0(DB))}{mu_{i=1}^{l}(U_k(DB)) \cdot c_k^0}. \tag{22}$$

The metric information gain is estimated by l first objects $l = 20$ in the database, as shown in Figure 12. The four first first values which correspond to the ratio of one, two, three and four objects in different spaces are taken out and the mean value is computed from the $4 - 20$ remaining values, $\iota_1^0 \approx 1.0913$ $\iota_2^1 \approx 1.2017$, $\iota_3^2 \approx 1.1490$ (see Figure 12).

The resulting $d[U_2(DB)]_n \approx d^*[U_2(DB)]_n = \iota_3^2 \cdot d[U_3(DB)]_n$, and $d[U_1(DB)]_n \approx d^*[U_1(DB)]_n = \iota_2^1 \cdot d^*[U_2(DB)]_n$ and $d[U_0(DB)]_n \approx d^*[U_0(DB)]_n = \iota_2^0 \cdot d^*[U_1(\overline{y})]_n$ providing an approximation with about 80% accuracy, saving about 99% of computation time (see Figure 13). The savings correspond to the proportion $dim(U_0)$ to $dim(U_3)$. Because $d^*[U_3(DB)]_{745} = 466.02 \cdot 60 = 27961$ and
$U_0(\sigma) = |U_0(DB[DB])_{27961}|$=133, $U_1(\sigma) = 203$ and $U_2(\sigma) = 504$ satisfying also the Equation 7.

It follows that to get the 133 most similar images, $U_0(DB[DB])_{23175}$, the computation costs according to Equation 9 are:

$$(43200 \cdot 203 + 1200 \cdot 504 + 48 \cdot 745 + 12 \cdot 1000) \cdot 3 = 9422400 \ (12021360) \cdot 3$$

which is 4.5848 (3.5936) less complex then list matching, which requires $43200 \cdot 3 \cdot 1000$ operations.

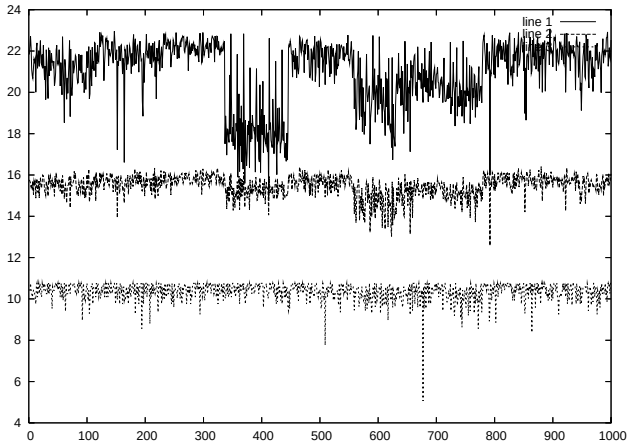

Figure 11: Graph representing the image entropy of $U_1(DB)$, $U_2(DB)$ and $U_3(DB)$. Less entropy means, less information. Smaller values indicate less information. The mapping from U_1 to U_2 and U_2 to U_3 reduce the entropy; fluctuations become more and more washed-out because there is not enough entropy information. $U_0(DB) \approx U_1(DB)$ it is not shown.

4.5 Errors

By the metric information loss of the transition from $U_0(DB)$ to $U_k(DB)$ errors emerge as indicated for example by $d[U_k^0(y)]_n$ with $y = x^{(4)}$, see Equation 15, for $n \in \{1..s\}$ in Figure 14 and for $n \in \{1..30\}$ in Figure 15.

