A Pivot-based Index Structure for Combination of Feature Vectors

Benjamin Bustos Daniel Keim Tobias Schreck
Department of Computer and Information Science, University of Konstanz
Universitätstr. 10 Box D78, 78457 Konstanz, Germany
{bustos,keim,schreck}@informatik.uni-konstanz.de

ABSTRACT

We present a novel indexing schema that provides efficient nearest-neighbor queries in multimedia databases consisting of objects described by multiple feature vectors. The benefits of the simultaneous usage of several (statically or dynamically) weighted feature vectors with respect to retrieval *effectiveness* have been previously demonstrated. Support for *efficient* multi-feature vector similarity queries is an open problem, as existing indexing methods do not support dynamically parameterized distance functions. We present a solution for this problem relying on a combination of several pivot-based metric indices. We define the index structure, present algorithms for performing nearest-neighbor queries on these structures, and demonstrate the feasibility by experiments conducted on two real-world image databases. The experimental results show a significant performance improvement over existing access methods.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content analysis and indexing—*indexing methods*

Keywords

Content-based indexing and retrieval, combination of features, nearest neighbor queries

1. INTRODUCTION

The development of multimedia search systems is an important research issue, due to the growing amount of digital audio-visual information. For example, in the case of images and video, the growth of digital data has been observed since the introduction of 2D capture devices. Also, the acquisition technology of 3D models by means of 3D scanners is constantly improving. As we see progress in the fields of acquisition, storage, and dissemination of various multimedia formats, the development of effective and efficient database management systems that handle these formats is needed.

SAC'05 March 13-17, 2005, Santa Fe, New Mexico, USA

Some of the most important tasks in multimedia databases are clustering, classification, and retrieval. These tasks all rely on the definition of a similarity measure between multimedia objects, which depends on the similarity of the objects' content. For describing object content, it is possible to use annotation information, which represents the content of an object in textual form, or to use characteristics of the multimedia data itself, the so-called *content-based approach*. The latter is the more promising approach, because in general textual descriptions are manually created, which is prohibitively expensive, and they are subject to the opinion of the person who creates them. In contrast, content-based search algorithms allow an implementation of fully automatic retrieval systems.

To describe multimedia objects under the feature vector approach, numerical values are extracted from each object to form *feature vectors* of typically high dimensionality. For many multimedia data types (e.g., images, 3D models, audio tracks), a number of extraction algorithms have already been proposed. In recent studies, e.g., [3], it has been shown that the usage of *combinations of feature vectors* can lead to significant improvements on the effectiveness of the similarity search, but the efficiency problem was not addressed.

Index structures for vector spaces are surveyed in [2]. These data structures were primarily designed to index single feature vectors, and they cannot be directly used to index set of features. Even if one concatenates the feature vectors and applies standard indexing techniques, the efficiency of these indices will be poor due to the *curse of dimensionality* [2]. The main contribution of this paper is to propose an index structure based on the so-called *pivots*, that can be used to improve the efficiency of similarity search algorithms in multimedia databases, where each object is described by a set of different features vectors.

The paper is organized as follows. In Section 2, we motivate the use of combinations of features and we describe the canonical pivot-based index structure. In Section 3, we present our proposed index, describing the nearest-neighbor (NN) algorithm, and we also show how to use an R*-tree (in some restricted case) to index the combination of features. Section 4 presents the experimental study. Section 5 concludes and outlines our future work.

2. PROBLEM DEFINITION

In this section, we motivate the use of combinations of feature vectors and depict the canonical index based on pivots, which will be used as the basis for the proposed index structure.

2.1 Combination of feature vectors

By their definition, individual feature vectors (also referred to as signatures or descriptors) rely on certain aspects of multimedia objects, which are sampled from the objects to form vectors of real values. Usually, two different feature extraction algorithms describe different, and often complementary information of an object. The use of combinations of descriptors can improve the retrieval effectiveness of the similarity search [7, 3]. This approach avoids the disadvantages of using a single feature only, which captures only certain characteristics of an object, and leads to a more complete search, usually resulting in higher retrieval precision.

