Buoy Indexing of Metric Feature Spaces for Fast Approximate Image Queries

Stephan Volmer

Fraunhofer IGD, Rundeturmstr. 6, 64283 Darmstadt, Germany
stephan.volmer@igd.fhg.de

Abstract. A novel indexing scheme for solving the problem of nearest neighbor queries in generic metric feature spaces for content-based image retrieval is proposed to break the "dimensionality curse." The basis for the proposed method is the partitioning of the feature dataset into clusters that are represented by single buoys. Upon submission of a query request, only a small number of clusters whose buoys are close to the query object are considered for the approximate query result, effectively cutting down the amount of data to be processed enormously. Results concerning the retrieval accuracy from extensive experimentation with a real image archive are given. The influence of control parameters is investigated with respect to the tradeoff between retrieval accuracy and computational cost.

1 Introduction

Interest in digital images has increased enormously over the last few years, fuelled at least in part by the ubiquity of digital media, the availability of large image archives, and the rapid growth of the Internet infrastructure. Users in many professional fields exploit the opportunities offered by the ability to access and manipulate remotely stored images in all kinds of new and exciting ways. Finding an image whose content is truly relevant to the user's need has become the focal point of recent research in information technology.

Problems with traditional methods of image retrieval [5] have led to the rise of techniques for retrieving images on the basis of content descriptors for perceptual features such as color, texture, shape, structure, and spatial relationship – a technology now generally referred to as *content-based image retrieval* (CBIR). CBIR systems employ unsupervised automatic feature extraction algorithms on images by analyzing their pixel distributions. This analysis results in compact feature descriptors, which convey specific aspects of the image's most salient visual properties [8, 10, 13].

Upon presentation of a query, the feature descriptor of the query image is compared with all corresponding descriptors in the database by some well-defined similarity measure. A sequential search through all potential descriptors contained in the database would be very time-consuming and inefficient. Therefore, an indexing scheme becomes necessary in order to limit the number of potential target descriptors from the database and reduce the computational effort needed to sequentially determine their similarity to the query descriptor. This task is generally referred to as *similarity indexing* [14]. The goal of similarity indexing is to reduce the amount of data to be processed by categorizing or grouping similar objects together.

2 Preliminaries

In this paper, we focus on providing a general purpose spatial indexing scheme applicable to any feature derived from images that complies with the postulates of the *metric feature model*. The common framework for the metric feature model is defined in the remainder of this section.

2.1 Metric Feature Model

The metric feature model is based on the assumption that human similarity perception corresponds with a measurement of an appropriate distance between *features* that model the images' characteristic properties.

Let Δ be a *feature extraction algorithm* that transforms images $\boldsymbol{I}$ into compact *feature descriptors* ω:

$$\boldsymbol{I} \stackrel{\Delta}{\longmapsto} \omega \tag{1}$$

Then (Ω, δ) is called a generic *feature space*, where Ω is a – possibly infinite – set called the *feature domain* whose elements are feature descriptors ω, and δ is a metric on Ω called the *dissimilarity measure*. The metric δ must satisfy the following properties:

Positivity

$$\delta(\omega_i, \omega_j) \geq 0 \tag{2}$$

Self-similarity

$$\delta(\omega_i, \omega_i) = 0 \tag{3}$$

Symmetry

$$\delta(\omega_i, \omega_j) = \delta(\omega_j, \omega_i) \tag{4}$$

Triangle Inequality

$$\delta(\omega_i, \omega_j) + \delta(\omega_j, \omega_l) \geq \delta(\omega_i, \omega_l) \tag{5}$$

It is assumed that the definition of the feature extraction algorithm Δ and its associated dissimilarity measure δ is only based on pre-attentive human similarity. This means they depend only on the perceived stimuli of the visual content of the images, without accounting for any previous knowledge, interpretation or reasoning.

This common framework includes the definition of ubiquitous d-dimensional feature vector spaces ($\Omega \equiv \mathbb{R}^d$), but is not necessarily limited to them.

2.2 Feature Dataset

Let

$$\mathcal{S} = \{\, \omega_1, \omega_2, \ldots, \omega_N \} \tag{6}$$

be a finite subset of a feature domain Ω called the *feature dataset* whose elements are the feature descriptors from a set of N images.

