SOM-Based R*-Tree for Similarity Retrieval

Kun-seok Oh, Yaokai Feng, Kunihiko Kaneko, Akifumi Makinouchi
Department of Intelligent Systems
Graduate School of Information Science & Electrical Engineering
Kyushu University
6-10-1 Hakozaki Higashi-ku Fukuoka, Japan 812-8581
{okseok,kaneko,akifumi}@db.is.kyushu-u.ac.jp

Sang-hyun BAE
Department of Computer Science and Statistics
College of Natural Science Chosun University Korea
375 Seosuk-dong Tong-gu Kwangju, Republic of Korea
shbae@mail.chosun.ac.kr

Abstract

Feature-based similarity retrieval has become an important research issue in multimedia database systems. The features of multimedia data are useful for discriminating between multimedia objects (e.g., documents, images, video, music score, etc.). For example, images are represented by their color histograms, texture vectors, and shape descriptors. A feature vector is a vector that represents a set of features, and are usually high-dimensional data. The performance of conventional multidimensional data structures (e.g., R-tree family, K-D-B tree, grid file, TV-tree) tends to deteriorate as the number of dimensions of feature vectors increases. The R-tree is the most successful variant of the R-tree. In this paper, we propose a SOM-based R*-tree as a new indexing method for high-dimensional feature vectors. The SOM-based R*-tree combines SOM and R*-tree to achieve search performance more scalable to high dimensionalities. Self-Organizing Maps (SOMs) provide mapping from high-dimensional feature vectors onto a two-dimensional space. The mapping preserves the topology of the feature vectors. The map is called a topological feature map, and preserves the mutual relationships (similarity) in the feature spaces of input data, clustering mutually similar feature vectors in neighboring nodes. Each node of the topological feature map holds a codebook vector. A best-matching-image-list (BMIL) holds similar images that are closest to each codebook vector. In a topological feature map, there are empty nodes in which no image is classified. When we build an R*-tree, we use codebook vectors of topological feature map which eliminates the empty nodes that cause unnecessary disk access and degrade retrieval performance. We experimentally compare the retrieval time cost of a SOM-based R*-tree with that of an SOM and an R*-tree using color feature vectors extracted from 40,000 images. The results show that the SOM-based R*-tree outperforms both the SOM and R*-tree due to the reduction of the number of nodes required to build R*-tree and retrieval time cost.*

1. Introduction

With the increasing use of new database applications for dealing with highly multidimensional data sets, technology to support effective query processing with such a data set is considered an important research area. Such applications include multimedia databases, medical databases, scientific databases, time-series matching, and data analysis/data mining. For example, in the case of image searches, a typical query of content-based image retrieval [15, 12, 39, 47, 31, 7] is "*find images with similar colors, texture, or shapes in a collection of color images*". The features used in this query are useful for discriminating between multimedia objects (e.g., documents, images, video, music score etc.). A feature vector is a vector that contains a set of features, and usually hold high-dimensional data. Many indexing techniques [1, 38, 3, 50, 42, 27, 21, 5, 8] have been proposed to access such high-dimensional feature vectors effectively, referred to as high-dimensional index trees. These index trees work effectively in low to medium dimensionality space (up to 20-30 dimensions). However,

0-7695-0996-7/01 $10.00 © 2001 IEEE

even a simple sequential scan performs better at higher dimensionalities [8].

In this paper, we propose a SOM-based R*-tree as a new indexing method for high-dimensional feature vectors. The SOM-based R*-tree combines a Self-Organizing Map (SOM) [23, 44] and an R*-tree [3] to achieve search performance more scalable to high dimensionalities. In this paper, we consider a similarity search for image data. SOM, which is a kind of neural network, provides mapping from a data set of high-dimensional feature vectors onto a usually two-dimensional space. The mapping preserves the topology of the feature vector, and is called a *topological feature map*. The vectors contained in each node of the topological feature map are usually called *codebook vectors*.