Let $F = \{f_1, \ldots, f_N\}$ be a set of feature vectors, each of them associated with a distance function $d_i,\ i = 1, \ldots, N$. We define the *combined distance function* $D(x, y)$ as the normalized linear combination of the distances between x and y,

$$D(x, y) = \sum_{i=1}^{N} w_i \cdot \frac{d_i(x_i, y_i)}{nFactor_i}.$$

The *normalizing factors* $nFactor_i$ are necessary in case that the feature vectors have different dimensionalities or component scales. For example, one can define $nFactor_i$ as the maximum distance between two objects in the feature space defined by f_i. The *weights* w_i allow us to give more importance to those feature vectors that are more relevant to the similarity search. The basic case is to assign $w_i = 1, 1 \leq i \leq N$, i.e., all the vectors are equally important for the combined distance computation. The computation of w_i can also be *dynamically* done, i.e., their values depend on the query object [3]. Also, in interactive systems, user relevance feedback may be obtained in order to adjust feature vector weightings. If all weights are positive and all distance functions are metrics, then it follows that D is also a metric. We assume this property for the rest of the paper.

A simple linear scan is enough to perform NN queries using the combined distance function D, but using an index structure the search should be faster. Unfortunately, index structures designed for single feature vectors cannot be directly used, in the general case, on sets of features vectors.

2.2 Pivot-based indexing

There are many similarity search indices based on *pivots* [5], which are selected objects from the database. Here we describe the canonical index structure based on pivots and the algorithm for performing range searches using this index. In Section 3.3 we will discuss how to adapt this algorithm to implement a nearest-neighbor search. The pivot-based index will be the basis for our proposed index structure.

Let $(\mathbb{X}, d)$ be a feature space (usually a vector space), where $\mathbb{X}$ is the universe of valid *objects* and d is a metric on the space, that is, d satisfies the properties of *strict positiveness* ($d(x, y) \geq 0$ and $d(x, y) = 0 \Leftrightarrow x = y$), *symmetry* ($d(x, y) = d(y, x)$), and the *triangle inequality* ($d(x, z) \leq d(x, y) + d(y, z)$). Let $\mathbb{U} \subseteq \mathbb{X}$ be a set of objects (a database), with size $|\mathbb{U}| = n$. Given a query object $q \in \mathbb{X}$, a *range query* $(q, r)_d$ is defined as the objects in $\mathbb{U}$ that are within distance r to q, that is $(q, r)_d = \{u \in \mathbb{U},\ d(u, q) \leq r\}$.

Given a query $(q, r)_d$ and a set of k pivots $\mathbb{P} = \{p_1, \ldots, p_k\}$, $p_i \in \mathbb{U}$, by the triangle inequality it follows that $d(p_i, x) \leq d(p_i, q) + d(q, x)$, and also that $d(p_i, q) \leq d(p_i, x) + d(x, q)$ for any $x \in \mathbb{X}$. From both inequalities, it follows that a lower bound on $d(q, x)$ is $d(q, x) \geq |d(p_i, x) - d(p_i, q)|$. The objects $u \in \mathbb{U}$ of interest are those that satisfy $d(q, u) \leq r$, so all the objects that satisfy the *exclusion condition* (1) can be discarded, without actually evaluating $d(q, u)$.

$$|d(p_i, u) - d(p_i, q)| > r \quad \text{for some pivot } p_i. \tag{1}$$

The pivot-based index consists of the kn precomputed distances $d(p_i, u)$ between every pivot and every object of the database. Therefore, at query time it is only necessary to compute the k distances between the pivots and the query q, $d(p_i, q)$, in order to apply the exclusion condition (1). The list of candidate objects $\{u_1, \ldots, u_m\} \subseteq \mathbb{U}$ that cannot be discarded with the exclusion condition (1) must be directly checked against the query object.

The way how pivots are selected affects the efficiency of the search algorithms. We will use the *incremental selection technique* described in [4] to select sets of "good pivots", as well as *random selected* pivots.

Note that this canonical pivot-based algorithm (and all its variants) *does not allow* the usage of dynamic weights in the distance function.

3. PROPOSED INDEX STRUCTURE

We will study two different cases: When the weights are fixed (*fixed-weighted combination*) and when the weights are dynamic and may change on each query (*dynamic-weighted combination*).

