Faster Exact Histogram Intersection on Large Data Collections Using Inverted VA-Files

Wolfgang Müller and Andreas Henrich

Universität Bayreuth, 95448 Bayreuth, Germany,
Wolfgang.Mueller2@uni-bayreuth.de,
http://ai1.inf.uni-bayreuth.de/mitarbeiter

Abstract. Most indexing structures for high-dimensional vectors used in multimedia retrieval today rely on determining the importance of each vector component at indexing time in order to create the index. However for Histogram Intersection and other important distance measures this is not possible because the importance of vector components depends on the query. We present an indexing structure inspired by VA-file and Inverted file that does not need to determine the importance at indexing time in order to perform well. Instead, our structure adapts to the importance of vector components at query processing time.
Success of this approach is demonstrated in experiments on feature data extracted from a large image collection.

1 Introduction

The research area of Content Based Image Retrieval (CBIR) investigates methods to help users find images in large collections.

Interaction modes and details vary, but most CBIR (and related) applications are driven by *(a)* a *feature extraction* that transforms each image of the collection into a real valued multidimensional feature vector, *(b)* a *distance measure or a similarity measure* that permits evaluating the similarity of a pair of images, *(c)* an *indexing structure* that permits rapid k-NN (k-nearest neighbor) searches in the high-dimensional vector space, as well as *(d)* a *learning method* that permits learning from user (*e.g.* relevance) feedback during use of the CBIR system.

Unfortunately, most publications about CBIR treat the CBIR specific issues (*(a)*, *(b)*, and *(d)*), seeing indexing structures *(c)* just from the user's perspective. This is not helped by the fact that literature about indexing structures is typically uniquely using Euclidean distance, not making sure that the needs of CBIR and related fields are met (*i.e.* they are treating just *(c)*). Few publications address the relation between indexing and feedback in CBIR (for example [6,7]). The present paper intends to be one of these few. It is about a disk-based indexing structure adapted to CBIR-specific distance measures.

In most CBIR systems, as in this paper, the image collection is viewed as a large collection of data points in a high-dimensional real-valued vector space. When performing a query by visual example (QBvE), a query image $\mathbf{q}$ is a point in that space, and we are looking for the k data points $\mathbf{r}_i$ within the collection

closest to the query. Closeness between data points $\mathbf{r}_1, \mathbf{r}_2$ is defined via a distance measure (Distance of $\mathbf{r}_2$ given $\mathbf{r}_1$) $\boldsymbol{\Delta}(\mathbf{r}_2|\mathbf{r}_1)$[1].

Most indexing structures adapted to high-dimensional vectors known to us perform well in the scenario where the distance measure $\boldsymbol{\Delta}(.|.)$ is chosen once, when indexing the data. In particular, it is assumed that the discriminative power of each vector component can be determined at indexing time. A successful example of such an indexing structure is the VA-file (vector approximation file, [8]) for k-NN queries on real-valued vectors. It uses vector approximations for filtering candidates, *i.e.* vectors. The full precision values of all candidates will then be tested in order to find out if each candidate is within the k-NNs. However, for some popular CBIR distance measures, (*e.g.* Histogram Intersection, Kullback-Leibler Divergence, Cosine Distance) the discriminative power of a vector component varies depending on the query. We will call the discriminative power of a vector component henceforth the *Importance* of a vector component, to be defined more formally below.

While in VA-files the quality of the approximation is fixed at indexing time for each vector component, *our contribution is an indexing structure that uses the importance of vector components for choosing the quality of the approximation at every query.* Our structure, the Inverted Vector Approximation (IVA) file combines the advantages of the inverted file (a well-known data structure from information retrieval, also used in CBIR [6]) with the advantages of the VA-file.

This paper is organized as follows: in the next section we describe the VA-file. Then we describe the query model for which IVA-files are optimized, as well as the notion of vector component importance that is used for choosing the appropriate approximation level (section 3). Then, in section 4, we describe Inverted VA files as VA-files with changed storage and evaluation order. In section 5 we give a method to provide diverse levels of approximation for components of color histograms. Lastly, we present experiments, related work and future work.