In spite of all the benefits of SOM described above, its application has been limited by some drawbacks in terms of the interpretability of a trained SOM and the complexity of search time for a best-matching node. A best-matching-image-list (BMIL) holds similar images that are closest to each codebook vector. Whereas more than one similar image is classified in each node of the topological feature map, there are empty nodes in which no image is classified. These empty nodes result in unnecessary disk access, thus degrading retrieval performance. When we build an R*-tree, we use codebook vectors of the topological feature map which eliminates the empty nodes and allows as to build an index using fewer nodes than the number of nodes in the SOM. Therefore, using our similarity search technique, we expect the search time to be faster than the search time of SOM.

This paper is organized as follows: In Section 2, we provide an overview of related work. In Section 3, we present the algorithm of the SOM and R*-tree, and describe the SOM-based R*-tree proposed in this research. We experiment in order to compare the SOM-based R*-tree with the SOM and R*-tree alone in terms of retrieval time cost using color feature vectors extracted from 40,000 images. The experimental results are discussed in Section 4, and Section 5 presents the concluding remarks.

2. Related Works

In this Section, we describe the related work on clustering methods and high-dimensional index structures.

Clustering There are supervised and unsupervised clustering methods for clustering similar data [10]. During the training phase in supervised clustering, both the input data and the desired output are presented to the neural network. If the output of the neural network differs from the desired output, the internal weights of the neural network are adjusted. In unsupervised clustering, the neural network is given only the input vectors and the neural network is used to create abstractions in the input space.

SOM is an unsupervised self-organizing neural network that is widely used to visualize and interpret large high-dimensional datasets [36, 24, 22, 9, 19, 20]. In this study, the reasons for using SOM are as follows: *(i)* No prior assumption is needed for distribution of data, *(ii)* the learning algorithm is simple, *(iii)* we do not need an external supervised signal for input and it learns self-organizationally, and *(iv)* similarity can be found automatically from multidimensional feature vector, and similar feature vectors are mapped onto neighboring regions on the topological feature map, in particular, highly similar feature vectors are mapped on the same node.

High-dimensional index structure Many techniques, including R-tree [1], R^+-tree [38], SS-tree [50], SR-tree [21], X-tree [42], TV-tree [27], and Hybrid tree [8] have been proposed to index feature vectors for similarity search. Despite various attempts at accessing high-dimensional feature vectors effectively, the current solutions are far from satisfactory [18]. Although these index structures can scale to medium dimensionalities, above a certain dimensionality they are outperformed by a simple sequential scan through the database. This occurs because the data space becomes sparse at high dimensionalities, causing the bounding regions to become large [8]. We selected the R*-tree from among other index techniques for the following reasons: *(i)* This index structure is the most successful variant of the R-tree, *(ii)* it can be applied to spatial data such as geography and CAD data, and *(iii)* it can be used as an index structure for feature space such as image retrieval currently.

In order to realize an SOM-based R*-Tree, we built an R*-tree using a topological feature map and a BMIL by learning the SOM. The use of an SOM-based R*-tree avoids unnecessary disk access during searching by eliminating the empty nodes on the topological feature map. In addition, our proposed technique can build R*-tree using fewer nodes than the number of nodes in SOM. Therefore, the number of node to visit decreases, and search time is shortened. To the best of our knowledge, there has not been until now notable indexing techniques like our method.

3. SOM-Based R*-Tree

3.1. Self-Organizing Maps

Self-Organizing Maps (SOMs) are unsupervised neural networks that provide mapping from high-dimensional input space to a usually two-dimensional regular grid while preserving topological relations as faithfully as possible. The SOM consists of a set of i nodes arranged in a two-dimensional grid, with a weight vector $m_i \in \Re^n$ attached to each node. Elements from the high-dimensional input

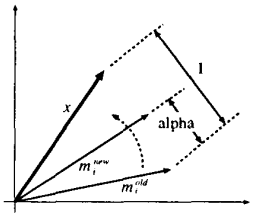

Figure 1. Update SOM. x is input vector which is feature vector in this paper. m_i^{old} and m_i^{new} represent before- and after-modification of weight vector respectively.

