An Efficient Bitmap Indexing Method for Similarity Search in High Dimensional Multimedia Databases

Jinguk Jeong, Jongho Nang
Dept. of Computer Science, Sogang University
jhnang@ccs.sogang.ac.kr

Abstract

This paper[1] proposes a new indexing mechanism for the similarity search in high-dimensional multimedia database that quickly filter-out the irrelevant objects using bitmap index, in which the characteristic of each object is approximated as a bit-string. The bits in a bit-string that set to '1' denote *the representative dimensions* of object that their attribute values are relatively larger value than others. Since two objects are dissimilar if their representative dimensions are different so much, the degree of dissimilarity could be computed easily by XORing the bit-strings of two objects and counting the number of '1' s in the resulting bit-string. Experimental results with more than 100,000 images show that a remarkable speed-up could be obtained with the proposed indexing method compared to VA-file and linear scan because of the simple XORing operation in the filtering process, although there are some losses in the search accuracy.

1. Introduction

An important problem of CBMR (Content Based Multimedia Retrieval) systems is to quickly find the *k*NN (k Nearest Neighbors) of a query object. Various approaches such as multidimensional indexing [1, 4] have been proposed for the *k*NN search in multimedia database. So far, however, their performances are known to degrade as the number of dimensions increases because of the problem called "curse of dimensionality" [3]. Some filtering approaches [2, 3] have been proposed to overcome this problem by firstly excluding the irrelevant objects from the candidates of *k*NN search using the approximated values of objects. However, these indexing mechanisms require a lot of computational resources because the distance between approximated values of query and all objects in database should be computed first to filter-out the irrelevant objects and the distance calculation itself is not so trivial (for example, an Euclidean-distance used in VA-file [3]), although they guarantee the exact nearest neighbor search.

This paper proposes a new indexing mechanism following the filtering approach that significantly improves the speed of *k*NN search in high-dimensional multimedia database. In the proposed indexing mechanism, the attribute value of the object at each dimension is approximated as a bit that indicates whether it is a representative dimension or not, rather than its exact value. The *representative dimensions* of the object are the ones that the attribute values of object at those dimensions are relatively larger than others. Using this approximation, the object could be represented as a string of bits each of which indicates whether it is a representative dimension or not. Since two objects are dissimilar if their representative dimensions are different so much, the degree of dissimilarity could be computed easily in the proposed indexing mechanism by XORing the bit-string of two objects and counting the number of '1' s in the resulting bit-string. Although the exact nearest neighbors could not be computed because of some losses caused by a rough approximation and distance metric (XORing), the filtering process itself could be speed-up very much by filtering-out the irrelevant objects using just XORing that is much simpler than others.

We experiment the proposed indexing method with the bitmap indexes of more than 100,000 images that are represented with ColorStructure descriptor [5] of 256 dimensions. The performance results show that the proposed indexing mechanism guarantees a remarkable speed-up over the VA-file and linear scan with accuracy of 90%. It could be used to build a large scale CBMR system with a fast response time since the 100% accuracy of *k*NN search is not required in the high dimensional data space as argued in [6].

2. Bitmap Indexing

Similarity search on multimedia databases is typically measured not on objects directly, but rather on abstractions of objects termed features that are often points in n dimensional vector space where n is the number of used attributes. If the attribute values of object at some dimensions are larger than those of other dimensions (for example, the value of "*red*" bin in a color histogram of image is larger than the values of other bins), the dimensions themselves rather than their values could be an abstraction of the object. These dimensions are called *the representative dimensions* of the object, and could be used to measure the similarity of two objects. That is, if the representative dimensions of two objects are different, we can say they are dissimilar. If we represent this object abstraction as a string of bits of size n each of which denotes whether it is a representative dimension or not, the similarity of two objects could be measured by simply XORing of the bit-strings of two objects. If there are a lot of '1' s in the resulting bit-string, we can

[1] This work was supported by the Basic Research Program of the Korea Science and Engineering Foundation (grant No. R01-2002-000-00141-0).

say they are different so much since their representative dimensions are different so much. Let us formalize this abstraction mechanism in more detail.

