An Application of LSI and M-tree in Image Retrieval

Tomáš Skopal, Václav Snášel

[1]Charles University in Prague, FMP, Department of Software Engineering,
Malostranské nám. 25, 118 00 Prague, Czech Republic
[2]Technical University of Ostrava, FEECS, Department of Computer Science,
tř. 17. listopadu 15, 708 33 Ostrava, Czech Republic
tomas.skopal@mff.cuni.cz, vaclav.snasel@vsb.cz

Abstract. When dealing with image databases, we often need to solve the problem of how to retrieve a desired set of images effectively and efficiently. As a representation of images, there are commonly used some high-dimensional vectors of extracted features, since in such a way the content-based image retrieval is turned into a geometric-search problem. In this article we present a case study of feature extraction from raw image data by means of the LSI method (singular-value decomposition, respectively). Simultaneously, we show how such a kind of feature extraction can be used for efficient and effective similarity retrieval using the M-tree index. Because of the application to image retrieval, we also show some interesting effects of LSI, which are not directly obvious in the area of text retrieval (where LSI came from).

LSI, similarity search in image databases, M-tree

1 Introduction

With the continuously emerging information technology, the volume of multimedia databases (collections of text, image, audio and video documents/objects) increases rapidly, while there is a strong need for an appropriate representation of such a kind of databases, in order to provide an efficient and effective way of content-based multimedia retrieval [7]. A multimedia object (an image, in our case) itself is often represented by a high-dimensional vector, because then semantics of content-based retrieval from a database (transformed into a vector set) can be defined as a geometric problem. Given a query image (its vector respectively), the vector set is examined by use of a similarity function which is used to assign a relevance of each examined particular image to the query.

Up to this day, there exist many methods for a particular image retrieval domain, but none of them stands for a way how to generally build up an *effective* geometric retrieval model (i.e. the feature vector extraction method and the design of similarity function), which could be successfully applicable to any given image database domain. Furthermore, the most frequent method of similarity

search in a vector set employs the simple sequential scan, where the query vector is compared against every vector in the set. Although this approach is sufficient for small data, for large image databases the sequential scan becomes *inefficient* or even impracticable.

Fig. 1. Several images from the collection of buildings.

In this article we propose an M-tree-based indexing of images, the feature vectors of which were constructed using the LSI method (singular-value decomposition) from the raw pixel data. With this approach we address both issues, the effectiveness (i.e. the quality of retrieval, commonly measured by the precision and recall) as well as the efficiency (i.e. the performance of retrieval). In Figure 1 see a sample from collection of 730 images [20], which will be used in the following as well as for the experiments.

We need to say this paper has not an ambition to compete with the powerful techniques of image recognition and indexing [7] (that's why we do not include the state-of-the-art section, it would be too huge), we intend this paper rather as a expedition into the "guts" of SVD and LSI by means of visualization. The visualized objects (e.g. singular vectors) should provide the reader by a hint why to use (or not use) the LSI techniques in various areas of information retrieval.

2 Feature Extraction using LSI

During the last two decades, there have been developed very many methods for feature extraction from images (hence forming the vector representation), the most often methods extract various histograms (of colors, contrast, etc) [3], textures [18], shapes [14], color layout [9], and so on. For domain specific image databases (e.g. fingerprints, irises, faces) we can find many other, specialized, approaches. For an overview of current methods of image feature extraction we refer to [4, 13].

In our approach we use completely different view to the problem [11, 12], where the whole image of size $x \cdot y$ pixels (their levels of brightness, respectively) is treated as a vector of dimension $x \cdot y$. The image-to-vector transformation is very simple, we just concatenate the pixel rows of the image into a single one, long pixel row (the vector of pixel brightnesses). After this preparation[1] we are able to represent the entire collection of n images by a matrix A of order $x \cdot y \times n$, where the individual image vectors are the matrix columns (see Figure 2, $x = 80$, $y = 60$, $n = 730$, i.e. the matrix A is of order 4800×730).