2 VA-Files

The VA-file is motivated by the observation that the *curse of dimensionality* makes it impossible for tree-based multi-dimensional indexing structures (*e.g.* [4] and others) to obtain better than linear complexity for k-NN search in spaces with dimensionality $n > 10$ [8]: Tree-based indexing structures have to consider $c \cdot N$ (with $c \approx 1$) vectors when querying a collection of N vectors. The basic idea of the VA-file is to accept this as a fact, and to accept linear complexity, but to minimize the overhead in order to be faster than tree-based indexing structures. This is achieved in a two-pass process.

[1] For brevity, we do not introduce an additional notation for *similarity measures*. The difference between distance and similarity measures is that for distances smaller values indicate better match whereas for similarity measures higher values indicate better match. We always write Δ and $\Delta(y|x) < \Delta(z|x)$, thinking in terms of distances, *i.e.* x is matches y better than z.

The base assumption in optimizing the VA-file's performance is that the time complexity of one query is determined by the number of data blocks that have to be read from disk during the query. If for processing the query, $c \cdot N$ vectors need to be looked at, then the number of data blocks to be read during a query process is proportional to the total number of bits needed for representing a data vector within the context of our query.

Typical tree-based indexing structures store the data vectors in full precision. Each vector component corresponds to a 32 bit `float` value. The VA-file however, stores each vector twice: once in full (`float`) precision, once its approximation ($b \ll 32$ bits per component). In the approximation each b-bit value designates a float interval.

A k-NN query is processed in a two-pass process: in the first pass, approximations are used for obtaining so called *candidates*, *i.e.* data points for which we cannot rule out the possibility that they are among the query result (*i.e.* the k-NN) by looking at the approximation alone. For some of the candidates we need to look at the full precision vectors, possibly discarding more candidates from the candidate set while we are finding more and more of the k nearest neighbors, because we get new bounds for being among the k-NN during this process.

The performance of the VA-file is determined by the following factors:

Number of bits needed per approximation component: As *all* approximations are read during the first pass of the query processing, the number of blocks read in this phase is proportional to the number of bits to be read per vector component.

Size of the candidate set: Of course, the number of candidates influences how many data vectors have to be read in full precision. In fact, as hard disks are block-oriented devices, we will have to read statistically *almost one entire block* per full precision data vector we look at.

Block size: The data block that needs to be read to verify the candidate status of a data vector will possibly contain a large number of data vectors (≈ 10 vectors) that are not member of the candidate set, *i.e.* that are not interesting for our query process. Although these data points will not be considered, they are still read, and still are part of the cost of the query process.

Note that the number of bits needed per component approximation and the size of the candidate set are directly related to the quality of the approximation. There is some literature on choosing useful approximations for skewed distributions [3,9]. Both methods, as well as the initial VA-file approximation method have in common that the approximation is fixed at indexing time. There is no possibility to adapt approximation quality to the query without updating the index. That is, the methods adapt to the data, however costly and slowly.

3 Queries and Component Importance

What we need to do when processing a query is to find the k vectors $\mathbf{r}_l$, $l \in \{1, \ldots, k\}$ closest with respect to a query $\mathbf{q}$ from which a distance measure

$\mathbf{\Delta}(.|\mathbf{q})$ (given the query) is inferred. We require that the distance measure can be expressed as a component wise sum $\mathbf{\Delta}(\mathbf{r}_i|\mathbf{q}) = \sum_{j=1}^{n} \Delta_j(r_{i,j}|\mathbf{q})$.