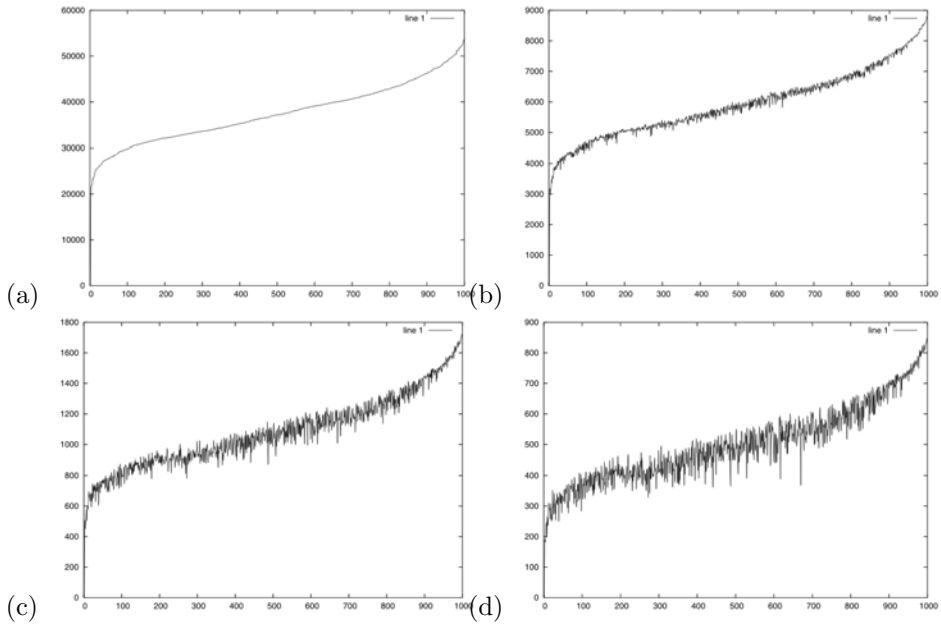

Figure 14: For $n \in \{1..s\}$, $d[U_k^0(y)]_n$ with $y = x^{(4)}$. (a) The original $d[U_0(y)]_n$, (b) $d[U_2^0(y)]_n$, (c) $d[U_2^0(y)]_n$, (d) $d[U_3^0(y)]_n$.

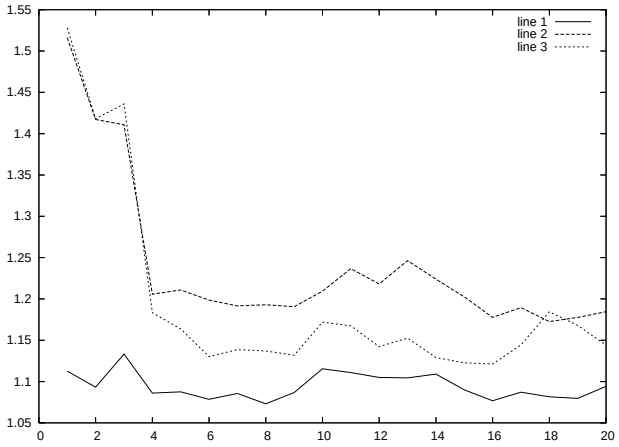

Figure 12: The metric information gain is estimated by l first objects $l = 20$ in the database, line 1=ι_1^0, line 2=ι_2^1 and line 3= ι_3^2