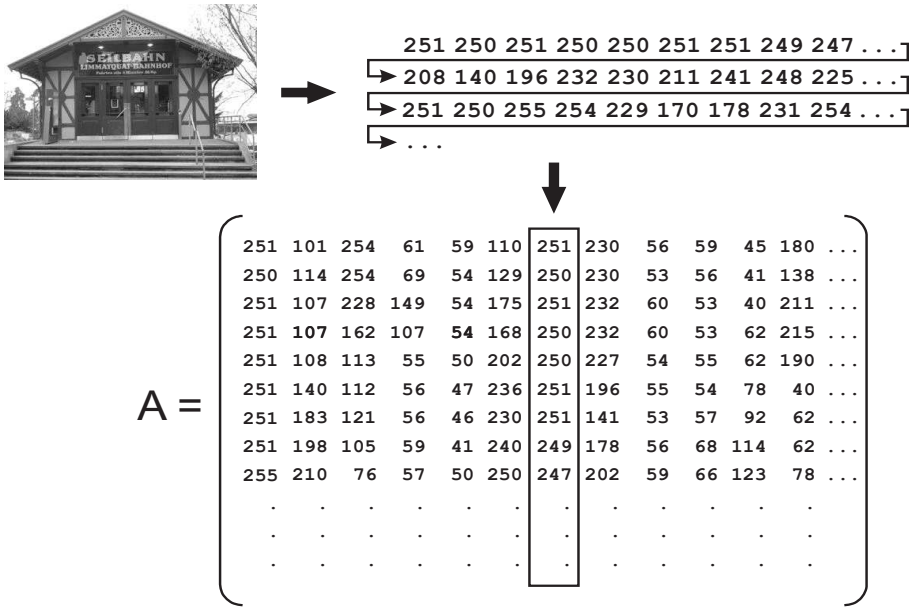

Fig. 2. The image-to-vector transformation and the placement in the matrix A.

[1] We cannot speak about a kind of feature extraction here, because the entire image information (brightness information, actually) is preserved.

2.1 The Classic LSI Method as an Extension to the Vector Model

The origin of LSI (latent semantics indexing) method is in the area of text retrieval [1], while the main intent was an attempt to eliminate the negative aspects of classic vector model [8, 2]. A text collection consists of m unique terms and each of the n documents in the collection is represented by an m-dimensional vector of frequencies (or weights) of terms in that document. The entire collection is represented by a matrix A, in a similar way as we have mentioned above. Using the singular-value decomposition (SVD) of the matrix A

$$A = U\Sigma V^T$$

we obtain so-called *concept vectors* (left-singular vectors – the columns in U), which can be interpreted as individual (semantic) topics present in the collection. The concept vectors form a basis in the original high-dimensional vector space, while they are actually linear combinations of terms (the terms are supposed as independent). An important property of the SVD is a fact that concept vectors are ordered according to their "significance", which is defined by values of the singular values σ_i stored in ascending order in the diagonal matrix Σ. Informally, the concept significance says in what quantity is the appropriate concept globally present (or missing) in the collection. It also says which concepts are semantically important (that is where the "latent semantics" comes from) and which are not – such unimportant concepts are, in fact a "semantic noise". The columns of ΣV^T contain document vectors (the *pseudo-document vectors*), but which are now represented in the basis U, i.e. in the *concept basis* (unlike the original term basis). Every pseudo-document vector describes a linear combination of the concept vectors, i.e. the appropriate document consists somehow (positively or negatively) of every concept found.

Since only first k concepts can be considered are semantic important (the singular values are high), we can approximate the decomposition as

$$A \approx U_k \Sigma_k V_k^T$$

where U_k contains the first k most important concept vectors, Σ_k contains the respective singular values and $\Sigma_k V_k^T$ contains the pseudo-document vectors represented using the first k concept vectors (see Figure 3). In other words, by SVD the original m-dimensional vectors are projected into a vector space of dimension k ($k \ll m$). The SVD approximation (so-called *rank-k SVD*) can be created either by "trimming" the full-SVD matrices or by usage of a special method designed to perform directly rank-k SVD, like Lanczos or Arnoldi.