2.3 K-Nearest Neighbor Query

By far the most common query of a CBIR system is a request like "find the first K images most similar to the query example." Such a request can be formulated as a K-*nearest neighbor query* (K-NN query) in metric space: Given an query object $\omega_Q \in \Omega$ and an integer $K \geq 1$, the K-NN query $\mathcal{S}_{NN}(\omega_Q, K)$ selects the K elements from the feature dataset $\mathcal{S}$ which have the smallest distance from ω_Q with the following properties:

$$\begin{aligned} &\text{(i)} \quad \mathcal{S}_{NN}(\omega_Q, K) \subset \mathcal{S} \\ &\text{(ii)} \quad |\mathcal{S}_{NN}(\omega_Q, K)| = K \\ &\text{(iii)} \; \forall\, \omega \in \mathcal{S}_{NN}(\omega_Q, K) \;\; \nexists\, \omega' \in \mathcal{S} \setminus \mathcal{S}_{NN}(\omega_Q, K) \\ &\qquad \text{with} \;\; \delta(\omega_Q, \omega') < \delta(\omega_Q, \omega) \end{aligned} \tag{7}$$

3 State-Of-The-Art

The history of recent research on similarity indexing techniques can be traced back to the middle 70's when hierarchical tree structures (e.g. k-d tree) for indexing multi-dimensional vector spaces were first introduced. In 1984, Guttman proposed the R-tree indexing structure [7], which was the basis for the development of many other variants. Sellis et al. proposed the R^+-tree [12], and Beckman et al. proposed the best dynamic R-tree variant, the R^*-tree [2] in the following years.

A very extensive review and comparison of various spatial indexing techniques for feature vector spaces can be found in [14]. Motivated by k-d tree and R-tree, they proposed the VAM k-d tree and the VAMSplit R-tree. Experimentally, they found that the VAMSplit R-tree provided the best performance, however, this was at the loss of the dynamic nature of the R-tree.

Common to all of the cited research is the idea that feature descriptors are stored at the leaf level of a hierarchical index tree structure. Each leaf corresponds to a partition of the feature space and each node to a convex subspace spanning the partitions created by its children. During a similarity query, the search space is reduced by pruning tree branches at nodes that do not meet certain distance requirements. The main problem with this approach is that it requires the calculation of the minimum distance from the query point to the arbitrarily shaped convex subspace represented by the node being examined. The most common approach for simplifying this problem is that partitions are split into sub-partitions along a single axis of the vector space, ultimately creating hyper-rectangular partitions whose sides are aligned parallel to the spanning axes of the underlying feature space.

Most of the hierarchical spatial indexing methods work satisfactorily for lower dimensions, but suffer from the dimensionality curse [11] when applied to feature vectors in medium- or high-dimensional feature spaces ($d > 20$). The dimensionality curse is strictly related to the distribution of the dissimilarity measures between the feature dataset and the query object. If the variance of the dissimilarities for a given query object is low, then conducting an indexed K-NN query becomes a difficult task. A way to

obviate this situation is to conduct queries that come up with an approximate solution of the K-NN query problem [1, 3].

In recent research, there have been many attempts to get a grip on the problem of the dimensionality curse – one of them is the reduction of the dimensionality of the underlying feature domain with a principal component analysis (PCA) or its variants. In [9], Ng and Sedighain followed this approach to reduce the dimensionality, and in [6] Faloutsos and Lin proposed a fast approximation of the Karhunen-Loeve Transform (KLT) to perform the dimension reduction. However, even though experimental results from their research showed that some real feature datasets can be considerably reduced in dimension without significant degradation in retrieval quality, the image queries become proportionally less accurate with the loss of dimensions.

The biggest shortcoming of the techniques mentioned is that they are inherently only applicable to feature vector spaces, that is, each feature descriptor can be suitably represented by an adequate vector of fixed dimensionality.

In the following section an indexing scheme is proposed that is applicable to feature domains beyond the traditional vector spaces and enables fast approximate similarity queries through a very simplistic indexing paradigm.

4 Buoy Indexing

The proposed indexing scheme is based on the idea that the feature dataset is decomposed into disjoint non-empty partitions of arbitrary convex shape. However, each partition is NOT represented by a complex description of its extension or its boundaries in the feature domain but rather by a single prototype element that is an element of the feature domain itself. The prototype element serves as a *buoy* for its associated partition. The membership of an element of the feature dataset to a specific partition is solely determined by its metric distances to all buoys placed in feature space – a feature descriptor exclusively belongs to the partition with the closest associated buoy in the feature space. Ideally, each partition should have the same number of feature descriptors as members, and the number of partitions should be an order of magnitude smaller than the number of feature descriptors in the dataset. The buoys for a specific dataset are stored in a simple list or in a more complex hierarchical structure (e.g. M-tree [4]).