Now, let us imagine for an instant, we would like to perform an inexact k-NN query on our data collection. When calculating each of the distances $\mathbf{\Delta}(\mathbf{r}_i|\mathbf{q})$ we would not evaluate the distance for all vector components $\sum_{j=1}^{n} \Delta_j(r_{i,j}|\mathbf{q})$, but rather only for a subset $J \subset \{1,\ldots,n\}$ of vector components $\sum_{j\in J} \Delta_j(r_{i,j}|\mathbf{q})$. It is intuitively evident that for two sets of components $J_1 \subset \{1,\ldots,n\}$ and $J_2 \subset \{1,\ldots,n\}$ the result will differ most of the time if $J_1 \neq J_2$. Moreover, we will assume that ignoring some components will change the results more strongly with respect to the results of the full evaluation than ignoring other ones.

This is what importance of vector components is about. An important component is a vector component that, if not evaluated, changes the ranking strongly. We define the importance of vector component j given the query $\mathbf{q}$ as:

$$I(j|\mathbf{q}) = \max_{i=1}^{N}(\Delta_j(r_{i,j}|\mathbf{q})) - \min_{i=1}^{N}(\Delta_j(r_{i,j}|\mathbf{q})) \tag{1}$$

That is, the Importance of component j is given by the variability of its contribution to $\mathbf{\Delta}$, *i.e.* its discriminative power.

The implementation of VA-files involves the approximation of vector components. That is, we evaluate the distance for all components, but the evaluation is not exact. The concept of component importance is also important here. It allows us to choose how precise or how coarse we can make the approximation of a given component. Components of high importance will be better approximated than those of low importance. However, it is important here to note that neither the VA-file nor our variant, the IVA-file, perform inexact k-NN queries, because the inexact first pass is supplemented by the second pass exactly checking the candidates.

In this publication we are interested in non-Gaussian data distributions and distance measures other than Euclidean distance.[2] The definition of importance we gave in Eq. 1 proves very useful in these cases, as the following example shows:

Assume documents represented as n-bin histograms, and assume they are going to be ranked using histogram intersection $\mathbf{\Delta}(\mathbf{r}|\mathbf{q}) := \sum_{j=1}^{n} \min(q_j, r_j)$

Let histogram bin (*i.e.* vector component) number 42 represent the color *"purple"*. Furthermore, assume the query $\mathbf{q^{42}}$ to be a vector with $q_j^{42} = 1$ for $j = 42$ and $q_j^{42} = 0$ otherwise. $\mathbf{q^{42}}$ would correspond to a uniformly purple query image. In this case $I(j|\mathbf{q^{42}}) = I_{42}$ for $j = 42$ and $I(j|\mathbf{q^{42}}) = 0$ otherwise, for some constant $I_{42} > 0$ whose exact value does not matter here.

What's relevant for us here is that when a vector $\mathbf{r}_i$ is ranked with respect to a uniformly colored image: $\mathbf{q^{42}}$, no histogram components matter at all (*i.e.*

[2] In the case of Gaussian distribution of data points and Euclidean distance the way to quantify and use importance is to perform a principal components analysis (PCA), *i.e.* finding a basis of the vector space that consists of the Eigenvectors of the covariance matrix. In this case, the least important vector components are those corresponding to the Eigenvectors with the smallest Eigenvalues. VA$^+$-files [3] exploit PCA in order to reduce the size of the approximation.

they have *zero importance*) except the one that is present in the query image. Indeed evaluating just $\Delta_{42}(.|q_{42}^{42})$ suffices for obtaining the correct k-NN query result, because $\Delta_{42}(r_{i,42}|\mathbf{q}^{42}) = \boldsymbol{\Delta}(\mathbf{r}_i|\mathbf{q}^{42})$ for all $\mathbf{r}_i$.

In contrast, a PCA-based method would evaluate the importance of vector components based on their variance over the whole collection. As a consequence, a component c, $c \neq 42$ could be assigned a larger importance than component 42 due to the large variance of $r_{i,c}$. Clearly this is not useful in our example, as the importance of vector components is solely determined by the query.