For the retrieval of documents we compare the pseudo-document vectors with the *pseudo-query vector* using a similarity measure (e.g. the *cosine measure*, used in the classic vector model as well as by LSI). In order to compare the pseudo-document vectors, we need to project the query vector q (which is represented the same way as the document vectors in A) into the concept basis, i.e. by $U_k^T q$.

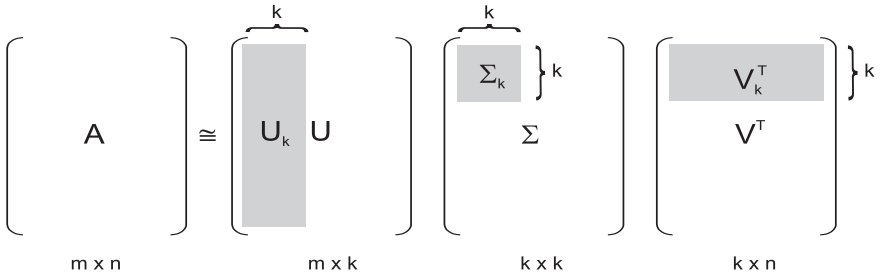

Fig. 3. The SVD approximation (rank-k SVD).

2.2 An LSI for image databases

Since the SVD can be applied to any matrix (i.e. to any static vector set of fixed dimension), the LSI method is applicable to feature vectors of any kind of data. The interpretation of matrix A as a term-by-document matrix is purely a matter of application, i.e. an application to the vector model in text retrieval. However, we can generally adopt the LSI method for indexing of any multimedia database which is representable by a set of vectors (of equal dimension). Moreover, it does not matter which kind of extraction was used to construct the vectors – we only require to build the matrix A such that the feature vectors of the indexed objects are stored as columns.

In our case, there is no obstacle to build the matrix A from the "brightness vectors", as we have described at the beginning of the section. Using SVD we decompose the matrix A the same way as a text collection is indexed, i.e. we get the decomposition approximation $A \approx U_k \Sigma_k V_k^T$. What is interesting specifically to this application is the interpretation and mainly visualization of the concept vectors (the U basis), which represents some set of "base images", from which every image in the database is composed (see Figure 4). In other words, with a suitable linear combination of the base images we can reconstruct any of the images in the database (the higher k, the more precise image reconstruction).

At this point we can notice a certain connection with the discrete cosine transformation (DCT), used in the JPEG compression, where e.g. 64 base images are used to represent image blocks of size 8×8, however, these 64 base images are fixed (combinations of discrete cosines with different frequencies) and can be generated independently.

In Figure 4 see several most significant base images – these can be interpreted as follows. The first base image is just the average brightness in the entire database (average brightness of buildings). The next bases represent the coarse shapes (building silhouettes, transition between buildings and the background or sky), and the bases with lowest singular values σ_i (i.e. with the higher order i) constantly include more and more of finer details present in images (e.g. window and door shapes).