In general, the task of partitioning a particular feature dataset $\mathcal{S}$ into k disjoint non-empty subsets $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k$ (also called *clusters*)

$$\bigcup_{i=1}^{k} \mathcal{S}_i = \mathcal{S} \tag{8}$$

$$\mathcal{S}_i \neq \emptyset \quad \forall \quad 1 \leq i \leq k \qquad \wedge \qquad \mathcal{S}_i \cap \mathcal{S}_j = \emptyset \quad \forall \quad 1 \leq i, j \leq k, i \neq j \tag{9}$$

is performed by any k-clustering algorithm. The number of clusters k is assumed to be fixed and each descriptor of the feature dataset belongs to exactly one cluster (crisp membership). By far the most common type of k-clustering algorithm is the *optimization algorithm*.

The optimization algorithm defines a cost criterion

$$c : \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k\} \rightarrow \mathbb{R}_0^+ \tag{10}$$

which associates a non-negative cost with each cluster. The goal of the optimization algorithm is then to minimize the global cost

$$c\left(\mathcal{S}\right) = \sum_{i=1}^{k} c\left(\mathcal{S}_i\right) \tag{11}$$

for a given feature dataset.

If each cluster $\mathcal{S}_i$ is represented by a buoy $\hat{\omega}_i$ that is an element of the feature domain Ω itself, then, the cost criterion of a cluster can be defined as

$$c(\mathcal{S}_i) = \sum_{m=1}^{|\mathcal{S}_i|} \delta(\hat{\omega}_i, \omega_{im}) \tag{12}$$

where ω_{im} is the mth element of $\mathcal{S}_i$, and $|\mathcal{S}_i|$ is the number of elements in $\mathcal{S}_i$.

Naturally, the centroid of the cluster would be chosen to be the buoy $\hat{\omega}_i$ (*k-means clustering algorithm*). However, since many types of dataset do not belong to feature spaces in which the mean is defined (the mean of two elements of the feature domain is required to be an element of the feature domain itself – this is NOT always the case for feature spaces that are not vector spaces), a different buoy for clusters must be chosen for a more general type of feature space (see Sect. 2). Intuitively, the median of each cluster is selected as its representative buoy (*k-medians clustering algorithm*). Note that $\hat{\omega}_i \in \mathcal{S}_i \subset \mathcal{S} \subset \Omega$ and that $\hat{\omega}_i$ is chosen to minimize the cost $c(\mathcal{S}_i)$ of the cluster itself.

4.1 Building the Index

The classic implementation of the optimization problem is an algorithm that tries to minimize (11) iteratively. The algorithm converges if $c(\mathcal{S})$ remains constant for two consecutive iterations. The result is usually a local minimum of the optimization problem. Techniques like simulated annealing can be employed further to improve the result.

The pure k-medians clustering algorithm produces clusters with sizes $0 < |\mathcal{S}_i| \leq N$. In order to support the development of clusters of approximately the same size, additional constraints on the cluster sizes are imposed during each iteration:

$$S_{\min} \leq |\mathcal{S}_i| \leq S_{\max} \tag{13}$$

If any cluster's size exceeds the constraint of (13), the smallest cluster is deleted and accordingly the largest cluster is split randomly into two equally sized clusters. The member descriptors of the deleted cluster are assigned to the clusters with the closest associated buoys. A high level description of the iterative optimization algorithm is shown in Fig. 1.

4.2 Updating the Index

Considering that CBIR systems today are dynamic since new images are continuously added to the image archive, the feature dataset increases with time also. Intuitively, the

```
Initialize clusters by assigning descriptors of dataset
Initialize buoys of clusters
Calculate global cost
repeat
      for all clusters whose size exceed constraints
            Find smallest cluster
            Redistribute descriptors of smallest cluster to other clusters
              according to the descriptors' distance to the clusters' buoys
            Delete smallest cluster
            Split largest cluster randomly
            Update buoys of split clusters
      Reassign descriptors of the dataset to clusters
        according to the descriptors' distance to the clusters' buoys
      Update buoys of clusters
      Calculate global cost
until global cost remains constant
```

Fig. 1. Constrained k-Medians Clustering Algorithm

newly added descriptors are assigned to the cluster with the closest buoy. However, since the buoys are not modified during this process, some clusters might grow extensively, while others might not grow at all. This necessitates an infrequent periodic update to the index in order to compensate for the newly added feature descriptors. The clusters are then simply initialized with the buoys of the old index to avoid starting the iterative algorithm from scratch again.