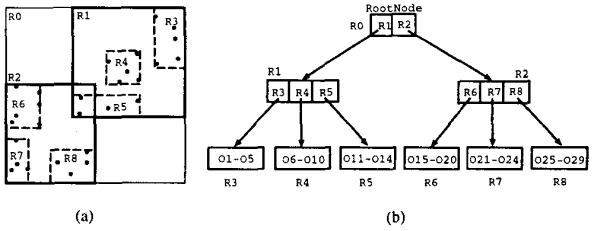

Figure 2. An example of R*-tree structure; (a) space of point data, (b) tree access structure.

space, referred to as input vector $x \in \Re^n$, are presented to the SOM and the best-match-node (BMN) for the presented input vector is calculated using the Euclidean distance between the weight vector of the node and the input vector. In the next step, the weight vector of the BMN (i.e. the node with the smallest Euclidean distance) is selected as the 'winner' and is modified so as to more closely resemble the presented input vector. The weight vector of the winner is moved towards the presented input vector by a certain fraction of the Euclidean distance as indicated by a time-decreasing learning rate $\alpha(t)$, shown in Figure 1, where t=1,2,3,... is an integer representing the discrete-time coordinate. $\alpha(t)(0 < \alpha(t) < 1)$ is a parameter adjusting what extent close to the input vector x. If α=1, m_i is completely equal to x. $\alpha(t)$ begins with a value close to unity, thereafter decreasing monotonically. Thus, this BMN will be even higher the next time the same input vector is presented. Furthermore, the weight vectors of nodes in the neighborhood of the winner, described by a time-decreasing neighborhood function $\sigma(t)$, are modified accordingly, although to a lesser than the winner. This learning procedure finally leads to a topologically-ordered mapping of the presented input vectors. Similar input data is mapped onto neighboring regions on the map [23]. The map is called a *topological feature map*, and a weight vector held by a node in the topological feature map is called a *codebook vector*. The topological feature map preserves the mutual relationships (similarity) of the input data in feature spaces, and clusters mutually similar feature vectors in neighboring nodes. During the training phase, the input vectors become ordered on the grid such that similar input vectors are close to each other and dissimilar input vectors are far apart. Despite these advantages, SOM applications have been limited. The search for the BMN dominates the computing time of the SOM algorithm, making it computationally expensive for high input dimensionalities or large SOM networks. The basic algorithm uses full search, where all the nodes must be considered to find the BMN, increasing the complexity of the search.

3.2. R*-Tree

In the field of database systems, various index structures (see Section 2) have been proposed recently as multidimensional indexing techniques. As one of these techniques, the R*-tree improves the performance of the R-tree by modifying the insertion and split algorithms and by introducing the forced reinsertions mechanism [3]. The R*-tree is proposed as an index structure for spatial data such as geographical and CAD data. Currently, it is used as an index structure of feature space, such as for image retrieval [13]. The R*-tree, the most successful variant of the R-tree, is a multidimensional index structure for rectangular data, it is a height-balanced tree corresponding to a hierarchy of nested rectangles. Nodes and leaves correspond to rectangles in the hierarchy and a disk page is allocated for each. Each internal node contains an array of (p, μ) entries, where p is a pointer to one child node of this internal node, and μ is the minimum bounding rectangle (MBR) of the child node pointed to by the pointer p. Each leaf node contains an array of (OID, μ) for spatial objects, where OID is an object identifier, and μ is the MBR of the object identified by OID. Therefore, the rectangle of the root node corresponds to the MBR of all the data entries, while the rectangle of an internal node corresponds to the MBR of the data entries contained in its lower leaves. The regions of the R*-tree are allowed to overlap each other, and because sibling regions can overlap each other, the search time for a point query depends on the amount of overlap and not the height of the tree.

The number of entries in each node is called a *fanout*. The fanout of a root node is at least 2 unless it is a leaf. The fanout of the other nodes is between m and M, where $2 < m < M/2$. The M value of leaf nodes may be different from that of internal nodes when objects are point data. Figure 2

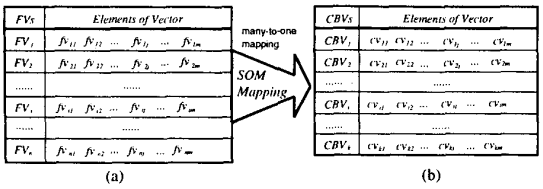

Figure 3. Relationship between feature vector and codebook vector; (a) Feature vectors extracted from images, (b) Codebook vectors generated by SOM.