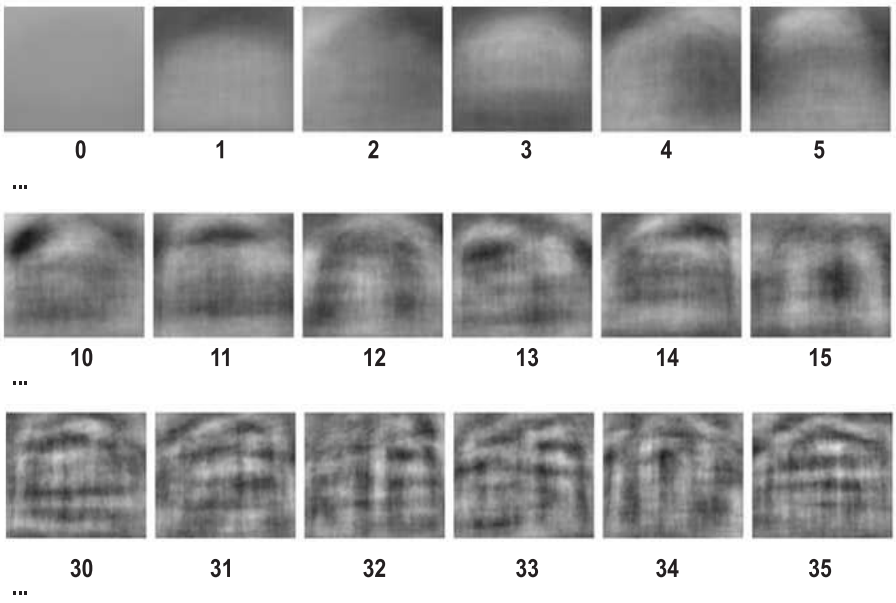

Fig. 4. Base images (visualization of concepts – singular vectors, respectively) with the appropriate order of singular values σ_i.

Image reconstruction In order to get an idea of how big (or small) k is yet (still) sufficient to describe the semantic content of an image, we can reconstruct and visualize the decomposition of A by $A \approx U_k \Sigma_k V_k^T$.

In Figure 5a we see the original images, which have been perfectly reconstructed by the full SVD $A = U\Sigma V^T$ (where k is maximal, i.e. the rank of the matrix A). On the other side, for very small k the reconstruction is very bad, see Figure 5b, where $k = 15$. However, even this imperfect reconstruction gives an information about silhouettes of the buildings. We must realize that the reconstructed image is a combination of only 15 base images, i.e. instead of 4800 pixels we need just 15 concept weights! In case of $k = 50$ (see Figure 5c) the reconstruction is better, we can recognize some coarse details, e.g. the windows. For $k = 250$ the reconstruction is on such level that we can identify the original images (see Figure 5d).

The above example gives us some guidelines for the content-based retrieval. If we want to consider just the coarse shapes, it is appropriate to use first few coordinates of pseudo-image vectors. On the other side, in case we want to search according to details, we choose a greater number of coordinates, i.e. a higher k.

Note: The example also shows the capability of SVD as a compression method for the entire image database – the advantage over JPEG (DCT, respectively) is a fact that the base images are made-to-measure to the particular database being

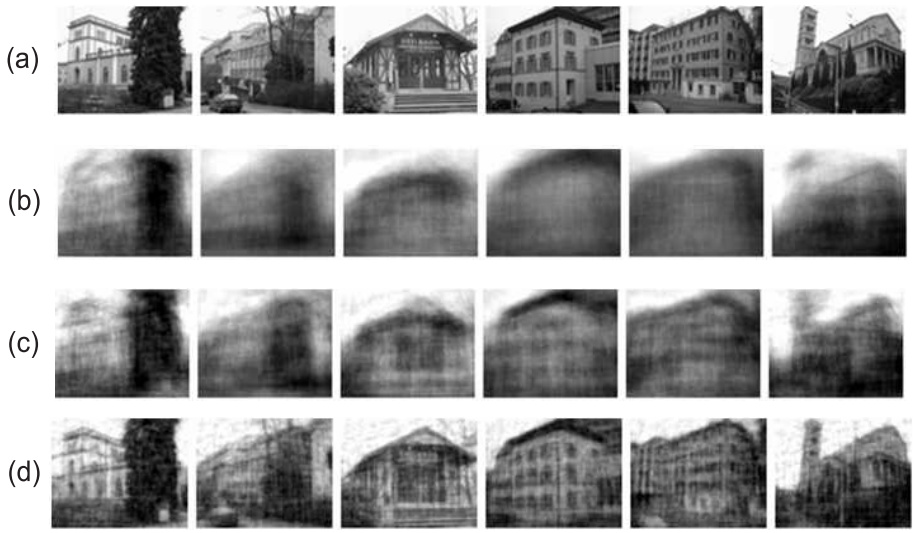

Fig. 5. (a) A sample of the original images. Reconstruction of the images from the rank-k approximation $A \approx U_k \Sigma_k V_k^T$ for (b) $k = 15$, (c) $k = 50$, and (d) $k = 250$.

compressed (not just simple combinations of cosines), which should be reflected in a smaller number of base images needed for a good quality reconstruction. A disadvantage is the need to store the "base image lexicon" together with the pseudo-image vectors.