With the number of descriptors increasing, it might even become necessary to increase the overall number of clusters. In this case, empty clusters are added which are then removed by the size constraint in following iterations.

4.3 Indexed Queries

Upon submission of an indexed query request, the first task is to conduct a k'-NN query on the set of buoys $\hat{\mathcal{S}} = \{\hat{\omega}_1, \ldots, \hat{\omega}_k\}$ of the index in order to find the $k' \ll k$ closest buoys $\hat{\omega}'_i$ to the query object

$$\hat{\mathcal{S}}_{NN}(\omega_Q, k') \subset \hat{\mathcal{S}} \tag{14}$$

Then, the second task is to perform an approximate K-NN query $\mathcal{S}_{ANN}(\omega_Q, K, k')$ on the joint union of all member descriptors of the clusters $\mathcal{S}'_i$ associated with the k' closest buoys $\hat{\omega}'_i \in \hat{\mathcal{S}}_{NN}(\omega_Q, k')$

$$\mathcal{S}_{ANN}(\omega_Q, K, k') \subset \bigcup_{i=1}^{k'} \mathcal{S}'_i \tag{15}$$

where K is the number of results the query is supposed to return (typically $K = \{10, 20, 50, 100\}$). The result is an approximation of the correct result given by a sequential K-NN query based on the whole dataset. The accuracy of $\mathcal{S}_{ANN}(\omega_Q, K, k')$

mainly depends on the selection of parameter k'. This is due to the fact that query objects might be located in areas of the feature space that are less populated by the dataset. As a result, the variance of the distance distribution to the dataset descriptors for this particular query object is low. This yields many potential candidate descriptors with approximately the same distance to the query object. Because the feature space is intentionally limited to a small number of clusters k' prior to the search, some potentially relevant descriptors might not be returned during an indexed query.

Practically, this indexing paradigm yields a number of consecutive sequential NN-searches. However, the overall number of comparisons necessary for an approximate NN-search can be reduced significantly by a smart selection of the parameters k and k' with respect to the total number of descriptors N present in the dataset. The interaction between computational cost and retrieval accuracy is investigated in the next section.

5 Results

The overall retrieval accuracy of an approximate K-NN query $\mathcal{S}_{ANN}(\omega_Q, K, k')$ with the proposed indexing scheme depends on the following features:

- the statistical distribution of a given feature dataset $\mathcal{S}$ in the feature domain Ω
- the specific characteristics of the metric δ
- the number of clusters k and the initialization conditions of the constrained k-medians clustering algorithm
- the query object ω_Q and the parameters k' and K of an indexed K-NN query

The first and second features are directly associated with the selection of a particular feature extraction algorithm and its associated dissimilarity measure. The third is related to the visual content of the images that are inserted into the dataset, and therefore, has a random aspect. As a result, their influence cannot be made quantitatively tangible. However, if the feature space and the dataset are selected carefully for an experimental analysis, a general statement about the method's performance can be made.

5.1 Experimental Setup

Database The image database consisted of a total of $N = 25000$ color JPEG images in screen preview quality (approximate size 300×200). The images were taken from CD image catalogues with a variety of topics, e.g. people, sports, art, travel, animals, nature, industry, and business. Thus, the visual content of the database was quite heterogeneous and can be considered domain independent.

Feature Space The feature domain was generated by extracting the color histogram with a total of 116 distinct color bins obtained by fuzzy quantization of the cylindrical HSL colorspace. The L_1-metric was used for comparing the color histograms in terms of similarity.

Clustering The feature dataset was clustered into $k = 500$ clusters. The expected average cluster size is $\overline{|\mathcal{S}_i|} = \frac{N}{k} = 50$. The influence of the number of clusters on the retrieval accuracy is not investigated in this paper.

Queries A total sample of 500 query images that were not part of the image database itself were submitted as requests to the query engine in order to collect the experimental results.