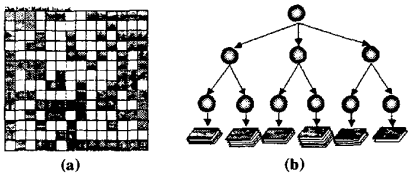

Figure 4. SOM-based R*-tree structure;(a) example of topological feature map, empty grids refer to empty nodes, (b) SOM-based R*-tree structure using topological feature map after eliminating empty nodes.

shows an example of an R*-tree.

3.3. SOM-Based R*-Tree

The construction of a SOM-based R*-tree consists of two processes; *clustering similar images* and *construction of R*-tree*, as follows:

Clustering similar images We first generate the topological feature map using the SOM. We generate the BMIL by computing the distances between the feature vectors and codebook vectors from the topological feature map. In order to generate topological feature map, given a learning parameter (e.g., learning-rate, neighborhood radius, size of map layer, learning iteration), we perform SOM learning. The SOM is trained iteratively. At each training step, a feature vector is sequentially chosen from the input data set. The distance between the feature vector and all weight vectors are computed. The BMN (node with minimum distance) is chosen from the map nodes. Next, the weight vectors are updated. The BMN and its topological neighbors are moved closer to the input feature vector. As a result of learning, the vector, which is generated on each node of the map, is called a codebook vector, and is represent by

$$\boldsymbol{CBV_i} = [cv_{i1}, cv_{i2}, ..., cv_{ij}, ..., cv_{im}]^T,$$

where $i(1 \leq i \leq k)$ is the node number of the map, m is the number of input nodes, i.e., the dimensionality of the feature vector, and k is the number of map nodes.

Using the topological feature map, we classify similar images to the nearest node, which has the minimum distance between a given feature vector and all codebook vectors. This classified similar image of each node is called the best-matching-image-list (BMIL). Similarity between feature vectors and codebook vectors is calculated by the Euclidean distance. Best-match-node BMN_i is

$$BMN_i = \min_i \{||\boldsymbol{FV} - \boldsymbol{CBV_i}||\},$$

where $\boldsymbol{FV}$ is a feature vector. The relationship between feature vectors and codebook vectors is shown in Figure 3. Between these two kinds of vectors, there are many-to-one relationships based on the similarity between each feature vector. In an ideal situation, there should be a one-to-one correspondence between the feature vectors of images and the codebook vectors of SOM nodes in the map. This is not, however, generally the case and some map nodes still hold multiple images [26]. This means that empty nodes occur in a topological feature map when the BMIL is generated. Empty nodes refer to the portion of the node (vector) spaces that contains no matching feature vectors. Empty node indexing causes unnecessary disk access, thus degrading search performance. The space requirement can be reduced by indexing only live nodes (in contrast to empty nodes).

Construction of R*-tree In the construction of a SOM-based R*-tree, we use the R*-tree algorithm [3]. Let one point on the n-dimensional space correspond to each codebook vector of the topological feature map, and the space covering all codebook vectors corresponds to the root node. In order to construct the R*-tree, we select a codebook vector from the topological feature map as an entry. If it is an empty node, we select the next codebook vector. Otherwise, determine the leaf node which insert codebook vector. To determine the most suitable to accommodate the new entry, i.e., codebook vector by choosing a subtree whose centroid is the nearest to the new entry. When a node or a leaf has space, the entry is added, otherwise, perform reinsertions or split algorithm. A leaf of the SOM-based R*-tree has the following structure:

$$L : (E_1, ..., E_i, ..., E_p) \quad (m \leq p \leq M)$$
$$E_i : (OID, \mu)$$

A leaf L consists of entries $E_1, ..., E_i, ..., E_p (m \leq p \leq M)$, where m and M are the minimum and the maximum

number of entries in a leaf. Each entry contains an OID and its MBR μ. The node structure of the SOM-based R*-tree is the same as that of the R*-tree (see 3.2) as shown in Figure 4.

4. Experiments

We performed experiments to compare the SOM-based R*-tree with a normal SOM and R*-tree. Our image database contains still color images. The experimental image database currently consists of 40,000 artificial/natural images, including landscapes, animals, buildings, people, plants, CG, etc., from H^{2}soft[1] and Stanford University[2]. We fixed the image size at 128×128 pixels. All experiments were performed on a COMPAQ DESKPRO (OS: FreeBSD 3.4-STABLE) with 128 MBytes of memory, and all data was stored on its local disk.