2.3 Measuring the Image Similarity

After we get the set of pseudo-image vectors, we need to compare the vectors with the pseudo-query vector, in order to evaluate a similarity query. Generally, there exist many similarity measures in the area of similarity search, some of them even have no analytic representation, just an algorithmical one.

In case of color/brightness histograms on images there is often used the quadratic form distance [9, 15], where the relation between different colors is captured by correlation weights between individual vector coordinates. The quadratic form distance is, in fact, a generalization of the well-known Euclidean distance where the correlation between different coordinates are zero, i.e. all dimensions are considered as independent.

Since the images are modeled by pseudo-image vectors which dimensions are inherently independent (the coordinates contain weights of concepts – the concepts form a basis, so they are linearly independent), and so it is sufficient to use the Euclidean distance, or another Minkowski (L_p) metric. The similarity modeled by distance is interpreted such that distant vectors are not very similar, while close vectors are highly similar. The zero distance means the respective

two images (or an image and a query image) are identical. The Euclidean distance is advantageous in satisfying the metric properties, thus the entire image database (the derived vector set, respectively) can be indexed by *metric access methods* [19, 5], which have been designed to provide an efficient retrieval from multimedia databases modeled in metric spaces. The efficiency of retrieval is usually measured by two factors – the number of I/O operations (disk access costs) and the number of distance computation (computation costs).

3 M-tree

The M-tree [6, 10, 17, 19] is one of the metric access methods, allowing to index a collection of objects modeled in a metric space $\mathcal{M} = (\mathbb{U}, d)$, where $\mathbb{U}$ is an object universe (e.g. a vector space, in our case) and d is a metric. Like other indexing structures, also the structure of M-tree is based on the concept of B^+-tree, i.e. it is a balanced, dynamic and paged (easily persistent) structure. A particular M-tree index represents a hierarchy of spherical metric regions (each M-tree node maps to a single region), and the whole M-tree is a hierarchy of that regions.

3.1 Structure of M-tree

The M-tree leaf nodes store so-called *ground entries* $grnd(O_i)$ (the indexed objects themselves), while the inner nodes store *routing entries* $rout(O_i)$. Each routing entry describes a spherical *metric region*, which is an area in the metric space where the objects indexed in the appropriate subtree are located. The shape and location of every metric region is given by a hyper-sphere centered in an data object O_i and bounded by a covering radius r_{O_i}. In Figure 6 see an example of hierarchy of metric regions (using Euclidean distance and 2D space) and the respective M-tree.

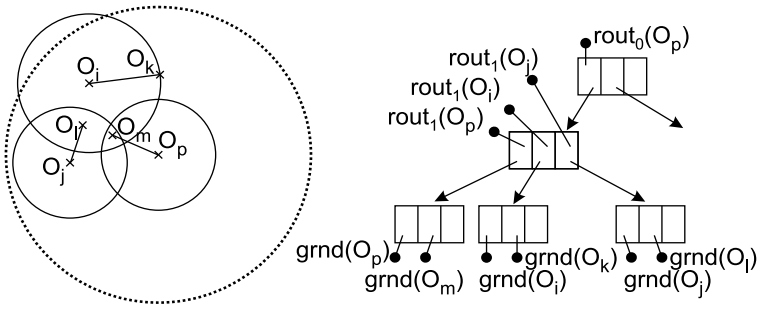

Fig. 6. A hierarchy of metric regions and the respective M-tree.

3.2 Querying the M-tree

The indexing of objects in M-tree is based purely on mutual object-to-object distances (using the metric d), due to which we can easily implement the two basic types of similarity queries – the *range query* and the *h nearest neighbors query* (h-NN query)[2]. The range query selects all such indexed objects for which the distance to the query object is smaller than a distance threshold (or query radius). The h-NN query selects the h closest objects to the query object.