Consider a multimedia database Λ ($\Lambda = \{ \vec{O}_k \mid 1 \le k \le n \}$, where $\vec{O}_k$ is the k^{th} object.) consisting of n image objects. When the number of dimensions is m, we can define $\vec{O}_k$ as $\vec{O}_k = (h_k^1, h_k^2, \ldots, h_k^m)$, where h_k^i is the attribute value at i^{th} dimension of the k^{th} image object. Since the meaning of each dimension could be different (so have different value range), the feature vector should be normalized first. Let $\vec{Q} = \{q^1, q^2, \ldots, q^m\}$ be a normalization vector where q^i ($1 \le i \le m$) represents the maximum attribute value at i^{th} dimension. The normalized feature vector of k^{th} image object, $\vec{N}_k$, could computed as follows;

$$\vec{N}_k = (n_k^1, n_k^2, \ldots, n_k^m), n_k^i = h_k^i / q^i, \forall i, 1 \le i \le m \qquad (1)$$

The representative dimensions of image object could be identified by comparing the normalized attribute values of image object. If ε_k is the T^{th} largest normalized attribute value of the feature vector $\vec{N}_k$, then the representative dimension of image object $\vec{O}_k$ would be the ones whose normalized attribute values are larger than ε_k. It leads a way to generate the bit-string of image object $\vec{O}_k$, B_k, as follows;

$$B_k = b_k^1\ b_k^2\ b_k^3 \ldots\ b_k^m, \qquad (2)$$

$$\begin{cases} if\ n_k^i \ge \varepsilon_k, then\ b_k^i = 1 \\ else\ b_k^i = 0 \end{cases}, \text{ for } \forall i, 1 \le i \le m$$

<Figure 1> shows an example of the proposed bitmap indexing, in which the dimensionality of the data space is 8, and let T be 3. In this example, since the 3rd largest attribute value of $\vec{O}_1$, ε_1, is 0.7 at 5th dimension, the 2nd, 4th, and 5th bits of bit-string are set to '1'. The resulting bit-string of $\vec{O}_1$ is '01011000', and represented with just one byte as shown in <Figure 1>. Note that the size of bitmap index for database with n objects is $\left\lceil \frac{n \times m}{8} \right\rceil$ bytes.

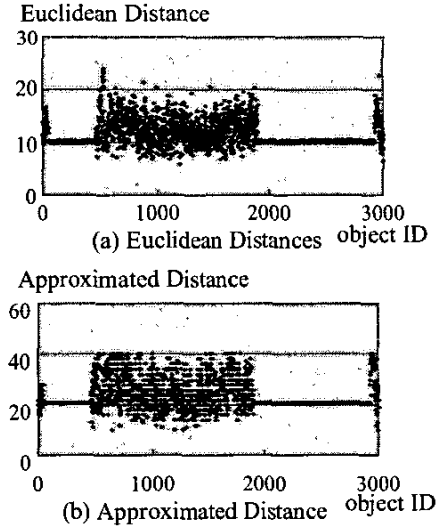
The bit-string is generated for all image objects, and collected to form a bitmap index for multimedia database. The bit-string of query object is compared to the bit-string of all objects in database to filter-out the irrelevant objects. The dissimilarity of between the query and image objects in database could be computed in the proposed indexing mechanism by simply XORing their bit-strings and counting the number of '1' in result bit-string. This simple dissimilarity calculation is based on the characteristic of L_p-norm that the dominant components of L_p-norm are the dimensions on which the points are farthest apart. Although there are some errors in the distance calculation using bit-strings, it would not be a problem since it is used to just filter-out the irrelevant objects as shown in the experimental results.

Object \ Dimension	1	2	3	4	5	6	7	8	ε_k	Bitmap Index
O_1	0.1	0.9	0	0.8	0.7	0	0.2	0.3	0.7	01011000
O_2	0.4	0.6	0.9	0	0	0.5	0.3	0.1	0.5	01100100
O_3	0.2	0	0.5	0	0	0.7	0	0.9	0.5	00100101
O_4	0	0.1	0	0.9	0.5	0.8	0.7	0	0.7	00010110

<Figure 1> An Example Bitmap Index

3. Search Algorithm

The effectiveness of proposed indexing and distance calculation mechanisms would be dependent on how well the bit-string approximates the characteristics of image object. In order to show an effectiveness of the proposed approximation mechanism, we have made an experiment with 3,000 image objects selected from COIL-100 (Columbia Object Image Library), and its results are shown in <Figure 2>, in which the Euclidean and approximated distances to a query image are presented. The attribute values of image object are represented with MPEG-7 ColorStructure descriptor [5] whose dimension size is 256. As shown in this figure, the dissimilarities (or distances) computed by Euclidean and approximated distance calculations are similar to each other, although the maximum distance in approximated calculation is limited to 40 because the most representative 20 dimensions are used in this experiment.