4.1. Experimental Methodology

Feature Extraction In this study, we extract color features from the image data, and use it in the experiments. To compute feature vectors, we use Haar wavelets [28], which are a kind of wavelet transform. Haar wavelets provide the fastest computations and have been found to perform well in practice [7]. One disadvantage of using Haar wavelets is that the computation tends to produce blocky image artifacts in the most important subbands. However, this drawback does not noticeably affect similarity retrieval [31].

The color space used in this paper for feature vectors is the YIQ-space (NTSC transmission primaries) [40] with luminance and chrominance information. We computed 5-level two-dimensional wavelet transforms for each of the three color spaces using Haar wavelets. Extracting the lowest submatrix for the color feature, we generated this submatrix as part of the feature vector. Each element of this feature vector represents an average of 32×32 pixels of the original image. The color feature vector has 48 dimensions (=4×4×3, where 3 is the three channels of YIQ-space).

Construction of SOM-based R*-tree As shown in Table 1, the map size is almost the same as the number of images. We generated the topological feature map using color feature vectors via the learning of the SOM, and the BMIL is generated using this feature map. The empty nodes occupied 53% to 60% of the original map size. As the map size becomes larger, the number of empty nodes increases. The existence of empty nodes indecates that images of high similarity are classified in the same node, regardless of map size. Therefore, reducing the number of nodes and speeding up search time can be realized by eliminating empty nodes; an R*-tree built with this pruned set of nodes will have a smaller overall index size.

[1] H^{2}soft, http://www.h2soft.co.jp.

[2] Stanford University, http://WWW-DB.Stanford.EDU/IMAGE/.

Table 1. Map size vs. empty nodes

Data set	Map size	Empty nodes	Ratio (%)
1000	32 × 32	551	53
5000	70 × 70	2634	53
10000	100 × 100	5395	54
20000	140 × 140	10819	55
30000	175 × 175	18377	60
40000	200 × 200	23706	59

Table 2 compares the structure of a normal R*-tree with that of the SOM-based R*-tree for each data set. The height of the tree is not that different, however both the total number of nodes and the time cost of building the index decrease. These observations reduce memory usage and retrieval access time. The larger the data set, the more efficient the index, as can be clearly seen in Table 3.

4.2. Experimental Results

To measure search time, we experimented with four types of searches; search for *(i)* normal SOM including empty nodes, *(ii)* normal SOM with eliminated empty nodes, *(iii)* normal R*-tree, and *(iv)* SOM-based R*-tree with eliminated empty nodes. The data set size was from 1,000 to 40,000 images. The search method used was the k-Nearest Neighbor(NN) [37] method, which searches for $k(k > 1)$ objects nearest to the given query. In SOM, an exhaustive search of the topological feature map is performed, and finding $k(k$=10) nodes nearest to the given query. In the same manner, the normal R*-tree and SOM-based R*-tree are applied using the k-NN (k=10) search.

A comparison of retrieval time cost is shown in Figures 5 and 6. In both figures, the horizontal axis is the dataset size. As shown in Figure 5, the retrieval time of SOM with empty nodes, as compared to the SOM without empty nodes, grows drastically as the dataset size increases, over 5 times the retrieval time cost at 40,000 images. Thus, eliminating empty nodes significantly reduces retrieval time by removing unnecessary distance computations.

We also compared the performance of the SOM-based R*-tree with that of the R*-tree based on 10-NN retrieval time cost, as shown in Figure 6. In this comparison, the nearest OID was obtained for a given query. The retrieval time of the SOM-based R*-tree is far shorter compared to the R*-tree, by 3 to 15 times. The results show that building

Table 2. Tree Structure

		Data set (×1000)					
		1	5	10	20	30	40
Total No. of nodes	R*-tree	119	499	972	2089	3042	3980
	SOM-based R*-tree	51	241	483	928	1300	1705
Height of tree	R*-tree	3	4	4	5	5	5
	SOM-based R*-tree	3	4	4	4	4	5
Time cost (sec)	R*-tree	7.55	50.44	111.89	233.89	350.66	476.49
	SOM-based R*-tree	3.27	19.12	40.76	81.95	115.49	153.92