The higher efficiency (performance) of searching with M-tree, when related to the simple sequential scan, relies in filtering of those branches of M-tree index, which do not intersect the query region (and thus cannot contain any relevant objects). The correctness of filtering is guaranteed by metric properties, especially by the triangular inequality, which is the essential property used by all metric access methods.

3.3 An Application of M-tree for Image Retrieval

The M-tree can index any collection which is represented in metric space, so we can safely use M-tree also for indexing of the pseudo-image vectors according to the Euclidean distance. The similarity search in the image database is accomplished by range and h-NN queries.

4 Experimental Results

We have performed several retrieval effectiveness and efficiency experiments on the database of 730 building images. Regarding the effectiveness point of view, we have examined the benefits of the feature extraction using LSI, i.e. what quality of retrieval can be achieved when using pseudo-image vectors and the Euclidean distance. The efficiency experiments were aimed to measure the I/O and computation costs when querying using the M-tree.

4.1 Retrieval Effectiveness

The image database consisted of 146 groups of buildings, each building was shot 5 times from different angles of view (see an example in Figure 7). Therefore, we have decided for the identification task, which is one of the hardest ones. Naturally, the entire database of 730 images was indexed, the clustering into groups was just logical (or semantic).

We have randomly selected 50 groups of buildings, from each group a single image was randomly chosen and a 4-NN query executed against this image. We have expected that rest of the images in each respective group will be returned by the query. So, if the result included all 4 images, the precision of the answer was 100%, if only 3 correct images were included, the precision was 75%, and so on.

[2] The parameter is usually labeled as k, i.e. we speak about k-NN queries, however, we have introduced k for rank-k SVD, so for queries we will use h to avoid a confusion.

Fig. 7. A group of images showing the same building from different angles of view.

Although the order of images in the query result matters, in our "identification" scenario it was sufficient to have the correct image in the result, regardless of the similarity order, because the query result was quite small. The recall of the answer have always matched the precision, since the size of query result was equal to the number of relevant images in the database, i.e. 4.

Fig. 8. The query result for the first query.

In Figure 8 see an example of query, for different values of k (i.e. for image description using k base images). The precision of answer was 50% for $k = 50$, moreover, for $k = 250$ the desired building occupied the first two places in the ordering.

In Figure 9 see a 4-NN query for another image. The precision of answer was again 50% for $k = 15$, however, for $k = 50$ and $k = 250$ it was only 25%. This result could seem to be in contrast with what we have said about the quality of reconstruction, i.e. where the higher k was better for the quality of a reconstructed image – so we would expect that a higher k will lead also to a better precision when searching. However, such an expectation is not quite well-founded, because (as we have discussed in section 2.2), the more significant

(i.e. frequent) base images represent the coarse shapes, while the less significant ones represent various details. In the second query, the higher k caused a mismatch between LSI and the human perception of similarity, because the details percieved by human are of another kind than those captured by high-order base images obtained by SVD. On the other side, the same effect is responsible for ordering the really relevant image down to the fourth place.