The above example is clearly extreme. However, also in realistic settings with color histograms calculated from real data collections, the essential ingredient stays the same: the importance of the vector components depends on the distance measure Δ, the query $\mathbf{q}$ and the data collection $\mathcal{C}$. This is, why we write the importance of component j for query $\mathbf{q}$ as $I(j|\mathbf{q}, \mathcal{C}, \Delta)$ in the following.

4 From VA to Inverted VA

In VA-files, vector approximations are stored and read "line by line", *i.e.* document by document, vector by vector. It is not possible to read partial vectors because hard disks are block-oriented devices: typically, the approximations for a number of vectors will fit into one block. Reading only parts of one block won't bring any efficiency gain. This observation leads us to changing the storage and evaluation order of the VA-file and then looking at the possible benefits.

Essentially, Inverted VA-files are VA-files with a changed storage and evaluation order. In IVA-files, vector approximations are stored and read column by column. That is, when processing a query, the query processor first reads and processes the first components $r_{i,1}$ for each data point $\mathbf{r}_i$, before then processing the second components $r_{i,2}$, and so forth. The second phase of IVA-queries is the same as for queries using VA-files.

While this might appear only a minor change, this change of storage order allows us to *choose which components to read*. We use this newly-acquired liberty of what to read as follows: We store each component of each data vector $\mathbf{r}_i$ at several levels of approximation quality. On processing a query we choose for each component the level of approximation quality needed. After having chosen, we will read the corresponding column from the IVA-file.

5 Approximations for Histogram Intersection

Now the only ingredient missing for successful use of the IVA-file for color histograms is the approximation itself. The approximation is simply an array mapping of a b bit integer to 2^b real-valued intervals. While the methods presented up to this point in the paper have been generic, now we will concentrate on the histogram intersection distance.

Let us come back to our example given in section 2. Here, we showed that the importance is zero for each component for which the query component q_j

is zero. More generally, the importance of the j-th component is limited by the value of q_j: $I(j|\mathbf{q}) = \max_{i=1}^{N}(\min(q_j, r_{i,j})) - \min_{i=1}^{N}(\min(q_j, r_{i,j}))$.

In other words, for a given document $\mathbf{r}_i$, the contribution of component j to $\mathbf{\Delta}(\mathbf{q}, \mathbf{r}_i)$, $\Delta(q_j, r_{i,j}) := \min(q_j, r_{i,j})$ is limited by q_j. We can use this fact for the approximation of $\mathbf{r}_i$.

For generating the approximations that performed best in our experiments (we also tried the ones suggested initially for Euclidean distance in [3]), we chose a small number β and cut the interval between minimum ($r_{-,j} := \min_i r_{i,j}$) and maximum ($r_{+,j} := \max_i r_{i,j} + \epsilon$) value of a component within the collection into 2^β non-overlapping equal-sized chunks, obtaining a set of intervals (R abbreviates "Region", and $0 \leq \ell < 2^\beta$):

$$R(\ell, \beta, \beta, r_{+,j}, r_{-,j}) := \left[r_{-,j} + \frac{\ell}{2^\beta} \cdot (r_{+,j} - r_{-,j}), r_{-,j} + \frac{\ell+1}{2^\beta} \cdot (r_{+,j} - r_{-,j}) \right) \quad (2)$$

$$:= [R_{\min(\ell,\beta,\beta,r_{+,j},r_{-,j})}, R_{\max(\ell,\beta,\beta,r_{+,j},r_{-,j})}) \quad (3)$$

Eq. 3 is just shorthand for Eq. 2. We now define for $b < \beta$ bits:

$$R(\ell, b, \beta, r_{+,j}, r_{-,j}) := \begin{cases} R(\ell, b, \beta, r_{+,j}, r_{-,j}) & : \ell < 2^b - 1 \\ \left[r_{-,j} + \frac{2^b-1}{2^\beta} \cdot (r_{+,j} - r_{-,j}), r_{+,j} \right) & : otherwise \end{cases} \quad (4)$$

That is, the first $2^b - 1$ intervals are exactly the same as with $\beta > b$, and the remaining interval covers the complete rest.