Table 3. R*-tree file size (real data)

	Data set (×1000)					
	1	5	10	20	30	40
Original feature vector (MB)	1.38	5.78	11.27	24.22	35.26	46.13
SOM-based R*-tree (MB)	0.59	2.79	5.59	10.75	15.07	19.76
Ratio (%)	42	48	49	44	42	42

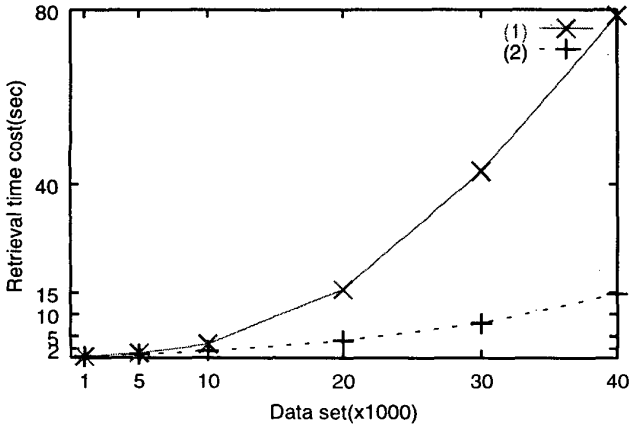

Figure 5. Retrieval time cost. (1) retrieval from SOM with empty nodes, and (2) retrieval from SOM without empty nodes.

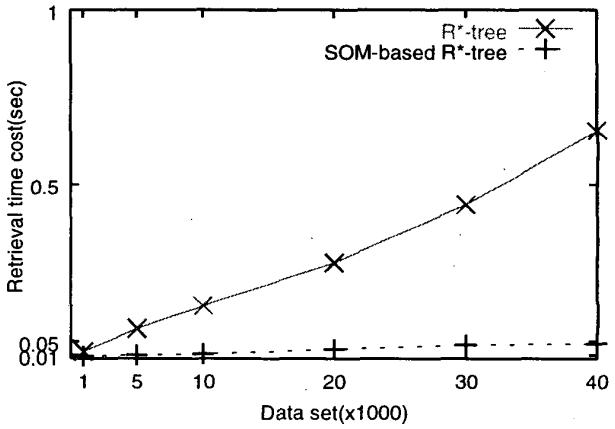

Figure 6. Comparison of retrieval time cost between SOM-based R*-tree and R*-tree.

the R*-tree with overall original feature vectors improves retrieval performance. Furthermore, the SOM-based R*-tree performs much better than SOM alone, which sequentially searches feature vectors. These experimental results clearly show that a SOM-based R*-tree is more efficient for similarity retrieval.

5. Conclusions

In this study, we proposed a SOM-based R*-tree for dealing with similarity retrieval from high-dimensional data sets. Using a topological feature map and a best-matching-image-list (BMIL) obtained via the learning of a SOM, we constructed an R*-Tree. The major finding of this study is that building an R*-tree in which the empty nodes in the

topological feature map are removed yields an R*-tree with fewer nodes, thus enhancing performance by decreasing unnecessary access, node visits, and overall search times.

In an experiment, we performed a similarity search using real image data and compared the performance of the SOM-based R*-tree with a normal SOM and R*-tree, based on retrieval time cost. The R*-tree with fewer nodes experimentally verified to shorter search time, and search efficiency was improved due to the use of a k-NN search, compared to SOM.

Acknowledgments.

This work was supported in part by a Grant-in-Aid for Scientific Research (10308012) from the Ministry of Education, Science, Sports and Culture of Japan.

References

[1] A.Guttman. R-tree: a dynamic index structure for spatial searching. In *Proc. of ACM SIGMOD Int. Conf. on Management of Data*, pages 45–57, 1984.

[2] E. Albuz, E. Kocalar, and A. A. Khokhar. Scalable image indexing and retrieval using wavelets. Technical report, SCAPAL Technical Report, 1998.

[3] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger. R*-tree: an efficient and robust access method for points and rectangles. In *Proc. of ACM SIGMOD Int. Conf. on Management of Data*, pages 322–331, Atlantic City, NJ, May 1990.