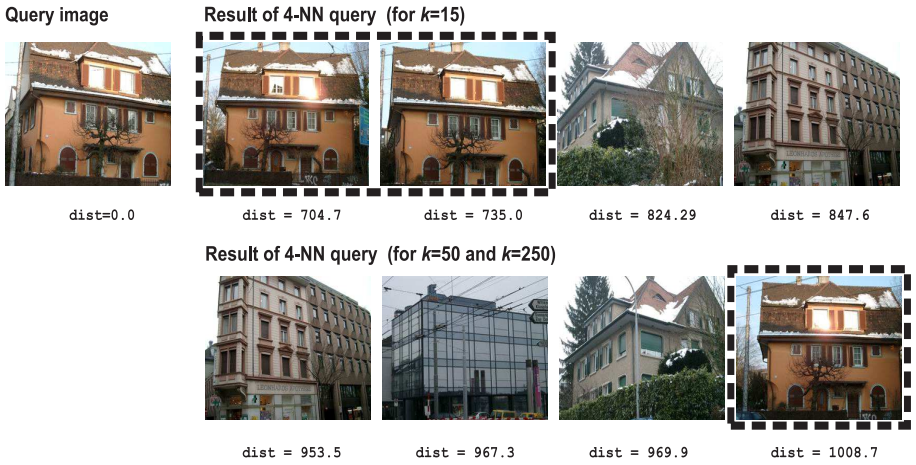

Fig. 9. The query result for the second query.

Summary The average precision for the 50 4-NN queries was 43% (i.e. 1–2 correct images in every query result), while the best results were achieved with $k = 15$. The precision 43% can be evaluated as very good if we realize that every building was often shot from very different angles of view. This observation also leads us to a hypothesis that LSI is quite resistant to spatial transformation in the images. If we relax the identification requirement such that it is sufficient to have a single occurence of the correct building in the query result, we would get almost 100% identification precision.

4.2 Retrieval Efficiency

In the second set of experiments we have examined the retrieval costs when using M-tree. A tree M-tree indices were built, for $k = 15, k = 50, k = 250$, respectively, i.e. each k-dimensional vector set was indexed by a single index. The M-tree index construction was guided by `MinMax + MultiWay` method (for details we refer to [17, 16]). The results of query processing are presented in Table 1. The node capacity was set to 10 vectors, while the average node utilization reached about 70% (denoted as UTIL in the table).

We have measured the number of disk accesses to the nodes (denoted as I/O in the table) and the number of distance computation (denoted as COMP in the table). The costs are represented in percentage, standing for a proportion of costs needed by (a) the entire M-tree traversal, or (b) the simple sequential scan over the vector set. We measured average, minimum and maximum values of costs (for the 50 4-NN queries).

Table 1. The efficiency of search using M-tree.

k	UTIL %	4-NN queries											
		I/O			COMP			I/O			COMP		
		% of M-tree			% of M-tree			% of seq.			% of seq.		
		Avg	*Min*	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*	*Max*
15	68	**54**	17	76	**41**	11	58	**93**	29	131	**48**	13	68
50	71	**63**	29	88	**49**	20	74	**104**	48	145	**58**	24	88
250	70	**65**	24	91	**51**	16	75	**112**	41	157	**60**	19	88

Summary As we can see in the table, the best results were achieved for $k = 15$, where the half of the M-tree index was needed to access, which is equivalent to 93% of the sequential file. The number of distance computation was lower, 48% computations needed by sequential scanning. The efficiency presented result are not quite convincing, however, it must be recalled that the image database was very small (just 730 images). In the future we would like to try this method on much larger databases, say of tens or thousands images, where the potential of M-tree could be enforced in much greater extent.

5 Conclusions

In this article we have introduced specific application of LSI into the area of image retrieval, together with the usage of M-tree serving as an indexing structure for efficient retrieval. Due to the application to image databases, we have also visualized some interesting aspects of LSI, which are not directly obvious when using LSI in the original area of text retrieval.

Acknowledgments

This research has been partially supported by grants GAČR 201/05/P036 and "Information Society" 1ET100300419.