3.1 Fixed-weighted combinations

In this case, the combined distance has the form

$$D_{fix}(x, y) = \sum_{i=1}^{N} W_i \cdot \frac{d_i(x_i, y_i)}{nFactor_i},$$

where the weights W_i are constant values. As the combined distance D is a metric and a static function, the canonical pivot-based index can be used without any modification.

3.2 Dynamic-weighted combinations

In this case, the combined distance has the form

$$D_{dyn}(x, y, w) = \sum_{i=1}^{N} w_i(\cdot) \cdot \frac{d_i(x_i, y_i)}{nFactor_i},$$

that is, the weights are dynamically assigned on each query. It follows that the distance function *is not static* and depends on the query object. Therefore, *it is not possible* to precompute the distance matrix between pivots and objects, because we do not know *a priori* the set of weights and they may change with each query.

To overcome this problem, we propose a novel index structure that builds the distance matrix at query time. The index consists of N matrices of the form

$$M_i = \frac{1}{nFactor_i} \cdot \begin{bmatrix} d_i(p_1, u_1) & \ldots & d_i(p_k, u_1) \\ \vdots & \ddots & \vdots \\ d_i(p_1, u_n) & \ldots & d_i(p_k, u_n) \end{bmatrix}.$$

It follows that the combined distance between pivot p_s, $1 \leq s \leq k$, and object u_t, $1 \leq t \leq n$, can be computed as

$$D_{dyn}(u_t, p_s, w) = \sum_{i=1}^{N} w_i(\cdot) \cdot M_i[s,t].$$

Intuitively, at query time we dynamically build a pivot index table that reflects the submitted combination weights, and we then use this table to discard objects in order to save computation time.

3.3 Nearest-neighbor search algorithm

We use a modification of the NN algorithm sketched in [5] to perform this type of queries using our proposed index. This algorithm can be easily modified to implement k-NN queries.

The first algorithm (Figure 1) is used in the case of fixed-weighted combination of feature vectors. The second algorithm (Figure 2) is used in the case of dynamic-weighted combination of feature vectors.

```
    NN-Search(𝕌, ℙ, Index, q)
 1  Compute D_fix(p_i, q), 1 ≤ i ≤ k
 2  mindist ← min_{i=1}^{k} {D_fix(p_i, q)}
 3  NN ← p_{arg min_{i=1}^{k} {D_fix(p_i,q)}}
 4  for each u ∈ U − P do
 5    for each p ∈ P do
 6      if |D_fix(p, q) − D_fix(p, u)| > mindist then
 7        Discard object u and break
 8      endif
 9    endfor
10    if u not discarded and D_fix(q, u) < mindist
      then
11      mindist ← D_fix(q, u)
12      NN ← u
13    endif
14  endfor
15  return NN
```

Figure 1: Nearest-neighbor search algorithm, fixed-weighted combination.

The idea of the NN search algorithm is as follows. Firstly, we compute the distances between all pivots and the query object q, and the pivot whose distance to q is minimum (*mindist*) will be the first *NN candidate*. Then, for each object $u \in \mathbb{U}$ that it is not a pivot, the exclusion criterion is applied, using as tolerance radius the distance from the candidate NN to the query object. If u cannot be discarded, we compute the distance between u and q. If this distance is smaller than *mindist*, then we set u as the new NN candidate and we update *mindist*. The process ends when all the objects from $\mathbb{U}$ have been checked.

3.4 Combination of feature vectors and spatial access methods

It is possible to use a spatial access method (e.g., the R*-tree [1]) in the case of fixed-weighted combination of features, but with one restriction: The distance function must be the Manhattan distance $l_1(x,y) = \sum_{i=1}^{t} |x_i - y_i|$ for all feature vectors. Then, the combined distance function is defined as

```
    NN-Search(𝕌, ℙ, Index, q, w)
 1  Compute D_dyn(p_i, q, w), 1 ≤ i ≤ k
 2  mindist ← min_{i=1}^{k} {D_dyn(p_i, q, w)}
 3  NN ← p_{arg min_{i=1}^{k} {D_dyn(p_i,q,w)}}
 4  for each u ∈ U − P do
 5    for each p ∈ P do
 6      Compute D_dyn(p, u, w)
 7      if |D_dyn(p, q, w) − D_dyn(p, u, w)| > mindist
      then
 8        Discard object u and break
 9      endif
10    endfor
11    if u not discarded and D_dyn(q, u, w) <
      mindist then
12      mindist ← D_dyn(q, u, w)
13      NN ← u
14    endif
15  endfor
16  return NN
```