<Figure 2> Comparison of the Euclidean and Approximated Distances

The proposed bit-string that represents the position of representative dimensions is a bit-encoded approximation of image object. The kNN search

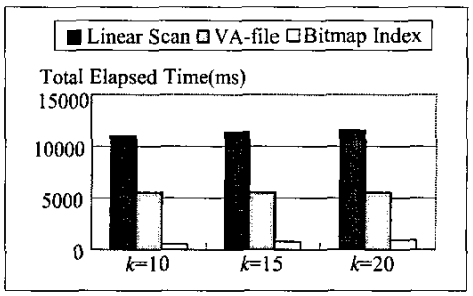
algorithm reflecting this characteristic of bitmap index is described in <Figure 3>. In this algorithm, the bit-strings of all image objects are computed in advance and form a bitmap index for the image database. When the query image is submitted, its attribute values ($\vec{Q}$) and bit-string (Q') are generated. It is compared with all bit-strings in bitmap index to make a candidate objects set (Ω), as shown in procedure **Filtering()**. It returns a set of candidate objects of size P that has the lowest number of '1' s in the resulting bit-strings (r_k), where P is the number of candidate objects and usually a multiple of k. The exact distances (for example, Euclidean distance) between query and the objects in the candidate set is computed in procedure **FinalSearch()** to return the most similar k image objects.

```
// O_k : the attribute values of k-th objects
// B_k : the bit-string of O_k
// Q : the attribute values of query object
// Q' : the bit-string of Q
// Ω : a candidate objects set
// kNN : a set of nearest k image objects
Procedure BitMapSearch( Q ) {
   Q' = BitString( Q ); // Make a bit string by Eq.(2)
   Ω= Filtering( Q' ); // Make a candidate objects set
   kNN = FinalSearch( Q , Ω); // Exact search with Ω
  return (kNN);
}
Procedure Filtering( Q' ) {
   for all B_k (1 ≤ k ≤ n) in BitmapIndex do
     r_k = Q' XOR B_k ;
   end_for
   Select P objects that have the lowest number of '1'
     in r_k , and return them;
}
Procedure FinalSearch( Q , Ω) {
 for all O_k (1 ≤ k ≤ P) in Ω do
    Calculate the Euclidean distance E_k = |O_k - Q| ;
 end_for
 Select k objects that have the lowest Euclidean
      distance, and return them;
}
```

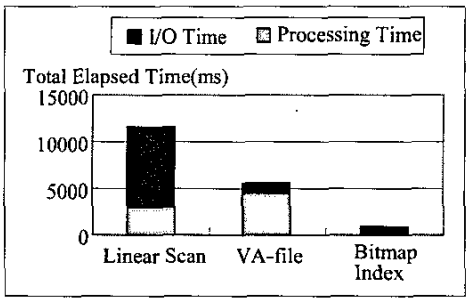
<Figure 3> An Overview of *k*NN Search Algorithm with Bitmap Index

4. Experiments

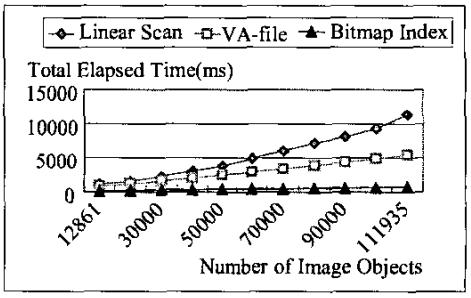
To demonstrate the practical effectiveness and efficiency of proposed bitmap index, we experimented the proposed indexing algorithm with more than 100,000 image objects (12,861 COIL-100 images and 99,074 COREL DRAW images), and compared the performance with the indexing algorithm with VA-file and linear scan algorithm. In this experiment, the attribute values of image objects are represented with MPEG-7 ColorStructure descriptor [5] whose dimension size is 256, as in experiments in <Figure 2>.



(a) Total Elapsed Times of Linear Scan, VA-file, and Bitmap Index when n=111,935 and k=10, 15,20



(b) Profile of Elapsed Times of Linear Scan, VA-file, and Bitmap Index when n=111,935 and k=15



(c) Total Elapsed Time of Linear Scan, VA-file, and Bitmap Index with respect to the number of image objects (n)

<Figure 4> Total Elapsed Times of Linear Scan, VA-file, and Bitmap Index

Our experiments have been computed under the Microsoft Windows XP on Intel Pentium IV 1.5GHz CPU with 512MB of main memory.

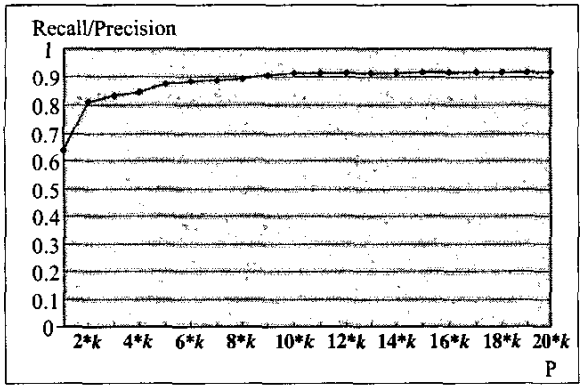
Let us compare the performances of the linear scan, the VA-file, and the bitmap index with respect to the total elapsed time and the precision of *k*NN search. <Figure 4> -(a) shows the total elapsed times of these search algorithms when the number of image object (n) is 111,935, k is 10,15, and 20, and P is k*10. <Figure 4>-(b) shows their profiles when k=15. Let us explain the efficiency of three indexing algorithm with these figures. In the case of *k*NN search with linear scan algorithm, since the attribute values of all objects

should be retrieved for the *k*NN search, a lot of I/O time is required. On the other hand, much more CPU time is required in the VA-file because it should calculate the Euclidean distances for all image objects to make a candidate objects set. Although it helps to reduce the I/O time because only the attribute values of candidate objects are retrieved to find the final *k*NN, it could not reduce the total elapsed time so much as shown in <Figure 4>-(a) and (b).