References

1. Baeza-Yates, R. A., Ribeiro-Neto B.: Modern Information Retrieval, Addison-Wesley Longman Publishing Co., Inc., 1999
2. Berry, M. W., Browne, M.: Understanding search engines: mathematical modeling and text retrieval, Society for Industrial and Applied Mathematics, 1999
3. Carson, C., Thomas, M., Belongie, S., Hellerstein, J. M., Malik, J.: Blobworld: A system for region-based image indexing and retrieval, Third International Conference on Visual Information Systems, Springer, 1999
4. Castelli, V., Bergman, L. D.: Image Databases : Search and Retrieval of Digital Imagery, Wiley-Interscience, 2002
5. Chávez, E., Navarro, G., Baeza-Yates, R., Marroquín, J. L.: Searching in metric spaces, ACM Compututing Surveys 33(1):273–321, 2001
6. Ciaccia, P., Patella, M., Zezula, P.: M-tree: An Efficient Access Method for Similarity Search in Metric Spaces, Proceedings of the 23rd Athens Intern. Conf. on VLDB, Morgan Kaufmann, 1997
7. Deb, S.: Multimedia Systems and Content-Based Image Retrieval, Information Science Publishing, 2003
8. Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., Harshman, R. A.: Indexing by Latent Semantic Analysis, Journal of the American Society of Information Science 41(6):391–407, 1990
9. Faloutsos, C., Barber, R., Flickner, M., Hafner, J., Niblack, W., Petkovic, D., Equitz, W.: Efficient and Effective Querying by Image Content, Journal of Intelligent Information Systems (JIIS) 3(3/4):231–262, 1994
10. Patella, M.: Similarity Search in Multimedia Databases, Dipartmento di Elettronica Informatica e Sistemistica, Bologna, 1999, `http://www-db.deis.unibo.it/Mtree/index.html`
11. Praks, P., Dvorský, J., Snášel, V.: Latent Semantic Indexing for Image Retrieval Systems, SIAM Conference on Applied Linear Algebra (LA03), The College of William and Mary, Williamsburg, USA, 2003
12. Praks, P., Machala, L., Snášel, V.: Iris Recognition Using the SVD-Free Latent Semantic Indexing, Workshop on Multimedia Data Mining (MDM/KDD 2004), Seattle, WA, USA, 2004
13. Rui, Y., Huang, T. S., Chang S.: Image retrieval: current techniques, promising directions and open issues, Journal of Visual Communication and Image Representation 10(4):39–62, 1999
14. Rui, Y., She, A. C., Huang, T. S.: Modified Fourier Descriptors for Shape Representation – a practical approach, First International Workshop on Image Databases and Multimedia Search, 1996
15. Seidl, T., Kriegel, H.P.: Efficient User-Adaptable Similarity Search in Large Multimedia Databases, Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB), pp. 506–515, 1997
16. Skopal, T.: Metric Indexing in Information Retrieval, Technical University of Ostrava, 2004, `http://urtax.ms.mff.cuni.cz/skopal/phd/thesis.pdf`
17. Skopal, T., Pokorný, J., Krátký, M., Snášel, V.: Revisiting M-tree Building Principles, Proceedings of the 7th East-European conference on Advances in Databases and Informations Systems (ADBIS), LNCS 2798, pp. 148–162, Springer-Verlag, Dresden, Germany, 2003
18. Tamura, H., Mori, S., Yamawaki, T.: Textual features corresponding to visual perception, IEEE Transactions on Systems, Man, and Cybernetics 8(6), 1978

19. Zezula, P., Amato, G., Dohnal, V., Batko, M.: Similarity Search – The Metric Space Approach, Springer, 2005
20. Zurich Building Image Database, `http://www.vision.ee.ethz.ch/showroom/zubud/index.en.html`

Tomáš Skopal received the B.S. and M.S. degrees in computer science from Palacky University in Olomouc, Czech Republic in 1999 (2001, respectively), and the Ph.D. degree in computer science and applied mathematics from Technical University of Ostrava, Czech Republic in 2004. He is currently an assistant professor at the Department of Software Engineering, Faculty of Mathematics and Physics, Charles University in Prague. His current research interests are similarity search in multimedia databases, database indexing, similarity modeling.

Václav Snášel received the M.S. degree in numerical mathematics from Palacky University in Olomouc, Czech Republic in 1981, and the Ph.D. degree in algebra and number theory from Masaryk University in Brno, Czech Republic in 1991. He is currently a professor at the Department of Computer Science, Faculty of Electrical Engineering and Computer Science, Technical University of Ostrava. His current research interests are multidimensional data and XML indexing, data compression, formal concept analysis, ordered sets, universal algebra.