[4] R. Bellman. *Adaptive Control Processes: A Guided Tour.* Princeton University Press, 1961.

[5] S. Berchtold, C. Bohm, and H.-P. Kriegal. The pyramid-technique: towards breaking the curse of dimensionality. In *Proc. of ACM SIGMOD int. conf. on Management of data*, pages 142–153, Seattle, WA USA, June 1998.

[6] K. Beyer, J. Goldstein, R. Ramakrishnan, and U.Shaft. When is "nearest neighbor" meaningful? In *ICDT*, pages 217–235, 1999.

[7] A. F. C. E. Jacobs and D. H. Salesin. Fast Multiresolution Image Querying. In *Proc. SIGGRAPH95*, pages 6–11, New York, August 1995. ACM SIGGRAPH.

[8] K. Chakrabarti and S. Mehrotra. High dimensional feature indexing using hybrid trees. In *Proc. of ICDE1999*, March 1999.

[9] G. Deboeck and T. Kohonen. *Visual Explorations in Finance with Self-Organizing Maps.* Springer-Verlag, London, 1998.

[10] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons, 1973.

[11] T. K. et.al. Organization of a massive document collection. *IEEE Transaction on NEURAL NETWORK*, 11(3):574–585, May 2000.

[12] C. Faloutsos, W. Equitz, M. Flickner, W. Niblack, D. Petkovic, and R. Barber. Efficient and Effective Query by Image Content. *J. of Intell. Inform. Syst.*, 3:231–262, 1994.

[13] M. Flickner, H. Sawhney, W. Niblack, 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*, 28(9):23–32, September 1995.

[14] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proc. of VLDB'99*, pages 518–529, Edinburgh, Scotland, 1999.

[15] V. N. Gudivada and V. V. Raghavan. Content-based Image Retrieval system. *IEEE Computer*, 28(9):18–22, September 1995.

[16] G. R. Hjaltason and H. Samet. Distance browsing in spatial databases. In *ACM Transactions on Database Systems*, volume 24, pages 265–318, 1999.

[17] S.-H. S. Huang and C. K. Wong. Binary search trees with local rotation. In *Proc. 20th Annual Allerton Conference on Communication, Control, and Computing*, pages 481–489, Monticello Illinois, USA, 1982.

[18] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. of STOC*, 1998.

[19] S. Kaski. Dimensionality reduction by random mapping: Fast similarity computation for clustering. In *Proc. of IJCNN'98*, pages 413–418, Piscataway, NJ, 1998.

[20] S, Kaski. Fast Winner Search for SOM-Based Monitoring and Retrieval of High-Dimensional Data. In *Proc. of ICANN'99*, Edinburge, UK, September 1999.

[21] N. Katayama and S. Satoh. The sr-tree: An index structure for high-dimensional nearest neighbor queries. In *Proc. of the 1997 ACM SIGMOD Int. Conf. on Management of Data*, pages 369–380, May 1997.

[22] T. Kohonen. Exploration of very large databases by self-organizing maps. In *Proc. of ICNN'97*, pages PL1–PL6, Piscataway, NJ, 1997. IEEE Service Center.

[23] T. Kohonen. *Self-Organizing Maps.* Springer, Berlin, 1997.

[24] T. Kohonen. Self-Organization of Very Large Document Collections:State of the Art. In *Proc. of ICNN98*, volume 1, pages 65–74, London, UK, 1998. Springer.

[25] T. Kohonen, J. Hynninen, and J. Laaksonen. SOM_PAK:the Self-Organizing Map Program Package. Technical Report A31, Helsinki University of Technology, Laboratory of Computer and Information science, Finland, January 1996.

[26] M. Koskelar. Content-Based Images Retrieval with Self-Organizing Maps. Master's thesis, Helsinki University of Technology, Department of Engineering Physics and Mathematics, 1999.

[27] K.-I. Lin, H. Jagadish, and C. Faloutsos. The tv-tree: An index structure for high-dimensional data. In *VLDB Journal*, pages 517–542, October 1994.

[28] S. G. Mallat. Multifrequency Channel Decompositions of Images and Wavelet Models. *IEEE. Trans., Acoust.,Speech and Signal Proc.*, 37(12):2091–2110, December 1989.