Figure 2: Nearest-neighbor search algorithm, dynamic-weighted combination.

$$D(x,y) = \sum_{i=1}^{N} W_i \cdot \frac{l_1(x_i, y_i)}{nFactor_i}.$$

It follows that $D(x,y)$ is equivalent to a concatenation of all *weighted and normalized* feature vectors followed by a computation of the Manhattan distance over the concatenated vectors (note that *this is not true* for the dynamic-weighted combination case!). Thus, it is guaranteed that the retrieved answer using the R*-tree will be the correct one. Note that the dimensionality of the vectors stored in the R*-tree will be equal to the sum of the dimensionalities of the N feature vectors that conforms the concatenation, which can result in a very high dimensionality.

4. EXPERIMENTAL RESULTS

We performed a number of NN queries using two real-world databases, and computed the average response time. We used the Manhattan distance as the distance function for all feature vectors. For constructing the pivot-based indices, we used random as well as good pivots. All feature vectors were normalized by the estimated maximum distance between two points in the space (for each feature vector, respectively). As the different weighting schemas affect the effectiveness but not the efficiency of the search, we only used uniform weighting in all experiments.

We implemented our proposed NN search algorithm and compared it against a linear scan and a R*-tree with bulk loading (best efficiency). We used the R*-tree implementation from the *Spatial Index Library* [6]. All indices were stored in main memory and optimized for this scenario. The platform on which the experiments were run is a PC with a Pentium IV 2.4 Ghz processor and 1 Gb of main memory. As efficiency measures, we used the CPU time needed to compute the NN queries and the number of discarded objects by the index.

4.1 Corel image features

The *Corel image features* contains features from 68,040 images extracted from a Corel image collection. The features are based on the color histogram (32-D), color histogram layout (32-D), co-occurrence texture (16-D), and color moments (9-D). This database is available at the *UCI KDD Archive* [8]. We used a subset of this database consisting on 66,615 images, because there were some missing features for some of the images (we included only those objects for which complete sets of feature vectors were available). We selected 10% of the images from the database at random to be used as query objects.

Figure 3 shows the results for both fixed-weighted and dynamic-weighted cases, using random and good pivots.

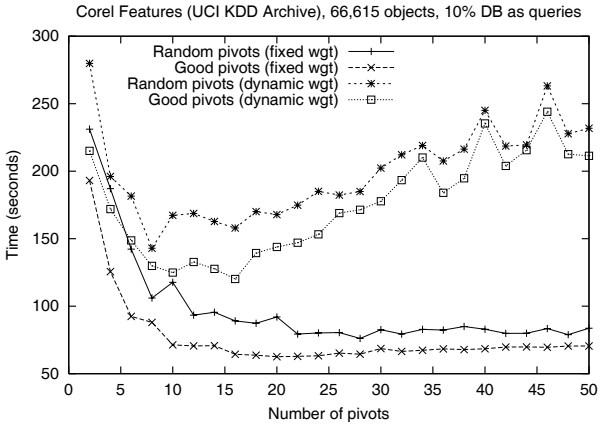

Figure 3: Total time to perform all queries with the Corel image features database.

Table 1 shows a comparison of the proposed index structure with a linear scan and the R*-tree. The results show that the R*-tree is more than 9 times slower than a simple linear scan, due to the high dimensionality of the concatenated feature vectors (89-D). In contrast, the proposed index structure shows a speed up of almost 7 times in the case of fixed-weighted combination, and a speed up of 3.5 times in the case of dynamic-weighted combination. The results also show that the best results are obtained using good pivots.