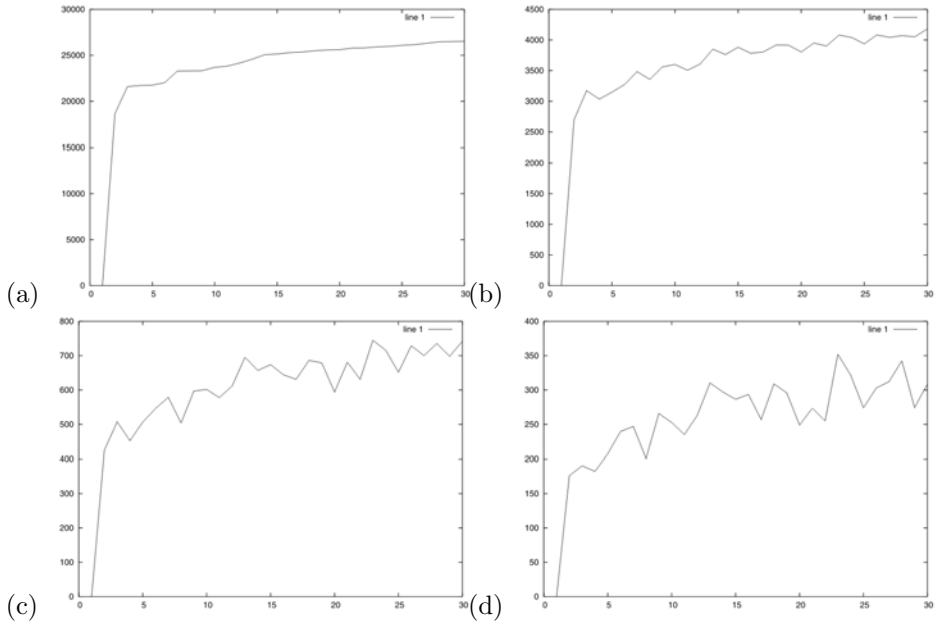

Figure 15: For $n \in \{1..30\}$, $d[U_k^0(y)]_n$ with $y = x^{(4)}$. (a) The original $d[U_0(y)]_n$, (b) $d[U_2^0(y)]_n$, (c) $d[U_2^0(y)]_n$, (d) $d[U_3^0(y)]_n$.

The resulting mean error in U_k is indicated by the distance of the ordered subsequencs in U_k from the sequence in U_0 according to Equation 16, the graph representing the mean error for $m \in \{1..s\}$ is shown in Figure 16. The $Error_1^0(1000) = 28968$, $Error_2^0(1000) = 67655$, $Error_3^0(1000) = 92143$. Surprisingly the resulting mean ordered sequence in U_0 using the so called Manhattan distance function $d = l_1$ in relation to the mean ordered sequence in U_0 using the Euclidian distance function $d = l_2$ distance metric produce a higher error, then $Error_1^0(1000)$, namely 33122.

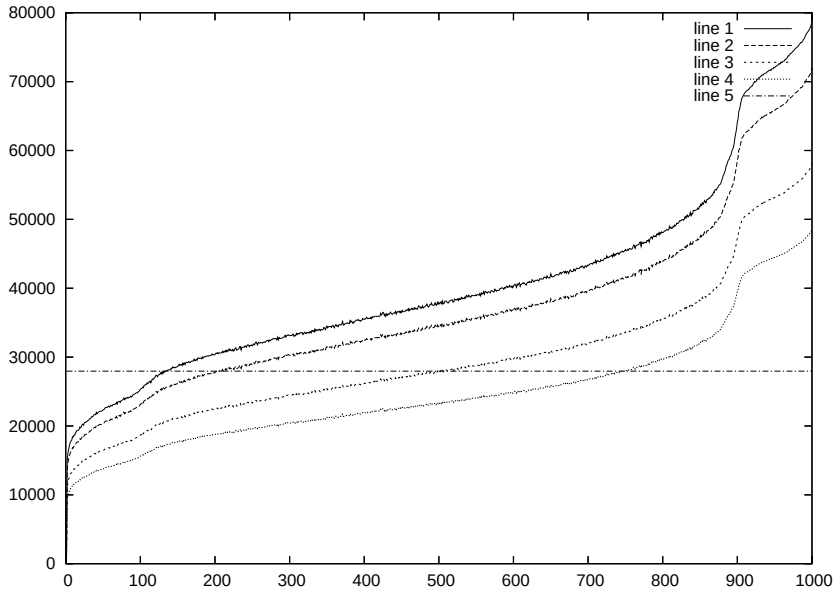

Figure 13: Characteristics of $d^*[U_0(DB)]_n$ =line 1, $d^*[U_1(DB)]_n$ = line 2, $d^*[U_2(DB)]_n$ = line 3 and $d[U_3(DB)]_n$ = line 4 U_0, and 745 = line 5, below the value 745 computing time is saved (around 80% of accuracy).