5.2 Retrieval Accuracy

A total of 3 different indices were evaluated in order to depict the influence of the size constraint on the k-medians clustering algorithm. Table 1 shows the corresponding parameters used for the generation of the single indices along with the final global cost. The resulting destribution of cluster sizes is depicted in graph (I) of Fig. 2. Clearly, the unconstrained index (IIa) tends to produce many small and few large clusters. This undesired effect is eliminated by introducing minimum and maximum size constraints on indices (IIb) and (IIc).

Table 1. Evaluated Indices

Index	k	$S_{\min}$	$S_{\max}$	$c(\mathcal{S})$
(IIa)	500	—	—	6580.0022
(IIb)	500	10	100	5976.8084
(IIc)	500	25	75	5879.8690

The retrieval accuracy $P(\omega_Q, K, k')$ for the experimental setup was determined by comparing the correct result $\mathcal{S}_{NN}(\omega_Q, K)$ of the non-indexed K-NN query with the approximate result $\mathcal{S}_{ANN}(\omega_Q, K, k')$ of the indexed K-NN query according to

$$P(\omega_Q, K, k') = \frac{|\mathcal{S}_{ANN}(\omega_Q, K, k') \cap \mathcal{S}_{NN}(\omega_Q, K)|}{K} \tag{16}$$

The graphs (IIa) through (IIc) in Fig. 2 show the retrieval accuracy's dependency on the number of clusters k' that were included in the approximate NN-query. It can be seen that the retrieval accuracy quickly approaches 100% as k' increases. The lower dashed line corresponds to the lower boundary with 90% confidence in the experimental results, that is, 90% of all results are expected to be better than this boundary. The same applies to the lower boundaries with 50% and 75% confidence respectively. In the majority of cases a retrieval accurary of more than 95% can be achieved for $k' = 30$. Furthermore, the retrieval accuracy for the constrained indices (IIb–c) is considerably better in comparison to the unconstrained index (IIa) for larger values of k'. This is due to the fact that the distribution of clusters in feature space is more balanced in terms of size.

5.3 Computational Cost

The computational cost of an NN-query with the proposed indexing scheme is

$$C(N, k, k') \approx k' \cdot (\overline{|\mathcal{S}_i|} + 1) \approx k' \cdot (\frac{N}{k} + 1) \tag{17}$$

$C(N, k, k')$ specifies the number of necessary computations of the metric distance between the query object and the indexed descriptors. In comparison, a linear search requires exactly N computations. A smart choice of the parameters k and k' can lead to a

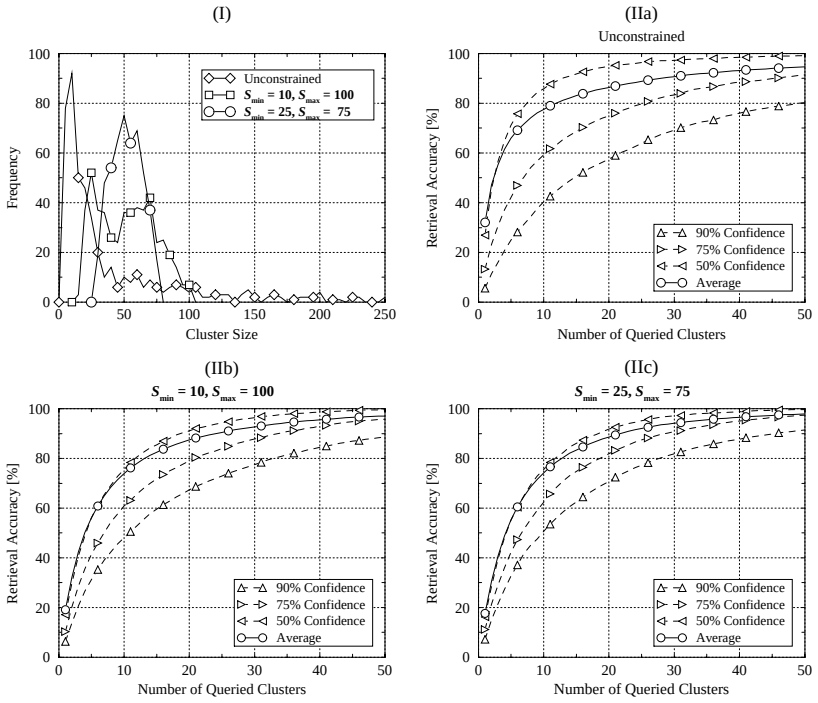

Fig. 2. Distribution of Cluster Sizes for Different Indices (I) and Retrieval Accuracy of Indexed K-NN-Queries (IIa–c) for $K = 100$

significantly lower computational cost while still resulting in a high retrieval accuracy. In the examined cases, the retrieval accuracy was almost 95% for most queries while requiring only 10% of the computational effort of a sequential search.