Now, how do we choose the level of approximation? Given a query $\mathbf{q}$ we can choose the minimal b such that for all ℓ the b-bit approximation produces the same histogram intersection values as the β-bit approximation:

$$\min(q_i, R_{\min(\ell,b,\beta,r_{+,j},r_{-,j})}) = \min(q_i, R_{\min(\ell,\beta,\beta,r_{+,j},r_{-,j})}) \quad (5)$$

$$\wedge \min(q_i, R_{\max(\ell,b,\beta,r_{+,j},r_{-,j})}) = \min(q_i, R_{\max(\ell,\beta,\beta,r_{+,j},r_{-,j})}) \quad (6)$$

This method works already fairly well, however we can improve on this by relaxing the constraint a bit. We choose a minimal b such that

$$\min(q_i, R_{\max(\ell,b,\beta,r_{+,j},r_{-,j})}) - \min(q_i, R_{\min(\ell,b,\beta,r_{+,j},r_{-,j})}) \leq \frac{(r_{+,j} - r_{-,j})}{2^\beta} \quad (7)$$

This makes the b-bit approximation (for some queries and documents) slightly worse than the β-bit approximation. However, this increases the number of times, where we do not have to read any approximation at all ($b = 0$-bit approximation).

Example: Let us continue our example already given in section 2. Let us assume that the minimum and maximum values of the 42nd component within the data collection are 0 and 1, respectively ($\min_{i=1}^{N} r_{i,42} = 0$ and $\max_{i=1}^{N} r_{i,42} = 1$), and let us choose $\beta = 5$.

With this choice,

$$R(\ell, 5, 5, 1, 0) = \left[\frac{\ell}{32}, \frac{\ell+1}{32} \right) \quad (8)$$

$$R(\ell, b, 5, 1, 0) := \begin{cases} \left[\frac{\ell}{32}, \frac{\ell+1}{32} \right) & : \ell < 2^b - 1 \\ \left[\frac{\ell}{32}, 1 \right) & : otherwise \end{cases} \quad (9)$$

We now choose values for q_{42} and give the corresponding b to be used for the approximation. We consider three cases:
(1) If $q_{42} = 1$, evidently, we need $b = \beta = 5$.
(2) If $q_{42} = 0$, then $b = 0$ yields $[0, 1)$ as only "approximation interval". For all $r_{i,42}$ within the collection

$$\begin{aligned} &\min(0,0) = \min(q_{42}, R_{\min(\ell,\underline{5},5,1,0)}) = \min(q_{42}, R_{\min(\ell,\underline{0},5,1,0)}) = \min(0,0) \\ \wedge &\min\left(0, \frac{1}{32}\right) = \min(q_{42}, R_{\max(\ell,\underline{5},5,1,0)}) = \min(q_{42}, R_{\max(\ell,\underline{0},5,1,0)}) = \min(0,1) \end{aligned}$$

So, we can choose $b = 0$ without degrading approximation quality.
(3) If $q_{42} = \frac{7}{64}$, $b = 2$ yields $\left[0, \frac{1}{32}\right), \left[\frac{1}{32}, \frac{1}{16}\right), \left[\frac{1}{16}, \frac{3}{32}\right), \left[\frac{3}{32}, 1\right)$ as only approximation intervals.

Now, for all $\ell = 0, 1, 2$ the resulting bounds will be exactly equal to those of the $b = \beta = 5$ bit approximation. So for $\ell = 0, 1, 2, 3$ a 2-bit approximation fulfills Eq. 5, Eq. 6. The interesting case is $\ell \geq 4$. Let us choose $\ell = 4$. We find:

$$\begin{aligned} \frac{7}{64} &= \min\left(q_{42}, R_{\min(4,\underline{5},5,1,0)}\right) \neq \min\left(q_{42}, R_{\min(4,\underline{2},5,1,0)}\right) = \frac{3}{32} \\ \frac{7}{64} &= \min\left(q_{42}, R_{\max(4,\underline{5},5,1,0)}\right) = \min\left(q_{42}, R_{\max(4,\underline{2},5,1,0)}\right) = \frac{7}{64} \end{aligned}$$