Method	# opt. pivots	Time (msec)	Improv.
Linear scan	-	64.84	-
R*-tree (fix)	-	597.5	-9.21x
Rnd. piv. (fix)	28	11.43	5.67x
Good piv. (fix)	20	9.39	6.91x
Rnd. piv. (dyn)	8	21.45	3.02x
Good piv. (dyn)	16	18.04	3.59x

Table 1: Corel image features.

Another measure for the effectiveness of the index is the *number of discarded objects* on each query. Figure 4 shows the average number of discarded objects per query. With 20 good pivots, the index is able to discard 50% of the objects, thus avoiding all those distance computations. Using more pivots does not pay off, because the extra comparisons against pivots are more expensive than computing a distance. Therefore, we could expect that the index will perform better with very high dimensional databases, where a distance computation is expensive.

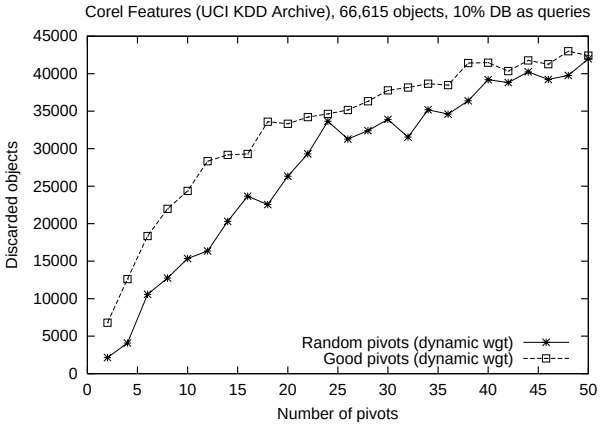

Figure 4: Average number of discarded objects, Corel image features database.

4.2 Corel images database

This database [10] contains several features obtained from images of a subset of the Corel Gallery 380,000 package. The database contains 6,192 images classified into 63 categories. Six features vectors of very high dimensionality (184-D, 165-D, 784-D, 625-D, 784-D, and 30-D) were computed for each image. The feature vectors include color histogram, texture, and convolution descriptors (see [10] and [9] for details on the feature vectors).

Figure 5 shows the results for both fixed-weighted and dynamic-weighted cases, using random and good pivots.

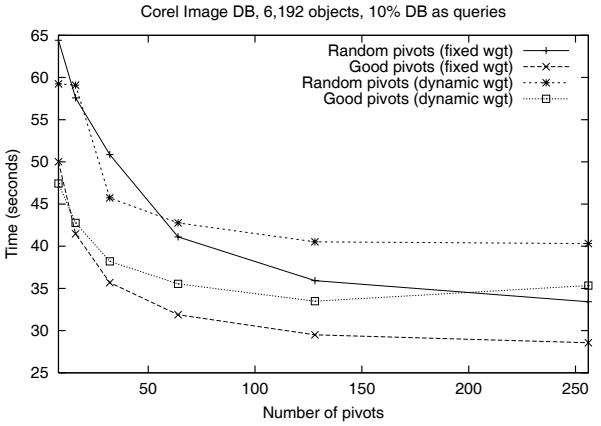

Figure 5: Total time to perform all queries with the Corel images database.

Table 2 shows a comparison of the proposed index structure with a linear scan and the R*-tree. The results are very similar to those presented in Section 4.1. Using the proposed pivot-based index, we obtained an improvement over a linear scan of almost 4 times for the fixed-weighted combination, and more than 3 times for the dynamic-weighted combination. Again, the best results were obtained using

good pivots. The R*-tree was almost 12 times slower than a linear scan on the database. The concatenation of the six feature vectors results in a combined vector of 2,572-D.

It is interesting to note, that on 2,572-D vectors the R*-Tree performs only 12 times slower than linear scan, considering it performs about 9 times slower on just 89-D vectors. This can be attributed to the fact that already with 89-D almost all MBRs in the index overlap, and the increase in overlap is only sublinear in the increasing dimensionality.

Method	# opt. pivots	Time (msec)	Improv.
Linear scan	-	175.12	-
R*-tree (fix)	-	2,048.76	-11.70x
Rnd. piv. (fix)	256	53.99	3.24x
Good piv. (fix)	256	46.14	3.80x
Rnd. piv. (dyn)	256	65.14	2.69x
Good piv. (dyn)	128	54.10	3.24x

Table 2: Corel images database.