In the case of *k*NN search with bitmap index, the filtering approach helps to reduce the I/O time for the retrieval of attribute values of image objects, and the simple XOR operations used in the filtering process helps to reduce the CPU time. It leads the total elapsed time of *k*NN search with the bitmap index is 15.3 times faster than the linear scan, and 7.5 times faster than VA-file when k=15. <Figure 4>-(c) shows the scalability of three *k*NN search algorithms, in which the total elapsed times of linear scan, VA-file, and bitmap index are presented as the functions of image database size. As shown in this experiment, the elapsed time of *k*NN search with bitmap index is not increased so much although the number of image objects are increased because some simple XORing operations are additionally required if more image objects are compared for *k*NN search.

The recall and precision of the proposed indexing mechanism would be dependent on the value of P that controls the number of candidate objects. If a larger P value is used in the filtering process, a higher recall/precision would be obtained as shown in <Figure 5> that shows the recall and precision values of the proposed indexing mechanism as a function of P when n=111,935 and k=15. From this experiment, we can find that the recall/precision values of proposed indexing mechanism are about 0.9, and there is a threshold in P. Of course, a more computation is required in **FinalSearch()** when a larger P value is used since more candidate objects are generated by **Filtering()** procedure in <Figure 3>.

Let us compare the additional storages for indexing with VA-file [3] and the proposed bitmap indexing. Of course, with linear scan algorithm, an additional storage is not required since it tries to compute the distances with all image objects in the database. When the attribute (in this descriptor, it is the histogram bin) value is stored as double precision (8bytes), the total number of bytes to store the attributes of all 111,935 image objects is 111,935(objects) * 256(bins) * 8(double) = 229,242,880 bytes. The size of storage required for the bitmap indexes of this image database is $\lceil (111{,}935 \times 256)/8 \rceil$ bytes that would not be a problem because it is only 1.5% overhead. If two bits per dimension are used in VA-file, the total number of bytes for VA-file is $\lceil (111{,}935 \times 256 \times 2)/8 \rceil$, and it is larger than the size of bitmap index.



<Figure 5> Recall and Precision of Bitmap Index

5. Conclusions

In this paper we proposed a new bitmap indexing mechanism for similarity queries in high-dimensional databases. It filters-out the irrelevant image objects by simple XORing of bit-strings of the query image and all images in the multimedia database. We have experimentally shown that the proposed indexing method produced a remarkable speed-up over the linear scan and the VA-file with accuracy of 90%. A more sophisticated normalization mechanism to rank the attribute values of image object would be required to improve the accuracy of the bitmap index. Although the exact nearest neighbors could not be computed with the proposed bitmap index, it would not be a big problem in the real multimedia applications because the retrieval process itself is usually iterative and there is an additional mechanism to help the retrieval process such as relevance feedback. The proposed indexing mechanism could be used to build a CBMR that guarantees a quick response time.

6. References

[1] N. Beckmann, H.P, Kriegal, R. Schneider, and B. Seeger, "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles," *Proceedings of ACM SIGMOD Conference*, 1990.

[2] G.H. Cha, X. Zhu, D. Petkovic, and C.W. Chung, "An Efficient Indexing Method for Nearest Neighbor Searches in High-Dimensional Image Databases," *IEEE Transaction on Multimedia*, Vol. 4, No.1, 2002.

[3] R. Weber, H.J. Schek, and S. Blott, "A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces," *Proceedings of ICVLDB*, 1998.

[4] P.O. Neil, and D. Quass, "Improved Query Performance with Variant Indexes," *Proceedings of ACM SIGMOD Conference*, 1997.

[5] A. Yamada, M. Pickering, S. Jeannin, L. Cieplinski, and Jens, *MPEG-7 Visual part of eXperimentation Model Verision 9.0*, ISO/IEC JTC1/ SC29/WG11/ N3914, January 2001.

[6] K.S. Beyer, J.Goldstein, R.Ramakrishan, and U.Shaft, "When is Nearest Neighbor Meaningful?," *Proceedings of Int'l Conf. on Database Theory*, 1999.