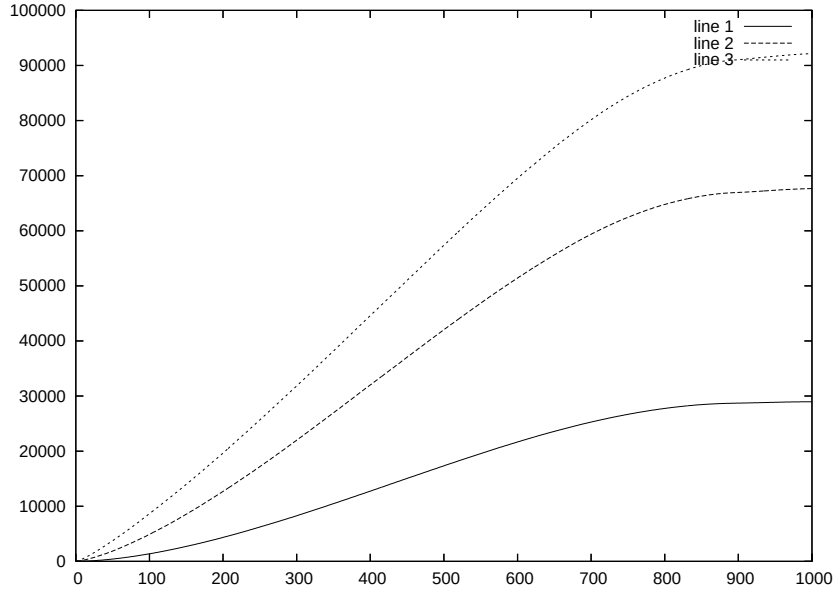

Figure 16: Graph representing the mean error, line 1 indicates $Error_1^0(n)$, line 2 indicates $Error_2^0(n)$ and line 3 indicates $Error_3^0(n)$.

5 Conclusion

Metric indexes trees operate efficiently when the number of dimensions is small. Growth of the number of dimensions has negative implications for the performance of multidimensional index trees. Tree indexing methods explode exponentially with the dimensionality, eventually reducing the search time to sequential scanning.

Based on the generic multimedia indexing approach and lower bounding methods an hierarchical subspace indexing method was described which does not suffer from the dimensionality problem. This is because no sphere or a minimum bounding rectangles are used.

The hierarchical subspace approach offers a fast search method for large content-based multimedia databases as shown by mathematical estimations. The introduced metric information loss differs from the image entropy, because it takes into account the metric properties of the images.

The developed framework was demonstrated on a practical example of fast searching methods for content-based image retrieval. The test database DB consisted of 1000 color images. In our image indexing method the subspaces correspond to different resolutions of the images. The search starts in the subspace with the lowest resolution of the images, where the set off all possible similar images is determined. In the next subspace, additional information corresponding to a higher resolution is used to reduce the set of the possible similar images. This procedure is repeated until the similar images can be determined in the space from the selected set eliminating the false candidates. Different resolutions of an image were computed by the arithmetic mean using the pixels in rectangle windows, corresponding to an orthogonal projection of these values onto a bisecting line.

We have shown empirically (software experiments) and theoretically that our method can save the computing time considerably.

6 Acknowledgements

This work was supported by the FCT Bolsa de Investigacao SFRH/BPD/20388/2004, "Development of Statistical language modeling for image retrieval in large databases".

References

[1] C. Böhm, S. Berchtold, and A. Keim Kei, D. Searching in high-dimensional spaces—index structures for improving the performance of multimedia databases. *ACM Computing Surveys*, 33(3):322–373, 2001.

[2] T. H. Cormen, C. E. Leiserson, L. R. Rivest, and C. Stein. *Introduction to Algorithms*. Second. MIT Press, 2001.

[3] Lynne Dunckley. *Multimedia Databases, An Object-Rational Approach*. Addison Wesley, 2003.

[4] Christos Faloutsos. Modern information retrieval. In Ricard Baeza-Yates and Berthier Ribeiro-Neto, editors, *Modern Information Retrieval*, chapter 12, pages 345–365. Addison-Wesley, 1999.

[5] Christos Faloutsos, Ron Barber, Myron Flickner, Jim Hafner, Wayne Niblack, Dragutin Petkovic, and Will Equitz. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, 3(3/4):231–262, 1994.

[6] M. Flickner, H. Sawhney, W. Niblck, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content the QBIC system. *IEEE Computer*, pages 23–32, September 1995.

[7] R. C. Gonzales and E. W. Woods. *Digital Image Processing*. Prentice Hall, second edition, 2001.

[8] Till Quack. Cortina: A systemfor large-scale, content-based webimage retrieval and the semantics within. Master's thesis, ETH Zurich, 2004.

[9] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, December 2000.

[10] J.Z. Wang, J. Li, and G. Wiederhold. Simplicity: Semantics-sensitive integrated matching for picture libraries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(9):947–963, 2001.