Figure 6 shows the average number of discarded objects per query. The index is able to discard more than 50% of the objects with 128 good pivots.

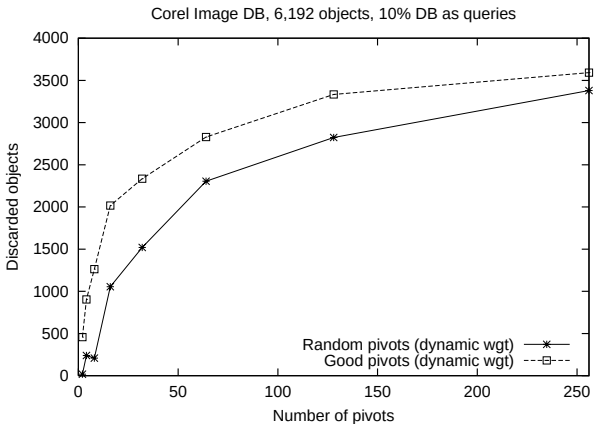

Figure 6: Average number of discarded objects, Corel images database.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a pivot-based index for combinations of feature vectors. It has been shown previously that combinations of features can improve the effectiveness of the similarity search. This work complements these studies, addressing the efficiency problem of searching using more than a single feature to describe a multimedia object. We described a novel index structure and a NN search algorithm for both the fix-weighted and the dynamic-weighted cases, respectively.

The experimental results show that for fix-weighted combinations, the proposed pivot-based index performs very well. We observed an improvement up to a factor 7x over linear scan in the experimental results. For dynamic-weighted combinations, the proposed index also improves the efficiency of the search up to a factor of 4x over linear scan. The R*-tree was an order of magnitude slower than linear scan. This can be explained for the extremely high dimensionality that the concatenated feature vector has. It is well known that the performance of all spatial access methods degrades with dimensionality, a fact known as the *curse of dimensionality*. Thus, we do not expect that any spatial access method will be able to deal appropriately with a concatenation of feature vectors.

We plan in our future work to propose a cost model for our pivot-based index, as well as to study the behavior of this index when stored in secondary memory.

Acknowledgments

This work was partially funded by the German Research Foundation (DFG), Project No. KE 740/6-1, within the strategic research initiative "Distributed Processing and Delivery of Digital Documents" (V3D2), SPP 1041. The first author is on leave from the Department of Computer Science, University of Chile.

We would also like to thank Stefan Rüger and Peter Howarth for kindly allowing us to use the Corel images database.

6. REFERENCES

[1] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *Proc. ACM International Conference on Management of Data (SIGMOD'90)*, pages 322–331. ACM Press, 1990.

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

[3] B. Bustos, D. Keim, D. Saupe, T. Schreck, and D. Vranić. Using entropy impurity for improved 3D object similarity search. In *Proc. IEEE International Conference on Multimedia and Expo (ICME'04)*, 2004.

[4] B. Bustos, G. Navarro, and E. Chávez. Pivot selection techniques for proximity searching in metric spaces. *Pattern Recognition Letters*, 24(14):2357–2366, 2003.

[5] E. Chávez, G. Navarro, R. Baeza-Yates, and J. Marroquín. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, 2001.

[6] M. Hadjieleftheriou. Spatial index library [http://www.cs.ucr.edu/~marioh/spatialindex/].

[7] D. Heesch and S. Rüger. Combining features for content-based sketch retrieval - a comparative evaluation of retrieval performance. In *Proc. 24th BCS-IRSG European Colloquium on IR Research*, pages 41–52. Springer-Verlag, 2002.

[8] S. Hettich and S. Bay. The UCI KDD archive [http://kdd.ics.uci.edu], 1999.

[9] P. Howarth and S. Rüger. Evaluation of texture features for content-based image retrieval. In *Proc. 3rd International Conference on Image and Video Retrieval (CIVR'04)*, LNCS 3115, pages 326–334. Springer, 2004.

[10] M. Pickering and S. Rüger. Evaluation of key frame-based retrieval techniques for video. *Computer Vision and Image Understanding*, 92:217–235, 2003.