[29] M. Minsky and S. Papert. *Perceptrons.* MIT Press, 1969.

[30] C. Nastar, M. Mitschke, and C. Meilhac. Efficient query refinement for image retrieval. In *Proc. of CVPR '98*, Santa Barbara, California, 1998.

[31] A. Natsev, R. Rastogi, and K. Shim. WALRUS: A Similarity Retrieval Algorithm for Image Databases. In *Proc. ACM SIGMOD International Conference on Management*

of Data, pages 396–406, Philadephia, PA, June 1999. ACM SIGMOD.

[32] K.-S. Oh, K. Kaneko, and A. Makinouchi. Image classification and retrieval based on wavelete-som. In *DANTE'99*, pages 164–167, Kyoto, Japan, 1999.

[33] K.-S. Oh, K. Kaneko, A. Makinouchi, and A. Ueno. Design, implementation and performance evaluation of similar image retrieval system based on self-organizing feature maps. *Technical Report of IEICE(DE2000)*, 100(31):9–16, 2000.

[34] E. Oja and S. Kaski. *KOHONEN MAPS*. ELSEVIER, Amsterdam, The Netherlands, 1999.

[35] A. Pentland, R. W. Picard, and S. Schlaroff. Photobook: Content-Based Manupulation of Image Databases. In *Storage and Retrieval for Image and Video Databases II*, San Joes, 1995. SPIE.

[36] A. Rauber. LabelSOM:On the Labeling of Self-Organizing Maps. In *Proc. of IJCNN'99*, Washington DC, July 1999.

[37] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest neighbor queries. In *Proc. of the ACM SIGMOD*, pages 71–79, San Jose CA, May 1995.

[38] T. S. N. Roussopoulos and C. Faloutsos. The r+-tree: A dynamic index for multi-dimensional objects. In *Proc. VLDB'87*, pages 507–518, Brighton, September 1987.

[39] Y. Rui, T. Huang, and S.-F. Chang. Image Retrieval:Current Techniques, Promising Directions, and Open Issues. *J. Visual Communication and Image Representation(JVCIR)*, 10(1):39–62, 1999.

[40] J. C. Russ. *The Image Processing Handbook*. CRC Press, Boca Raton, 1995.

[41] S. Santini and R. Jain. Similarity Matching. In *Proc. ACCV 95, Asian Conference on Computer Vision*, 1995.

[42] S.Berchtold, D.A.Keim, and H.-P.Kriegel. The x-tree: An index structure for high-dimensional data. In *Proc. VLDB'96*, 1996.

[43] J. R. Smith and S. F. Chang. VisualSEEK:A Fully Automated Content-Based Image Query System. In *Proc. ACM Multimedia Conference*, pages 87–98, Boston, November 1996.

[44] T.Kohonen. Self-organizing maps. *Proc.of The IEEE*, 78(9):1464–1480, 1990.

[45] A. Tversky. Features of Similarity. *Psychological review*, 84(4):327–352, July 1977.

[46] J. Vesanto and E. Alhoniemi. Clustering of the self-organizing map. *IEEE Transaction on NEURAL NETWORK*, 11(3):586–600, May 2000.

[47] J. Z. Wang, G. Wiederhold, O. Firschein, and S. X. Wei. Content-based Image Indexing and Searching Using Daubechies' Wavelets. *International Journal of Digital Libraries (IJODL)*, 1(4):311–328, April 1998.

[48] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proc. of the 24th VLDB Conf.*, New York, USA, September 1998.

[49] D. A. White and R. Jain. Similarity indexing: Algorithms and performance. In *Proc. of the SPIE*, volume 2670, pages 62–73, San Jose CA, 1996.

[50] D. A. White and R. Jain. Similarity indexing with the ss-tree. In *Proc. of ICDE'96*, pages 516–523, New Orleans, USA, February 1996.

[51] J. K. Wu. Content-Based Indexing of Multimedia Databases. *IEEE Trans. Knowledge and Data Eng.*, 9(6):978–989, June 1997.

[52] W.Y.Ma and B. S. Manjunath. Image Indexing Using a Texture Dictionary. In *Proc. of SPIE conf. on Image Storage and Archiving System*, volume 2606, pages 288–298, Philadelphia, Pennsylvania, October 1995.