That is, for $\ell = 4$, the approximation condition given in Eq. 5 is *not* fulfilled *i.e.* the approximation quality when using $b = 2$ bits is slightly degraded with respect to using $\beta = 5$-bit approximations in this case. However, Eq. 7 holds here, *i.e.* when using the second way of approximation, described in the previous section, we would use 2 bits. When using $b = 3$ bits, Eq. 5 and Eq. 6 both hold, *i.e.* when using the first way of approximation described in the previous section, we would use 3 bits.

As this example shows, we have the opportunity to use fewer bits by degrading approximation quality slightly. In our experiments, the loss of approximation quality (*i.e.* the increase in size of the candidate set) was offset by the savings in approximation size.

6 Experiments

In the experiments we present here we used a collection of 66616 32-bin color layout histograms (*i.e.* color histograms with 32 components), obtained from `kdd.ics.uci.edu/databases/CorelFeatures/CorelFeatures.html`. The site reports that the data has been used in [5]. These histograms have been extracted from the Corel image collection. They are free for scientific use. The 32 bins correspond to 4 levels of hue, and 2 levels of saturation in 4 image regions, capturing both color and layout in the histogram.

For our experiments, we varied β from 2 to 12 bits. Each time, we did 100 1-NN test queries using a VA-file and an IVA-file, respectively, and counted during the test queries the number of 8192-byte blocks read, calculating from the

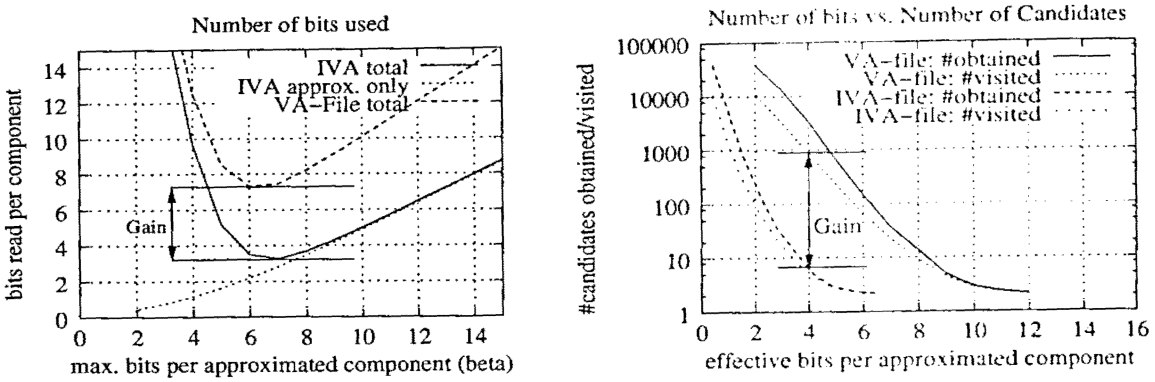

Fig. 1. Overall performance of 1-NN query on VA-file. On the left, the horizontal lines within the plot highlight the value that matters: the overall number of bits read per component, and the gain obtained by use of the IVA-file. Here, the IVA-file performs about 2 times better than the VA-file. On the right, we show that the number of candidates visited for an average approximation bit width of $b = 4$ is by two orders of magnitude smaller for the IVA-file than for the VA-file.

number of blocks the effective number of bits read. The average number of bits read per vector component on the test queries for one given β is one data point in our plots. The choice of β influences the precision of the approximation, and thus the number of candidates. The total number of bits read depends on β, and the number of candidates for which the exact vector has to be read, so we need a tradeoff for which the average overall number of bits read becomes minimal. The number of bits read per component at this minimum is the parameter that determines the performance of the VA-file or IVA-file. The comparative plot obtained using an IVA-file and a VA-file are shown on the left side of Fig. 1. The overall number of bits read per vector component is more than 2 times better for the IVA-file than the number of bits read using the VA-file.