In fact, with a suitable choice of the parameter k', any query request can be guaranteed to complete in a given time frame after submission (depending on the system's hardware). Of course, this is at the potential loss of some retrieval quality for small time frames.

6 Future Research

The influence of the parameters, namely the number of cluster buoys k, the minimum and maximum cluster constraints $S_{\min}$ and $S_{\max}$, and the number of queried clusters k' during an approximate NN-query, has to be examined more thoroughly in order to fully understand their relationship on retrieval accuracy and computational cost.

The generation of the index through the constrained k-medians clustering algorithm needs considerable processing time that increases with the number of clusters k and the number of feature descriptors N of the dataset. Although the generation is generally

computed off-line, there is a point where the load in terms of processing power and memory resources has to be distributed on multiple computers.

Acknowledgements

This work was supported by the European Commission with the ESPRIT Project #28773 COBWEB "Content-based Image Retrieval on the Web."

References

1. Arya, S., Mount, D.M., Netanyahu, N.S., Silverman, R., Wu, A.Y.: An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions. *Journal of the ACM*, 45(6):891–923, November 1998.
2. Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B.: The R^*-tree: An Efficient and Robust Access Method for Points and Rectangles. In *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, pp. 322–332, Atlantic City, USA, May 1990.
3. Ciaccia, P. and Patella, M.: Using the Distance Distribution for Approximate Similarity Queries in High-dimesional Metric Space. In *Proc. of the 10th Int'l Workshop on Database & Expert Systems Applications*, pp. 200–205, Florence, Italy, September 1999.
4. Ciaccia, P. and Patella, M. and Zezula P.: M-tree: an Efficient Access Method for Similarity Search in Metric Spaces. In *Proc. of the 23rd Int'l Conf. on Very Large Databases*, pp. 426–435, Athens, Greece, September 1997.
5. Enser, P.G.B.: Pictorial Information Retrieval. *Journal of Documentation*, 51(2):126–170, June 1995.
6. Faloutsos, C., Lin, K.I.: FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets. In *Proc. of the ACM SIGMOD Conf.*, pp. 163–174, San Jose, USA, May 1995.
7. Guttman, A.: R-tree: A Dynamic Indexing Structure for Spatial Searching. In *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, pp. 47–57, Boston, USA, June 1984.
8. Kovalev, V., Volmer, S.: Color Co-occurence Descriptors for Querying-by-Example. In *Proc. of the 5th Int'l Conf. on Multimedia Modeling*, pp. 32–38, Lausanne, Switzerland, October 1998.
9. Ng, R., Sedighian, A.: Evaluating Multi-dimensional Indexing Structures for Images Transformed by Principal Component Analysis. In *Storage and Retrieval for Image and Video Databases IV*, vol. 2670, pp. 50–61. SPIE, La Jolla, USA, January 1996.
10. Pass, G., Zabih, R.: Comparing Images Using Joint Histograms. *ACM Journal of Multimedia Systems*, 7(3):234–240, May 1999.
11. Pestov, V.: On the Geometry of Similarity Search: Dimensionality Curse and Concentration of Measure. *Information Processing Letters*, 73(1–2):47–51, January 2000.
12. Sellis, T., Roussopoulos, N., Faloutsos, C.: The R^+-tree: A Dynamic Index for Multi-dimensional Objects. In *Proc. 13rd Int'l Conf. on Very Large Data Bases*, pp. 507–518, Brighton, England, September 1987.
13. Volmer, S.: Tracing Images in Large Databases by Comparison of Wavelet Fingerprints. In *Proc. of the 2nd Int'l Conf. on Visual Information Systems*, pp. 163–172, La Jolla, USA, December 1997.
14. White, D.A., Jain,R.: Similarity Indexing: Algorithms and Performance. In *Storage and Retrieval for Image and Video Databases IV*, vol. 2670, pp. 65–72. SPIE, La Jolla, USA, January 1996.