To highlight the increase in the quality of the approximation for a given number of bits effectively used, [3,9] plot the number of candidates obtained in the first (*i.e.* approximation) pass and the number of candidates actually visited versus the number of bits actually used in the approximation (average b plus overhead for blocking). The right side of Fig. 1 shows such a plot for our test data. We see two pairs of curves. Each pair depicts the number of candidates generated by the first pass, and (a little lower) the number of candidates that actually have to be visited. The pair of curves depicting the IVA-file is distinctly lower than that depicting the VA-file. In particular one can see that for a given number of bits per approximation (average b), the IVA-file needs to visit up to two orders of magnitude fewer candidates than the corresponding VA-file.

7 Related Work

There are a few groups that have worked with structures similar to the IVA-file. Squire *et al.* [6] used search pruning on inverted files to improve response time.

However, the process was a one-pass process yielding *approximately* the same result as full evaluation of the query. De Vries *et al.* [2] use full precision *Vertically Decomposed Data* (*i.e.* column vectors) in order to choose what parts of the query to evaluate as part of a two-pass query process. Here each tuple is read in full precision. However, only some (not all) tuples are read. They report savings of about $\frac{2}{3}$rds with respect to full evaluation (compared to savings of about $\frac{8}{9}$th reported in our experiments). The motivation for vertically decomposed data and pruning is cheap update. Lastly, Böhm *et al.* [1] report experiments in which they index feature data belonging to each group of features (color, texture *etc.*) in its own VA-file, merging the results of queries on each file.

8 Further Work

In the future, we want to explore the usefulness of our data structure for other distance measures as *e.g.* the cosine distance and the Kullback-Leibler Divergence. In addition to that we want to show the usefulness of our method also for other distance measures, if relevance feedback changes the importance of vector components at query time.

References

1. K. Böhm, M. Milvoncic, H.-J. Schek, and R. Weber. Fast Evaluation Techniques for Complex Similarity Queries. In *Proc. Intl. Conf. on VLDB*, 2001.
2. A. P. de Vries, N. Mamoulis, N. Nes, and M. L. Kersten. Efficient k-NN Search on Vertically Decomposed Data. In *Proc. SIGMOD*, Madison, WI, USA, June 2002.
3. H. Ferhatosmanoglu, E. Tuncel, D. Agrawal, and A. E. Abbadi. Vector approximation based indexing for non-uniform high dimensional data sets. In *CIKM: ACM Intl. Conf. on Information and Knowledge Management*. McLean, VA, USA, 2000.
4. A. Guttman. R-trees: A dynamic index structure for spatial searching. In B. Yormark, editor, *SIGMOD'84, Proc. of Annual Meeting, Boston, MA, 1984*
5. M. Ortega, Y. Rui, K. Chakrabarti, K. Porkaew, S. Mehrotra, and T. S. Huang. Supporting ranked boolean similarity queries in MARS. *IEEE Transactions on Knowledge and Data Engineering*, 10(6), December 1998.
6. D. M. Squire, H. Müller, and W. Müller. Improving response time by search pruning in a content-based image retrieval system, using inverted file techniques. In *IEEE Workshop on Content-based Access of Image and Video Libraries (CBAIVL'99)*, Fort Collins, CO, USA, 1999.
7. J. Tesic and B. S. Manjunath. Nearest Neighbor Search for Relevance Feedback. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, Madison, WI, USA, June 2003.
8. R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proc. Intl. Conf. on VLDB*, New York, USA, 1998.
9. P. Wu, B. Manjunath, and S. Chandrasekaran. An Adaptive Index Structure for High-dimensional Similarity Search. In *Proc. IEEE Pacific-Rim Conf. on Multimedia*, Advances in Multimedia Information (PCM '01), Beijing, China, 2001.